## APPENDIX A
## MOVEMENT EXTRACTION ALGORITHMS AND EVALUATION

### A. Problem Definition

We first consider a continuously measured trajectory $\Gamma$ of a mobile user during a time period $T$. $\Gamma$ is defined by the set of spatiotemporal records in $T$: $\Gamma = \bigcup_{t\in T} < t, \ell(t) >$ where $\ell(t)$ denotes the location of the user at time $t$. Any continuous sub-trajectories of $\Gamma$ are defined as the trips of $\Gamma$. For example, the trip $\gamma$ during the time period $\tau \subseteq T$ is defined as $\gamma = \bigcup_{t\in\tau} < t, \ell(t) >$.

*Definition 1:* STAY/TRAVEL TRIP – for any trip $\gamma$ of a trajectory $\Gamma$ during the time period $\tau$:

(a) $\gamma$ is a **stay trip** if: $|\tau| \geq \Delta T$ and $||\ell(t_1) - \ell(t_2)|| < \Delta S$ $(\forall t_1, t_2 \in \tau)$;

(b) $\gamma$ is a **travel trip** if: all the sub-trajectories of $\gamma$ do not satisfy (a);

(c) The state of the urban trajectory $\Gamma$ at any time $t$ within the trip $\gamma$ ($t \in \tau$), denoted by $I_{S/T}(t)$, is *Stay* (*Travel*) if $\gamma$ is the stay (travel) trip.

Here $|\cdot|$ denotes the length of a time period, $||\cdot||$ is the $L_2$ norm that represents the spatial distance between two locations. $\Delta T$ and $\Delta S$ are the temporal and spatial thresholds in the definition.

In essence, Definition 1(a) models the stay trip as a sufficiently long time period ($\geq \Delta T$) when the trajectory is kept within a circular region of diameter $\Delta S$. This definition is consistent among all the previous literature on the stay point detection [2] [3]. On the other hand, based on the ground truth that a user can either stay or travel in any time point, the trip not overlapped with any stay trip is determined as a travel trip (Definition 1(b)).

The real-world human trajectory is hardly measured continuously, but consists of a list of discretely sampled records on certain time points ($t_1 < \cdots < t_L$): $\Gamma = \bigcup_{t\in\{t_1,\cdots,t_L\}} < t, \ell(t) >$. It has been shown in Ref. [6] that if the sampling process is dense, i.e., $\forall i \in [1, L), ||t_i - t_{i+1}|| << \Delta T/2$, any travel trip extracted from the discretely sampled trajectory by Definition 1(b) is very close to the travel trip detected on the underlying continuous trajectory. The extraction of travel trips from dense trajectories can be computed through an iteration-based algorithm by Definition 1 (Section A-B).

Nevertheless, in this work we are given temporally sparse trajectories with an average record interval of 2.5 hours (compared with a default parameter of $\Delta T = 15min$). This research problem is formulated as:

PROBLEM: TRIP EXTRACTION ON SPARSE TRAJECTORY

**Given:** (1) *a set of users*; (2) *each user's sparse trajectory* $\Gamma = \bigcup_{i\in[1,L]} < t_i, \ell(t_i) >$;

**Detect:** *the stay/travel trips in each sparse trajectory by inferring the state of each record* $I_{S/T}(t_i), \forall i \in [1, L]$.

The problem of the trip extraction on the sparse trajectory is difficult. Take a uniformly sampled trajectory with the average consecutive record interval larger than the time threshold $\Delta T$ as an example. No state information can be inferred confidently on any of the records. Before or after each record, there is an unobserved time period with a length of at least $\Delta T$. Observing this time period will probably classify the state of the record as stay if Definition 1(a) is met. The path inference algorithms [5] [4] can potentially recover the unobserved trajectory, but these algorithms generally work for an average time interval of seconds or a few minutes, and fails in our case with a typical time threshold of $\Delta T = 30$ minutes. Meanwhile, having a large average time interval would not necessarily lead to an unsolvable state inference problem. As shown in the empirical study, our urban trajectory dataset has a long-tailed sparsity, which is positive in inferring the state of the trips inside a trajectory. There are many cases that the trajectory is densely measured in its sub-trajectories despite of its overall sparsity. In these sub-trajectories, the stay/travel trips can be confidently inferred, e.g., when the record intervals are as small as a few seconds. Below we define the metric that captures the sparsity of the trajectory, subject to the feasibility for the trip inference.

**Algorithm 1:** The exact algorithm on dense trajectories.

**Input :** $\Gamma = \bigcup_{i\in[1,L]} < t_i, \ell(t_i) >, t_1 < \cdots < t_L$ (dense trajectory), $\Delta T$, $\Delta S$ (stay/travel trip threshold)
**Output:** $I_{S/T}(t_i), \forall i \in [1, L]$ (state of each record)

```
begin
    for head ← [1, L−1] do
        for cursor ← [head + 1, L] do
            /* iterate candidate stay trips */
            if t_cursor − t_head ≥ ΔT then
                for i ← [head, cursor − 1] do
                    for j ← [i + 1, cursor] do
                        if ||ℓ(t_i) − ℓ(t_j)|| ≥ ΔS then
                            Stay ← False, Break
                if Stay! = False then
                    for i ← [head, cursor] do
                        I_{S/T}(t_i) ← S
    /* the remaining records are travel trips */
    for i ← [1, L] do
        if I_{S/T}(t_i)! = S then
            I_{S/T}(t_i) ← T
    return I_{S/T}(t_i), i = [1, L]
```

*Definition 2:* SPARSITY OF THE TRAJECTORY – for any trajectory $\Gamma$ observed at $t \in \{t_1, \cdots, t_L\}$:

(a) the **global sparsity** of $\Gamma$ is proportional to the average time interval between the consecutive records observed on the trajectory: $\xi_G(\Gamma) = E(t_{i+1} - t_i), \forall i \in [1, L-1]$;

Here $E(\cdot)$ denote the mean function of a set. Note that the global sparsity is irrelevant to the time threshold $\Delta T$.

### B. Algorithm

We start from the trip extraction problem on the densely sampled trajectory. The problem can be exactly solved by

an iteration-based algorithm (Algorithm 1). The algorithm first detects all the stay trips by Definition 1(a) and then checks the remaining sub-trajectories to detect the travel trips by Definition 1(b). The iteration-based algorithm needs to scan all the possible sub-trajectories, thus leading to a $O(L^4)$ computational complexity where $L$ is the number of records in a trajectory. This is costly given that there are a huge number of trajectories with an average length of several hundred records.

According to the long-tailed sparsity pattern discovered in this work, we design a new algorithm to infer stay/travel trips from a single long-tailed sparse trajectory, called Slice & Doubly Sliding (SDS). As shown in Algorithm 2, the algorithm first slices the trajectory into multiple dense segments at all the intervals larger than $\Delta T$ (L2). On each dense segment $\gamma$, the stay/travel segments are detected respectively (L3∼10, L11∼19). In particular, to avoid the worst-case $O(L^4)$ complexity in stay trip extractions, we introduce a doubly sliding window data structure which keeps track of the currently checked segment. The key of the algorithm lies in that, when one pair of records no closer than $\Delta S/3$ are found (L6), all the segments containing this pair of records will be pruned early in the detection and the sliding window will advance aggressively (L10). The travel detection follows Theorem 3 which is detailed and proved in Ref. [6]. The average-case complexity of SDS is $O(L \cdot W)$ where $W$ is the average number of records in a maximal stay trip.

*Theorem 3:* CONTINUOUS MOBILITY OF TRAVEL RECORDS – Consider a discrete trajectory $\Gamma$ defined in the time series $\omega = \{t_1, \cdots, t_L\}$:

(a) any record at time $t_i$ $(1 < i < L)$ is in the travel trip by the continuous model of Definition 1(b) under the parameters of $\Delta S$ and $\Delta T$ if only there exist $1 \le p < i < q \le L$ that: 1) $||\ell(t_i) - \ell(t_p)|| \ge \Delta S$; 2) $||\ell(t_i) - \ell(t_q)|| \ge \Delta S$; 3) $t_q - t_p \le \Delta T$;

(b) any record at time $t_i$ can be inferred as in the travel segment by Definition 1(b) under the parameters of $\Delta S$ and $\Delta T$ only if there exist $1 \le p < i < q \le L$ that: 1) $||\ell(t_i) - \ell(t_p)|| \ge \Delta S/2$; 2) $||\ell(t_i) - \ell(t_q)|| \ge \Delta S/2$; 3) $t_q - t_p \le \Delta T$.

The SDS algorithm guarantees a 100% precision in the detection of both stay and travel trips. By Theorem 3(b), the lower bound of recall in detecting travel trips is $\frac{SDS(\Gamma,T,\Delta S,\Delta T)}{SDS(\Gamma,T,\Delta S/2,\Delta T)}$, where $SDS(\Gamma,T,\Delta S,\Delta T)$ is the number of travel records detected by the SDS algorithm from $\Gamma$ under the parameters of $\Delta S$ and $\Delta T$. Note that the recall is defined on all the stay and travel records that can be detected given the single sparse trajectory, not on the continuous stay/travel records given the full trajectory information.

### C. Experiment

We apply the SDS algorithm to the dataset of Beijing. Figure 1 summarizes the percentage of records inferred as in the stay and travel trips respectively. All the curves are bell-shaped with only one peak: the highest ratio of stay is found at the global sparsity around 1.6 min (97.3%∼98.5%, Figure 1(a)); the highest ratio of travel is found at the global sparsity from 5 min to 10 min, which increases with $\Delta T$ (0.34%∼3.8%, Figure 1(b)).

In the stay/travel trip definition, the thresholds of $\Delta S$ and $\Delta T$ need to be determined. In fact, these thresholds provide the flexibility to capture the multi-scale mobility in the human trajectory. Inside the city boundary, $\Delta T$ and $\Delta S$ can be minutes and meters to describe the short-term stays and travels; while in the state level, $\Delta T$ and $\Delta S$ can be days and hundreds of kilometers to characterize the stay in a city and the travel between cities.

We focus on the detection of intra-city travels because the number of travel records is much fewer than the stay and the detection of stay is relatively insensitive to the parameter change (Figure 1(a)(c)). The goal is to detect more travels while keeping the mobility definition reasonable. According to

**Algorithm 2:** SDS algorithm on long-tailed sparse trajectories.

**Input :** $\Gamma = \bigcup_{i\in[1,L]} < t_i, \ell(t_i) >, t_1 < \cdots < t_L$ (sparse trajectory), $\Delta T$, $\Delta S$ (stay/travel trip threshold)
**Output:** $I_{S/T}(t_i), \forall i \in [1,L]$ (state of each record)

1 **begin**
  /* slice $\Gamma$ into $M$ segments ($\gamma_j$) at every interval larger than $\Delta T$ */
2  $\{\gamma_j = \{< t_{j,k}, \ell(t_{j,k}) >\}_{k\in[1,L_j]}\}_{j=[1,M]} \leftarrow Divide(\Gamma, \Delta T)$
3  **for** $j \leftarrow [1,M]$, $head \leftarrow 1$ **do**
    /* detect all the stay trips on $\gamma_j$ */
4  **for** $cursor \leftarrow [2, L_j]$ **do**
      /* iterate all the records backward from $cursor-1$ */
5  **for** $anchor \leftarrow [cursor-1, head]$ **do**
        /* cut at the first escape outside a range of $\frac{\Delta S}{3}$ */
6  **if** $||\ell(t_{j,cursor}) - \ell(t_{j,anchor})|| \ge \frac{\Delta S}{3}$ **then**
          /* stay trip */
7  **if** $t_{j,cursor-1} - t_{j,head} \ge \Delta T$ **then**
8  **for** $k \leftarrow [head, cursor-1]$ **do**
9  $I_{S/T}(t_{j,k}) \leftarrow$ S
10  $head \leftarrow anchor + 1$, **Break**
    /* detect all the travel records on $\gamma_j$ */
11  **for** $cursor \leftarrow [2, L_j - 1]$ && $I_{S/T}(t_{j,cursor}) \ne S$ **do**
      /* find the first record on the left outside a range of $\Delta S$ */
12  **for** $l \leftarrow [cursor-1, 1]$ **do**
13  **if** $||\ell(t_{j,cursor}) - \ell(t_{j,l})|| \ge \Delta S$ **then**
14  $left \leftarrow l$, **Break**
      /* find the first record on the right outside a range of $\Delta S$ */
15  **for** $r \leftarrow [cursor+1, L_j]$ **do**
16  **if** $||\ell(t_{j,cursor}) - \ell(t_{j,r})|| \ge \Delta S$ **then**
17  $right \leftarrow r$, **Break**
18  **if** $t_{j,right} - t_{j,left} \le \Delta T$ **then**
19  $I_{S/T}(t_{j,cursor}) \leftarrow T$
20  **return** $I_{S/T}(t_i), i = [1,L]$

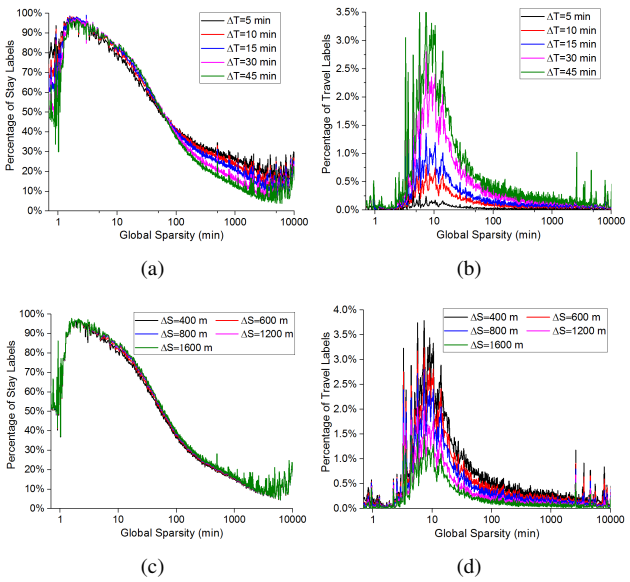



Fig. 1: The performance of the SDS algorithm on the dataset of Beijing: (a)(b) the percentage of stay/travel records detected by the global sparsity of the trajectory and $\Delta T$ ($\Delta S$=800 m); (c)(d) the percentage of stay/travel records detected by the global sparsity of the trajectory and $\Delta S$ ($\Delta T$=30 min).

Figure 1(b), we pick $\Delta T = 30min$ because the detected ratio of travel does not increase much when switching to $\Delta T = 45min$ and it does not impose a strict stay definition in Definition 1(a). Similarly, according to Figure 1(d), we pick $\Delta S = 800m$ which maximizes the recall of travel ($\frac{SDS(\Gamma,T,\Delta S,\Delta T)}{SDS(\Gamma,T,\Delta S/2,\Delta T)}$) and allows a mild stay definition compared with $\Delta S = 400m$. The parameters of $\Delta T = 30min$ and $\Delta S = 800m$ are consistent with the empirical settings of Ref. [2] [3].

## Appendix B
## User Requirement Collected from Expert Interviews

In a pilot study, we interviewed three groups of domain experts, including urban planners, public safety officers, and business analysts. First, we introduced the urban trajectory data collected in this work and the tentative visualization design for presenting the data to them. Next, the experts were asked several questions and required to provide their answer or feedback to the questions. After the interview, experts were paid with consulting fee for their time. Note that there might be more than one people interviewed in each group of experts and each expert will respond to all the questions. Below we report answers to three key questions from each group of experts by combining people from the same group together.

### A. Urban Planner

*Q1: What are the characteristics of this type of trajectory data from your experience with urban analytics?*

A: I think I would champion its fine granularity in the measurement of urban movements. Each individual's location is reported at reasonable precision (we caution that at this point of the interview, we have not fully figured out the temporal sparsity of the data set). Before, we mostly rely on census data from traffic surveys or high-level traffic measurement for urban planning, e.g., road-side sensors reporting traffic volume. This is probably a new opportunity, though you do not have more attributes of these urban users due to the privacy issue, e.g., their genders, professions, etc.

*Q2: What kind of tasks you think this data and its visualization can help for your everyday job?*

A: Many tasks! I could link many tasks with this data. The most immediate thought is to help understand the commute traffic of the city, e.g., the major incoming/outgoing areas in the everyday commute, the highly utilized urban roads, the bottleneck of the commuting road network. Notably, it will be interesting to evaluate the work-home balance of citizens using your data. Previously this could be hard because of the limited spatial and/or temporal granularity of urban data.

*Q3: What is the current practice on the tasks you mentioned, is there any solution using visualization?*

A: As I said, we use traffic census data, i.e., the volume of traffic among districts (blocks) of the city. Some visualizations have been applied, notably the "jump wire" display as we often call them, imagine some arrows among the districts. This simple visualization could help in the high level, but can not show the detailed road utilization in the city during a particular time.

### B. Public Safety Officer

*Q1: What are the characteristics of this type of trajectory data from your experience with urban analytics?*

A: Big data I think. It is a typical source of urban big data and might be one of the earliest big data source for urban analytics as we know. There are also several other data source from government or data companies. An interesting idea will be to fuse these data sources for urban analytics.

*Q2: What kind of tasks you think this data and its visualization can help for your everyday job?*

A: For my job responsibility, situation awareness of the city is of current concern related to visualization. We have very large screens showing live data streams from multiple data sources. Most of these data sources are video streams for status of urban transportation or situation in key roads, regions, or buildings. There is a lack of the kind of visualization showing the overall situation of the city, maybe fusing multiple data sources. I think your data might be a good candidate to serve this need.

*Q3: What is the current practice on the tasks you mentioned, is there any solution using visualization?*

A: Visualization of urban data in large screens. The challenge is mainly from the techniques to process and present multiple sources of big urban data.

### C. Business Analyst

*Q1: What are the characteristics of this type of trajectory data from your experience with urban analytics?*

A: This is probably the first time in our company to use the full-scale data of a city for visualization (Note that these business analysts are from the same company that provides us the urban trajectory data. Their job responsibility is to work with company's customers to solve customer problems with the collected urban data). The scale and granularity of this kind of urban data could help to understand the microcirculation of people in modern cities.

*Q2: What kind of tasks you think this data and its visualization can help for your everyday job?*

A: We work with many customers to innovate with our data, in almost all kinds of economic sectors. For example, we work with real estate companies to help them make business decisions in the pricing and purchasing of certain real estate projects. A visualization of fine-grained urban movements around the candidate projects would be a reference for the decision-making of our customers.

*Q3: What is the current practice on the tasks you mentioned, is there any solution using visualization?*

A: We mainly use data mining and machine learning methods now, but these methods are not so intuitive in presenting to our customers. Visualization could be a big plus for us.

## APPENDIX C
## COMPARATIVE STUDY WITH CARTOGRAPHIC FLOW MAP VISUALIZATION

UrbanMotion is an aggregation-based visualization technique to display massive urban movements. From the methodology point of view, it is similar to several existing methods using spatial aggregation, notably the cartographic flow map visualization proposed by Andrienko and Andrienko [1] (FlowMap in short). The fundamental difference lies in that UrbanMotion is designed for trajectory data with long-tailed temporal sparsity, in which most travel trips extracted are only tiny segments of the full trajectory and are extremely short in time (mostly in minutes). We propose UrbanMotion techniques to extract, aggregate, and visualize both global and local urban movements out of these tiny segments. To interpret this subtle difference and understand the pros and cons of UrbanMotion for movement visualization, we conducted a lab experiment to compare the visualization result of UrbanMotion and FlowMap on the same long-tailed sparse trajectory data set.

The FlowMap visualization method is implemented according to Ref. [1] and also adapted to take the new trajectory data. First, travel trips are detected by the same method in UrbanMotion (Algorithm 2). Note that large intervals between consecutive location records will make it impossible to know the actual route between these records. We use a smaller interval of one minute to cut the original trajectory in Line 2 of Algorithm 2 (15 minutes by default for UrbanMotion). This is consistent with the suggestion in Ref. [1] where the record intervals in their data are mostly between 30 and 45 seconds. The travel trips detected are processed by an optional map-matching stage and then used as the input trajectory data of FlowMap method. Second, characteristic points are extracted from input trajectories and then grouped to form Voronoi tessellation. Finally, trajectories are aggregated by the Voronoi tessellation into a mobility network among Voronoi cells, and displayed by classical flow map visualization. The line thickness indicates the volume of each flow. A default setting of $MaxRadius = 3km$ is applied according to the recommendation in Ref. [1] and can be adjusted for level-of-detail viewing.

The visualization results of urban movements in Beijing on 9AM-10AM of July 5th are given in Figure 2, (a)(b)(c)(d)(f) by FlowMap and (g)(h) by UrbanMotion. Different color themes are applied because the white color theme of UrbanMotion is not good for presenting flow volume due to the simultaneous coding of movement speeds. It can be found that comparing the overview visualization by FlowMap (Figure 2(a) using a default parameter of $MaxRadius = 3km$) and UrbanMotion (Figure 2(g)), the output of FlowMap represents a higher level of abstraction, while the output of UrbanMotion better corresponds to the actual road network of the city of Beijing (Figure 2(e)). Both methods in comparison do not apply map-matching. We drill down to examine why the overview of the two methods are different. It is discovered that the visualization by FlowMap in Figure 2(a) only represents 5.42% of all the raw travel trips. The remaining 94.58% of travel trips detected are very short and stay within the same Voronoi cell because of the long-tailed sparse nature of the trajectory data. In other words, for each trajectory of a urban user, multiple travel trips are detected, most of which lie within the boundary of a single Voronoi cell. Meanwhile, UrbanMotion uses a much smaller cell setting in the aggregation of movements. More travel trips are visualized among spatial cells.

Applying the map-matching technique, the FlowMap visualization result in Figure 2(b) is mildly changed from the result over unmatched movement data (Figure 2(a)). This is because the measurement error of location records is much smaller than the parameter of *MaxRadius* in FlowMap (e.g., 3km). The movement aggregation is less affected by the map-matching, except the Voronoi tessellation computed from the distribution of map-matched records. By design, FlowMap with map-matching still connects the generating point of Voronoi cells and remains a higher level of abstraction compared with the UrbanMotion visualization.

We try to increase the granularity of FlowMap method to obtain an overview that better covers of the data set. Figure 2(c) depicts the result under $MaxRadius = 1km$ (covering 13.05% of all the raw travel trips), which however multiplies the visual complexity. When the bottom 80% flows by volume are filtered out, the remaining flows can not represent a good overview of movements in the city (Figure 2(d)). In comparison, UrbanMotion applies a default cell size of 500m (approximately $MaxRadius = 250m$), which ensures a much higher ratio of cross-cell travel trips. The direction of within-cell movements are also considered in the trajectory analysis. The movement clustering and flow tracing algorithms further help to extract the backbone of global movement flows in the city while preserving a moderate visual complexity. FlowMap

also displays within-cell movements by charting their overall flow volume, but the movement direction of within-cell movements are not shown (accounting for 94.58% movements under $MaxRadius = 3km$). In addition, as proposed in the main document of this paper, applying map-matching to UrbanMotion helps to examine local movement patterns in focused areas of a city. The detailed UrbanMotion visualization in Figure 2(h) illustrates the utilization of major roads in the Wangjing district of Beijing.

Through our experiments, we also identify two major disadvantages of UrbanMotion in comparison to FlowMap. First, UrbanMotion adopts a fix-sized gridding of the city map with relatively small cells. Though it better covers the long-tailed sparse trajectory data, in areas with few movements, the technique is less efficient as the same number of cells need to be processed. FlowMap resolves this issue by introducing Voronoi tessellation. Through characteristic point detection and grouping, areas with low density of movements will have larger Voronoi cells while areas with high density of movements will have smaller Voronoi cells. The spatial data aggregation is more efficient under uneven movement distribution. Second, FlowMap is more suitable for online movement visualization over streaming data. The Voronoi tessellation can be computed offline using historical data. The online processing only needs to map each characteristic point into a corresponding Voronoi cell. In comparison, UrbanMotion computes movement clustering on each cell or map-matching on each location record. These stages are both costly, as shown in the implementation details of Sec. 4.5 and the performance result in Table 2. For now, UrbanMotion is mainly used for the replay of existing trajectory data due to the limitation in computational complexity.

## REFERENCES


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

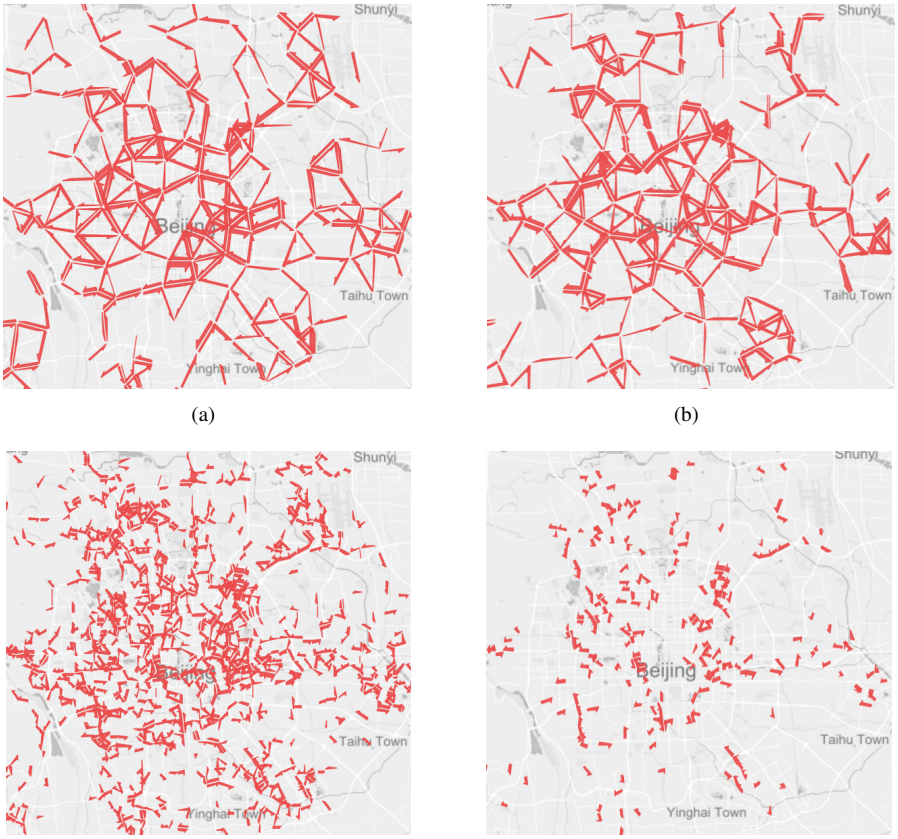
[2] F. Calabrese, F. C. Pereira, G. Di Lorenzo, L. Liu, and C. Ratti. The geography of taste: analyzing cell-phone mobility and social events. In *PERCOM'10*, pp. 22–37, 2010.

[3] S. Jiang, G. A. Fiore, Y. Yang, J. Ferreira Jr, E. Frazzoli, and M. C. González. A review of urban computing for mobile phone traces: current methods, challenges and opportunities. In *UrbComp'13*, p. 2, 2013.

[4] W. Li, D. Nie, D. Wilkie, and M. C. Lin. Citywide estimation of traffic dynamics via sparse gps traces. *IEEE Intelligent Transportation Systems Magazine*, 9(3):100–113, 2017.

[5] M. Rahmani and H. N. Koutsopoulos. Path inference from sparse floating car data for urban networks. *Transportation Research Part C: Emerging Technologies*, 30:41–54, 2013.

[6] L. Shi. Mobility inference from long-tailed sparse trajectories. *arXiv preprint arXiv:2001.07636*, 2020.




(a) (b)

(c) (d)

(e) (f)

(g) (h)

Fig. 2: Visual comparison of FlowMap and UrbanMotion on the same sparse trajectory data set: (a) abstract overview by FlowMap without map-matching ($MaxRadius = 3km$); (b) abstract overview by FlowMap using map-matched movement vectors as input data ($MaxRadius = 3km$); (c) more fine-grained visualization by FlowMap ($MaxRadius = 1km$, without map-matching); (d) filter to show top 20% flows ($MaxRadius = 1km$); (e) the road network of the city of Beijing; (f) local movements in the Wangjing district by FlowMap using map-matching; (g) abstract overview by UrbanMotion without map-matching; (h) local movements by UrbanMotion using map-matching.