RouteFlow: Trajectory-Aware Animated Transitions

Duan Li*
School of Software
Tsinghua University
Beijing, China
liduan429@gmail.com

Xinyuan Guo*
School of Software
Tsinghua University
Beijing, China
yczddgj@126.com

Xinhuan Shu
School of Computing
Newcastle University
Newcastle
Upon Tyne, United Kingdom
xinhuan.shu@gmail.com

Lanxi Xiao
Academy of Arts and Design
Tsinghua University
Beijing, China
tarolancy@gmail.com

Lingyun Yu
School of Advanced Technology
Xi'an Jiaotong-Liverpool University
Suzhou, Jiangsu, China
Lingyun.Yu@xjtlu.edu.cn

Shixia Liu†
School of Software
Tsinghua University
Beijing, China
shixia@tsinghua.edu.cn

Abstract

Animating objects' movements is widely used to facilitate tracking changes and observing both the global trend and local hotspots where objects converge or diverge. Existing methods, however, often obscure critical local hotspots by only considering the start and end positions of objects' trajectories. To address this gap, we propose RouteFlow, a trajectory-aware animated transition method that effectively balances the global trend and local hotspots while minimizing occlusion. RouteFlow is inspired by a real-world bus route analogy: objects are regarded as passengers traveling together, with local hotspots representing bus stops where these passengers get on and off. Based on this analogy, animation paths are generated like bus routes, with the object layout generated similarly to seat allocation according to their destinations. Compared with state-of-the-art methods, RouteFlow better facilitates identifying the global trend and locating local hotspots while performing comparably in tracking objects' movements.

CCS Concepts

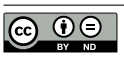
• **Human-centered computing** → **Visualization techniques**; **Information visualization**.

Keywords

trajectory data, animation, edge bundling

ACM Reference Format:
Duan Li, Xinyuan Guo, Xinhuan Shu, Lanxi Xiao, Lingyun Yu, and Shixia Liu. 2025. RouteFlow: Trajectory-Aware Animated Transitions. In *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26–May 01, 2025, Yokohama, Japan.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3706598.3714300

*Both authors contributed equally to this research.
†Shixia Liu is the corresponding author.



CHI '25, Yokohama, Japan

ACM ISBN 979-8-4007-1394-1/25/04
https://doi.org/10.1145/3706598.3714300

1 Introduction

Animating objects' movements is widely used to facilitate tracking changes and observing both the global trend and local hotspots where objects converge or diverge [14, 43]. For example, by animating bird migration data [46], users can observe birds' movements, understand the migration trend, and identify highly active locations where birds converge to cross the straits or diverge to bypass mountains (Fig. 1(a)). Here, the global trend provides valuable insights into broader movement patterns, while the local hotspots serve as strategic locations for observation and analysis [30, 32].

Many research efforts have been directed toward developing techniques for animated transitions, aimed at helping users track objects' movements. These efforts mainly focus on adjusting various animation parameters from temporal (e.g., speed [15], staging [26], staggering [13]) and spatial (e.g., animation paths [17, 49]) perspectives. Recent studies have further advanced these techniques. Zheng *et al.* [55] divided transitions into groups and animated them sequentially, thereby breaking down complex animations into simpler ones (Fig. 1(b)). Wang *et al.* [49] used vector fields to coordinate group movements and reduce occlusion by spatially separating animation paths (Fig. 1(c)). However, all these methods only consider the start and end positions in the objects' trajectories. Although effective in conveying global trends, they often obscure critical local hotspots along the movement trajectories.

Recognizing this gap, we aim to design an animated transition method that considers the movement trajectories of objects. By using these trajectories, the animations can effectively reveal both the global trend and local hotspots. Thus, our method provides a clearer understanding of local areas of high activity in their global context. However, designing such animations is non-trivial. First, balancing the global trend and local hotspots in animation remains challenging. Overemphasizing local hotspots may result in excessive branching areas, impeding the identification of the global trend. Conversely, stressing the global trend heavily may obscure important local hotspots. Second, reducing occlusion in animated transitions is imperative yet difficult, especially when multiple objects move simultaneously. They may occlude each other, thus significantly increasing the difficulty of tracking their movements. Occlusion becomes even more severe in local hotspots, where many objects converge or diverge.

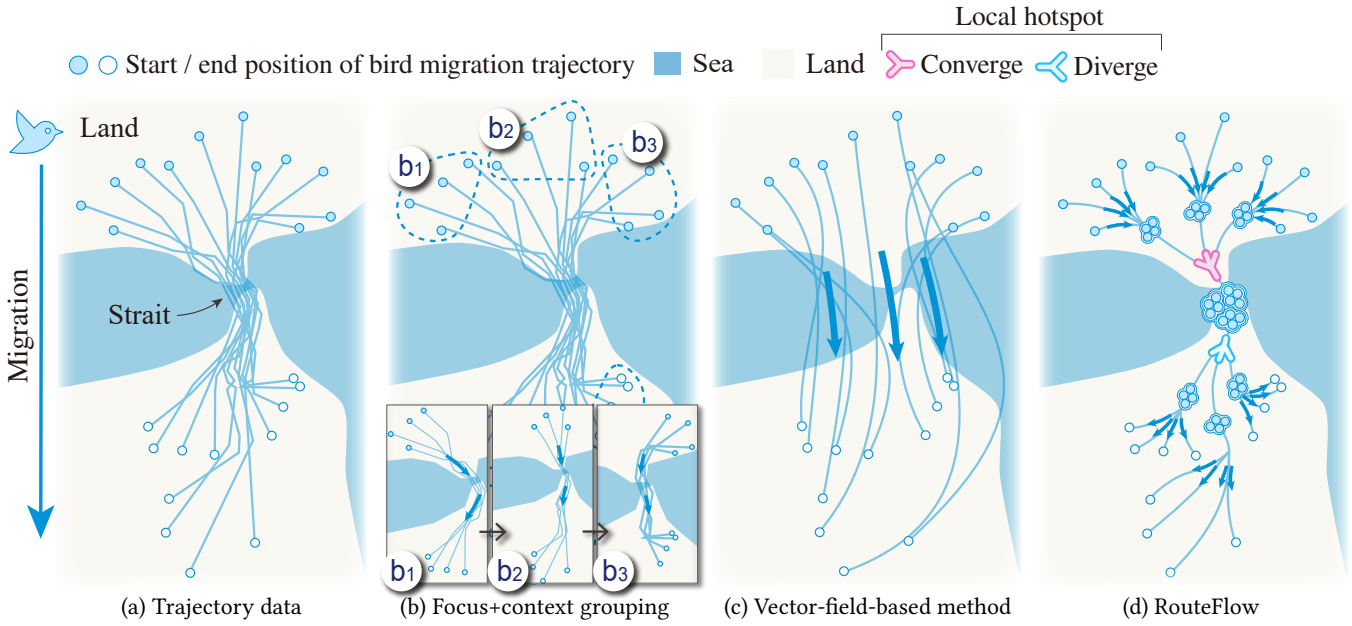


Fig. 1. A comparison of three animated transition methods on a bird migration example.

To address these challenges, our animation design utilizes a real-world bus route analogy: groups of passengers board the same bus at different stops, travel together along the shared routes, and disembark at designated stops. We regard objects as passengers traveling together, with local hotspots representing various bus stops. Based on this, we animate objects following the shared paths, converging or diverging at local hotspots. As such, users can observe the global trend and identify local hotspots, similar to observing overall bus routes and identifying frequently visited stops. In this analogy, we regard 1) achieving a balance between the global trend and local hotspots as planning bus routes for efficiency and effectiveness. These bus routes should not only be of minimal length but also meet passengers' travel demands. Besides, we consider 2) reducing occlusion by allocating passengers to respective seats in the process. Consequently, we formulate the problem of designing animated transitions for trajectory data as a sequential optimization of two sub-problems: bus routing and seat allocation.

Based on this formulation, we propose RouteFlow, a trajectory-aware animated transition method comprising two steps: trajectory-driven path generation and object layout generation. As shown in Fig. 1(d), we create "bundled" animation paths for groups of objects that share similar movement trajectories. These animation paths are generated by a bottom-up hierarchical edge bundling algorithm, which progressively bundles similar trajectories, level by level, effectively capturing both the global trend and local hotspots. To minimize occlusion, we apply an incremental circle packing algorithm, sequentially generating the layout at each local hotspot. The animation is then rendered using an interpolation-based method.

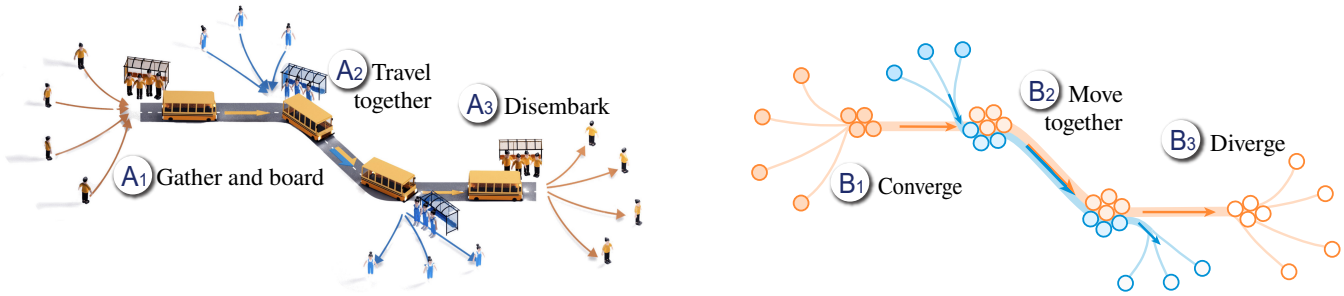
We evaluate RouteFlow through a quantitative experiment on real-world data and a controlled user study. The results indicate that compared with the state-of-the-art methods, RouteFlow better facilitates identifying the global trend and locating local hotspots while performing comparably in tracking objects' movements. The main contributions of our work include:

- A formulation of designing animated transitions as a sequential optimization of bus routing and seat allocation problems.
- RouteFlow, a trajectory-aware animated transition method that consists of a bottom-up hierarchical edge bundling algorithm and an incremental circle packing algorithm. The open-source implementation is available at https://github.com/Trajectory-Anim/Trajectory-Aware-Animated-Transitions.
- A quantitative experiment and a user study evaluating performance on tracking objects' movements, identifying the global trend, and locating local hotspots.

2 Related Work

There are two main tasks for animated transitions: **tracking objects' movements** and **identifying the trend**. Most existing efforts focus on **tracking objects' movements** from temporal and spatial perspectives. The temporal perspective includes adjustments such as refining movement speed [15], staging [24, 26, 55], and staggering [13]. The spatial perspective focuses on the animation paths of objects [17, 26, 49, 50]. Our work falls into the latter perspective.

Animation paths play a crucial role in **tracking objects' movements** [18, 37]. According to Heer and Robertson [26], simple trajectories are effective in minimizing confusion and enhancing predictability, thereby making it easier for users to track objects' movements. A direct method to achieve simplicity is to use straight lines connecting the start and end positions of the movement [11]. To provide more natural and engaging movements while maintaining simplicity, later research used smooth curves, such as arcs [16, 50] and B-splines [6]. Building on these advancements, Du *et al.* [17]



(a) Passengers gather and board the bus, travel together, and disembark

(b) Objects converge, move together, and diverge

Fig. 2. Illustration of our analogy.

explored bundling animation paths to coordinate group movements, which improved group tracking but could introduce the occlusion issue. To address this, Wang *et al.* [49] utilized vector fields to coordinate movements for each group and separated the animation paths mutually to reduce occlusion among them (Fig. 1(c)). However, this separation can cause much deviation from the input trajectories. In contrast, RouteFlow creates bundled animation paths for objects with similar trajectories and reduces occlusion by applying non-overlapping constraints on the object layout.

In addition to tracking objects, animated transitions are widely used to **identify the global trend** [40]. Empirical studies have discussed the potential of careful animation designs for trend identification [4, 9]. Recently, Zheng *et al.* [55] proposed the focus+context grouping method for animated transition to simultaneously track objects and identify the global trend. This method grouped objects with similar trends together and animated these groups sequentially (Fig. 1(b)). It simplifies complex animated transitions by dividing them into a sequence of simpler groups, facilitating easier tracking of objects while also revealing the global trend. However, this method groups transitions solely based on the start and end positions of the objects, ignoring their trajectories. As a result, it may fail to capture important patterns throughout trajectories, e.g., the local hotspots where objects converge or diverge. To overcome this limitation, RouteFlow considers the movement trajectories of objects, aiming to balance both the global trend and local hotspots.

3 Problem Formulation

In this section, we introduce the problem formulation, including the bus route analogy and two sub-problems derived from this analogy.

3.1 The Bus Route Analogy

We illustrate the objects' movements in animation using the real-world bus route analogy, where passengers travel along different bus routes to reach their destinations. As shown in Fig. 2(a), passengers gather at bus stops and board the same bus (A_1). They then travel together along shared routes (A_2). Eventually, they disembark at designated stops when approaching their destinations or transferring to other routes (A_3). As such, we apply this analogy to guide the design of our animation. As shown in Fig. 2(b), groups of objects with similar movement trajectories converge at local hotspots (B_1), analogous to bus stops, and then move together along the shared animation paths (B_2), much like passengers on the same bus. As the animation progresses, these objects may diverge to reach their destinations separately (B_3).

Based on this analogy, we design our animation, RouteFlow, to capture both the global trend and local hotspots. By grouping objects with similar trajectories and moving them along shared animation paths, we reveal the global trend, just as the bus routes that passengers travel along. Meanwhile, objects converge or diverge at specific local hotspots, similar to passengers boarding and disembarking at bus stops. This allows us to simplify complex and cluttered trajectories in animation while ensuring that critical convergence and divergence points are preserved.

Our animation leverages the Gestalt principles of *Common Fate* and *Proximity* [45, 47] to shape the perception of grouping. The *Common Fate* principle states that visual elements moving together are perceived as a group [9]. Accordingly, objects moving together along the same animation path are interpreted as a cohesive group. The *Proximity* principle states that visual elements close to one another are perceived as part of the same group [47]. In this case, we position objects with similar trajectories in close proximity, simulating passengers on the same bus.

To create the animation, we should generate the animation paths in a way that is similar to planning bus routes. Furthermore, since multiple objects often move simultaneously along the same animation path, we should minimize occlusion in animation, ensuring that each object has its own position, like passengers having individual seats on a bus. In this process, the seat allocation depends on the bus routes, as the bus routes determine which passengers are on the bus and where they board and disembark. This dependency naturally lends itself to sequential optimization [2]. In sequential optimization, the overall problem is decomposed into smaller, manageable sub-problems that are solved in sequence. The solution to each sub-problem then informs and serves as the input for the subsequent one, ensuring a cohesive and efficient resolution of the entire problem. Accordingly, we decompose the problem into two sub-problems: trajectory-driven path generation (bus routing) and object layout generation (seat allocation). Next, we detail these two sub-problems and their respective optimization goals.

3.2 Trajectory-Driven Path Generation

In the context of the bus routing problem, there are two main optimization goals: efficiency and effectiveness [31]. Efficiency involves minimizing operational costs, such as reducing the total length

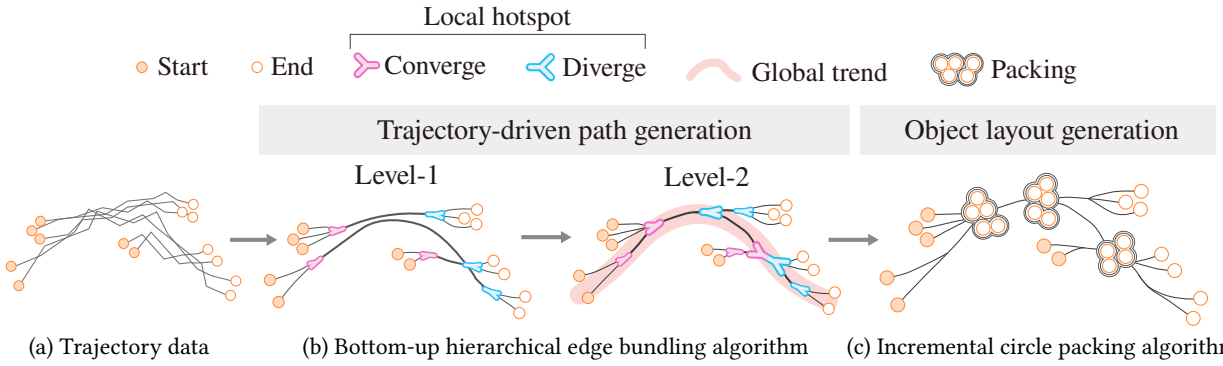


Fig. 3. The pipeline of our method.

of the bus routes. One of the most effective strategies is route aggregation. This strategy encourages passengers to share the same route as much as possible during their journeys. The key is to identify groups of passengers with similar trajectories and then design routes that accommodate these shared trajectories. Likewise, in animation, we should group objects with similar trajectories and share their animation paths to reduce the total path length. This consolidates similar movements, allowing users to easily perceive the global trend in animation. However, overemphasizing efficiency can lead to excessive route aggregation, forcing some passengers to deviate far from their intended trajectories and causing significant detours. On the other hand, effectiveness focuses on meeting passengers' travel demands by ensuring they can successfully reach their destination without excessive detours. This requires minimizing the deviation between the aggregated route and each passenger's intended trajectory. One solution is to set up proper bus stops in high-demand locations to satisfy more passengers' demands. In animation, the resulting animation paths should align closely with the input trajectories of the objects and thus better reveal critical local hotspots where objects converge or diverge.

As such, we derive two primary optimization goals for trajectory-driven path generation:

- **Minimize the total animation path length** by aggregating animation paths for groups of objects with similar trajectories.
- **Minimize the deviation from the input trajectories** by constraining the distance between the input trajectories to their aggregated paths.

3.3 Object Layout Generation

There are two main optimization goals when allocating seats: maximize capacity and avoid overcrowding. The first optimization goal is to maximize capacity. Similar to buses efficiently filling seats, the object layout should be designed to reduce empty space to enhance compactness. To achieve this, we strive to position similar objects close together to reduce gaps between them. This not only optimizes space utilization but also fosters a sense of group cohesion among closely placed objects, aligning with the *Proximity* principle. The second optimization goal is to avoid overcrowding, which can be addressed through three strategies. First, when passengers are on the same bus, each should have their own seat to avoid interfering with others. Second, co-travelers who board or disembark together should sit close to maintain group cohesion and avoid mixing with the crowd. Third, passengers who disembark first should sit closest to the exit (the principle of "first out, closest to the exit"), facilitating a smoother queueing process and mitigating potential overcrowding during disembarkation. Correspondingly, in our animation: 1) objects moving along the same path should remain visible and not overlap; 2) objects that converge or diverge together should be grouped closely to avoid mixing with other groups; and 3) objects should be placed based on their disembarking order and positions. Based on the analysis above, this problem involves two primary optimization goals:

- **Maximizing compactness** by reducing empty space in the layout.
- **Minimizing occlusion** by 1) applying the non-overlapping constraint within a group of objects moving together, 2) keeping objects that converge or diverge together as a group, and 3) following the principle of "first out, closest to the exit."

4 Method

Fig. 3 shows the pipeline of our method. Given trajectory data as input, it consists of two modules: **trajectory-driven path generation** and **object layout generation**.

4.1 Trajectory-Driven Path Generation

An edge bundling algorithm can effectively minimize both the total path length and the deviation from the input trajectories in generating the animation paths. However, aggregating all the trajectories simultaneously presents two issues. First, it fails to identify local hotspots at multiple levels of granularity, which are pervasive in real-world applications [35]. Second, the real-time animated transitions require high scalability of the algorithm. To address these issues, we develop a bottom-up hierarchical edge bundling algorithm that progressively bundles similar trajectories, level by level. As shown in Fig 3(b), it captures local hotspots across multiple levels of granularity while revealing the global trend. At each level, we adopt a force-directed strategy [27, 42] to bundle the edges. The core of our algorithm lies in the design of the forces that drive the bundling process, along with a bottom-up bundling that progressively bundles trajectories.

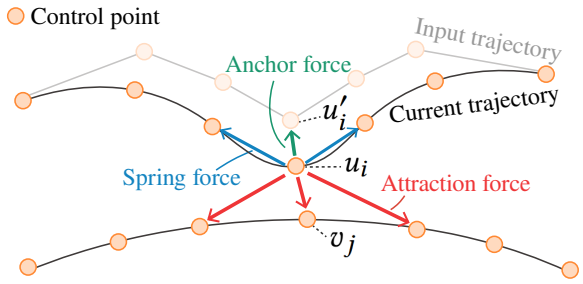


Fig. 4. Illustration of three types of forces in our algorithm.

4.1.1 Force Design. Existing force-directed edge bundling algorithms model trajectories as a series of control points and apply forces to adjust their positions [27, 42]. They typically adopt two types of forces: attraction force and spring force. However, they often fail to preserve local hotspots because these forces ignore the original positions of these input trajectories. To address this issue, we introduce a new force, the anchor force, to reduce deviation from the input trajectories. Fig. 4 illustrates how our algorithm incorporates these three types of forces. Given the trajectory set S and a pair of trajectories u and v, the three types of forces are defined as follows:

- **Attraction force** (F_{att}) is applied between control points on different trajectories to draw them closer together. This force bundles similar trajectories. According to Selassie *et al.* [42], F_{att} is defined as:

$$F_{att}(u_i, v_j) = \frac{\eta(v_j - u_i)}{C_v(\eta^2 + ||u_i - v_j||^2)^2}, \tag{1}$$

  where u_i and v_j represent the i-th and j-th control points on these trajectories, and $||u_i - v_j||$ denotes the Euclidean distance between them. The weighting parameter η controls the rate at which the force diminishes with increasing distance. A larger η causes F_{att} to decrease slower, thereby extending its influence range. C_v denotes the number of control points on trajectory v.
- **Spring force** (F_{spr}) is applied between adjacent control points on the same trajectory. This force promotes uniform distribution of control points along the trajectory and avoids highly curved trajectories. According to Holten *et al.* [27], F_{spr} is defined as:

$$F_{spr}(u_i) = C_u(u_{i+1} + u_{i-1} - 2u_i), \tag{2}$$

  where C_u is the number of control points on trajectory u.
- **Anchor force** (F_{anc}) is applied to each control point, pulling it back toward its position in the input trajectories. This force prevents the current trajectories from deviating too far from the input trajectories. F_{anc} is defined as:

$$F_{anc}(u_i) = ||u'_i - u_i||^2 \cdot \frac{u'_i - u_i}{||u'_i - u_i||}, \tag{3}$$

  where u'_i denotes the original position of u_i, and $\frac{u'_i - u_i}{||u'_i - u_i||}$ is a unit vector indicating the direction of the force.

Based on the above force analysis, the resultant force on the i-th control point of trajectory u is calculated as:

$$F(u_i) = (\sum_{v \in \Gamma_u} \sum_{j=1}^{C_v} F_{att}(u_i, v_j)) + \alpha F_{spr}(u_i) + \beta F_{anc}(u_i). \tag{4}$$

Here, Γ_u denotes the set of top-k similar trajectories of u. The parameters α and β balance the three types of forces. In our implementation, they are determined as 5 and 1 through a grid search.

4.1.2 Bottom-Up Hierarchical Edge Bundling. Progressively bundling similar trajectories at each level of granularity involves two key aspects. The first is how to select the most similar trajectories to consider when applying forces at each level. Existing edge bundling algorithms assess edge similarities through compatibility metrics, which consider factors such as topology [42] and importance [39] but often fail to capture trajectory similarities. To better capture trajectory similarities, we design our compatibility metric based on dynamic time warping (DTW) [36], a widely accepted metric for assessing trajectory similarity [54]. DTW calculates the distance between two trajectories by finding the optimal alignment between points on them, thereby capturing the overall similarity between the entire trajectories [36]. Given two trajectories (u, v) and their DTW distance (DTW(u, v)), the compatibility between u and v is defined as:

$$\text{compatibility}(u, v) = 1 - \text{norm}(\text{DTW}(u, v)), \tag{5}$$

where norm($\cdot$) denotes the min-max normalization to scale the distance into the range [0,1]. To reduce computational complexity, we only consider the top-k similar trajectories according to the

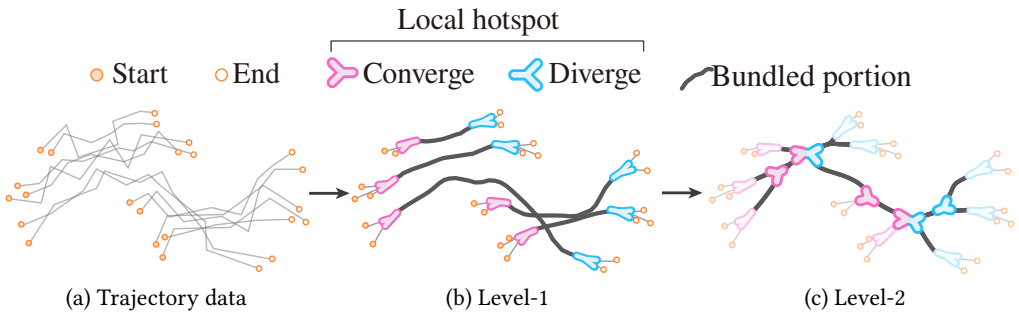


Fig. 5. Illustration of our bottom-up hierarchical edge bundling algorithm.

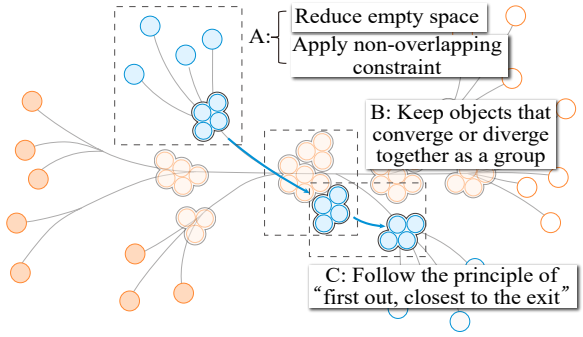


Fig. 6. Illustration of the goals in our object layout generation.

compatibility metrics. In our implementation, k is a user-specified parameter that is set as five by default.

The second is how to hierarchically bundle these trajectories. At each level, similar trajectories are bundled, revealing local hotspots where they converge and diverge. To identify local hotspots across multiple levels of granularity, it is crucial to preserve the convergence and divergence identified at lower levels. Therefore, we determine the bundled portions of trajectories at each level by measuring the distances between control points, as shown in Fig. 5. Each bundled portion will be merged into a single trajectory, which serves as input for the next level, where they are further bundled. Throughout this process, the identified local hotspots remain unchanged, as they are excluded from the bundled portions and unaffected by further bundling. At each level, the three types of forces are applied. When moving to the next level, the attraction force is increased tenfold to adapt to the sparser distribution of trajectories.

To evaluate the effectiveness of our algorithm, we compare it with two representative edge bundling algorithms, divided edge bundling (DEB) [42] and multilevel agglomerative edge bundling (MAEB) [22]. We assess these methods based on their efficiency in reducing total path length and deviation from the original paths. The results show that our algorithm achieves the lowest deviation and performs comparably with the baseline algorithms in terms of ink ratio. Details can be found in Appendix A.

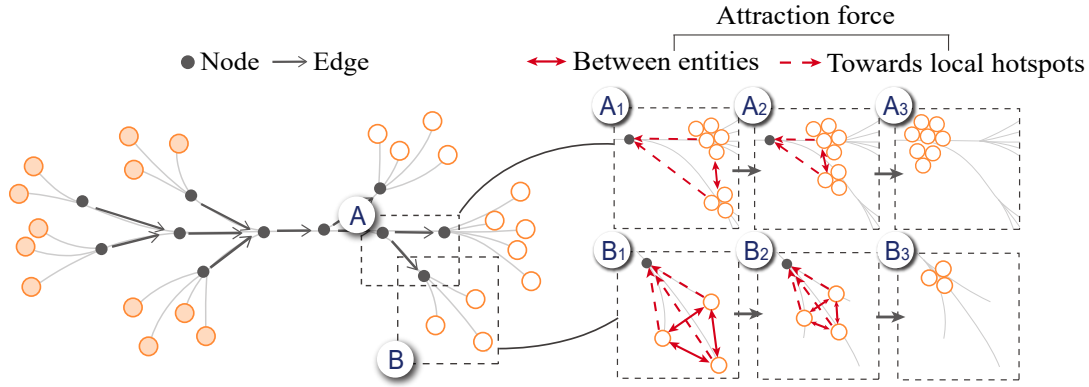
4.2 Object Layout Generation

The optimization goals described in Sec. 3.3 are achieved in three ways. First, to reduce the empty space and satisfy the non-overlapping constraint for a group of objects (Fig. 6A), we use a circle packing algorithm [51] to generate the object layout. Second, to keep objects that converge or diverge together as a group (Fig. 6B), an incremental circle packing algorithm is developed. Third, to follow the principle of "first out, closest to the exit"(Fig. 6C), we place objects based on their disembarking order and positions.

Just as passengers only adjust their seats when boarding or disembarking along the bus route, we update the layout incrementally only at the local hotspots. To achieve this, we first determine the order of local hotspots for layout generation by constructing a directed acyclic graph (DAG, Fig. 7(a)) and then incrementally generate the layout at each local hotspot (Fig. 7(b)). In the DAG, nodes represent local hotspots, and directed edges indicate the movements of objects between these local hotspots. We perform a reverse topological sort on the graph to generate the order of local hotspots. Then, we incrementally generate the layout at local hotspots according to their order. The basic idea of generating the layout at each local hotspot is to generate a layout for each newly arriving group and then pack these new layouts with those of previous groups. As shown in Fig. 7, when packing the objects at a given local hotspot A, they are placed to preserve their relative positions. This prevents occlusion during disembarking. Inspired by Görtler *et al.* [23], we adopt a force-directed algorithm and apply two types of attraction forces (Fig. 7(b)). The first force moves all objects/groups toward the current local hotspot, while the second attracts neighboring objects/groups together. To avoid occlusion, we model objects/groups as rigid bodies and use the Box2D engine [7] for implementation.

4.3 Implementation

After generating the animation paths and object layout at all the local hotspots, we use an interpolation-based method to render smooth animations. This method synchronizes the movements based on the timing of the local hotspots and the objects' start and end points. To simplify, we refer to these collectively as "point." We ensure that 1) groups of objects that converge or diverge together



(a) Build a DAG to determine the order of local hotspots (b) Generate the layout at each local hotspot

Fig. 7. Illustration of our incremental circle packing algorithm.

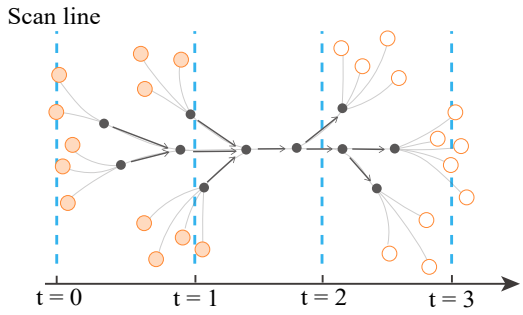


Fig. 8. Illustration of the scan line moving through all the local hotspots and the objects' start and end points.

arrive at or leave the local hotspots at the same time and 2) excessively fast speeds are avoided. As shown in Fig. 8, we use a scan line that moves through all points, assigning their timing as when they intersect with the scan line. The movement direction of the scan line is determined by the vector formed between the average start and end positions of all animation paths. However, paths that form a large angle with the scan line's movement direction can result in excessively high speeds of objects. To address this, we iteratively adjust the timing of points until the maximum speed is less than twice the minimum speed. If we detect excessively high speed between two connected points, we adjust the timing by either delaying the latter point or advancing the former at random. We interpolate between points to render the animation after completing the iterative adjustment. To further enhance smoothness, we apply the slow-in, slow-out technique [15].

5 Evaluation

To demonstrate the effectiveness of RouteFlow, we conducted a quantitative experiment and a user study.

5.1 Quantitative Evaluation

Datasets. The quantitative evaluation was conducted on seven datasets from real-world applications: Taxi [52, 53], BirdMap [29], Railway [25], MEIBook [8], OpenSkyAirline [38], US Migration [27], DanishAIS [3]. These datasets were collected from three common areas in trajectory data analysis, including transportation, sociology, and ecology. We preprocessed the raw data through several steps, including noise filtering, trajectory compression, and merging of redundant trajectories.

Baseline methods. We selected two state-of-the-art animated transition methods for comparison. The first method, the focus+context grouping method, simultaneously facilitates tracking objects' movements and identifying the global trend by breaking down transitions into groups [55]. We used the default parameters reported in the paper. The second method, the vector-field-based method, is the state-of-the-art method in terms of tracking objects' movements by utilizing vector fields to generate smooth, non-linear paths [49]. As the original paper did not provide specific parameter settings, we performed a grid search to find the optimal parameters. Moreover, since the vector-field-based method requires predefined groups, we used the grouping results from the focus+context grouping method for consistency.

Evaluation criteria. Previous studies classified the metrics into three types: occlusion, deformation, and dispersion [13, 15, 17, 49]. We adopted the metrics summarized by Wang *et al.* [49] as they are tailored for objects with groups.

We denote all the frames in the animation as T, a particular frame as t, and all the objects as P.

Occlusion measures the overlap between objects. This metric is useful for evaluating the capability of an animation in facilitating the tracking of objects' movements. High occlusion reduces the visibility and distinguishability of moving objects, making these objects harder to distinguish and track [13, 15].

Specifically, *overall occlusion* (occlusion_o) measures the overlap between all objects during the entire animation.

$$\text{occlusion}_o(T) = \frac{1}{|T|}\sum_{t\in T}\frac{\sum_{p,q\in P, p\neq q}\text{overlap}(p,q,t)}{|P|(|P|-1)},$$

where $\text{overlap}(p,q,t)$ is an indicator function with value 1 if objects p and q overlap at frame t, and 0 otherwise.

Within-group occlusion (occlusion_w) measures the overlap between objects in the same group.

$$\text{occlusion}_w(T) = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{|T_{G_i}|}\sum_{t\in T_{G_i}}\frac{\sum_{p,q\in G_i, p\neq q}\text{overlap}(p,q,t)}{|G_i|(|G_i|-1)},$$

where K is the number of groups, G_i is the set of objects in the i-th group, and T_{G_i} is the frames of the group G_i.

Table 1: Comparison between different methods, including the focus+context grouping method (F+C), the vector-field-based method (VF), and RouteFlow. For all four metrics, lower values are better.

Dataset	Overall occlusion			Within-group occlusion			Deformation			Dispersion		
	F+C	VF	RouteFlow	F+C	VF	RouteFlow	F+C	VF	RouteFlow	F+C	VF	RouteFlow
Taxi [52, 53]	0.00123	0.00180	**0.00048**	0.01963	0.00626	**0.00606**	0.00081	0.00107	**0.00062**	0.05948	0.06766	**0.02606**
BirdMap [29]	0.00277	0.00958	**0.00213**	0.02773	0.02607	**0.00904**	0.00152	0.00160	**0.00064**	0.13762	0.12494	**0.03258**
Railway [25]	0.00229	0.01351	**0.00142**	0.11960	0.10420	**0.02875**	0.00128	0.00140	**0.00063**	0.10629	0.10950	**0.03090**
MEIBook [8]	0.00132	0.00566	**0.00066**	0.01362	0.01196	**0.00675**	0.00079	0.00105	**0.00064**	0.06699	0.06817	**0.02430**
OpenSkyAirline [38]	0.00076	0.00394	**0.00055**	0.01934	0.01814	**0.00571**	0.00082	0.00117	**0.00061**	0.08047	0.08216	**0.03007**
US Migration [27]	0.00115	0.00408	**0.00072**	0.01107	0.01027	**0.00295**	0.00097	0.00147	**0.00066**	0.13015	0.13345	**0.03802**
DanishAIS [3]	0.00047	0.00120	**0.00012**	0.01435	0.00460	**0.00266**	0.00122	0.00089	**0.00087**	0.11688	0.12816	**0.08611**
Average	0.00148	0.00561	**0.00088**	0.03097	0.02469	**0.00842**	0.00122	0.00141	**0.00065**	0.10627	0.10858	**0.03727**

Deformation measures the changes in distance between objects within the same group across consecutive time frames. Lower deformation indicates that the relative object positions within the group remain stable, making it easier to track the objects of interest.

$$\text{deformation}(T) = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{|T_{G_i}|}\sum_{t\in T_{G_i},t>0}\frac{\sum_{p,q\in G,p\neq q}|\text{dist}(p,q,t)-\text{dist}(p,q,t-1)|}{|G_i|(|G_i|-1)}.$$

$dist(p, q, t)$ is the distance between objects p and q at frame t.

Dispersion measures how spread out objects in the same group are. Lower dispersion indicates that the members of a group are moving more closely together, enhancing the perception of the group as a whole. This facilitates more effective tracking of the group's collective movements, thereby enhancing the identification of the global trend.

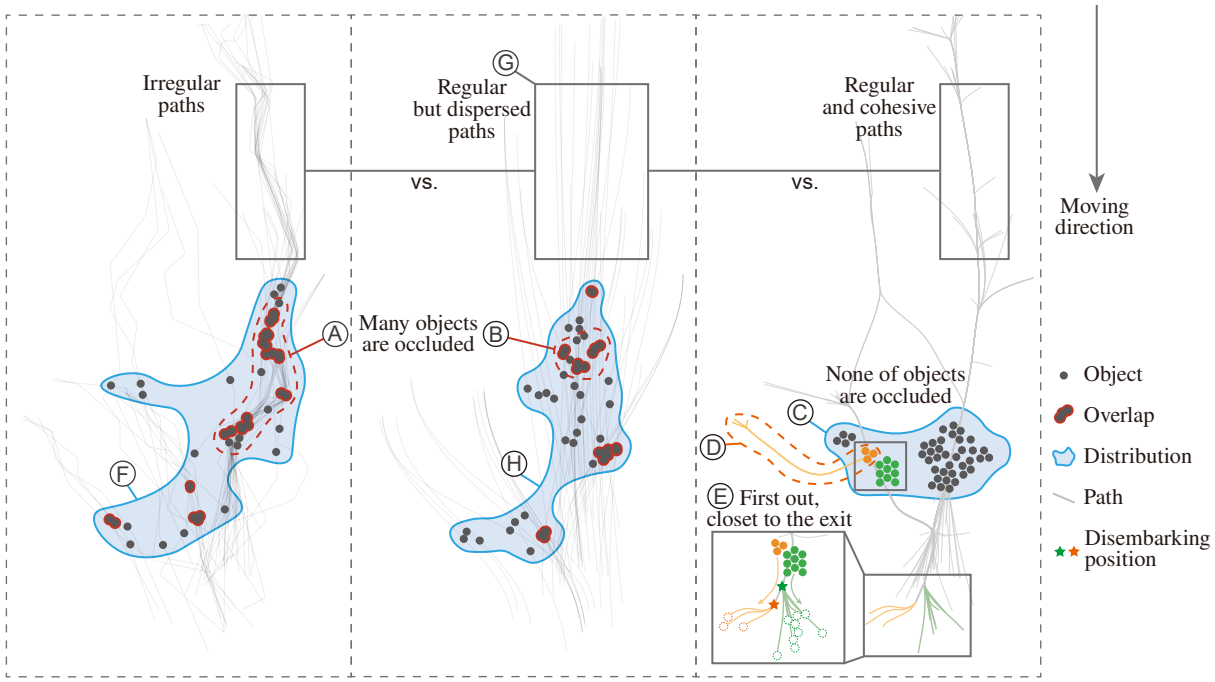
$$\text{dispersion}(T) = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{|T_{G_i}|}\sum_{t\in T_{G_i}}\frac{\sum_{p,q\in G_i,p\neq q}\text{dist}(p,q,t)}{|G_i|(|G_i|-1)}.$$

The aforementioned three metrics focus on tracking objects' movements and identifying the global trend. To the best of our knowledge, there is no existing metric that adequately measures the preservation of local hotspots in animation. Additionally, the employed real-world datasets lack ground truth for local hotspots. As a result, we supplement the evaluation of preserving local hotspots with a user study using several synthetic datasets, which is described in Sec. 5.2.

Results. Table 1 presents the comparison results between RouteFlow and the baseline methods. RouteFlow performs the best on all datasets and all metrics.

Occlusion. RouteFlow achieves lower overall occlusion and within-group occlusion scores compared to the two baseline methods. This improvement is mainly due to differences in object layout. The baseline methods do not explicitly optimize the overlaps within groups (Fig. 9A and Fig. 9B), and in particular, the vector-field-based method may even introduce overlaps between groups as it moves all objects simultaneously. In contrast, our incremental circle packing algorithm reduces overlaps by employing three strategies: 1) applying a non-overlap constraint to minimize occlusion between objects (Fig. 9C), 2) keeping objects that converge or diverge together as a group (Fig. 9D), and 3) following the principle of "first out, closest to the exit" (Fig. 9E).

Deformation. Compared to the two baseline methods, RouteFlow exhibits the least deformation. The focus+context grouping



(a) Focus+context grouping (b) Vector-field-based method (c) RouteFlow

Fig. 9. Object positions at a specific frame in the animations generated by three methods on the BirdMap dataset. Here, overlaps are highlighted as red strokes, and the distributions of objects are shown as blue contours. The metric values for these frames are displayed below each sub-figure. The detailed analysis of these values is provided in Appendix B.

Table 2: The result of running time on different modules.

Dataset	Attribute	Time cost (second)		
	Size	Trajectory-driven path generation	Object layout generation	Total
Taxi [52, 53]	73	0.06	0.03	0.09
BirdMap [29]	109	0.10	0.06	0.16
Railway [25]	129	0.11	0.06	0.17
MEIBook [8]	174	0.11	0.10	0.21
OpenSkyAirline [38]	187	0.14	0.11	0.25
US Migration [27]	258	0.23	0.22	0.45
DanishAIS [3]	316	0.46	0.37	0.83

method changes the layout of every frame as objects follow their input trajectories. Similarly, the vector-field-based method causes unsynchronized movements due to varying velocity vectors among objects at different positions, leading to layout changes in subsequent frames. Conversely, RouteFlow incrementally updates the layout only at local hotspots, with the aim of balancing readability and stability in these refinements. This balance greatly reduces deformation during animated transitions.

Dispersion. RouteFlow achieves lower dispersion compared to the other methods. The focus+context grouping method, which groups objects based solely on their start and end positions, records the highest dispersion. This method might group objects with different trajectories together due to their similar start and end positions, increasing dispersion (Fig. 9F). The vector-field-based method, which generates the animation paths of objects using a vector field, tends to disperse the input trajectories (Fig. 9G), leading to a less compact layout (Fig. 9H) and high dispersion. In contrast, our incremental circle packing algorithm keeps the objects compact (Fig. 9C), resulting in lower dispersion.

Running Time. Table 2 shows the average running times for each module of our animation method on real-world datasets, where object sizes vary from 73 to 316. The performance tests were conducted on a Windows PC with an Intel i9-13900K CPU. We averaged results over five trials to minimize randomness. The average running time per dataset is within 1 second, which is fast enough for designing animated transitions. The object layout generation module is the most time-consuming because it requires incremental generation for the layout of each local hotspot. In contrast, the trajectory-driven path generation module is less demanding and achieves stable results in 300 iterations.

5.2 User Study

We conducted a user study to evaluate how effectively people use RouteFlow to track objects' movements and identify the global trend and local hotspots. We formulated three hypotheses: participants perform more accurately with RouteFlow in tracking objects' movements (**H1**), identifying the global trend (**H2**), and locating local hotspots (**H3**) compared to two baseline methods, the focus+context grouping method and the vector-field-based method.

5.2.1 *Study Setup.*

Participants. We recruited 15 participants (12 males and 3 females, denoted as P1-P15) from local universities. They were graduate students majoring in computer science (12) and information design (3), aged from 22 to 32 years ($mean$ = 24.47, SD = 2.53). All of them reported to have normal vision and no color deficiencies. Upon completion, each participant received a $30 compensation, independent of their performance.

Apparatus. The user study was conducted on a personal computer equipped with a 27-inch display with a resolution of 3840 × 2160 pixels and a 60 Hz refresh rate. Objects were presented as circles with a radius of 9 pixels (0.20 cm), filled in black color, following the previous practice [17, 49]. The animation window measured 1250 × 1250 pixels (27.0 × 27.0 cm) with a white background. Participants were seated at a distance of 40 cm from the display.

Datasets. We used synthetic data for a controlled study setting instead of real-world data, which may lack ground truth for the global trend and local hotspots. This follows the common practice [17, 49]. Fig. 10 shows our dataset generation process, involving three steps. First, we generated a smooth global trend trajectory using B-splines. Next, we determined local hotspots by sampling points along the

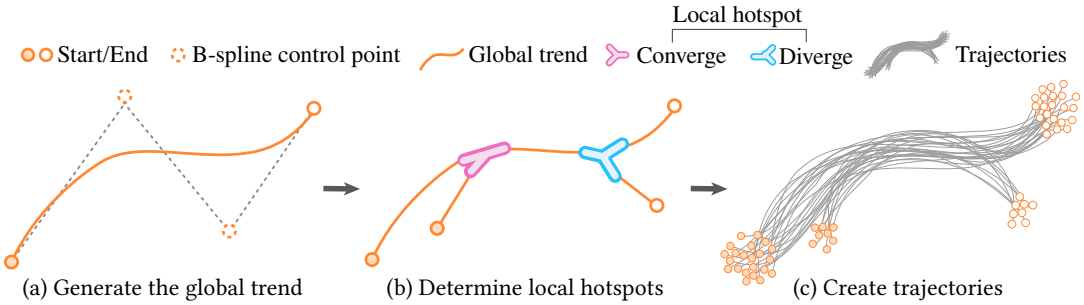


Fig. 10. The data generation pipeline.

global trajectory and randomly classifying them as converging or diverging points. Finally, we created trajectories for objects by adding random perturbations to the global trend, avoiding overlap between start and end positions. To achieve better diversity and a certain level of complexity, we finalized our design through several iterations. In the final iteration, there were two types of global trend (one or two bends in the B-spline), and three types of local hotspot assignment (1 convergence + 1 divergence, 2 convergences + 1 divergence, and 1 convergence + 2 divergences). Each dataset included 30 trajectories. To control the experiment duration and keep participants focused, we generated datasets for each combination of trend type and local hotspot type and conducted two repetitions per combination, resulting in 12 datasets (2 types of the global trend $\times$ 3 types of local hotspot assignment $\times$ 2 repetitions).

Task design. Our study consisted of three tasks, each iterated and refined through small-scale pilot studies. For each task, participants were asked questions with four options (one correct, three incorrect) along with an additional option for "I am not sure."

T1—Tracking objects' movements: Participants were required to track the movement of target objects to identify their end positions, and then select one answer from five options. This task design referred to the previous practices [17, 49]. We set the number of target objects to three, all from the same group, to simplify the task. We generated the incorrect options by randomly replacing the correct targets with their nearest neighbors based on their end positions.

T2—Identifying the global trend: Participants were asked to observe the overall movement of all objects to identify the global trend, and then select one answer from five options. Initially, we set the background to be fully white, whereas feedback from the pilot study indicated difficulty in observing and locating the movements. To alleviate this issue, the background canvas was divided into 8 $\times$ 8 grid, colored alternatively in white and grey. The three incorrect options were generated by adding random perturbations to the correct trend, ensuring that they passed through different grids to be distinguishable from the correct option.

T3—Locating local hotspots: Participants were asked to identify the grids containing local hotspots. Similar to *T2*, to facilitate locating local hotspots, we employed a white and grey background canvas. Each answer option included two marked grids: one for convergence and one for divergence. Incorrect options were generated from the correct option by randomly replacing one correct grid with a neighboring grid. To simplify the task, participants were allowed to click and mark grids that might assist them while viewing the animation and refer to these marks when answering.

Study protocol. Participants started by signing consent and watching a tutorial video about the study procedure and tasks. We then provided three brief videos, each explaining a different animation method. The study adopted a within-subjects design, requiring each participant to complete all three tasks using three different methods.

For each task, we designed a practice session and a test session. The practice session familiarized participants with the interface and tasks, through six trials, two for each method. In each trial, participants initially saw all objects in grey points. Particularly, we highlighted the target objects in red for the tracking task. They then clicked to start the animation. All objects in the tracking task transitioned to grey and then to black within the first 0.5 seconds. In the other two tasks, the objects turned black directly. This allowed participants to recognize the target objects and prepare to follow their movements. After the animation, participants clicked to start the question and could not review the animation anymore. In the practice session, we provided correct answers to help participants check their understanding and encouraged them to ask questions.

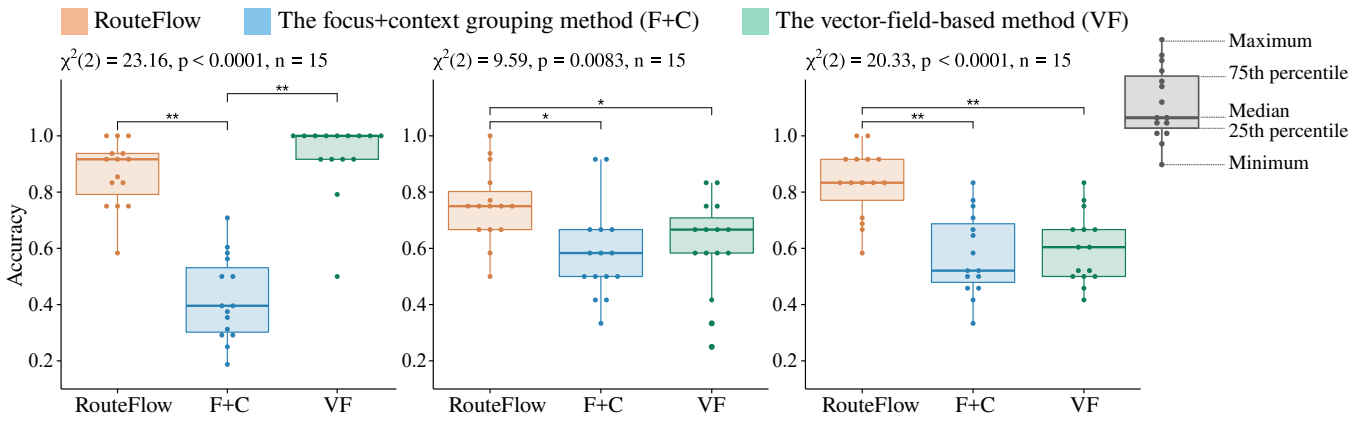
After completing the practice session and confirming that they fully understood the tasks and methods, participants advanced to the test session, which consisted of 36 trials (12 datasets $\times$ 3 methods). Unlike the practice session, correct answers were no longer provided during the test session. To counterbalance the order of methods, we divided 15 participants into five groups, three for each group. Within each group, we used an expanded Latin square and applied a cyclic shift to the method order for each participant. Additionally, to alleviate the learning effect, we randomly mirrored and rotated the datasets. Participants were allowed to take short breaks after each task or whenever they requested one.

In total, each participant finished 108 trials (3 tasks $\times$ 3 methods $\times$ 12 datasets), leading to 1,620 total trials (15 participants $\times$ 108 trials). After finishing each task, we assessed participants' workload and fatigue levels using NASA's Task Load Index [41] and asked about their preferred methods and the reasons for their preferences. For all trials, we recorded participants' answers and completion times. The entire study lasted 75-90 minutes. Additional study details are provided in Appendix C, and results are provided in the supplemental material.

5.2.2 Result Analysis. We analyzed three types of data: the accuracy of multi-choice questions, participants' subjective ratings for workload, and their stated preferences.

Accuracy. For each task, we computed participants' average accuracy across different methods. As data is not normally distributed, we conducted Friedman tests for each task, followed by Wilcoxon signed-rank tests with Bonferroni correction for pairwise comparisons of different methods. We report the statistical test results and the box plots in Fig. 11. The Friedman test results indicate significant differences among the methods in all three tasks: tracking objects' movements ($\chi^2(2) = 23.16, p < 0.0001$), identifying the global trend ($\chi^2(2) = 9.59, p = 0.00083$), and locating local hotspots ($\chi^2(2) = 20.33, p < 0.0001$). In the subsequent analysis, we focus on the pairwise comparisons.

T1—Tracking objects' movements: In pairwise comparisons, our RouteFlow method significantly outperforms the focus+context grouping method (average: 0.865 vs. 0.421, $p = 0.001$). However, RouteFlow only shows a small difference compared to the vector-field-based method in average accuracy (0.865 vs. 0.930), and the paired test indicates no significant difference ($p = 0.275$). Therefore, the results can **partially support H1**. Participants using the focus+context grouping method show a significantly lowest accuracy. This is because both our method and the vector-field-based method optimize the animation paths to reduce complexity, while the focus+context grouping method directly uses the input trajectories as its animation paths. Correspondingly, the majority of participants (10 out of 15) reported that the animation paths in the focus+context grouping method were more complex and had greater occlusion compared to the other two methods. This feedback aligns with our comparison results in Sec. 5.1, where the focus+context grouping method achieves the highest within-group occlusion. In addition,



(a) T1 — tracking objects' movements (b) T2 — identifying the global trend (c) T3 — locating local hotspots

Fig. 11. User study results on three tasks. Here, * indicating $p < 0.05$, ** indicating $p < 0.01$.

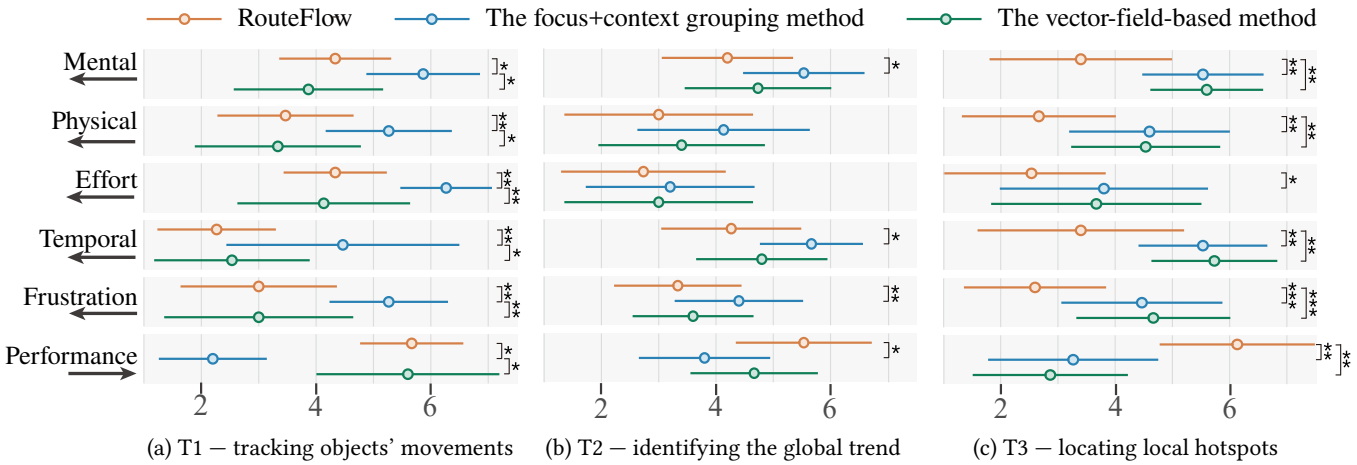
we asked participants about their different performances using RouteFlow and the vector-field-based method. P12 appreciated the grouping in RouteFlow for tracking, saying *"The relative positions of objects are stable within the group. I can track one or two targets inside the group and use them as a reference to locate other targets."* However, we also received feedback expressing different opinions. For instance, P2 commented, *"The objects in RouteFlow are grouped closely. I sometimes confuse the targets with other objects."* P2 preferred the vector-field-based method, noting *"objects are relatively spread out. Even when occlusion occurs, it is easy to distinguish the targets by following their movements."*

T2—Identifying the global trend: RouteFlow significantly outperforms the focus+context grouping method ($p = 0.021$) and the vector-field-based method ($p = 0.028$), thereby **supporting H2**. Participants described RouteFlow as having a sense of *"unity"* (P1, P2, P4, P10, and P11) and showing *"a clear trend"* (P1, P4, P5, P6, and P11). Besides, they explained their performances using the two baseline methods related to the dispersion. P2 complemented that *"Although [the focus+context grouping method] depicts a clear trend of each group, it requires memorizing and comparing the movements of several groups to get the global trend, which brings extra burden."* P4 mentioned the vector-field-based method, saying *"It is distracting to follow the objects that are dispersed in different locations but move synchronously."* This feedback aligns with the quantitative result in Sec. 5.1, where these baseline methods showed higher dispersion compared to RouteFlow. Additionally, P1 commented that the animation design for groups of objects in RouteFlow can be a double-edged sword for capturing the global trend: *"[It allows me] to infer the global trend based on groups' movements. However, it may be disrupted due to the convergence and divergence [of the groups]."* This further highlighted the importance of balancing the global trend and local hotspots.

T3—Locating local hotspots: RouteFlow achieves an average accuracy of 0.832, which is significantly better than the focus+context grouping method ($p = 0.002$) and the vector-field-based method ($p = 0.002$). Therefore, the results can **support H3**. Participants described the convergence and divergence of objects in RouteFlow as *"clearly noticeable"* (P1, P2, P12, P13, and P15) and *"apparent"* (P6, P8, and P10). In contrast, the other two methods required extra cognitive effort. Five participants (P2, P6, P11, P12, and P15) complained that animating different groups separately in the focus+context grouping method made it difficult to distinguish the locations where objects converge or diverge. Although the vector-field-based method animated all objects together, users still found it difficult to identify local hotspots. P11 commented from the spatial perspective, *"Objects are dispersed. I cannot confidently tell the locations [of the local hotspots]."* P2 noticed *"Objects do not pass through their shared positions simultaneously."* This issue arises because, while the vector-field-based method animates objects concurrently, the local hotspot locations along their individual trajectories do not always align. Thus, participants noted temporal discrepancies in the animation as objects passed through these local hotspots.

Workload. Fig. 12 summarizes participants' workload and fatigue levels using NASA's Task Load Index [41], including mental demand, physical demand, temporal demand, effort, frustration, and performance. We conducted Friedman tests for each dimension in each task, followed by Wilcoxon signed-rank tests with Bonferroni correction for pairwise comparison among different methods. The Friedman test results show significant differences among the methods across all six dimensions for each task, except the effort dimension for *T2—Identifying the global trend*. Next, we focus on analyzing the pairwise comparison results.

T1—Tracking objects' movements: The Wilcoxon signed-rank tests indicate that both RouteFlow and the vector-field-based method perform better than the focus+context grouping method on all six dimensions, whereas there are no significant differences between RouteFlow and the vector-field-based method. Further analysis of mean values and confidence intervals reveals a subtle difference: the vector-field-based method shows slightly lower mental and physical demands and effort than RouteFlow. This could be due to the relatively simpler animation paths for individual objects in this



(a) T1 — tracking objects' movements (b) T2 — identifying the global trend (c) T3 — locating local hotspots

Fig. 12. Participants' workload and fatigue levels according to NASA's Task Load Index. Here, error bars show the 95% confidence intervals. ← indicates that a smaller score is better, and vice versa. The results of pairwise Wilcoxon signed-rank tests are also shown, with * indicating $p < 0.05$, ** indicating $p < 0.01$, and * indicating $p < 0.001$.**

method [49], which might require less cognitive load for tracking compared to the bundled paths in RouteFlow.

T2—Identifying the global trend: The Wilcoxon signed-rank tests indicate that RouteFlow outperforms the focus+context grouping method in terms of mental demand, temporal demand, frustration, and performance dimensions. This is because the focus+context grouping method demands extra memory and analytical burden to synthesize the movements of multiple groups into the global trend, which often leads to frustration.

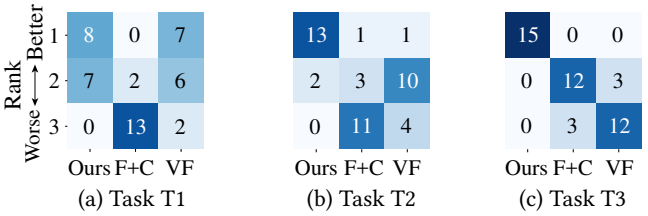
T3—Locating local hotspots: The Wilcoxon signed-rank tests indicate that RouteFlow outperforms the other two methods on the five dimensions, except for effort, where RouteFlow only outperforms the focus+context grouping method. These advantages align with participants' accuracy in completing the task. Regarding the effort dimension, participants found the background grids during the animation helpful, as they reduced the effort needed to observe and locate movement.

Preference. We also collected participants' preferences for the three methods in three tasks, as summarized in Fig. 13.

T1—Tracking objects' movements: Participants' opinions varied. Eight participants preferred RouteFlow, and seven preferred the vector-field-based method. This was slightly different from the overall subjective workload ratings, where participants thought RouteFlow brought a bit more mental and physical burden and effort. We noticed that P10 and P15 performed equally accurately with RouteFlow and the vector-field-based method but reported higher subjective performance and preference for RouteFlow. They commented that objects in RouteFlow were *"less occluded"*, making them *"more convinced."*

T2—Identifying the global trend: 13 participants preferred RouteFlow. Specifically, P1 appreciated that RouteFlow gradually revealed the global trend along with the group movement, just like *"painting with a brush."* In contrast, P6 preferred the focus+context grouping method, explaining *"It is easier to connect the movements of different groups into a global trend."* P10 liked the vector-field-based method and noted that *"I need to memorize the positions objects pass through to identify the global trend. [The vector-field-based method] requires less memory burden, as I can observe objects moving simultaneously in the global trend."*

T3—Locating local hotspots: All participants ranked RouteFlow at the top, indicating a strong advantage of our method. Their reasons mainly focused on the clear and noticeable animated transitions, showing groups that were converging and diverging.



(a) Task T1 (b) Task T2 (c) Task T3

Fig. 13. Participants' preference ranks of different methods on three tasks. Smaller ranks indicate stronger preferences.

6 Discussion and Future Work

As demonstrated by the results from both the quantitative evaluation and user study, the primary benefit of RouteFlow lies in its capacity to effectively reveal the global trend and identify local hotspots. It also maintains comparable performance in tracking objects. The participants generally gave positive feedback on its usability. Nonetheless, they also pointed out several limitations that deserve further investigation.

Introducing additional visual channels. RouteFlow delivers an elaborate animation by planning animation paths and generating a compact and non-occluded object layout. In the user study, P10, who majored in information design, suggested including more visual channels to enhance the expressiveness of animations. For instance,

using different colors and brightness can help users track the targeted objects [28]. Moreover, objects can be encoded in varying sizes based on their importance, making them more easily distinguishable. A recent study has also validated that the variations in object brightness and size have a positive impact on perceiving groups [9]. In the future, we can apply a dynamic color palette to our generated object layout to improve the animation [12].

Interactive animation adjustment. Our method automatically generates animations based on the input trajectories, easing the effort required for manual design. On top of the automated method, RouteFlow can be enhanced by supporting users to interactively design and refine animations, addressing their unique needs [49]. For example, users can directly drag to refine the animation paths or modify the corresponding object groups and layout, and our method can propagate the changes to the whole animation. Therefore, users do not need to manipulate each object in detail. Besides, our method can be integrated with dynamic parameter adjustment to balance different types of forces in our hierarchical edge bundling for animation path generation. As such, we can analyze data features such as trajectory density, local hotspot distribution, or occlusion levels and adjust parameters on targeted datasets or areas to identify different local hotspots. Users can also interactively specify parameters on demands.

Supporting more trajectory patterns. While our current method effectively highlights the global trend and local hotspots, trajectory data often contains other patterns that can provide valuable insights [54]. For instance, periodic patterns represent movements that repeat at regular intervals, such as daily commutes or seasonal migrations [5]. Anomalies refer to movements that deviate significantly from the norm, representing unusual or rare behaviors, such as a migratory bird taking an abnormal path due to environmental disruptions [33, 34]. A practical solution is integrating existing pattern detection techniques [10] and designing animations to communicate these patterns effectively to users. For example, trajectories that exhibit periodic patterns could be animated with pulsating effects or cyclic color changes [1], clearly highlighting their repetitive nature over time.

Scalability. RouteFlow faces scalability issues due to both algorithmic capabilities and visual constraints. From the algorithmic perspective, RouteFlow can process hundreds of moving objects in real time. When scaled to one thousand objects, RouteFlow completes processing in around 10 seconds, making real-time animation impractical at this scale. From the visual perspective, the number of displayed objects is constrained by both screen space and human perception. Especially, previous studies have shown that people can only track a limited subset of moving objects [19, 20], making human perception a more limiting factor. A potential solution to address these limitations is to combine a sampling method with our animation method. The key challenge is to identify a set of representative samples that effectively capture both the global trend and the local hotspots.

7 Conclusion

In this paper, we present RouteFlow, a trajectory-aware animated transition method to enhance the analysis of the movement trend and object tracking in the animation process. By analogizing animation paths to bus routes and the object layout to the seat allocation, RouteFlow offers clear and smooth animations of moving objects along their trajectories. The key feature of RouteFlow lies in its balanced depiction of the global trend and local hotspots, coupled with its ability to minimize occlusion. This balance improves users' capability to track movements and understand complex interactions within the objects. The quantitative evaluation and user study further validate that RouteFlow performs better than existing methods in identifying the global trend and locating local hotspots.

Acknowledgments

Duan Li, Xinyuan Guo, and Shixia Liu are supported by the National Natural Science Foundation of China under grants U21A20469, 61936002 and in part by Tsinghua University-China Telecom Wanwei Joint Research Center. Lingyun Yu is supported by the National Natural Science Foundation of China under grant 62272396. The authors would like to thank Jiangning Zhu, Zhen Li, Jiashu Chen, Yukai Guo, and Prof. Weikai Yang for their valuable contributions to the discussions and comments.

References

[1] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. 2011. *Visualization of Time-Oriented Data.* Vol. 4. Springer.

[2] Augusto Aubry, Antonio De Maio, Alessio Zappone, Meisam Razaviyayn, and Zhi-Quan Luo. 2018. A New Sequential Optimization Procedure and Its Applications to Resource Allocation for Wireless Systems. *IEEE Transactions on Signal Processing* 66, 24 (2018), 6518–6533.

[3] Danish Maritime Authority. 2020. Data from the Danish AIS system. Retrieved July 4, 2024 from https://www.dma.dk/safety-at-sea/navigational-information/download-data

[4] Matthew Brehmer, Bongshin Lee, Petra Isenberg, and Eun Kyoung Choe. 2020. A Comparative Evaluation of Animation and Small Multiples for Trend Visualization on Mobile Phones. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 364–374.

[5] Huiping Cao, Nikos Mamoulis, and David W. Cheung. 2007. Discovery of Periodic Patterns in Spatiotemporal Sequences. *IEEE Transactions on Knowledge and Data Engineering* 19, 4 (2007), 453–467.

[6] Nan Cao, David Gotz, Jimeng Sun, and Huamin Qu. 2011. DICON: Interactive Visual Analysis of Multidimensional Clusters. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2581–2590.

[7] Erin Catto. 2010. Box2D. https://box2d.org/

[8] C. CERL. 2015. MEI - Material Evidence in Incunabula. Retrieved July 4, 2024 from https://data.cerl.org/mei/_search

[9] Amira Chalbi, Jacob Ritchie, Deokgun Park, Jungu Choi, Nicolas Roussel, Niklas Elmqvist, and Fanny Chevalier. 2020. Common Fate for Animated Transitions in Visualization. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 386–396.

[10] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A survey. *ACM Computing Surveys* 41, 3 (2009), 1–58.

[11] Bay-Wei Chang and David Ungar. 1993. Animation: From Cartoons to the User Interface. In *Proceedings of ACM Symposium on User Interface Software and Technology*. 45–55.

[12] Jiashu Chen, Weikai Yang, Zelin Jia, Lanxi Xiao, and Shixia Liu. 2025. Dynamic Color Assignment for Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics* 31, 1 (2025), 338–348.

[13] Fanny Chevalier, Pierre Dragicevic, and Steven Franconeri. 2014. The Not-so-Staggering Effect of Staggered Animated Transitions on Visual Tracking. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2241–2250.

[14] Weiwei Cui, Shixia Liu, Li Tan, Conglei Shi, Yangqiu Song, Zekai Gao, Huamin Qu, and Xin Tong. 2011. Textflow: Towards Better Understanding of Evolving Topics in Text. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2412–2421.

[15] Pierre Dragicevic, Anastasia Bezerianos, Waqas Javed, Niklas Elmqvist, and Jean-Daniel Fekete. 2011. Temporal Distortion for Animated Transitions. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 2009–2018.

[16] Pierre Dragicevic, Stéphane Huot, and Fanny Chevalier. 2011. Gliimpse: Animating from Markup Code to Rendered Documents and Vice Versa. In *Proceedings of ACM Symposium on User Interface Software and Technology*. 257–262.

[17] Fan Du, Nan Cao, Jian Zhao, and Yu-Ru Lin. 2015. Trajectory Bundling for Animated Transitions. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 289–298.

[18] David E. Fencsik, Sarah B. Klieger, and Todd S. Horowitz. 2007. The Role of Location and Motion Information in the Tracking and Recovery of Moving Objects. *Perception & Psychophysics* 69 (2007), 567–577.

[19] Cary S. Feria. 2012. The effects of distractors in multiple object tracking are modulated by the similarity of distractor and target features. *Perception* 41, 3 (2012), 287–304.

[20] S. L. Franconeri, S. V. Jonathan, and J. M. Scimeca. 2010. Tracking multiple objects is limited only by object spacing, not by speed, time, or capacity. *Psychological Science* 21, 7 (2010), 920–925.

[21] Petitjean François, Ketterlin Alain, and Gançarski Pierre. 2011. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition* 44, 3 (2011), 678–693.

[22] Emden R. Gansner, Yifan Hu, Stephen North, and Carlos Scheidegger. 2011. Multilevel Agglomerative Edge Bundling for Visualizing Large Graphs. In *Proceedings of IEEE Pacific Visualization Symposium*. 187–194.

[23] Jochen Görtler, Christoph Schulz, Daniel Weiskopf, and Oliver Deussen. 2018. Bubble Treemaps for Uncertainty Visualization. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 719–728.

[24] David Guilmaine, Christophe Viau, and Michael J. McGuffin. 2012. Hierarchically Animated Transitions in Visualizations of Tree Structures. In *International Working Conference on Advanced Visual Interfaces*. 514—-521.

[25] hao86. 2005. Railway statistics. Retrieved August 21, 2024 from https://train.hao86.com/

[26] Jeffrey Heer and George Robertson. 2007. Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1240–1247.

[27] Danny Holten and Jarke J. Van Wijk. 2009. Force-Directed Edge Bundling for Graph Visualization. *Computer Graphics Forum* 28, 3 (2009), 983–990.

[28] Yueqi Hu, Tom Polk, Jing Yang, Ye Zhao, and Shixia Liu. 2016. Spot-Tracking Lens: A Zoomable User Interface for Animated Bubble Charts. In *Proceedings of IEEE Pacific Visualization Symposium*. 16–23.

[29] Adojaan K, Sellis U, Väli Ü, Ojaste I, Denac K, L ohmus A, and Kuze J. 2019. BirdMap Data - GPS tracking of Storks, Cranes and birds of prey, breeding in Northern and Eastern Europe. Retrieved July 4, 2024 from https://doi.org/10.15468/vnwmrx Pluto F. Occurrence dataset.

[30] Aidi Li, Zhijie Xu, Jianqin Zhang, Taizeng Li, Xinyue Cheng, and Chaonan Hu. 2023. A Vector Field Visualization Method for Trajectory Big Data. *ISPRS International Journal of Geo-Information* 12, 10 (2023), 1–21.

[31] Leon YO Li and Zhuo Fu. 2002. The school bus routing problem: a case study. *Journal of the Operational Research Society* 53, 5 (2002), 552–558.

[32] Shixia Liu, Weiwei Cui, Yingcai Wu, and Mengchen Liu. 2014. A survey on Information Visualization: Recent Advances and Challenges. *The Visual Computer* 30 (2014), 1373–1393.

[33] Siyuan Liu, Lionel M. Ni, and Ramayya Krishnan. 2014. Fraud Detection From Taxis' Driving Behaviors. *IEEE Transactions on Vehicular Technology* 63, 1 (2014), 464–472.

[34] Wei Liu, Yu Zheng, Sanjay Chawla, Jing Yuan, and Xie Xing. 2011. Discovering Spatio-Temporal Causal Interactions in Traffic Data Streams. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1010—-1018.

[35] Ferdinando Di Martino, Witold Pedrycz, and Salvatore Sessa. 2019. Hierarchical granular hotspots detection. *Soft Computing* 24, 2 (2019), 1357–1376.

[36] Meinard Müller. 2007. *Dynamic Time Warping*. Springer, 69–84. https://doi.org/10.1007/978-3-540-74048-3_4

[37] Suganuma Mutsumi and Yokosawa Kazuhiko. 2006. Grouping and Trajectory Storage in Multiple Object Tracking: Impairments Due to Common Item Motions. *Perception* 35, 4 (2006), 483–495.

[38] OpenSky Network. 2020. Open Air Traffic Data for Research. Retrieved August 21, 2024 from https://opensky-network.org/datasets/states

[39] Nguyen Quan, Hong Seok-Hee, and Eades Peter. 2012. TGI-EB: A New Framework for Edge Bundling Integrating Topology, Geometry and Importance. In *Proceedings of Graph Drawing*. 123–135.

[40] George Robertson, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. 2008. Effectiveness of Animation in Trend Visualization. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008), 1325–1332.

[41] Hart Sandra G. 2006. Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50, 9 (2006), 904–908.

[42] David Selassie, Brandon Heller, and Jeffrey Heer. 2011. Divided Edge Bundling for Directional Network Data. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2354–2363.

[43] Jia Tao and Ji Zheng. 2017. Understanding the Functionality of Human Activity Hotspots from Their Scaling Pattern Using Trajectory Data. *ISPRS International Journal of Geo-Information* 6, 11 (2017), 1–16.

[44] Yaguang Tao, Alan Both, Rodrigo I. Silveira, Kevin Buchin, Stef Sijben, Ross S. Purves, Patrick Laube, Dongliang Peng, Kevin Toohey, and Matt Duckham. 2021. A comparative analysis of trajectory similarity measures. *GIScience & Remote Sensing* 58, 5 (2021), 643–669.

[45] Dejan Todorovic. 2008. Gestalt Principles. *Scholarpedia* 3, 12 (2008), 5345.

[46] 5D Vision. 2006. Retrieved August 8, 2024 from https://birdmap.5dvision.ee/EN A website demo of BirdMap dataset.

[47] Johan Wagemans, James H. Elder, Michael Kubovy, Stephen E. Palmer, Mary A. Peterson, Manish Singh, and Rüdiger von der Heydt. 2012. A Century of Gestalt Psychology in Visual Perception I. Perceptual Grouping and Figure-Ground Organization. *Psychological Bulletin* 138, 6 (2012), 1172–1217.

[48] Markus Wallinger, Daniel Archambault, David Auber, Martin Nöllenburg, and Jaakko Peltonen. 2022. Edge-Path Bundling: A Less Ambiguous Edge Bundling Approach. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022), 313–323.

[49] Yong Wang, Daniel Archambault, Carlos E Scheidegger, and Huamin Qu. 2018. A Vector Field Design Approach to Animated Transitions. *IEEE Transactions on Visualization and Computer Graphics* 24, 9 (2018), 2487–2500.

[50] Ka-Ping Yee, Danyel Fisher, Rachna Dhamija, and Marti Hearst. 2001. Animated Exploration of Dynamic Graphs with Radial Layout. In *Proceedings of IEEE Symposium on Information Visualization*. 43–50.

[51] Jun Yuan, Mengchen Liu, Fengyuan Tian, and Shixia Liu. 2023. Visual Analysis of Neural Architecture Spaces for Summarizing Design Principles. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2023), 288–298.

[52] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with Knowledge from the Physical World. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 316—-324.

[53] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-Drive: Driving Directions Based on Taxi Trajectories. In *Proceedings of SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 99—-108.

[54] Yu Zheng. 2015. Trajectory Data Mining: An Overview. *ACM Transactions on Intelligent Systems and Technology* 6, 3 (2015), 1–41.

[55] Yixian Zheng, Wenchao Wu, Nan Cao, Huamin Qu, and Lionel M Ni. 2018. Focus+Context Grouping for Animated Transitions. *Journal of Visual Languages & Computing* 48 (2018), 61–69.

A Quantitative Experiment on Edge Bundling

We conducted a quantitative experiment to compare the effectiveness of our trajectory-driven path generation algorithm with two representative edge bundling algorithms. The experiment aims to assess 1) the deviation of the bundled paths from the original ones, and 2) the efficiency in reducing the total path length. The same datasets used in Sec. 5.1 were employed for this experiment.

Baseline algorithms. We selected two representative edge bundling algorithms for comparison. The first, divided edge bundling (DEB) [42], employs the attraction and spring forces to bundle edges with similar positions and directions. The second, multilevel agglomerative edge bundling (MAEB) [22], hierarchically bundles similar edges to minimize the total path length. For both algorithms, we used the default parameters reported in their papers.

Evaluation criteria. We adopted two metrics, deviation and ink ratio [48], which assess the deviation from the original paths and the efficiency in reducing the total path length, respectively. Let the original path set be $S = \{s_1, \ldots, s_n\}$ and the bundled path set be $S' = \{s'_1, \ldots, s'_n\}$.

Deviation measures the misalignment of the bundled paths with the original ones. Following the common practice [21, 44], the misalignment of a bundled path s'_i and its original path s_i is measured by their dynamic time warping distance $\mathrm{DTW}(s_i, s'_i)$. The deviation is then calculated as the average dynamic time warping distance across all pairs of bundled and original paths:

$$\mathrm{Deviation}(S, S') = \frac{1}{n}\sum_{i=1}^{n} \mathrm{DTW}(s_i, s'_i)$$

Ink ratio measures the efficiency in reducing the total path length. It is calculated as the ink of the bundled path set S' divided by that of the original path set S:

$$\text{Ink ratio}(S, S') = \frac{\mathrm{Ink}(S')}{\mathrm{Ink}(S)},$$

where $\mathrm{Ink}(S)$ is the number of pixels occupied by the paths in S [48].

Results

Table 3 shows the comparison results on seven datasets. These results indicate that our algorithm achieves the lowest deviation and performs comparably with the baseline algorithms in terms of ink ratio.

Table 3: Comparison between different edge bundling algorithms, including divided edge bundling (DEB) [42], multilevel agglomerative edge bundling (MAEB) [22], and our algorithm (RouteFlow). Lower values are better for both metrics.

Dataset	Deviation			Ink ratio		
	DEB	MAEB	RouteFlow	DEB	MAEB	RouteFlow
Taxi [52, 53]	0.023	0.017	**0.012**	0.387	**0.349**	0.378
BirdMap [29]	0.048	0.035	**0.016**	0.237	**0.235**	0.239
Railway [25]	0.038	0.031	**0.018**	0.243	0.309	**0.234**
MEIBook [8]	0.034	0.018	**0.017**	0.397	**0.327**	0.388
OpenSkyAirline [38]	0.043	**0.014**	**0.014**	0.393	0.342	**0.339**
US Migration [27]	0.045	0.023	**0.019**	0.432	0.272	**0.250**
DanishAIS [3]	0.071	0.076	**0.024**	0.213	**0.201**	0.252
Average	0.043	0.030	**0.017**	0.329	**0.291**	0.296

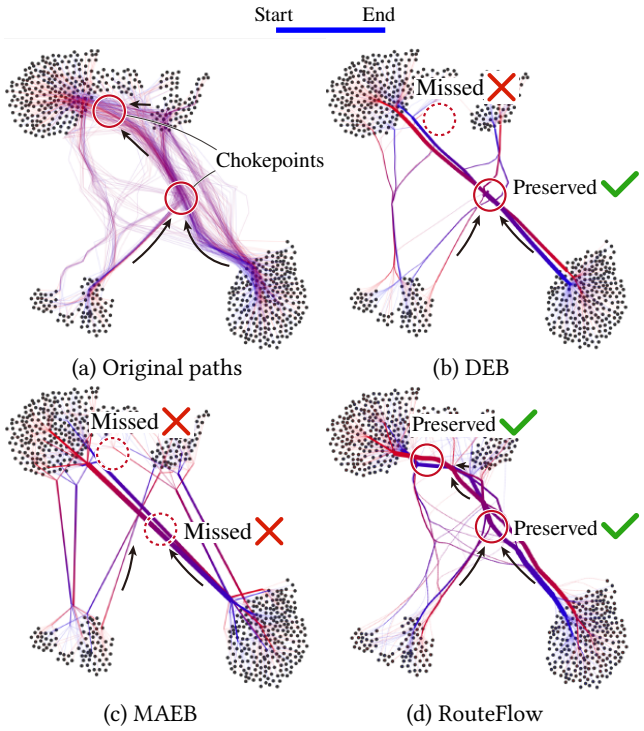


Fig. 14. Original paths and the edge bundling results on the DanishAIS dataset.

Deviation. The baseline algorithms perform worse in terms of deviation because they do not explicitly preserve the original paths and often excessively aggregate paths. This leads to larger deviation from the original paths and hinders the identification of local hotspots. For example, as shown in Fig. 14(a), the DanishAIS maritime transportation dataset contains two chokepoints that significantly impact maritime traffic efficiency, making them strategic hotspots for route optimization. However, the baseline algorithms fail to preserve these chokepoints (Figs. 14(b) and (c)). In contrast, RouteFlow reduces the deviation from the original paths using an anchor force and effectively preserves these two chokepoints (Fig. 14(d)). This preservation enables more accurate and effective route optimization.

Ink ratio. Out of the seven datasets, MAEB achieves the lowest ink ratio in four, while RouteFlow achieves the lowest in three. MAEB's better performance is due to its focus on optimizing ink ratio, but this comes at the cost of higher deviation from the original paths. In contrast, RouteFlow balances ink ratio and deviation, sacrificing a small amount of ink ratio to better align the bundled paths with the original ones. This balance is important in generating clear and reliable animation. For example, as shown in Fig. 15(a), migratory birds in the BirdMap dataset take detour paths along the Great Rift Valley to access water or suitable habitats during their journey from Europe to Africa. These detour paths are critical to understanding the birds' movement patterns in the context of geographical features. However, MAEB distorts the original paths and bundles them into straight paths to minimize the ink ratio, which obscures

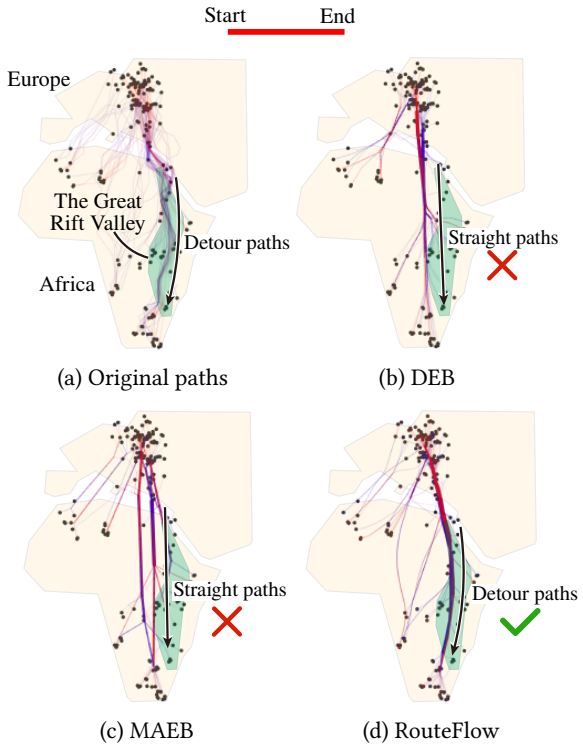


Fig. 15. Original paths and the edge bundling results on the BirdMap dataset.

the birds' actual movement patterns (Fig. 15(c)). Although DEB does not explicitly minimize the ink ratio, it also bundles the paths into straight paths and obscures the movement patterns in this example (Fig. 15(b)). In contrast, RouteFlow effectively bundles the paths while preserving the detour paths (Fig. 15(d)). This provides a clearer and more reliable visual summary of the birds' movements.

B Detailed Animation Result Analysis

To facilitate interpreting the magnitudes of the measured values, Fig. 16 compares the animation results with different measured values, which are generated by three methods: RouteFlow, the focus+context grouping method (F+C) and the vector-field-based method (VF). These methods are the same as in Sec. 5.1. Fig. 16(a) shows the positions of all objects, the same as in Fig. 9. For easy comparison, we select a subset of 10 objects in a group (with the orange contour), and calculate their values for each metric in this frame (Figs. 16(c)-(e)), except for overall occlusion, which is calculated on all objects (Fig. 16(b)).

Occlusion. Figs. 16(b) and (c) compare the animation results generated by the three methods in terms of the overall and within-group occlusion. RouteFlow prevents overlap between objects when they move together as a group, thereby achieving an occlusion score of 0.00000 for both metrics at this frame. However, overlaps remain unavoidable at local hotspots where objects converge and diverge. In contrast, the focus+context grouping method and the vector-field-based method result in object overlap to different degrees. For example, the focus+context grouping method achieves an overall

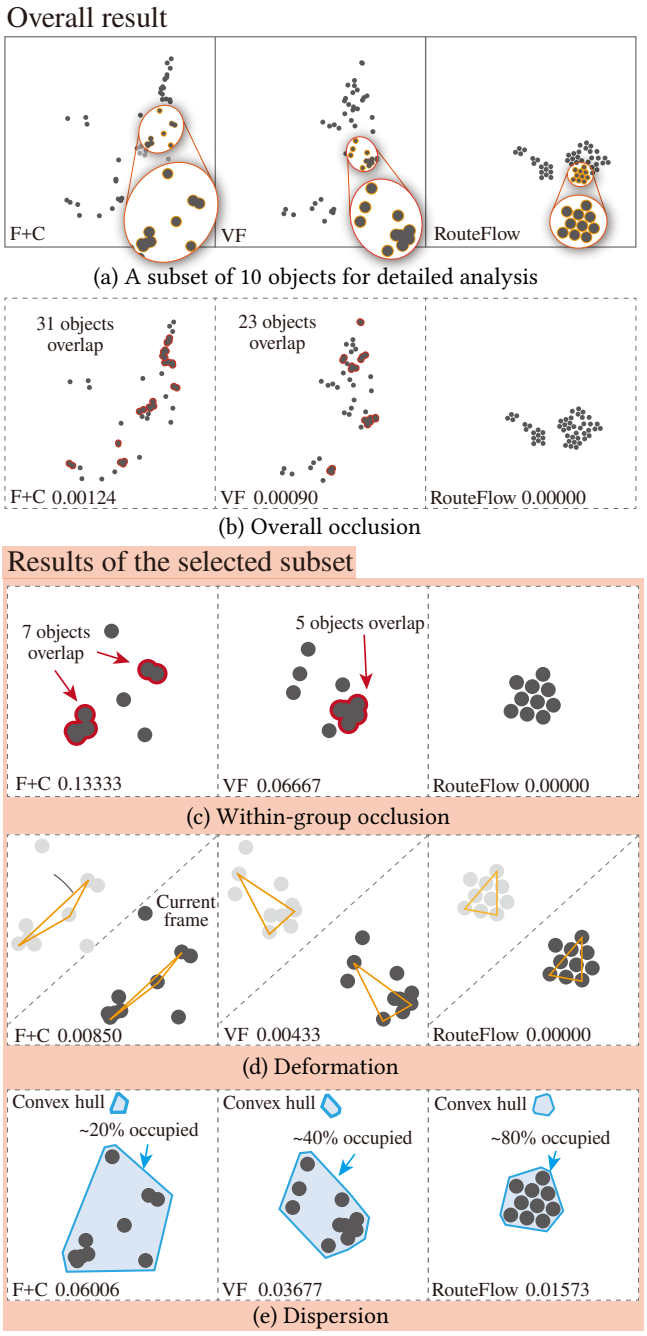


Fig. 16. Comparisons of the three animation results with different metric values, which are generated by the three methods. Here, (b)-(e) show the different metric values of the 10 objects.

occlusion score of 0.00124, with 31 out of 51 objects overlapping, and a within-group occlusion score of 0.13333, with 7 of the 10 objects overlapping.

Deformation. Fig. 16(d) compares the animation results generated by the three methods in terms of deformation. This metric

measures the changes in distance between objects from the previous frame (grey) to the current frame (dark grey). RouteFlow organizes the objects that move together into a cohesive group, with their relative positions changing only at local hotspots. As a result, the deformation score is 0.00000, indicating no changes in the distance between objects. In contrast, the focus+context grouping method and the vector-field-based method do not explicitly manage the object layout, leading to larger deformation scores. For example, the focus+context grouping method achieves a deformation score of 0.00850, indicating that the average distance between objects has changed by approximately 1.46 times the object's radius.

Dispersion. Fig. 16(e) compares the animation results generated by the three methods in terms of dispersion. RouteFlow closely groups the objects that move together, resulting in the lowest dispersion score of 0.01573, with the objects occupying approximately 80% of the space within their convex hull. In contrast, the focus+context grouping method and the vector-field-based method exhibit higher dispersion scores. For example, the focus+context grouping method achieves a dispersion score of 0.06006, with the objects occupying approximately 20% of the space within their convex hull.

C User Study Settings

C.1 Synthetic Dataset Generation

As illustrated in Fig. 10, our dataset generation process involves three steps: generate the global trend, determine local hotspots, and create trajectories.

Since the global trend can be depicted by representative trajectories [54], we generate a smooth trajectory using B-splines as the ground truth. We include one or two bends in these B-splines, leading to 2 types of the global trend in our dataset. To achieve this, we randomly select two points as the start and end points and sample one (for one bend) or two (for two bends) intermediate points as B-spline control points within the region between them. To avoid highly curved trajectories, we ensure that the angle between any three consecutive points (start, control points, and end) is greater than 135 degrees.

Next, to determine local hotspots, we randomly sample points along the generated B-spline and designate these points as ground truth for local hotspots. Feedback from our pilot study indicates that to balance complexity and diversity, the number of local hotspots should be limited to two or three, with each dataset containing at least one of the local hotspot types: convergence or divergence. Each local hotspot is then randomly assigned as either a converging or diverging point, resulting in three possible assignments: 1 convergence + 1 divergence, 2 convergences + 1 divergence, and 1 convergence + 2 divergences. Considering the types of the global trend and the local hotspot assignments, we generate 6 dataset types (2 types of the global trend $\times$ 3 local hotspot assignments). To simplify the evaluation, we 1) limit the number of branches at the local hotspots to 2; 2) select one B-spline as the global trend and generate another branch at the local hotspots along this B-spline; and 3) avoid crossing between the two branches of each local hotspot.

Finally, we generate the trajectories for the objects. Following the user feedback and the setting of previous studies [55], we set the number of trajectories to 30. The trajectories are generated by adding random perturbations to the global trend and the branches of the local hotspots.

C.2 Method Counterbalance

In our user study, we counterbalanced the order of different methods. We divided 15 participants into five groups, with three participants in each group. Within each group, we used an expanded Latin square and applied a cyclic shift to the method order for each participant. The three methods are denoted as A, B, and C. In the test scenario, the order in which these methods were presented to the participants in each group was as follows:

- A B C, B C A, C A B, A B C, B C A, C A B, A B C, B C A, C A B, A B C, B C A, C A B
- B C A, C A B, A B C, B C A, C A B, A B C, B C A, C A B, A B C, B C A, C A B, A B C
- C A B, A B C, B C A, C A B, A B C, B C A, C A B, A B C, B C A, C A B, A B C, B C A