Urban Trajectory Timeline Visualization

Zuchao Wang
Key Laboratory of Machine Perception
(Ministry of Education)
and School of EECS
Peking University, Peking, China, 100871
Email: zuchao.wang@pku.edu.cn

Xiaoru Yuan
Key Laboratory of Machine Perception
(Ministry of Education)
and School of EECS
and Center for Computational Science and Engineering
Peking University, Peking, China, 100871
Email: xiaoru.yuan@pku.edu.cn

***Abstract*—In this paper, we propose using timelines for 2D trajectory comparison. Trajectories directly rendered on a map do not show temporal information well, and are cluttered and unaligned. This make them difficult to compare. We convert trajectories to timelines, which naturally shows time, and are more compact and easy to align. In addition to simply showing how an attribute varies along the time, we further propose some novel timelines to show spatial-temporal features. We provide some use cases to show the benefit of our method.**

I. INTRODUCTION

With the advances of sensing technology, the trajectory data is pervasive now. Taxis, ships, airplanes, hurricanes, cell phone users and animals are all leaving trajectory data record. This gives us the opportunity to explore and understand complex social, economical and scientific phenomena. However, we face the challenge to effectively analyze this data, and extract knowledge from it. Trajectory visualization is one important strategy for such analysis. It is able to provide us with an intuitive overview of various aspect of the trajectories, help validate what is known and reveal what is unknown.

Trajectory visualization usually consists of three different but related aspects: space, time and attribute. The focus in this paper is the temporal aspect. It is hard to visualize and analyze, especially when the number of trajectory is over several hundreds. We explore the timeline visualization of trajectories. Timeline is a widely used visual metaphor for temporal visualization. When used in trajectory visualization, it has the following advantages:

- It intuitively and effectively show the temporal information, easy for navigation
- It can be packed side by side, therefore do not have visual cluttering
- It is convenient for analysis: compare, sort, cluster and align are easy

Our method will be restricted to 2D trajectories. That is, trajectories without height information. In order to apply our method to 3D trajectories, we can first transform them to 2D by dropping the height information. This is reasonable in many cases.

In this paper, we show how to construct a set of timelines from one trajectory, based on different criteria. Our timelines are not limited to those showing how an attribute change with time. We also design timelines to show the change of spatial features, such as curvature, straightness, turns and stops. This is one attempt to combine the spatial and temporal aspect in one visualization, which is usually considered important yet difficult. We arrange the timelines in a 2D heatmap view and a 3D terrain view. Besides, we link our timeline visualizations to other spatial and attribute visualization, and make some real analysis. Our major contributions are:

- Design some new forms of timeline visualization of 2D trajectory, such as turn-plot and stop-plot
- Build a system, and use timelines to analyze real trajectories

We will first review the related works. After that, we will present the timeline construction and interaction. We then show the interface of our system, and provide with some results on real data. Finally it's the discussion and conclusion.

II. RELATED WORK

Timeline is a widely used visualization technique for temporal data visualization. Some researchers adopt timeline directly for trajectory data visualization, some use other metaphor to show the temporal information. There are still a few line representation techniques, which can produce results similar to timeline.

A. Timeline Visualization

The most popoulat visualization method for temporal data is arguably timeline visualization. Each object or event is visualized as a line, showing its start and end time [18]. In addition, attribute values can be depicted on the timeline to show how the attribute change over time. Interactive query and filtering are supported [13]. Timeline visualization usually works well when the number of time series data is small. As long as the number goes large, cluttering occurs and features are hidden. Aggregation [22] can be used to give an overview, but mainly start, end time and time duration. The trend of attribute change are hard to summarize. Without aggregation, one strategy to avoid this is to use focus+context [17] technique. Line plots are shown only when there's enough space, but it does not give a good overview. Another way is to use height [15] or color [7] to show the attribute value. We use height in our 3D terrain view, and color in our 2D heatmap view.

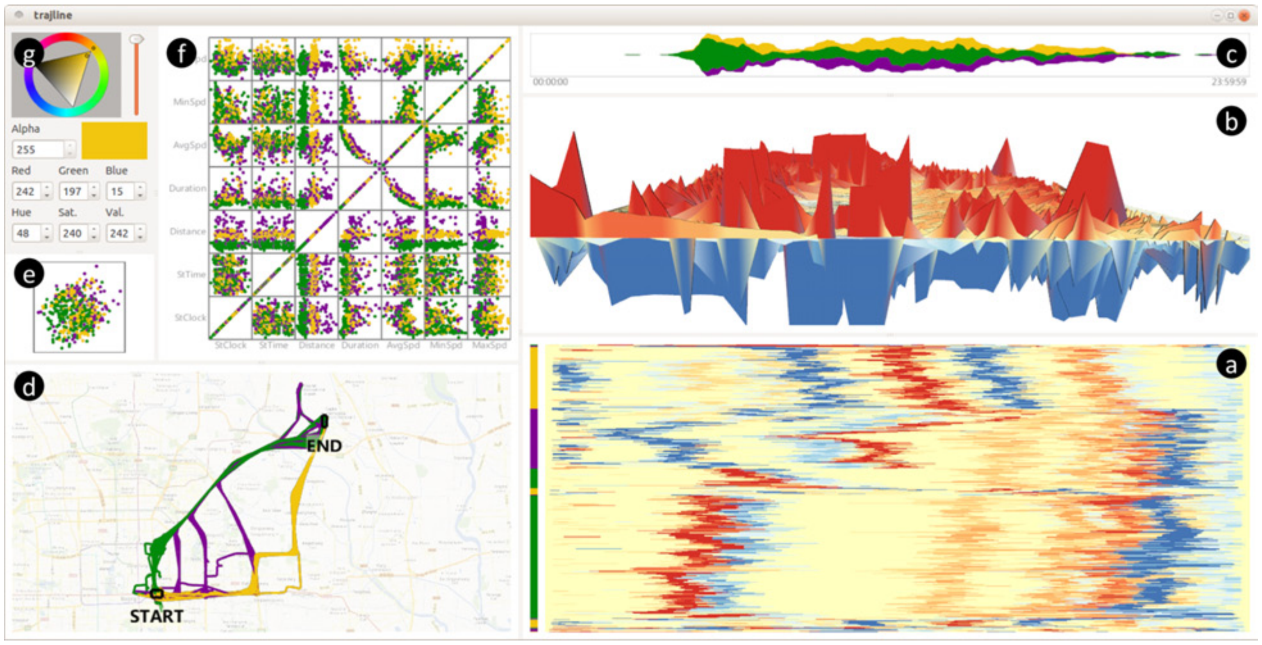

Alpha 255
Red 242 Green 197 Blue 15
Hue 48 Sat. 240 Val. 242

Fig. 1. Interface of our trajectory visualization system. (a) Timelines as 2D heatmap, visualizing trajectory curvature change over time; (b) Timelines as 3D terrain, visualizing trajectory curvature change over time; (c) ThemeRiver showing the temporal distribution of trajectories in a day; (d) Map view giving the spatial information of trajectories; (e) MDS projection view showing proximity of trajectories in attribute space; (f) Scatterplot matrix view showing trajectory attributes; (g) Color palette for group tagging. In this picture, 492 taxi trajectories are visualized. They are running from a railway station (labeled START) to an airport (labeled END), with passengers inside. Color on the right of the 2D heatmap timeline view (a) and color on of the 3D terrain view (b) encodes timeline value (curvature value here), red is for strong right turning, and blue is for strong left turning. On the left bar of the 2D heatmap view, and on other views (c-f), color encodes group belonging. Here 3 groups are tagged, assigned with color yellow, green and magenta.

B. *Temporal Visualization of Trajectories*

Most of the trajectory visualization focus on the spatial information, but there are still a few on the temporal information. Andrienko [2] et. al. have explored using timeline for trajectory visualization, but their result is rather simple, just basic ordering and alignment. Proximity visualization [8] looks similar to timeline. It projects the trajectory into an abstract space, defined by the distance to some critical spots. Especially, it can produce a time line representation, showing the distance change. However, it requires that the trajectories be in the same region, and some critical spots be predefined. Cluttering is serious when there are many trajectories.

Space time cube [11], [14], [4] draws the time as z-axis perpendicular to map. It precisely encodes the spatial and temporal information. However, similar to other 3D representation, it's hard for user to precisely decode the information. Besides, as the trajectories are just drawn as they were, visual cluttering becomes a serious problem, and comparison is difficult. Stacking [20] can show temporal ordering information on a spatial context. When trajectories are similar, visual cluttering is not that series. Some visualization use aggregation [3], [20], [10], [16] to give an overview of the temporal distribution of trajectories, or positions of trajectories at different time [19]. Our method shows individuals.

C. *Line Representation*

Line representation is easy for navigation and comparison, and is not cluttered. Therefore, many data are visualized as lines, even if their original forms are not lines. Most of them use deformation technique. Wu et. al. [23] adapt the metro map layout to make the user queried route straight. This makes navigation easier. Angelelli et.al. [5] straighten the flows in tubular structure for easy comparison. Potentially, trajectories can also be deformed to a line representation similar to timeline. One related work is route deformation [1]. Although its result is not a line, but this strategy can be used to indeed convert route into lines. Currently, our efforts are on timeline representation. We do not consider these deformation techniques.

III. Design

In this section, we first explain the design goal of trajectory timeline. After that, we show how we construct and interact with the timelines.

A. *Design Requirements and Decisions*

We expect our visualization to fulfill the following design requirements:

- (R0: Time) We hope to see temporal information of the trajectories. For the best we should have a time axis or time ordering axis.
- (R1: Navigable) We hope that a trajectory can be followed from start to end easily. This requires the general shape of each representation be simple and the representation be monotonous in some sense.
- (R2: Interpretable) We hope the representation is readable by human. Therefore we cannot calculate or train feature vectors as in machine learning and directly show it.

- (R3: Comparable) We hope that hundreds of trajectories can be compared. Therefore the representation must be simple and aligned.

Timeline naturally shows time with the x axis as time axis. Its linear form is easy to navigate and compare. The attribute value change is Interpretable as long as the attribute calculation is meaningful. Another promising method is deforming trajectories into line. In that case, the local shape is more meaningful than the position of the baseline. If we would like to have a time axis to make the time information clear, this will have conflict with the meaning of local shape. We think this is difficult, therefore we use timeline. One argument is that deformation may better preserve the spatial information than timeline conversion. However, to our observation, when a trajectory is strongly deformed to a line form, the spatial information are almost lost.

B. Timeline Construction

We describe five types of timeline attribute calculation. Each one will give a set of timelines. The former two types are naively showing some attribute, while the latter three are showing spatial features. Figure 2 shows different timelines for one trajectory. The line representations all have time as x axis, but with different y: velocity (a point attribute), segment average velocity (a segment attribute), curvature, straightness, significance of turn, significance of stop. Some critical points are labeled on the map manually, with their corresponding time one the left. From this figure we can see that only the stop-plot gives signal at stop. Curvature plot and straightness plot identify almost the same interesting time, but with different peak shape. Their peaks and turn plot's peaks correspond well with the spatial feature. We now describe them in detail.

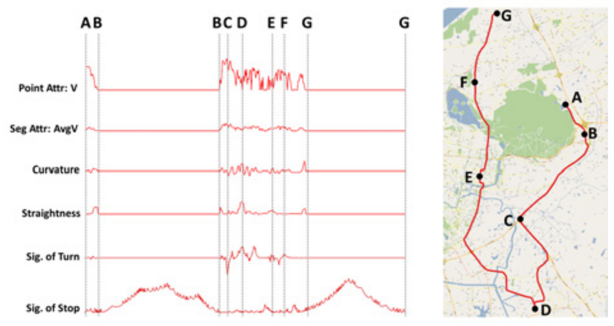


Fig. 2. Summary of different types of timeline construction.

1) Point/Segment Attribute Plot: We briefly call each trajectory sampling point as *point* and each part between two consecutive points *segment*. Both point and segment can have attributes, such as point speed or segment average speed. Given one point or segment attribute, we can show how it changes with the time. This gives a timeline. For point attribute, which is inherently associated with a time, this is direct. For segment attribute, we will assign it to the point starting the segment. For the last point, it will take the last segment attribute.

2) Curvature Plot and Straightness Plot: For each trajectory, we can show how its curvature change with time, and how its straightness change with time. This gives two timelines: curvature-timeline and straightness-timeline. The conversion is

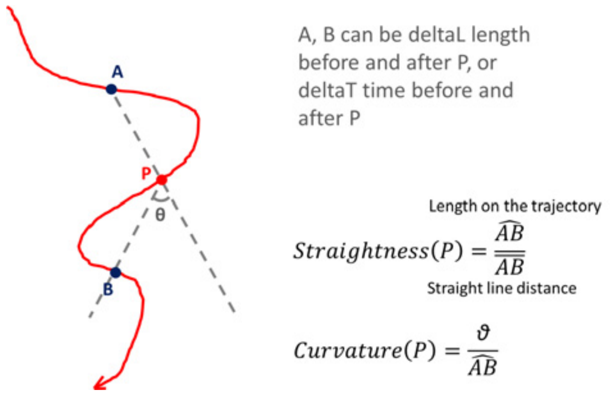

A
P
θ
B
A, B can be deltaL length before and after P, or deltaT time before and after P
Length on the trajectory
$Straightness(P) = \frac{\widehat{AB}}{\overline{AB}}$
Straight line distance
$Curvature(P) = \frac{\vartheta}{\widehat{AB}}$

Fig. 3. Definition of the curvature and straightness.

also point-wise. For each trajectory point P, we get a point A before it and a point B after it, and use A, P, B plus the trajectory between A and B to do the calculation. As figure 3 shows, the curvature is defined as the angle change over the distance along the trajectory. The curvature has a sign, positive for right turn and negative for left turn. The larger the absolute value of curvature, the shaper the turn. The straightness is defined as the distance along the trajectory over the straight line distance. It always larger than one. The large the straightness, the less straight it is. One thing to notice is that while the object is static for a long time, the calculation will be problematic. In this case, we will assign the point with a default value, 0 for curvature, 1 for straightness. Then it looks like that at this point there's no signal.

3) Turn Plot: For each trajectory, we show how its "turning" significance change over time. The term "turning significance" comes from the observation that, there's different scales of turns. Assume that we are studying a car. It can be a turning of the overall big direction, or a turning due to route selection, or a turn due to lane selection. We consider the former large scale turns more significant, while the later small scale turns less significant.

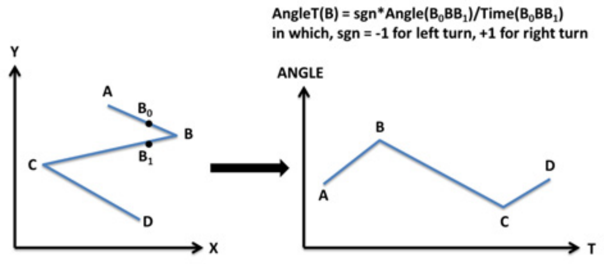

AngleT(B) = sgn*Angle(B_0BB_1)/Time(B_0BB_1)
in which, sgn = -1 for left turn, +1 for right turn
Y
A
B_0
B
B_1
C
D
X
ANGLE
B
A
C
D
T

Fig. 4. Definition of the angleT in turning significance calculation.

We have designed an algorithm to calculate this significance value. In order to capture multiscale turns, we try to get a multiscale representation of the trajectory geometry, and assume that turns on high level representations are more significant. We then calculate the turn value in a manner called angleT (Figure 4). Here the angle calculation is slightly different with the above curvature and straightness, but trying to show the same property. The division by time is according to the intuition that quick turns are more significant. After that, we get the representation for each scale, finally combine them back. Similar to curvature and straightness, if an object have not moved for a long time, it will not produce any signal remain a horizontal line.

Our algorithm includes six steps. In step 1 (Figure 5(a)) we split trajectory into trips: "stop"s and "move"s. These two kinds of subtrajectories will be treated differently: "stop"s will skip step 2 to 5 and be represented just as a straight horizontal line, "move"s will go through step 2 to 5. In step 2 (Figure 5(b)), for each "move", we make different scales of simplifications by an modified Douglas-Peucker algorithm [9], each level correspond to a different distance threshold. In step 3 (Figure 5(c)), for each trajectory, for each simplification scale, we convert trip into line representation by the angleT calculation (Figure 4). Fast turns with large angle will produce strong signal. In step 4 (Figure 5(d)), we interpolate between the control points with different strategy, the yellow rectangles highlight their difference. Direct interpolation on the first line between signal peaks can be problematic, because when no sampling point is recorded, it is less likely to have turns. Therefore it should produce very low signal. The lower two lines use parabolic curves and trigonometric curves to do the interpolation, which make the signal peaks more local. In step 5 (Figure 5(e)), for each trip, all representations from different scales of simplification are linearly combined, giving representations of large scales higher weight. In step 6 (Figure 5(f)), representation of all trips, whether "stop"s or "move"s are reconnected, forming the final representation.

Our angleT calculation has a sign, which result in positive and negative values on the timeline. This is intuitive, since left and right turns can be differentiated. However this can be problematic due to combination of different scales of representation at step 5. When positive peaks and negative valleys are combined, both of them disappear. This is very unfavorable. Therefore, we provide an option to turn off the sign of angleT calculation. However, this lead to left and right turns mixed together.

4) Stop Plot: From above we can see that stops are very important. In this type of timeline, we show how significance of stop change over time. By intuition, there are different scales of stops. Assume again we are studying cars. Considering temporal aspect, it can wait a red light for 2 minutes, can stuck in a traffic jam for half an hour, can park a car for half a day. Considering the spatial aspect, it can just stop somewhere, or continuously moving around a small region. Here we consider this issue. Our strategies are very similar to above when we calculate the significance of turn, but in this case we do not manually differentiate "move"s and "stop"s, we treat them together. This is partially because now significance of stop gives signal, so both "move" and "stop" give signal. Therefore we do not need step 1 and step 6. Another difference is in step 3. Now we do not calculate angleT for each point, instead we calculate the reciprocal of average speed for each segment, and the signal will be put at the middle time of the segment. Therefore we can see peaks on "stop" segments. All stop values are positive.

C. Timeline Interface

In order to show hundreds of timelines, we use a 2D heatmap view (Figure 1(a)) and a 3D terrain view (Figure 1(b)). The 2D view use color to encode the timeline

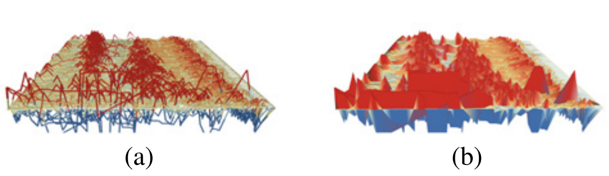


Fig. 6. Different styles of timeline rendering in 3D. (a) "line" like rendering; (b) "wall" like renedring.

attribute value. This is perhaps the only solution, since position encoding are used, and length encoding can not work in so small area. We use a red to blue color scale, with red encode positive value and blue encode negative value. The 3D view use height to encode the timeline attribute value, which is very natural. We have tested different styles of timeline rendering in 3D. Figure 6 shows two of them, the "wall" like rendering (b) gives better depth perception than the "line" like one (a), but results in more occlusion. Outliers are especially like to be occluded in the "wall" style. We use the "wall" like one by default. Occlusion can be reduced by rotating the 3D scene. In both views, the timelines can be sorted either by similarity, or by general measures such as total distance or total time. The sorting dramatically reveals patterns in the timelines, as shown in Figure 7.

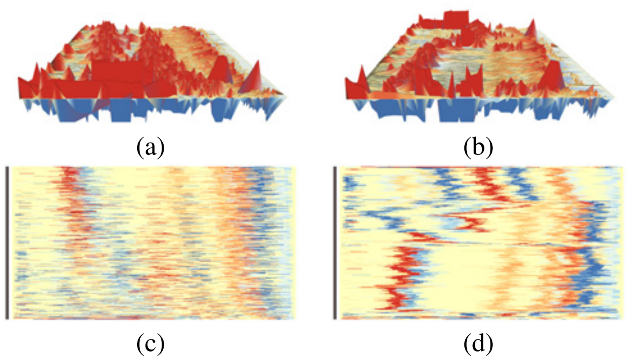
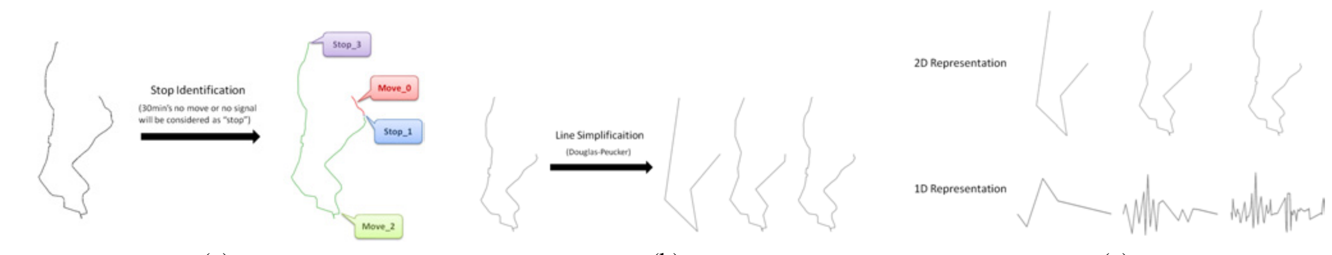


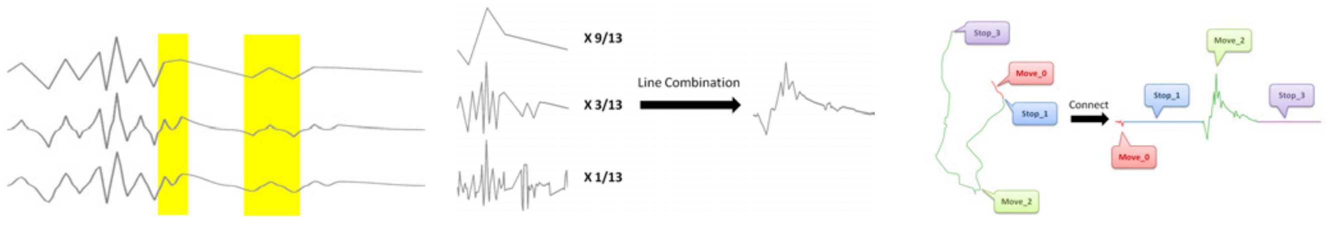
Fig. 7. Sorting by similarity greatly helps user to understand the general distribution of the timelines, both in 2D view and in 3D view. (a)(c) before sorting, (b)(d) after sorting.

IV. SYSTEM

The three aspect of trajectory visualization, time, space and attribute are strongly related. Therefore, in order to really use the timeline views, we build some other views for the spatial and attribute aspect, and link them together. As Figure 1 shows, our system has some other components in addition to the 2D heatmap (a) and 3D terrain (b) timeline views. In order to show temporal distribution of trajectories, we use a themeRiver [12] view (c). To show the exact spatial position of trajectories, we have a map (d). Further, we calculate some attribute for each trajectory and visualize the trajectories as high dimensional data. Notice here the attributes are on trajectory, while the attributes we are previously talking about are on point. They are different. Our high dimensional data views include an MDS projection view [21] (e) and a scatter plot matrix view [6] (f). A color palette (g) is provided. Each time user select a color from the palette, she can brush in other components, to tag interesting trajectories with the selected color. In our system, the color are implicitly used to show



(a) (b) (c)



(d) (e) (f)

Fig. 5. Algorithm to calculate the significance of turn.

grouping information. In order to comply with the color usage for grouping, our 2D heatmap timeline view has a bar on the left. Color on the bar is always the grouping color.

V. Results

In this section, we provide some results on real dataset. Our data consists of 492 taxis running from a railway station to an airport. It is extracted from a real taxi trajectory database. The data spans 24 days. The attributes associated with each sampling points are speed, direction, passenger state (whether there is a passenger inside the taxi). Our extraction only select those trajectories with passengers inside, so that trajectories have a clear destination. In this case, they will usually move to destination as fast a they can, and they will not prefer to drive randomly. We would like to compare them in terms of stop, speed and straightness.

A. Stops

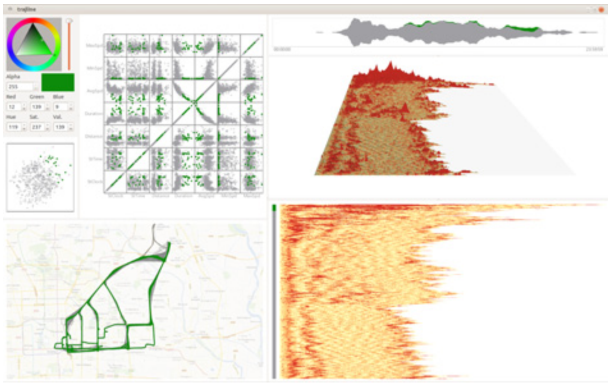
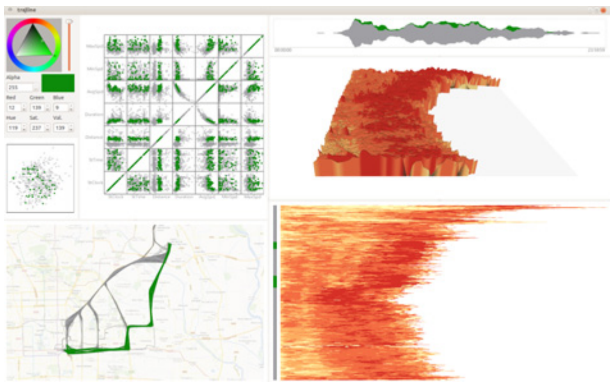


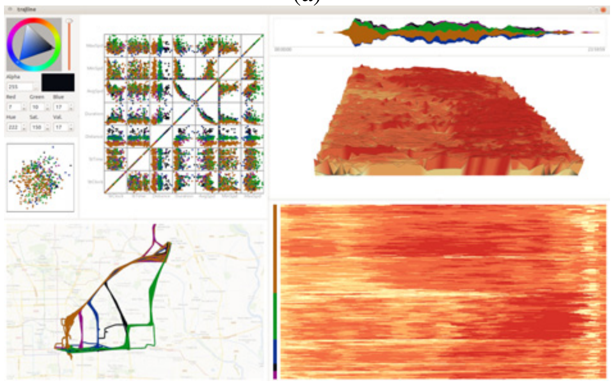
Fig. 8. Stops indeed result in long travel time, most of the stops happened between 17:00 to 19:00.

Stop is one major reason for slow taxi movement. It usually indicates traffic jams. In this case, we generate a stop-plot to show change of the significance of stops. The x axis are time. Timelines are sorted by similarity. Red color indicate significant stops, while yellow color represent smooth travel. We can see from Figure 8 that red color usually accompanies long timeline length (long travel time). We select the reddest ones on top with green color (see the color bar on the left of the 2D heatmap view), while the other unselected remain gray. We can see from the themeRiver that their temporal distribution is rather close, mostly between 17:00 to 19:00. This indicates that the traffic condition at this time period is bad.

B. Speed



(a)



(b)

Fig. 9. Overall the speed is faster at the later half of the time. Trajectories of different routes have different speed pattern.

Speed is another important factor for taxi movement, faster speed is usually preferred. In this case, we generate a point attribute plot to show the change of speed. The x axis is time (in Figure 9(a)) or normalized time (in Figure 9(b)). Timelines are sorted by similarity. Red color indicate high speed (preferred), while yellow color represent low speed. We can see from Figure 9 that red color usually comes at the later half of the travel time, this is due to the fact that the second part of the journey in on the airport express. We notice that some trajectories are fast at the end, and we select them with green color (Figure 9(a), see the color bar on the left of the 2D heatmap view). The other unselected remain gray. We can see that all these trajectories share the same route. This encourage us to assign one group (simultaneously assigning one color) to each route, then see if there are any difference between different routes. The result is that there are apparent difference (Figure 9(b)). The green trajectories indeed tend to have high speed at the end, while blue trajectories have high speed at middle. The dark yellow trajectories seem to consist different patterns.

C. Straightness

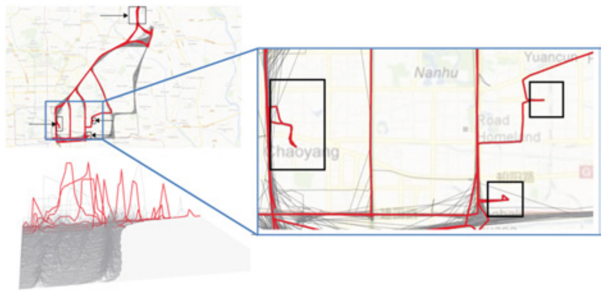


Fig. 10. Straightness plot (bottomLeft) can be used to detect strange peaks, which possibly indicates data error (topLeft, right).

Data error usually results in trajectories with strange shape, such as jumps or peaks. This will produce a very high straightness value, and therefore easy to be detected with straightness plot (Figure 10). The x axis is time. Timelines are sorted by total travel time. To view the outliers better, we use ”line” like rendering in the 3D view. The strange peaks are apparent.

VI. Conclusion and Future Work

In this paper, we have studied using timelines to make temporal analysis of trajectories. We have proposed a few new forms of trajectory timeline. We have built up a trajectory visualization system, and made some case studies on real dataset. The result shows that our method is promising. In the future, we would like to explore more forms of trajectory timeline, especially by deformation strategy. We would also allow more interactions on the timeline, such as selecting spatial positions from the timeline by brushing.

Acknowledgment

The authors thank Datatang for providing the data, and the anonymous reviewers for their valuable comments and suggestions. This work is supported by NSFC Project (No. 61170204).

References

[1] M. Agrawala and C. Stolte. Rendering effective route maps: improving usability through generalization. In *Proceedings of ACM SIGGRAPH 2001*, pages 241–249, 2001.

[2] G. Andrienko and N. Andrienko. Exploration of massive movement data: a visual analytics approach. In *AGILE'08*, 2008.

[3] G. Andrienko and N. Andrienko. Spatio-temporal aggregation for visual analysis of movements. In *Proceedings of IEEE VAST 2008*, pages 51–58, 2008.

[4] G. Andrienko and N. Andrienko. Poster: Dynamic time transformation for interpreting clusters of trajectories with space-time cube. In *Proceedings of IEEE VAST 2010*, pages 213 –214, 2010.

[5] P. Angelelli and H. Hauser. Straightening tubular flow for side-by-side visualization. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2063 –2070, 2011.

[6] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29:127–142, 1987.

[7] I. Boyandin, E. Bertini, P. Bak, and D. Lalanne. Flowstrates: An approach for visual exploration of temporal origin-destination data. *Comput. Graph. Forum*, 30(3):971–980, 2011.

[8] T. Crnovrsanin, C. Muelder, C. Correa, and K.-L. Ma. Proximity-based visualization of movement trace data. In *Proceedings of IEEE VAST 2009*, pages 11–18, 2009.

[9] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10:112–122, 1973.

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

[11] T. Hägerstrand. What about people in regional science? *Papers in Regional Science*, 24:6–21, 1970.

[12] S. Havre, E. G. Hetzler, P. Whitney, and L. T. Nowell. Themeriver: Visualizing thematic changes in large document collections. *IEEE Trans. Vis. Comput. Graph.*, 8(1):9–20, 2002.

[13] H. Hochheiser and B. Shneiderman. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.

[14] T. Kapler and W. Wright. Geotime information visualization. In *Proceedings of the IEEE InfoVis 2004*, pages 25–32, 2004.

[15] R. Kosara, F. Bendix, and H. Hauser. Timehistograms for large, time-dependent data. In *VisSym*, pages 45–54, 340, 2004.

[16] H. Liu, Y. Gao, L. Lu, S. Liu, H. Qu, and L. M. Ni. Visual analysis of route diversity. In *IEEE Conference on Visual Analytics Science and Technology (IEEE VAST'11)*, 2011.

[17] P. McLachlan, T. Munzner, E. Koutsofios, and S. North. Liverac: interactive visual exploration of system management time-series data. In *Proceeding of the ACM CHI 2008*, pages 1483–1492, 2008.

[18] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. Lifelines: visualizing personal histories. In *Proceedings of ACM Conference on Human Factors in Computing Systems*, pages 221–227, 1996.

[19] R. Scheepens, N. Willems, H. van de Wetering, and J. van Wijk. Interactive visualization of multivariate trajectory data with density maps. In *Proceedings of IEEE PacificVis 2011*, pages 147 –154, 2011.

[20] C. Tominski, H. Schumann, G. Andrienko, and N. Andrienko. Stacking-based visualization of trajectory attribute data. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2565–2574, 2012.

[21] P. C. Wong and R. D. Bergeron. Multivariate visualization using metric scaling. In *Proceedings of the IEEE Vis'97*, pages 111–118, 1997.

[22] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. Lifeflow: visualizing an overview of event sequences. In *Proceedings of ACM CHI 2011*, pages 1747–1756, 2011.

[23] H.-Y. Wu, S. Takahashi, C.-C. Lin, and H.-C. Yen. Travel-route-centered metro map layout and annotation. *Comput. Graph. Forum*, 31(3):925–934, 2012.