OD Morphing: Balancing Simplicity with Faithfulness for OD Bundling

Yan Lyu, Xu Liu, Hanyi Chen, Arpan Mangal, Kai Liu, *Member, IEEE*, Chao Chen, and Brian Lim

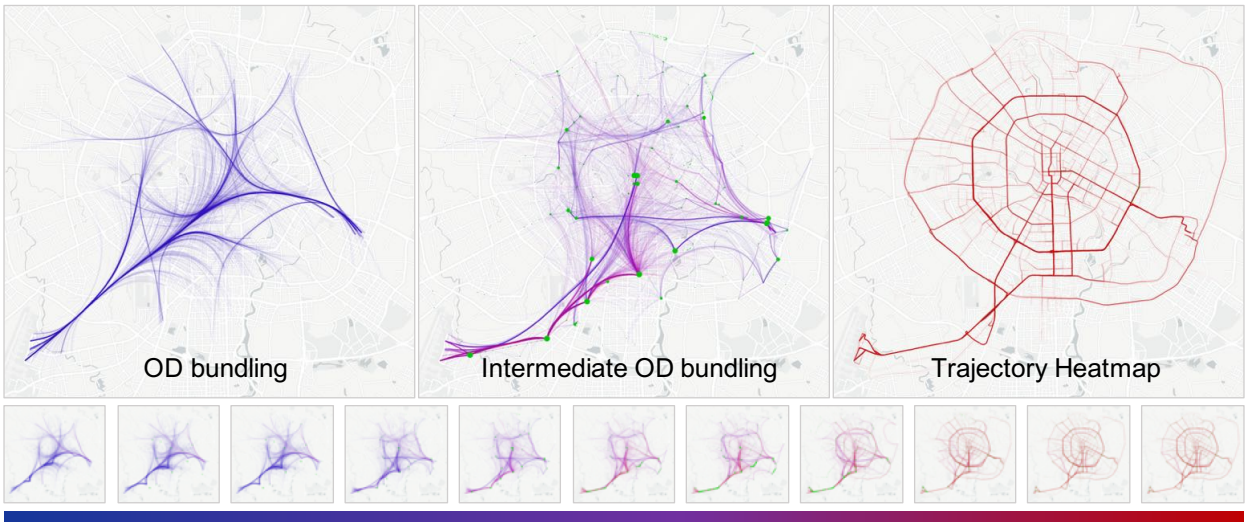


Fig. 1: Morphing transitions from OD bundling (left) to trajectory heatmap (right) improves faithfulness to actual movement paths.

Abstract—OD bundling is a promising method to identify key origin-destination (OD) patterns, but the bundling can mislead the interpretation of actual trajectories traveled. We present OD Morphing, an interactive OD bundling technique that improves geographical faithfulness to actual trajectories while preserving visual simplicity for OD patterns. OD Morphing iteratively identifies critical waypoints from the actual trajectory network with a min-cut algorithm and transitions OD bundles to pass through the identified waypoints with a smooth morphing method. Furthermore, we extend OD Morphing to support bundling at interaction speeds to enable users to interactively transition between degrees of faithfulness to aid sensemaking. We introduce metrics for faithfulness and simplicity to evaluate their trade-off achieved by OD morphed bundling. We demonstrate OD Morphing on real-world city-scale taxi trajectory and USA domestic planned flight datasets.

Index Terms—OD Visualization, Edge Bundling, Trajectory

1 Introduction

The widely used location acquisition technologies, such as GPS on vehicles and in mobile phones, have accumulated huge volumes of geographical mobility data. Visualizing origin-destination (OD) patterns of these data facilitate understanding in a variety of application domains, such as urban planning, transportation and social behavior analysis. However, OD visualization is challenging due to the edge-crossing clutter arising from the large number of OD connections and the requirement of maintaining geographical faithfulness.

- *Yan Lyu and Brian Lim are with the National University of Singapore. E-mail: dcslyuy@nus.edu.sg; brianlim@comp.nus.edu.sg.*
- *Xu Liu is with the Southeast University, China. E-mail: liuxu726@gmail.com.*
- *Hanyi Chen is with the Zhejiang University, China. E-mail: chanhanyi0923@gmail.com.*
- *Arpan Mangal is with the Indian Institute of Technology, Delhi. E-mail: mangalarpan@gmail.com*
- *Kai Liu and Chao Chen are with the Chongqing University, China. E-mail: liukai0807@gmail.com; cschaochen@cqu.edu.cn.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

OD bundling[1] is a popular method to summarize individual OD connections and show high-level connectivity patterns while maintaining geographical faithfulness of origins and destinations [22, 30]. OD connections (also called edges) that are similar with each other in terms of the location of origins and destinations and direction of travel are drawn as tightly bundled curves, providing a simplified connectivity overview. However, researchers argue that bundling techniques may be misleading because the bundled curves suggest unrealistic movement trajectories [58, 59].

For example, consider a visualization of trips from taxi data in a city in China. Fig. 2(a) shows an OD bundling visualization of the taxi trips. The emphasized blue curves suggest the trips between the north and south-east mostly travel along the curves, but their actual trajectories consist of two main paths (Fig. 2(b)): one closer to the city center towards the west and another detouring towards the east. The yellow paths deviate far from the bundled curves and clearly illustrate how OD bundling can be misleading. Although the OD end-points are geographically faithful, the bundled edges are not faithful to the true trajectories. This impedes the understanding of movement data, especially in the domains of transportation and urban planning, where researchers are concerned with not only the OD patterns, but also the movement paths between ODs [55]. Although there are some techniques focusing on path visualization, such as aggregating paths [7,

[1] also called edge bundling where each OD connection is considered as an edge in a graph.

(a) OD bundling (b) Actual paths

Fig. 2: OD bundling on an sample of taxi trips and their actual paths.

[8, 26] and constraining path bundles to follow a reference network [43], these techniques fail to provide a simplified overview for OD patterns.

To address the above issue, we propose OD Morphing, an interactive *morphing* visualization technique to make OD bundling less misleading (more geographical faithful) with actual paths while preserving visual simplicity for OD patterns. We utilize actual paths traveled between OD as the maximum geographical faithfulness. The paths can be obtained by map-matching real trajectories to a network (e.g., road network) and hence consist of sequences of waypoints (e.g., road junctions). To help see the representative and distinct paths of OD bundles, we discover critical waypoints that are visited by many paths. We also define intermediate OD bundles that pass through the critical waypoints, to enable users to distinguish different paths for bundled ODs, and allow us to split bundles into different parts. To help see more detailed subpaths of the OD bundles, we iteratively discover critical waypoints from subpaths and split intermediate OD bundles. This increases the degree of geographical faithfulness along a spectrum. Note also that *topological faithfulness* is increased, as the intermediate OD bundling network includes actual waypoints that flows transit through. To provide a smooth transition between different degrees of faithfulness, we developed a morphing technique that iteratively aligns each bundled OD curve to its waypoint. Furthermore, we proposed a heuristic grid-based min-cut algorithm to identify critical waypoints at interaction speeds. Users can interactively transition between degrees of faithfulness to aid sense-making and balance simplicity of OD patterns with faithfulness to actual paths. In summary, our contributions are:

- A visualization technique, OD Morphing, that enhances OD bundling to be more geographically faithful with actual paths, and provides user interaction for balancing simplicity for OD patterns with faithfulness for real paths.
- A hierarchical waypoint finding method to identify critical transit points from real paths, with a heuristic algorithm to improve computational efficiency.
- A smooth morphing algorithm to transition bundles to pass through waypoints to help users visually track the relationship between the more simple and more faithful intermediate states.
- Evaluation on two real-world datasets show that OD Morphing generates more faithful visualization with higher OD simplicity than pure trajectory bundling.

2 Related Work

There are numerous methods to visualize OD connections (e.g., [17, 50, 54], but we focus on discussing geographically aligned OD bundling methods and how to make them more geographically faithful. We also discuss how trajectory visualizations fail to identify OD patterns.

2.1 OD Bundling

Edge bundling has been proved to be effective in summarizing connectivity patterns and reducing visual clutter for large graphs. It is also known as "OD bundling" when applied to origin-to-destination flow data by taking each OD connection as an edge of a graph. It aggregates the connections that are geometrically close and directionally similar into bundles, providing a simplified overview for OD patterns.

Bundling methods include geometry-based methods that use a control mesh [12, 28, 34], hierarchical edge bundling [21], force directed edge bundling that models with the physics of springs and electrostatic forces [22, 35, 36, 40], and image-based methods [14, 25, 37]. While bundling is computationally expensive, recently, GPU computing has been leveraged in CUBu [44] and FFTEB [29] to process millions of edges in milliseconds. However, despite the popularity of these techniques, edge bundling can lead to biased and misleading results, because the curved edges could deviate far from their actual paths.

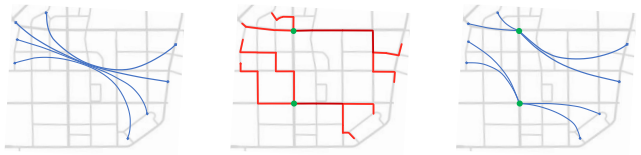
Misleadingness of bundling techniques. There are two ways that OD bundling can be misleading. First, the bundles mislead the perception of connections between unconnected locations as it is hard to distinguish individual connections from tightly bundled curves. This problem has been known as "*edge ambiguity*" [9, 34]. Second, the bundled curves are also misleading by suggesting unfaithful movement trajectories [58, 59], whereas moving objects have their own actual paths that may deviate far from the bundles.

Many solutions have been proposed to address the *edge ambiguity*, such as confluent drawing that only bundles topologically connected edges [9], and edge routing techniques to avoid edge-to-node overlapping [34, 39]. Wang et al. quantified the ambiguity in edge bundling based on the consistency of the bundled edges (i.e., distance, edge length similarity and parallelism) and the clarity of each bundle (i.e., edge curvature and intersection) [47]. Interaction is another way for users to untangle edge ambiguity, such as relaxing local bundling strength [21, 27], dispersing some edges away from a region to reveal underlying nodes [48, 49], and applying digging lens to separate overlapped edges [42].These methods disambiguate bundled edges from unrelated nodes, but may still mislead users to think that the edges are the movement paths tracking through certain detours or directions.

Other than disambiguating edges, we focus on tackling the misleading, unfaithful, curved paths of OD bundling visualizations. There have been a few proposed solutions. Thöny and Pajarola constrained the bundles to follow some reference network such as a road network [43]. Those bundles, however, follow the shortest paths rather than the actual paths of movements. It is noteworthy that some bundling methods have been applied to trajectories directly, instead of OD connections [24–26, 29, 44]. This simplifies trajectory visualization by aggregating similar trails. With proper tuning (e.g., setting small kernel size in KDEEB [25]), the bundled paths can lean towards the actual paths taken and become less misleading to some extend, but they still deviate from the actual path and are hence unfaithful. Zeng et al. further improved KDEEB by adding spatial constraints of roads so that the bundled trajectories can adhere to the major road network in a certain degree [56]. However, the OD connectivity patterns become ambiguous because the bundled trajectories may detour much more than the OD bundles and they may overlap with each other even though their origins or destinations are far from each other. To help users analyze OD patterns associated with path information, Zeng et al. introduced waypoints from paths into OD flow visualization [55]. Users can interactively select entry and exit waypoints in a transportation network. The system filters trajectories passing through the selected waypoints and presents the waypoints-constrained OD view. This method requires users to have domain knowledge to find good waypoints and only two waypoints can be specified. In contrast, in our proposed technique, the critical waypoints are generated in a data-driven manner, indicating the key transit points with high visiting frequency or significant direction changes. Moreover, the waypoints are iteratively selected and visualized to provide interactive degrees of faithfulness for real paths.

2.2 Trajectory Visualization

Different from OD bundling, trajectory visualization techniques can be much more faithful to actual movement paths. Existing techniques usually focus on the route or path information to display detailed motion structure [7, 8], such as traffic on roads [41]. To visualize trajectory big data, techniques such as aggregation, abstraction, and summarization have been proposed. Trajectories can be aggregated by extracting key characteristics, such as the sequential relation of between intermediate locations [7, 8] and geometry features, density and attributes [46]. For example, Andrienko et al. [8] divided trajectories into segments by extracting characteristic points and transformed them into aggregated flows between areas. Huang et al. [23] abstracted taxi trajectories into a traffic graph where a vertex represents a street or a region and an



(a) OD bundle (b) Waypoints and Paths (c) Intermediate OD bundle

Fig. 3: Examples of concepts.

edge indicates taxi traffic between streets or regions. Al-Dohuki [4] summarized trajectories by transforming them into text with reverse geo-coded POI name and speed meta data. By leveraging the rich details of trajectories, some visualization techniques focus on annotating with auxiliary details. For example, Tripvista [18] analyzes microscopic traffic patterns and abnormal behaviors on road intersections; Sun et al. [41] proposed a route-zooming framework by embedding spatio-temporal information into a map. However, all of these studies are concerned with the intermediate locations between origins and destinations. The OD connections are hard to perceive due to the data abstraction, aggregation, or the high level of detail, whereas, our work combines the intermediate locations and their ODs together.

3 Definitions and Requirements

In this section, we clarify key terms and discuss design requirements.

3.1 Definitions

OD Morphing uses the actual paths or real trajectories of movements to improve the geographical faithfulness of OD bundles. Hence, we are interested in both the OD connections and the path of movements.

An **OD connection** is a direct trip from an origin (O) node to a destination (D) node. It is usually drawn as a curve in bundling techniques. Fig. 3(a) shows a bundle of four OD connections.

A **Path** from O to D is a sequence of intermediate nodes from one node to another with links on a network (e.g., roads). A trajectory can be simplified into a path with map-matching techniques [32].

A **Waypoint** for a path is a node on the network that the path passes through. We measure the importance of a waypoint n based on three factors: 1) Number of visits, v_n, i.e., the number of paths passing through n; 2) Degree centrality, d_n, i.e., the number of other waypoints adjacent to n; 3) Average turning angle, a_n, of the paths passing through node n. Intuitively, the failure of a frequently visited node, (i.e., disconnected from the network) is likely to effect many paths; and the failure of a higher degree node would also impact a larger number of neighboring nodes [31, 51]. In transportation research, the number of visits has been used to approximate the capacity of a node [23]. Both the capacity and centrality have been verified as key indicators of node importance [31, 53] on a traffic network. The average turning angle at a junction node is a key measurement in traffic management which approximates the direction change of traffic flow [33, 52]. The larger turning angle, it is more likely to have a longer detour at the node. We also verify the importance of these criteria in our expert interviews, described later. Therefore, the importance, denoted by I_n, is measured by the weighted sum of those factor values, i.e.,

$$I_n = \omega_1 \frac{v_n}{\max v_n} + \omega_2 \frac{d_n}{\max d_n} + \omega_3 \frac{a_n}{\max a_n}, \quad (1)$$

Given a set of paths, we seek to identify a set of **critical waypoints** that minimizes the number of waypoints while maximizing their overall importance, under the constraint that each path visits at least one of the critical waypoints. Such a set of critical waypoints indicates key transit locations of movements. Fig. 3(b) illustrates two critical waypoints shared by actual paths of the four OD connections in Fig. 3(a). The details of identifying critical waypoints are in Section 4.2.

Intermediate OD bundle. An intermediate OD bundle is a result state of transforming an OD bundle to pass through critical waypoints discovered from their actual paths. This is used to distinguish representative paths of OD bundles. Fig. 3(c) shows an intermediate bundle transformed from the OD bundle in Fig. 3(a) to pass through the two critical waypoints, from which we can know two of the four trips share one common waypoint and the other two share another one, indicating there could be two common paths.

3.2 Requirements

From the previous section, we can see that while OD bundling seeks to simplify OD connections, it suffers from having misleading path representations. In contrast, trajectory visualizations faithfully represent actual paths, but fail to identify OD patterns. We seek to produce OD bundling visualizations that satisfy the following requirements:

1. *Simplicity*. The visualization should maintain the property of OD bundling visualizations that emphasize key patterns of OD flows by creating bundled curves that are easy to visually follow end to end.
2. *Faithfulness*. The visualization should accurately represent actual paths and waypoints so that viewers are not misled to perceiving non-existent paths.
3. *Balanced intermediate representation*. Simplicity and faithfulness are conflicting requirements; hence, the visualization should be able to be tuned towards a balance between both key criteria.
4. *Interactive and smooth morphing*. The difference in more simple or more faithful representations can lead to different degrees of interpretation.The visualization should be interactive with smooth transitions (also known as morphing) between different intermediate representations to support "details on demand" for user interaction [20].

4 OD Morphing

Satisfying the aforementioned requirements, we present OD Morphing that unifies the representation of OD bundling with varying degrees of simplicity and faithfulness, and smooth morphing between OD bundling of different degrees. Fig. 4 illustrates the iterative process of morphing the OD Morphing visualization from simple OD bundling to faithful paths. OD Morphing consists of a main pipeline (Fig. 4 middle row) where a simple OD bundling is iteratively morphed towards the faithful paths. The bundling is performed using KDEEB (Section 4.1), though other bundling methods [22] could also be used. We iteratively identify critical waypoints from the path network (Fig. 4 top row; Section 4.2) to represent path information at different degrees of faithfulness. In each iteration of morphing, the critical waypoints are added to previous OD bundles to produce more faithful OD bundles. Since the addition of waypoints to the OD bundling can be abrupt, we present smooth morphing (Fig. 4 bottom row; Section 4.3) to help users to track the transitions. We summarize the whole iterative process in Section 4.4. Finally, to allow interactive morphing between degrees, we extend OD Morphing to an approximate but faster variant with a grid-based min-cut algorithm, which will be introduced in Section 4.5.

4.1 OD bundling

We chose the bundling technique Kernel Density Estimation for Edge Bundling (KDEEB) [25], since it is more computationally efficient than other methods (e.g. [22]), is easy to implement and flexible to be extended for real-time calculations [24, 29, 44]. We briefly describe the standard method here.

In KDEEB, all the straight edges of the input graph are discretized into a set of points by a small sampling step. KDEEB first calculates the density at each edge point using kernel density estimation (KDE). Given a graph consisting of a set of edges $E = \{e_i\}$, KDEEB calculates the local density at an edge point x, denoted by $\rho(x)$, as

$$\rho(x) = \sum_{e_i \in E} \int_{y \in e_i} K(\frac{x-y}{h}), \quad (2)$$

where K is a Gaussian or quadratic kernel function with bandwidth parameter h. To compute $\rho(x)$, KDEEB adopts an image processing function, i.e., blending the edge point data on an 2D texture and reading $\rho(x)$ from the floating-point texture buffer. KDEEB then iteratively sharpens the density by advecting the edge points upwards in the density gradient, i.e., at step $t+1$, the location of $x(t+1)$ is

$$x(t+1) = x(t) + S(t) \frac{\nabla \rho(x,t)}{||\nabla \rho(x,t)|| + \varepsilon}. \quad (3)$$

where ε is a small constant to prevent division by zero. $\nabla \rho(x,t)$ is the density gradient at x estimated at step t. The step size $S(t)$ usually decreases over step t. By iteratively computing the density map ρ and shifting the edge points along the density gradient, the input graph becomes more tightly bundled.

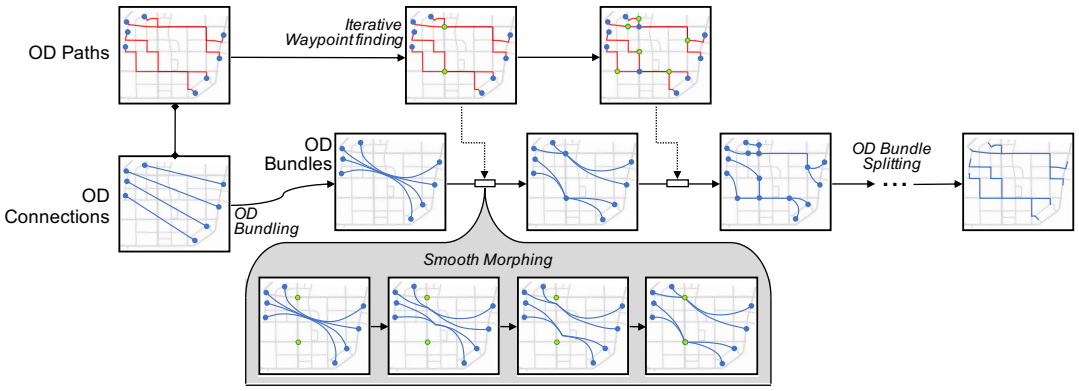


Fig. 4: OD Morphing Pipeline.

4.2 Waypoint Finding

As discussed in Section 3.1, we have defined the importance of a waypoint by the number of path visits, degree centrality and average path turning angles at this waypoint (Eq. 1). To identify a set of critical waypoints to represent the path flows, we first discuss criteria that they should satisfy and then propose a solution to find the set.

Each of the critical waypoints should have high importance, in other words, one objective is to maximize their total importance. Moreover, the number of the critical waypoints should be minimized in order to create a simple representation and clear visualization, but they should be also inclusive: each path should pass through at least one of the critical waypoints. Identifying such a set of critical waypoints as a simplified representation for path flows satisfies Requirement 3 to balance between visual simplicity and path faithfulness. We unify these criteria and define the following objective:

$$\text{minimize} \qquad |W_c| - \sum_{n \in W_c} I_n \tag{4}$$

$$\text{subject to} \qquad P_i \cap W_c \neq \emptyset, 1 \leq i \leq |\{P_i\}| \tag{5}$$

$$W_c \subseteq W \tag{6}$$

where W_c is a subset of critical waypoints from the set of all the waypoints W, and $|W_c|$ denotes the set size. I_n is the importance of a waypoint n, and can be calculated by Eq. 1, $I_n \in [0,1]$. $\{P_i\}$ is the set of paths. Eq. 5 constrains that each path P_i should visit at least one of the critical waypoints. Eq. 4 minimizes the set size of W_c and maximizes the sum of waypoint importance in W_c. It can also be written as

$$\text{minimize} \qquad \sum_{n \in W_c} (1 - I_n). \tag{7}$$

The problem depicted by Eqs. 5, 6 and 7 is actually a variant of min cut problem on a network flow. The min node cut problem aims to find a set of nodes with the minimum total weights (Eq. 7) that can break the flows from a source to a sink. That is, every path between them passes through some member of the cut (Eq. 5).

In order to leverage efficient and well-developed min-cut algorithms to find the set of critical waypoints, we made the following adaptations. First, we formulate a flow network G using the set of paths $\{P_i\}$, i.e., each waypoint on a path is a node of G and there is a direct edge between two nodes if there is a path passing through the two nodes successively. Each node has a weight of $1 - I_n$. Second, a dummy source and a dummy sink are introduced for all the origins and all the destinations, respectively. All the path flows start from the dummy source, visit their own origins immediately and travel along their own paths until their own destinations, and finally converge into the dummy sink. The minimum node cut that breaks the dummy source to the dummy sink is the set of critical waypoints. However, the existing classic min-cut algorithms [13] were designed for the minimum cut through edges. To find the min node cut, we convert the flow network G to the edge-to-node dual network form by converting each edge of G to a node and connecting two nodes with an edge if and only if the corresponding edges in G have a node in common. We apply the Boykov-Kolmogorov algorithm [10] to the dual network to find the minimum edge cut, which is actually the minimum node cut of G. Note that there could be the case that a node is an origin for some paths but also a destination for other paths. This node will be one of the critical waypoints but is unwanted because it is selected for cutting the dummy source to dummy sink rather than high importance. To avoid this case, we split this node into two nodes, one is an origin of some paths, and the other is a destination of other paths. The two nodes have the same geographical locations but there is no flow between them.

Waypoint hierarchy. The set of critical waypoints discovered from all paths could be too coarse for full path representation, because each path is represented by only one waypoint. To create more faithful path representations, we iteratively divide each path into subpaths at the critical waypoints and further search for a set of critical waypoints from subpaths, as depicted in the top row of Fig. 4. Specifically, for each path, we divide it into two subpaths at its critical waypoint. The waypoint becomes a destination for one subpath and an origin for another. We then recursively apply the waypoint-finding method for the newly generated subpaths to iteratively obtain the next set of finer-granularity critical waypoints. After some iterations, some subpaths will consist of endpoints only, i.e., no intermediate waypoints, and will be removed from further waypoint finding steps. The iteration terminates when no subpaths that contain intermediate vertices are found. The iterative discovery of critical waypoints builds a waypoint hierarchy. Each level in the hierarchy adds a set of critical waypoints to the previous OD bundling frame (one square in middle pipeline of Fig. 4). We denote each frame with index l.

4.3 Smooth Morphing

By incrementally adding critical waypoints to OD bundles, and performing further bundling, we incrementally create OD bundles that are more faithful. To aid visual tracking between degrees of faithfulness, OD Morphing applies smooth morphing to transition between OD bundle frames (e.g., see each frame in the middle pipeline in Fig. 4). The smooth morphing technique transitions the previous OD bundles to the next intermediate OD bundles by moving the points on the bundled curves toward the critical waypoint in small interpolation steps (see Fig. 4 bottom row). This animated transition helps keep viewers oriented to the changing positions of specific paths as they move from the OD bundled form to trajectories [19]. While it can still be difficult to track the full network as a whole, viewers can focus on one bundle or a few.

One challenge is to morph OD bundles from the previous frame $l-1$ to the next frame l. However, with KDEEB, the bundles are already stable with respect to the previous kernel density. We aim to move the curves towards the new critical waypoints by defining attractors towards these waypoints. Therefore, points of the bundled curves will be shifted in two directions: one towards the kernel density as the curves move, and another towards the new waypoints.

Consider Fig. 5 as an example to illustrate the details. Focusing on the thick curve e with endpoints o and d, x_i and x_j are the two arbitrary points on this curve, and w is the waypoint. c is the closest point on the curve to the waypoint, i.e., $c = \arg\min_{x_i \in e} ||\overrightarrow{x_i w}||$. The waypoint attraction to c, denoted by $\mathbf{F}(c)$, is directly towards the waypoint, i.e.,

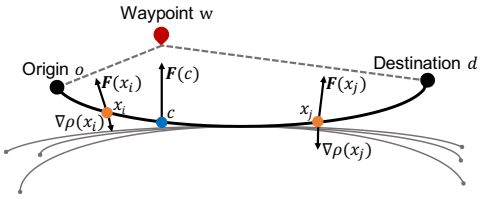


Fig. 5: An example of movement directions of points on a bundled curve. The point c is the closest point to the waypoint w and it will move toward directly to w. The rest of points on the curve, such as x_i and x_j, will move along a combined direction: one towards their projections on $\overrightarrow{ow}$ and $\overrightarrow{wd}$ and the other is local kernel density gradient.

$\overrightarrow{cw}$. For each point like x_i and x_j, the waypoint attraction, denoted by $\mathbf{F}(x_i)$ and $\mathbf{F}(x_j)$, is towards the projections of on the edge between the waypoint and the endpoint, namely, $\mathbf{F}(x_i) \perp \overrightarrow{ow}$ and $\mathbf{F}(x_j) \perp \overrightarrow{wd}$. Moving towards the projections on the straight (dotted) edge between the waypoint and the endpoints (o and d) helps the curve align to its straight subpath.

We combine the waypoint attraction $\mathbf{F}(x)$ with the kernel density gradient $\nabla\rho(x,t)$ to advect a point $x(t)$ to a new location $x(t+1)$, i.e.,

$$x(t+1) = x(t) + S_x(t)\left(r\frac{\nabla\rho(x,t)}{||\nabla\rho(x,t)|| + \varepsilon} + (1-r)\frac{\mathbf{F}(x)}{||\mathbf{F}(x)|| + \varepsilon}\right), \quad (8)$$

where r is for weighting the two directions, $S_x(t)$ is the step size, and ε is a small constant to prevent division by zero. As can be seen in Eq. 8, the bundles in each frame l in OD Morphing depends on the bundles drawn from the previous frame $l-1$. Thus, OD Morphing is not memory-less and bundles cannot be computed from an intermediate frame without knowing the previous less faithful frame.

The weight r is set under the intuition that the closer the point is to the waypoint, the more influential is the pull towards the waypoint, i.e., $\mathbf{F}(x)$; conversely, the closer the point is to the existing endpoint, the more influential is the pull towards the kernel density gradient, i.e.,

$$r = \frac{||\overrightarrow{wx}||}{||\overrightarrow{wx}|| + ||\overrightarrow{xp}||}, \quad (9)$$

where $||\overrightarrow{wx}||$ is the distance from x to waypoint w. $||\overrightarrow{xp}|| = \min(||\overrightarrow{xo}||, ||\overrightarrow{xd}||)$ is the distance from x to one of the endpoints (origin o or destination d, which ever is closer) p of the curve.

It is worth noting that the closest point c should not be affected by the kernel density gradient in order to guarantee that c will be at the location of w after a fixed number of steps. So we force c to move in a straight line towards w. The step size of c at step t, denoted by $S_c(t)$, is

$$S_c(t) = \frac{\tanh(T-t)}{\sum_{t=1}^{T}\tanh(T-t)}||\overrightarrow{cw}||, \quad (10)$$

where T is the total number of steps for the full smooth movement. The expression tanh moves c in a non-linear rate between each frame, such that the movement is at a relatively high speed in the beginning and middle, and slows down at the end. This visually emphasizes clear movement as the bundles deviate from the previous OD and converge to the new waypoints. The step size of other points such as x_i and x_j, is set to be proportional to $S_c(t)$, i.e.,

$$S_x(t) = S_c(t)\frac{||\overrightarrow{xx'}||}{||\overrightarrow{cw}||}, \quad (11)$$

where x' is the projection of x on $\overrightarrow{ow}$ or $\overrightarrow{wd}$. During each step, we implement Laplacian smoothing [15, 25] for each curve to remove small-scale zigzag curving. The smoothing avoids tuning the location of the closest point c to make sure it arrives its waypoint after T steps.

The bottom row of Fig. 4 illustrates four intermediate steps of the smooth morphing from the OD bundle to the intermediate OD bundles. Two of the four trips gradually shift to pass through one waypoint (in the green circle) and the other two trips move towards the other waypoint. The bundled shape is maintained in each step. From the transitions, users can distinguish that trips from the same O to the same D take two different paths via two critical waypoints. We also provide users interaction to control the transition. Users can freely go from bundled overview to split bundles for checking paths and vice versa.

The above computations have to be processed efficiently in order to produce a smooth transition. For each step t, the kernel density $\rho(x,t)$ is computed at a nearly real-time speed by leveraging openGL and GPU, as demonstrated by the real-time KDEEB implementation on dynamic graphs [24]. The movement of each point and Laplacian smoothing are also computed in parallel.

4.4 Iteration process for degrees of faithfulness

OD Morphing is an iterative process with multiple parallel processes. We summarize them here. Given a set of bundled OD curves $\{B\}$ and a set of paths $\{P\}$ taken between those ODs, the recursive scheme has the following steps:

(1) Formulate a flow network with $\{P\}$, identify a set of critical waypoints $\{w\}$ using min-cut algorithms, as described in Section 4.2.
(2) Morph $\{B\}$ to pass through the waypoints $\{w\}$ and formulate intermediate OD bundles $\{B_I\}$, as described in Section 4.3.
(3) Divide each path in $\{P\}$ into subpaths by its critical waypoint. The waypoint becomes a destination for one subpath and an origin for another. The set of all subpaths is denoted by $\{P_{sub}\}$.
(4) Divide each bundled curve in $\{B_I\}$ at the location of its critical waypoint. The set of all sub bundles is denoted by $\{B_{sub}\}$.
(5) Update paths $\{P\}$ with their subpaths and bundles $\{B\}$ with sub-bundles, i.e., $\{P\} \leftarrow \{P_{sub}\}$ and $\{B\} \leftarrow \{B_{sub}\}$.
(6) Remove subpaths that only consist of endpoints from $\{P\}$.
(7) Repeat the above steps until $\{P\}$ is empty.

Each iteration represents a frame in the OD Morphing pipeline. After some iterations, some subpaths will consist of endpoints only, i.e., no intermediate node on each subpath. In this case, no waypoints can be found on these subpaths and they will be removed from $\{P\}$. Meanwhile, their corresponding sub-bundles will be transitioned to the subpaths directly, i.e., the curve of each sub-bundle will be shifted directly toward to the subpath by making a dummy waypoint at the mid point of the subpath and following the same movement function (Eq. 8) where $r = 0$. After T steps, this curve overlaps with its subpath and will be removed from $\{B\}$. For the curves overlapped with subpaths, we cumulatively render them to a traffic heatmap where a red color gradient is applied to indicate the traffic volume, i.e., the number of subpaths. After all the subpaths are rendered, we obtain a trajectory heatmap, which has the maximum geographical faithfulness.

We provide user interaction to tune the degree of faithfulness, i.e., the number of iterations. The user can increase the degree from zero to the maximum. They can also reverse and decrease the degree to check OD connectivity, this can be achieved by caching the intermediate results in each iteration and rendering them backwards. By examining each visualized frame forward and backward, users can find a personal trade-off between simplification for OD patterns and geographical faithfulness for actual paths.

4.5 Interactive-Speed Grid-Based Waypoint Finding

In order to provide efficient interactions, the iterations in OD Morphing should be computed quickly (e.g., within 1 second). However, searching for min-cuts on a large graph has high time complexity, i.e., $O(|V||E|^2)$, where $|V|$ and $|E|$ are numbers of vertices and edges, respectively. Here we propose a heuristic algorithm in order to support computation at interactive speed.

Traditionally, min-cut algorithms are used for flow analysis. However, min-cut algorithms have also been adapted and extensively used in computer vision for pixel labeling problems such as image restoration and segmentation [10]. These image-specific algorithms usually parallelize min-cut search and have higher efficiency than traditional ones that are unparallelizable. This is feasible because the pixel graph is regular, i.e., each pixel node has four neighbors at four directions (up, down, left, right). Algorithms such as Push and Relabel [45] can be parallelized in one direction at a time.

Inspired by this, but applying to graph networks instead of image pixels, we transform the path flows over a network into a grid graph and parallelize the Push and Relabel algorithm to search the minimum grid cut. As demonstrated in Fig. 6 (a), we divide the 2D spatial space

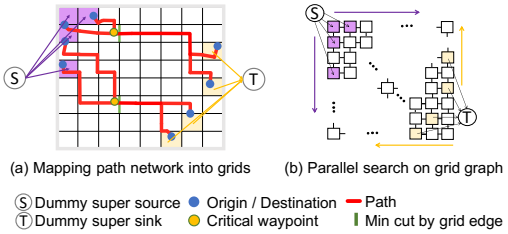


Fig. 6: An example of heuristic waypoint finding. (a) maps the path network into a graph and (b) shows four search directions of parallel Push and Relabel algorithm on the grid graph.

into equal-size squared grids. These grids formulate a regular graph in which each grid is a node and there is an edge between every two neighboring grids, as shown in Fig. 6 (b). Given a set of paths $\{P\}$, we map each path into the grids by their geo-locations with Bresenham's line algorithm [11]. The importance of an edge from grid i to neighbor grid j, denoted by I_{ij}, is set based on the number of the paths passing through the edge, i.e., $I_{ij} = v_{ij}/\max\{v_{ij}\}$, where v_{ij} is the number of the paths successively pass though grid i to j. Similar to our min-cut formulation in Section 4.2, we set the weight of each edge as $1 - I_{ij}$. We introduce a dummy super source node connecting to all the origin grids and a dummy super sink node connecting to all the destination grids. By applying parallel Push and Relabel algorithm on this grid graph, we can obtain a set of grid edges as the minimum cut (drawn as the green short lines in Fig. 6). For each path passing through a grid edge cut, we select the nearest node to the edge as the critical waypoint. Note that, while all waypoints found eventually are still the actual waypoints, they are not found in the same order as in Section 4.2. Grid-based waypoint finding may find less important waypoints first (as according to Eq. 1) and there may be more waypoints found for earlier frames. We define this discrepancy between the optimal approach (Section 4.2) and this grid-based approach as the **optimal gap**. However, these grid edges still indicate popular spatial regions visited by the paths, the distances between approximate waypoints and the optimal waypoints are not large. Moreover, this optimal gap will be compensated by efficiency improvements. We describe the details in Section 7.

5 Evaluation

We evaluate OD Morphing to demonstrate how it increases the faithfulness of OD bundling visualizations by showing examples from two real-world datasets and quantitatively by defining metrics for faithfulness and simplicity. We compare OD Morphing with a baseline and discuss trade-offs in using either method.

5.1 Metric

OD Morphing can generate visualizations that balance between the conflicting objectives of faithfulness and OD bundling simplicity. We formally define metrics for these objectives to allow for a quantitative comparison of OD Morphing with other OD bundling techniques.

5.1.1 Faithfulness

There are multiple ways to define path faithfulness and we consider three interpretations. First, faithfulness can be considered as a measure of how much of the OD bundling visualization shows actual paths and waypoints of the underlying path data. This is important since when viewers study the visualization, they will draw conclusions based on geographically associating the bundles with the base map. We call this first definition *overlap*, and denote a score $f_{overlap}$. If a bundled curve completely overlaps with its actual path, it has the maximum overlap ($f_{overlap} = 1$). If there are no actual waypoints shown in the visualization, it has no faithfulness ($f_{overlap} = 0$). Given a bundled curve e and its actual path P, and denoting their segment(s) of overlap as e_l, the proportion of overlap is defined by the ratio of length of the overlap to the total length of the path, i.e., $f_{overlap} = \text{Len}(e_l)/\text{Len}(P)$.

Another interpretation of faithfulness considers how little the bundles deviate from actual paths. We call this *closeness* and denote a score $f_{closeness}$. Note that closeness may still be misleading since no part of the visualized bundles may actually coincide with the actual paths on the map. Indeed, closeness faithfulness is well represented with trajectory bundling, which we use as a comparison baseline (Section 5.1.3). Similarly, if a bundled curve completely coincides with its actual path, it has maximum closeness ($f_{closeness} = 1$). Closeness is asymptotic and minimum closeness is unobtainable, but as a bundled curve deviates more from its actual path, closeness approaches the minimum ($f_{closeness} \to 0$). Given a bundled curve e and its actual path P, we define deviation of the curve e as the average distance from each sampling point on the curve e, denoted by x, to its corresponding closest point on the path, denoted by c, is $\text{Dev}(e) = \frac{1}{|e|}\sum_{x \in e} ||\vec{xc}||$. Closeness is the inverse of this distance and bounded in [0,1], so we define it as $f_{closeness} = \exp(-\lambda \text{Dev}(e))$, where λ is a tuning parameter and will be tuned for different datasets with various spatial scales.

Faithfulness can also be considered as how closely the bundles adhere to the general curves and shapes of the original trajectories. Although this is not important for our main analysis tasks of identifying detour paths and ODs, it is useful for summarize traffic structure. We define this as *shape* faithfulness and denote a score f_{shape} and compute it based on a shape similarity. The shape similarity between two curves can be measured with the Fréchet distance [5], denoted by $d_{frechet}$, that considers the location and ordering of the points along the curves. For a curve e and its actual path P, the *shape similarity* is obtained by calculating the Fréchet distance $d_{frechet}(e,P)$. We define and mapping it into $[0,1]$ where 1 is the maximum shape faithfulness, i.e., $f_{shape}(e) = \exp(-\gamma d_{frechet}(e,P))$, where γ is a tuning parameter for various spatial scales, such that $f_{shape} = 1$ for identical shapes and $f_{shape} \to 0$ as the edge diverges in shape from the actual trajectory.

5.1.2 Simplicity

The simplicity of edge bundling can be measured by the amount of ink used for rendering [16, 38]. The tighter of the edge bundles, the more ink savings. However, OD simplicity cannot be measured by ink alone, because OD patterns can also be hard to see when the tightly bundled edges detour too much. OD patterns would be clearer if the OD bundles minimize the total curvature as well as the amount of ink needed for rendering. For example, a curve that only bends clockwise is simpler than another (with the same O and D) that bends clockwise, counterclockwise, and clockwise again; a curve that loops back on itself is also less simple than one without loops. Hence, we define OD simplicity by two sub metrics: ink savings, denoted by s_{ink} and curvature, denoted by $s_{curvature}$.

The ink, denoted by I, is measured by rendering the bundles on a 8-bit gray-scale image and counting the pixels of curve drawings under the threshold of 100. We define the ink needed for rendering unbundled OD connections (i.e., node-link diagrams with straight edges) as the maximum ink, denoted by $I_{\max}$. The *ink savings* of bundles can be calculate by $s_{ink} = 1 - I/I_{\max}$.

We measure *curvature* for a curve by summing up the magnitude of turning angle at each sampling point on the curve. Given a point x_k, the turning angle at x_k, denoted by a_k, is $\arccos((x_k - x_{k-1}) \cdot (x_{k+1} - x_k)/(||x_k - x_{k-1}|| \cdot ||x_{k+1} - x_k||))$. The direction change of a curve e, denoted by $A(e)$, is the sum of the turning angles of all the sampling points. The sum of turning angle $A(e) = \sum_{x_k \in e}(a_k)$. Note that because $A(e)$ otherwise could be very large, we normalize it into [0,1] as $s_{angle}(e) = \exp(-A(e))$.

5.1.3 Baseline

Trajectory bundling (also known as trail bundling) has been regarded as a simplification for trajectories. The bundled trajectories visually emphasize dominant or popular trajectories by curving less dominant trajectories towards them. When applied more strongly to tighten the bundles, trajectory bundles may appear to look like OD bundling visualization with a strong degree of OD simplicity. With less tightening, trajectory bundles is more faithful to the original trajectories. Therefore, it can be used as a baseline for comparison. Specifically, we apply KDEEB to the trajectory and path data. By iterating the steps of 1) estimating kernel density, 2) shifting trajectory points towards

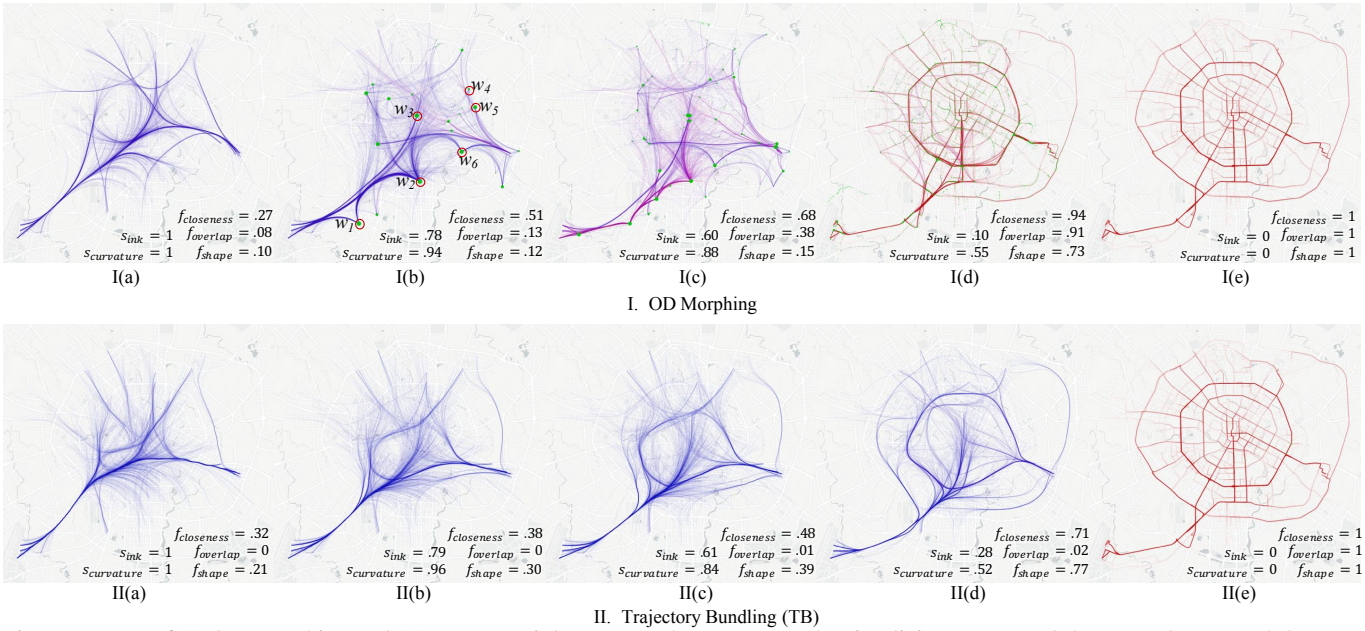


Fig. 7: Frames of (I) OD Morphing and (II) TB on taxi data. s_{ink} and $s_{curvature}$ are the simplicity scores, and $f_{closeness}$, $f_{overlap}$ and f_{shape} are faithfulness scores. The color of the bundles ranges from blue (more simple) to red (more faithful w.r.t. closeness and overlap). The green dots in frames (I) are the critical waypoints. The size of a dot is in proportion to its waypoint importance (Eq. 1).

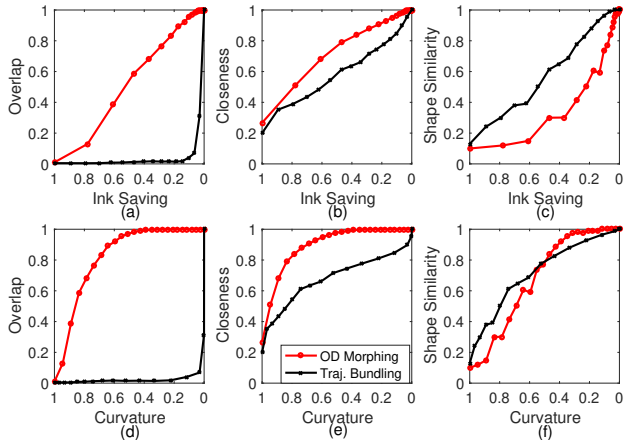


Fig. 8: Plots showing how faithfulness (overlap, closeness, and Fréchet similarity) vary with simplicity (ink saving and curvature), comparing OD Morphing with TB. Points closer to the top-left corner of the plots indicate more faithfulness for the same simplicity (a good result).

the direction of density gradient with a small step size and 3) smoothing, we obtain bundles with a certain tightness (i.e., simplicity) after each iteration, and compare them with corresponding intermediate OD bundles generated by OD Morphing with the same simplicity.

5.2 Application Datasets

We evaluate OD Morphing on two real-world datasets: taxi trajectory data and planned flight route data, and calculate the faithfulness and simplicity for each output frame of OD Morphing, i.e., starting from pure OD bundle to each intermediate OD bundles until pure trajectories visualization. We also compare it with Trajectory Bundling (TB).

5.2.1 Taxi trajectory data

The taxi trajectory data set was collected from Chengdu, China from August 3rd to 30th, 2014. There are 14K taxis with about 1.2 billion GPS records. We extracted 10 million passenger trajectories by detecting the pick-up (origin) and drop-off (destination) activities. After map-matching each passenger trajectory to the road network of Chengdu [32], we obtained actual paths for each OD trip and focus on the long taxi trips during morning peak hours, which formulates a flow graph with 28,174 nodes and 39,723 edges.

Fig. 7 I(a) shows OD bundles of the taxi trips and their actual paths are drawn in Fig. 7 I(e). From these two figures, it can be hard to tell which actual path maps to an OD bundle. OD Morphing addresses this problem by gradually transforming OD bundles to pass through critical waypoints on their roads. As the bundles iteratively pass through critical waypoints, they gradually align to their paths, hence the faithfulness (i.e., overlap and closeness) increases. The simplicity, in terms of ink saving and curvature, however, decreases as expected because the bundles became loose and detour much more when aligning to the roads. Fig. 7 I(b) shows an intermediate state where the first critical set of waypoints (green dots) are visible with their OD bundles are passing through them. We can see the trips between the airport (left bottom) and city center pass through waypoints w_1, w_2 and w_3, indicating these are important transit nodes for the airport transportation. From the trips passing through waypoints w_4, w_5, w_6, we interpret that these trips detour along the ring road as detour from city center to the east. This is further clarified by the later morphing steps I(c) to I(d) where those trips are gradually aligned to the ring roads.

Fig. 7 II shows the intermediate frames of applying KDEEB to the taxi paths in Fig. 7 II(e). From Figs. 7 II(a) to II(d), we can see the trajectory bundles are gradually getting close towards paths in shape. However, maintaining the shape of paths is not overlap faithful because the bundles shift as a whole from their paths. We can see the bundles that followed the ring-shaped highway are still tighten to city center in Figs. 7 II(b) and II(c)). These frames could tell users the bundles may split to or merge from different directions but they cannot provide accurate location on the roads where they split or merge. This explains why TB has lower overlap and closeness faithfulness than OD Morphing at the roughly the same ink saving and curvature scores.

Figs. 8 (a), (b), (d), (e) show how faithfulness varies with simplicity for different frames generated by OD Morphing and TB, respectively. We can see that when morphing from OD bundles to trajectories, both overlap and closeness faithfulness increase as the simplicity decreases for both methods, but at the same simplicity score (i.e., ink saving or curvature), OD Morphing is more faithful than TB.

Although our key metrics *overlap* and *closeness* indicate that OD Morphing is more faithful than TB, some viewers may still perceive TB to be more apparently faithful. This perceived faithfulness is due to the bundles maintaining the shape of original trajectories. The shape similarity describes this property and Figs. 8 (c) and (f) illustrate how TB has higher shape faithfulness and OD Morphing. Yet, quantitatively, TB is also misleading and uninformative regarding our key analysis

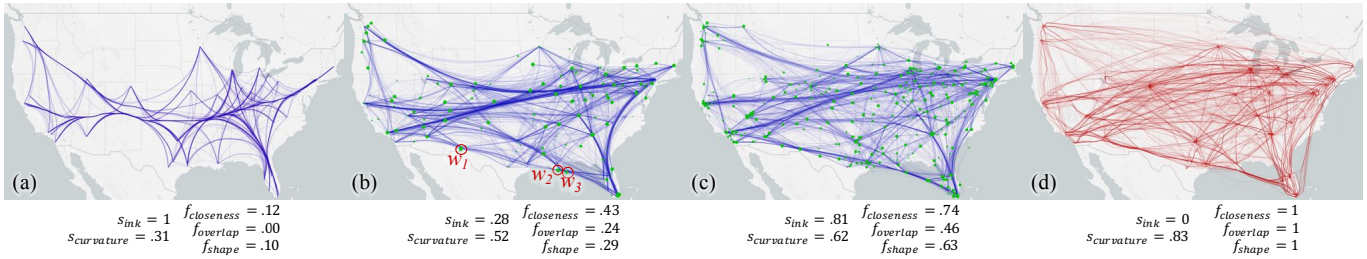


Fig. 9: Frames of OD Morphing on USA flight data. (b) shows the critical waypoints w_1 = ELP, w_2 = REDFN and w_3 = LEV, respectively.

tasks: i) the bundled trajectories are inaccurately placed, and therefore have low *closeness* and *overlap* scores; ii) TB is also unable to identify accurate locations where traffic merges or diverges and representative detour paths for OD patterns.

5.2.2 Planned flight path data

We further evaluate OD Morphing using a dataset of USA domestic planned flight paths collected from a flight tracking website FlightAware [1]. Note that although airplanes are not constrained to roads or train tracks, they still need to follow predefined airways within a margin of deviation. It consists of 4592 planned paths of 9921 USA domestic flights between the busiest 50 airports [2] in one day. Each path consists of a sequence of waypoints and reporting locations with a latitude and a longitude. These paths formulate an airway network with 5,740 nodes and 12,603 edges.

Fig. 9 (a) shows OD bundles of the flights and their planned paths are drawn in Fig. 9 (d). Fig. 9 (b) shows the OD bundles passing through the first critical set of waypoints discovered from their paths. Focusing on flights between southern California, Texas and Florida, we identified three critical waypoints w_1, w_2 and w_3. While OD bundling (Fig. 9 (a)) would suggest flights pass through the mid-continent via "n"-shape arches, OD Morphing (Fig. 9 (b) to (c)) indicates popular intermediate waypoints along somewhat straight paths (though not perfectly straight). Indeed, these waypoints (w_1=ELP, w_2=REDFN, w_3=LEV) serve as navigation aids to specify flight paths through certain airspaces. Depending on many factors (e.g., navigation charges, taxes, probability of delays and rerouting) [6], flights may take alternative paths or detours to travel between OD pairs. Therefore, OD Morphing can help to make these alternative paths clearer to visualize and potentially help with identifying congestion.

6 Evaluating Use Cases with Domain Experts

We qualitatively evaluated the usability and usefulness of OD Morphing with application use sessions and semi-structured interviews with three domain experts separately. These researchers working on transportation and urban planning, are not computing researchers and are not affiliated with the authors. The study serves four purposes: (i) to confirm the importance of our requirements, (ii) to evaluate the interpretability of the system, (iii) to provide real examples to showcase the capability of the system, and (iv) to provide user feedback on the system. Each session lasted for about 1.5 hours, which included two sessions: (1) an introduction to OD Morphing and datasets, followed by a tutorial and a user-practice session, (2) a usage session where experts were asked to use the system to explore patterns, interpret and discuss their findings, and (3) a post-use interview where experts commented on the system capabilities, and compared with the baselines (OD Bundling, Trajectory Heatmap, and Trajectory Bundling). In the following, we present the interpretation by experts, two observed use cases, and report the experts' perceived usefulness and feedback for improvement.

6.1 Interpretability and Usefulness

The experts were asked to use the system to explore both the taxi and flight data described in Section 5.2. A typical use is to first browse the whole morphing process from OD bundles to trajectory heatmap. The expert then quickly gets the initial understanding of "how these vehicles and airplanes detour when traveling" (Expert 1). Then the expert focuses on observing the intermediate states by sliding the scrub bar (scrubber) back and forth. Experts interpreted the intermediate states showing the important intersections and the commonly used roads between certain ODs. Expert 2 (E2) found the tool useful and said "this is very cool. From the beginning you can see the origins, destinations and their connections, and then from the intermediate states you can see where are the important intersections between OD." E1 remarked that "by moving forward and backward, I can see some popular routes also share similar OD". E3 commented that "this movement is quite interesting, it is simulating how the traffic flow gather from or to certain points on roads. The waypoints give me the impression that these are the most busy intersections on the network." These comments demonstrate the usefulness of OD Morphing. The experts confirmed that the proposed criteria for critical waypoints (Sec 3.1) are sufficient and help to identify bottlenecks in the traffic network, and that the requirements for OD Morphing (Sec 3.2) are sufficient.

6.2 Two use cases

We observed two use cases with OD Morphing.

1) Exploring travel detours and road usages. Experts pointed out the morphing process helped them to know how travelers choose certain parts of ring roads as detours. E3 mentioned that "from the morphing I find that ring roads actually bring the most traffic from city center to suburb." She also noticed a bundle of trips between north and east choose the outer ring road instead of a more direct route. She reasoned that "this ring road must be the high-speed road because it is much longer than the route close to the city center." They also noticed that some travelers choose different ring roads even through they have similar OD. As E2 pointed out, there is a group of travelers from airport to the north-west region, "some of them choose this outer ring road, and the others choose the second-ring road."

On the other hand, with Trajectory Bundling (TB), all experts agreed that they could not identify accurate detour paths for the ODs from TB, although it clearly shows the shape of corridors between regions. Some reported being confused. E1 misinterpreted that "the bundled trajectories should be sharing the same roads, but they are not." E2 and E3 found that the trajectory bundles were "too abstract to learn any information except the network structure" which is not useful to the traffic analysis. E3 also noticed that trajectory bundles split in different frames, but "the locations of the splitting keeps shifting which is quite wrong". In contrast, OD Morphing provides more accurate information such as "exact locations where traffic merge"[E3], which is useful to identify the detoured routes.

2) Identifying origins and destinations associated with busy road intersections. By visually tracking the movement of OD bundles to pass through waypoints, all the experts confirmed that the first few intermediate states clearly identified the busiest intersections (waypoints) that are likely to get congested. They could also identify which ODs cause the congestions. E1 mentioned that "the movement clearly shows me that most people from the airport need to go through this waypoint [w_2] to the city center", which demonstrates that the morphing steps can help E1 visually associate the OD bundle with the waypoint. The experts all agreed that OD information is fundamental for analyzing traffic and has been used to alleviate congestions [55]. E1 said "as a transportation researcher, it is important to know where is congestion, and it is nice to know if the congested location comes from certain ODs. Policy makers can do something about the OD." E3 said "[the] trajectory heatmap may be useful to study detailed traffic flow at a

Fig. 10: Critical waypoints generated by G-MC and BK-MC.

certain location, but for my research, I also care about OD because the land use of OD can fundamentally influence the traffic situation. I find OD Morphing is really useful because I can easily get two kinds of information at the same time."

In contrast, with OD bundling or Trajectory Heatmap, all the experts agreed that they could not get such information when using OD bundling and Trajectory Heatmap. E1 "thought the [OD] bundles are the paths but after checking the base map carefully I find that is not true, they did not correspond with any actual roads." While they found the trajectory heatmap is helpful to see the traffic volume, it failed to illustrate OD patterns. Instead, OD Morphing "nicely combines the two information and it is great to see busy intersections and roads taken by major ODs from the morphing in-between." [E2].

6.3 Feedback and Suggestions for Improvement

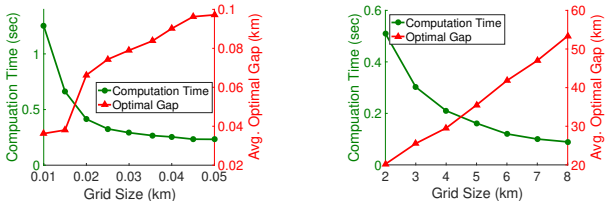
All the experts were confident that OD Morphing will be helpful for the exploration of mobility and transportation datasets. E3 mentioned that "it is very easy to use and will be useful if I can apply it with my subway transportation data. I think it can help me to know the popular OD stations and the important interchange station between them." E1 pointed out "although it takes me some effort to observe the morphing process, but I do get valuable information like commonly used roads under the context of OD." The experts also provided constructive suggestions such as using different colors to show different bundles and adding more interactions such as data selection, highlighting and zooming, which we will consider for the future work.

7 Efficiency Performance

We implemented OD Morphing on a visualization platform OmniSci [3] , which supports the off-screen GPU rendering for large-scale spatial data, and a workstation with an Intel Xeon E5-2650 CPU and a GeForce GTX 1080 Ti GPU. We programmed KDEEB and smooth morphing animation using C++ and OpenGL, and the min-cut algorithm (for waypoint finding) using the graph-tool python graph library.

OD Morphing takes 1.6 seconds and 0.82 seconds to generate a frame for the taxi dataset and flight dataset, respectively. The rendering time per frame takes 0.02 seconds on OmniSci. To evaluate performance with the proposed grid min-cut algorithm (G-MC) for waypoint finding, we compare it with the traditional Boykov-Kolmogorov min-cut algorithm (BK-MC) on the taxi and flight datasets. BK-MC takes around 2.674 seconds to find the minimum node cut on the flow graph, while G-MC takes around 1.2 seconds with a grid size of 0.01×0.01km. By setting a greater grid size, G-MC takes shorter time, as shown in Fig. 11 (a). We measure the gap between G-MC and BK-MC by the average distance between the optimal waypoint and its approximation on each path. The red line in Fig. 11 (a) shows the gap with different grid sizes. We can see the gap becomes larger with a larger grid size. We plot the two sets of waypoints generated by BK-MC and G-MC with grid size of 0.01km in Fig. 10. We can see most of waypoints from the two sets are overlapping with each other, indicating small gap.

On flight data, BK-MC takes around 2.437 seconds to find the minimum node cut, while G-MC takes around 0.5 seconds with a grid size of 2×2km. As shown in Fig. 11 (b), the computation time decreases while optimal gap increases as the grid size becomes larger. It is worth to note that the gap of the flight data is much larger than that of taxi data. This is because each flight path consists of very sparse nodes and flight paths are much longer than taxi paths. The average distance between two successive nodes on a flight path is 87 km.



(a) Taxi dataset (b) Flight dataset

Fig. 11: Efficiency vs optimal gap.

8 Discussion

We discuss how to generalize the use of OD Morphing to application domains, other datasets, and improvements to reducing misleadingness.

Application domains. By providing a seamless way to combine OD bundling and trajectory visualization, OD Morphing helps users to understand key origin-destination connections and the paths that connect them at various waypoints. This is useful for application domains such as urban or transportation planning to help planners find highly connected regions and their popular transportation spots to identify causes of congestion [57].

Generalization to other datasets. OD Morphing can be readily applied to, but not limited to, trajectories of any moving physical objects in 2D or 3D graphs, such as vehicle GPS trajectories on a road network and flight trajectories on an airway network. OD Morphing can also be applied to datasets without or with very sparse trajectory information, such as animal trafficking, plant pathogen and pest spread, and migration. Techniques such as shortest path matching [43] can be used to estimate likely paths taken. OD Morphing can then be applied to visualize popular paths from the OD bundling visualization, but with the caveat that these paths and waypoints were actually estimated rather than extracted from actual data. However, OD Morphing is not applicable to the broadcast communication networks and social relationship networks where a data item cannot be distinguished by an origin and a destination.

Other misleadingness. OD Morphing addresses the misleadingness problem of OD bundling by adding actual waypoints to OD bundling visualizations and improving path faithfulness. However, some ambiguity can remain when some critical waypoints are obscured or covered by other intermediate OD bundles. OD Morphing can be extended to include edge ambiguity methods, such as edge routing [39] to further shift curves to avoid revealed critical waypoints. We will also consider to use colors to differentiate OD directions by applying existing bundling techniques such as [40, 44].

9 Conclusion

We have presented a novel OD bundling technique, OD Morphing, that improves geographical faithfulness to actual paths while preserving visual simplicity for OD patterns. OD Morphing recursively identifies critical waypoints from the actual trajectory network with a min-cut algorithm and transitions OD bundles to pass through identified waypoints with a smooth morphing method. Next, we extended OD Morphing to support bundling at interaction speeds to enable users to interactively transition between degrees of faithfulness to aid sensemaking. We introduce metrics for faithfulness and simplicity to evaluate their trade-off achieved by OD morphed bundling. We evaluated OD Morphing on city-scale taxi trajectory and domestic USA flight datasets, showing that OD Morphing generates more faithful visualizations with higher OD simplicity than traditional trajectory bundling.

Acknowledgments

We thank the domain experts for their expert evaluation and feedback. We also thank Hangxin Lu, Shuqi Wang, Ashraf Abdul for their assistance. This work was supported in part by Ministry of Education, Singapore, and the National Natural Science Foundation of China under Grant No. 61872049.

REFERENCES

[1] https://flightaware.com/

[2] https://en.wikipedia.org/wiki/List_of_the_busiest_airports_in_the_United_States.

[3] https://www.omnisci.com/

[4] S. Al-Dohuki, Y. Wu, F. Kamw, J. Yang, X. Li, Y. Zhao, X. Ye, W. Chen, C. Ma, and F. Wang. Semantictraj: A new approach to interacting with massive taxi trajectories. *IEEE Transactions on Visualization & Computer Graphics*, 23(1):11–20, 2017.

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

[6] G. Andrienko, N. Andrienko, G. Fuchs, and J. M. C. Garcia. Clustering trajectories by relevant parts for air traffic analysis. *IEEE transactions on visualization and computer graphics*, 24(1):34–44, 2017.

[7] G. Andrienko, N. Andrienko, S. Rinzivillo, M. Nanni, D. Pedreschi, and F. Giannotti. Interactive visual clustering of large collections of trajectories. In *IEEE Symposium on Visual Analytics Science and Technology*, pages 3–10, 2009.

[8] N. V. Andrienko and G. L. Andrienko. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization & Computer Graphics*, 17(2):205–219, 2011.

[9] B. Bach, N. H. Riche, C. Hurter, K. Marriott, and T. Dwyer. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE Transactions on Visualization & Computer Graphics*, 23(1):541–550, 2017.

[10] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):1124–1137, 2004.

[11] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems journal*, 4(1):25–30, 1965.

[12] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization Computer Graphics*, 14(6):1277–1284, 2008.

[13] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[14] O. Ersoy, C. Hurter, F. Paulovich, G. Cantareiro, and A. Telea. Skeleton-based edge bundling for graph visualization. *IEEE Transactions on Visualization Computer Graphics*, 17(12):2364–2373, 2011.

[15] D. A. Field. Laplacian smoothing and delaunay triangulations. *Applied Numerical Mathematics*, 4(6):709–712, 1988.

[16] E. R. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. In *IEEE Pacific Visualization Symposium*, pages 187–194, 2011.

[17] D. Guo and X. Zhu. Origin-destination flow data smoothing and mapping. *IEEE Transactions on Visualization & Computer Graphics*, 20(12):2043–2052, 2014.

[18] H. Guo, Z. Wang, B. Yu, H. Zhao, and X. Yuan. Tripvista: Triple perspective visual trajectory analytics and its application on microscopic traffic data at a road intersection. In *IEEE Pacific Visualization Symposium*, pages 163–170, 2011.

[19] J. Heer and G. Robertson. Animated transitions in statistical data graphics. *IEEE transactions on visualization and computer graphics*, 13(6):1240–1247, 2007.

[20] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. *ACM Queue*, 10(2):30, 2012.

[21] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization & Computer Graphics*, 12(5):741–748, 2006.

[22] D. Holten and J. J. Van Wijk. Force-directed edge bundling for graph visualization. In *Computer graphics forum*, volume 28, pages 983–990. Eurographics, 2009.

[23] X. Huang, Y. Zhao, C. Ma, J. Yang, X. Ye, and C. Zhang. Trajgraph: A graph-based visual analytics approach to studying urban network centralities using taxi trajectory data. *IEEE Transactions on Visualization & Computer Graphics*, 22(1):160–169, 2016.

[24] C. Hurter, O. Ersoy, S. I. Fabrikant, T. R. Klein, and A. C. Telea. Bundled visualization of dynamic graph and trail data. *IEEE Transactions on Visualization & Computer Graphics*, 20(8):1141–1157, 2014.

[25] C. Hurter, O. Ersoy, and A. Telea. Graph bundling by kernel density estimation. In *Computer Graphics Forum*, volume 31, pages 865–874. Eurographics, 2012.

[26] C. Hurter, S. Puechmorel, F. Nicol, and A. Telea. Functional decomposition for bundled simplification of trail sets. *IEEE Transactions on Visualization & Computer Graphics*, 24(1):500–510, 2018.

[27] C. Hurter, A. Telea, and O. Ersoy. Moleview: An attribute and structure-based semantic lens for large element-based plots. *IEEE Transactions on Visualization & Computer Graphics*, 17(12):2600–2609, 2011.

[28] A. Lambert, R. Bourqui, and D. Auber. Winding roads: Routing edges into bundles. In *Computer Graphics Forum*, volume 29, pages 853–862. Eurographics, 2010.

[29] A. Lhuillier, C. Hurter, and A. Telea. Ffteb: Edge bundling of huge graphs by the fast fourier transform. In *IEEE Pacific Visualization Symposium*, pages 190–199, 2017.

[30] A. Lhuillier, C. Hurter, and A. Telea. State of the art in edge and trail bundling techniques. In *Computer Graphics Forum*, volume 36, pages 619–645. Eurographics, 2017.

[31] H. Liu, Y. Gao, L. Lu, S. Liu, H. Qu, and L. M. Ni. Visual analysis of route diversity. In *IEEE Symposium on Visual Analytics Science and Technology*, pages 171–180, 2011.

[32] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 352–361. ACM, 2009.

[33] A. Lozano, F. Granados, and A. Guzmán. Impacts of modifications on urban road infrastructure and traffic management: a case study. *Procedia-Social and Behavioral Sciences*, 162:368–377, 2014.

[34] S.-J. Luo, C.-L. Liu, B.-Y. Chen, and K.-L. Ma. Ambiguity-free edge-bundling for interactive graph visualization. *IEEE Transactions on Visualization Computer Graphics*, 18(5):810–821, 2012.

[35] Q. Nguyen, P. Eades, and S.-H. Hong. Streameb: Stream edge bundling. In *International Symposium on Graph Drawing*, pages 400–413. Springer, 2012.

[36] Q. Nguyen, S.-H. Hong, and P. Eades. Tgi-eb: A new framework for edge bundling integrating topology, geometry and importance. In *International Symposium on Graph Drawing*, pages 123–135. Springer, 2011.

[37] V. Peysakhovich, C. Hurter, and A. Telea. Attribute-driven edge bundling for general graphs with applications in trail analysis. In *IEEE Pacific Visualization Symposium*, pages 39–46, 2015.

[38] S. Pupyrev, L. Nachmanson, S. Bereg, and A. E. Holroyd. Edge routing with ordered bundles. In *International Symposium on Graph Drawing*, pages 136–147. Springer, 2011.

[39] S. Pupyrev, L. Nachmanson, S. Bereg, and A. E. Holroyd. Edge routing with ordered bundles. *Computational Geometry*, 52:18 – 33, 2016.

[40] D. Selassie, B. Heller, and J. Heer. Divided edge bundling for directional network data. *IEEE Transactions on Visualization Computer Graphics*, 17(12):2354–2363, 2011.

[41] G. Sun, R. Liang, H. Qu, and Y. Wu. Embedding spatio-temporal information into maps by route-zooming. *IEEE Transactions on Visualization & Computer Graphics*, (5):1506–1519, 2017.

[42] A. Telea and O. Ersoy. Image-based edge bundles: Simplified visualization of large graphs. In *Computer Graphics Forum*, volume 29, pages 843–852. Eurographics, 2010.

[43] M. Thöny and R. Pajarola. Vector map constrained path bundling in 3d environments. In *ACM SIGSPATIAL International Workshop on GeoStreaming*, pages 33–42, 2015.

[44] M. Van Der Zwan, V. Codreanu, and A. Telea. Cubu: Universal real-time bundling for large graphs. *IEEE Transactions on Visualization & Computer Graphics*, 22(12):2550–2563, 2016.

[45] V. Vineet and P. Narayanan. Cuda cuts: Fast graph cuts on the gpu. In *IEEE Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008.

[46] K. Vrotsou, H. Janetzko, C. Navarra, G. Fuchs, D. Spretke, F. Mansmann, N. Andrienko, and G. Andrienko. Simplifly: A methodology for simplification and thematic enhancement of trajectories. *IEEE Transactions on Visualization & Computer Graphics*, 21(1):107–121, 2015.

[47] Y. Wang, Q. Shen, D. Archambault, Z. Zhou, M. Zhu, S. Yang, and H. Qu. Ambiguityvis: Visualization of ambiguity in graph layouts. *IEEE Transactions on Visualization & Computer Graphics*, 22(1):359–368, 2016.

[48] N. Wong and S. Carpendale. Using edge plucking for interactive graph exploration. In *IEEE Symposium on Information Visualization*, pages 51–52, 2005.

[49] N. Wong, S. Carpendale, and S. Greenberg. Edgelens: An interactive method for managing edge congestion in graphs. In *IEEE Symposium on*

Information Visualization, pages 51–58, 2003.

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

[51] J. Wu, Z. Gao, and H. Sun. Effects of the cascading failures on scale-free traffic networks. *Physica A: Statistical Mechanics and its Applications*, 378(2):505–511, 2007.

[52] L. Xin, D. Yang, Y. Chen, and Z. Li. Traffic flow characteristic analysis at intersections from multi-layer spectral clustering of motion patterns using raw vehicle trajectory. In *IEEE Intelligent Transportation Systems Conference*, pages 513–519, 2011.

[53] M. Xu, J. Wu, M. Liu, Y. Xiao, H. Wang, and D. Hu. Discovery of critical nodes in road networks through mining from vehicle trajectories. *IEEE Intelligent Transportation Systems Conference*, (99):1–11, 2018.

[54] Y. Yang, T. Dwyer, S. Goodwin, and K. Marriott. Many-to-many geographically-embedded flow visualisation: an evaluation. *IEEE Transactions on Visualization & Computer Graphics*, 23(1):411–420, 2017.

[55] W. Zeng, C.-W. Fu, S. Müller Arisona, A. Erath, and H. Qu. Visualizing waypoints-constrained origin-destination patterns for massive transportation data. In *Computer Graphics Forum*, volume 35, pages 95–107. Eurographics, 2016.

[56] W. Zeng, Q. Shen, Y. Jiang, and A. Telea. Route-aware edge bundling for visualizing origin-destination trails in urban traffic. *Computer Graphics Forum*, 2019.

[57] Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. In *ACM International Conference on Ubiquitous Computing*, pages 89–98, 2011.

[58] Z. Zhou, L. Meng, C. Tang, Y. Zhao, Z. Guo, M. Hu, and W. Chen. Visual abstraction of large scale geospatial origin-destination movement data. *IEEE Transactions on Visualization & Computer Graphics*, 25(1):43–53, 2019.

[59] D. Zielasko, B. Weyers, B. Hentschel, and T. W. Kuhlen. Interactive 3d force-directed edge bundling. In *Computer Graphics Forum*, volume 35, pages 51–60. Eurographics, 2016.