Visual Traffic Jam Analysis Based on Trajectory Data

Zuchao Wang, Min Lu, Xiaoru Yuan, *Member, IEEE*, Junping Zhang, *Member, IEEE*, and Huub van de Wetering

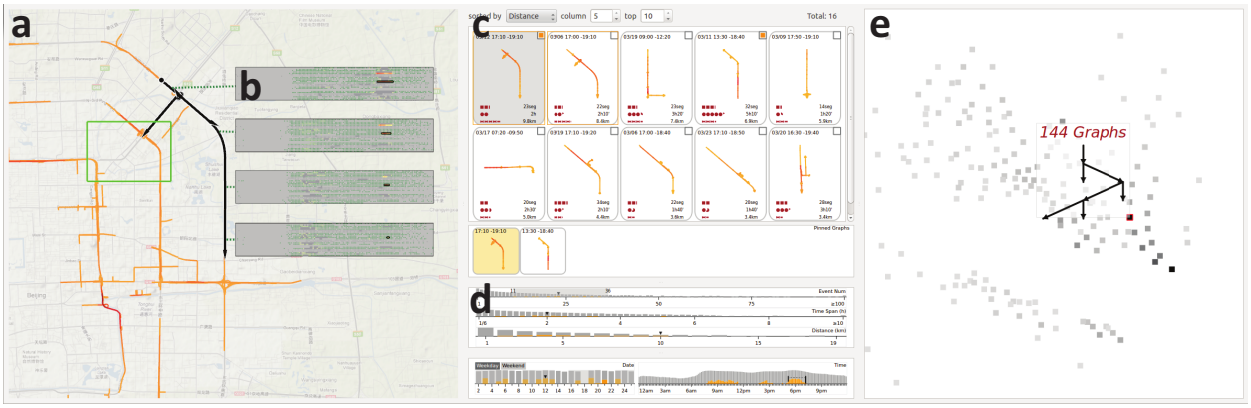

Fig. 1. An overview of our system. (a) The spatial view shows the traffic jam density on each road of Beijing by color, and one traffic jam propagation graph is highlighted in black. (b) The embedded road speed views show the speed patterns of four roads in the highlighted black propagation graph. (c) The graph list view shows a list of sorted traffic jam propagation graphs. (d) The multi-faceted filter view allows filtering of propagation graphs by time and size. (e) The graph projection view shows the topological relationship of graph clusters, where graphs in the same cluster have very similar topology.

Abstract—In this work, we present an interactive system for visual analysis of urban traffic congestion based on GPS trajectories. For these trajectories we develop strategies to extract and derive traffic jam information. After cleaning the trajectories, they are matched to a road network. Subsequently, traffic speed on each road segment is computed and traffic jam events are automatically detected. Spatially and temporally related events are concatenated in, so-called, traffic jam propagation graphs. These graphs form a high-level description of a traffic jam and its propagation in time and space. Our system provides multiple views for visually exploring and analyzing the traffic condition of a large city as a whole, on the level of propagation graphs, and on road segment level. Case studies with 24 days of taxi GPS trajectories collected in Beijing demonstrate the effectiveness of our system.

Index Terms—Traffic visualization, traffic jam propagation

1 Introduction

Traffic jams form a serious problem in modern cities. They bring about considerable economic loss, increase travel times and aggravate pollution. Governments spend a great amount of money trying to monitor and understand traffic jams, but this seems difficult due to the complex nature of traffic jams. One of the complexities is unpredictability. Sometimes traffic jams occur, sometimes not. Another complexity is that traffic jams are dynamic and interrelated. Traffic jams can, for instance, propagate from one road on to other roads. Due to these complexities, a fully automatic analysis of traffic jams is hard, requiring considerable experience and knowledge. In this work, we present a visual analysis system to study the patterns of traffic jams and their propagation. Our system combines automatic computation and human knowledge. We first extract traffic jams from GPS trajectories, from which we construct propagation graphs. Then we design a visual interface to explore both traffic jam patterns and the propagation of traffic jams. As far as we know, there is no previous work in visual analytics that deeply studies these traffic jam aspects.

Traditional traffic jam detection methods are based on road side sensors, like induction loops or radar [31] and monitor only a few critical points [25]. A GPS based method, however, can theoretically monitor a complete road network. This enables us to better study traffic jam propagation. Furthermore, the installation of expensive road side devices is not required. Previous GPS based traffic jam detection methods either just study separate jams [10, 34], therefore giving scattered traffic information of the road network, or being unable to attribute traffic jams to specific roads [29, 15]. Traffic jam data from these works are not suitable for visual exploration. In this work, we derive a road bound traffic jam dataset from GPS trajectory data, and structure the detected traffic jams by building propagation graphs. Our data is more suitable for visual exploration.

In the visual interface, as shown in Figure 1, we allow users to make multilevel exploration, from traffic patterns on a single road to the traffic jam condition in a whole city. We support various filtering techniques to query specific kinds of propagation graphs, and we allow users to compare them.

Our major contributions are:

- We present a process to automatically extract traffic jams from

- *Zuchao Wang is with Key Laboratory of Machine Perception (Ministry of Education), and School of EECS, Peking University. E-mail: zuchao.wang@pku.edu.cn.*
- *Min Lu is with Key Laboratory of Machine Perception (Ministry of Education), School of EECS, and Center for Computational Science and Engineering, Peking University. E-mail: lumin.vis@gmail.com.*
- *Xiaoru Yuan is with Key Laboratory of Machine Perception (Ministry of Education), School of EECS, and Center for Computational Science and Engineering, Peking University. E-mail: xiaoru.yuan@pku.edu.cn.*
- *Junping Zhang is with Shanghai Key Laboratory of Intelligent Information Processing, and School of Computer Science, Fudan University. E-mail: jpzhang@fudan.edu.cn.*
- *Huub van de Wetering is with Department of Mathematics and Computer Science, Technische Universiteit Eindhoven. E-mail: h.v.d.wetering@tue.nl.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

1077-2626/13/$31.00 © 2013 IEEE Published by the IEEE Computer Society

noisy GPS trajectory data. Our data is structured and road bound, therefore suitable for visual exploration.

- We design a visual interface to explore the traffic jams and their propagation. The exploration is multilevel, and supports filtering and comparison of traffic jam propagation graphs.

2 Related Work

Our related work section is split into an analysis subsection on traffic event detection and two visual subsections on traffic visualization and propagation graph visualization.

2.1 Traffic Event Detection

Perhaps the most commercialized technique to detect traffic events is by radar like sensors [31]. Analyzing video streams from a roadside camera also helps to understand the traffic [53, 22]. However, both techniques require the installation of high-cost devices, and can only monitor at fixed positions along the road. In contrast, the GPS technique is cheaper and able to monitor the whole road network. Therefore, much of recent research focuses on analyzing GPS trajectories.

The data mining community has long been working on trajectory data. They have studied different kinds of patterns [23, 26]. See Zheng et al.'s book [54] for an overview.

We are most interested in traffic jam detection. Traffic jams are important traffic events. They are usually characterized by long travel time, or, equivalently, low speed. Many traffic jam detection algorithms are based on road speed calculation [18], or low speed vehicle cluster detection [10, 34]. Bauze et al.'s traffic monitor system also detects traffic jams by low speed [13]. We use a road speed calculation based method to detect traffic jams. It provides traffic situation data, not only in a few congested places and during a few periods, but it does so in much larger regions and periods. Such data is more suitable for free exploration. Our work is different from the works cited above, in that we focus on the propagation of traffic jams.

Krogh et al. [25] have studied green waves on road stretches with signalized intersections. However, this differs from traffic jam propagation, and their scenario is much simpler than our city network. Recently, Zheng et al. published a paper [29] studying the causal interactions of traffic outliers. They first segment a city into medium sized regions, then study the traffic flows on the links between the regions. Outliers are detected and arranged as outlier trees. Our propagation graph construction uses the same idea, but our focus is on traffic jam events, not on outliers. A more important difference is that, traffic jams originally happen on roads, not on links between regions. So when users detect an anomalous link, it is difficult to explore and explain their results. Although in later work [15] they correlate the anomalous links to anomalous routes, it is still unclear where anomalies occur on long routes. In our results, we can directly see traffic jams propagating on roads, as they actually are. Our model is more suitable for visual analysis.

The traffic condition in road networks can also be modelled by Probabilistic Graph Models (PGM) [27, 37]. This technique is able to learn the temporal change of traffic conditions for each road, and the spatial dependency between roads from historical data. Therefore, it can simulate the macro traffic, and can make predictions of future traffic conditions. However, here we mainly want to summarize the historical data, and support user explorations. Our traffic jam detection and propagation graph construction algorithm already summarize the historical traffic and their results are easy to understand and explore. In contrast, a PGM, although being more generic, has parameters that are harder to tune, and is harder to understand, explore and evaluate.

2.2 Traffic Visualization

A major type of traffic data is trajectory data. In this case, all trajectory visualization techniques can be used. An overview of all trajectories is the first step in their visual analysis. It often requires aggregation. A density map [51] provides an overview by visual aggregation. It plots the trajectory density and helps identify "hot" spots. Density maps may also show the density of multi-variant trajectories [42] and extracted events [41]. Different from density maps, techniques such as spatial aggregation [12] and spatial-temporal aggregation [6, 43] provide overview by data aggregation. They discretize the spatial and temporal dimension into many regions, flows, or bins. Statistics are performed on each discrete spatial-temporal unit, e.g. a region in a time bin. This aggregated information is then visualized.

Micro-behavior analysis is another common task. In this case, trajectories have to be treated individually. Hurter et al. [21] show how to select trajectories with specific position and attributes. Guo et al.[19] present a system to analyze the traffic at a road intersection. Liu et al. [28] present a system to study the route diversity in a city.

Temporal information is critical in trajectory visualization. Space time cube [20, 24, 7] uses z-axis to represent the time, but suffers from visual clutter. A trajectory may also be represented as a timeline [9, 16], but the spatial information is then largely lost. Events can be extracted from these time series [11].

For trajectory attributes, Tominski et al. [46] and von Landesberger et al. [49] have addressed their visual analysis problem.

Some of the above cited works study the events of trajectories. However, none of them focuses on the interactions of these events, and none of them focuses on traffic jams. Our work aims to deeply study traffic jams and their interactions.

Although most of the traffic visualizations use trajectory data, some use other types of sensor data. Pack et al. [35] study traffic incidents data. They design a linked view interface to visualize the spatial, temporal and multi-dimensional aspects of the incidents. Users are allowed to select, filter and cluster these incidents. Piringer et al. [38] study the surveillance videos in a tunnel. They automatically detect and prioritize different types of events and mark them in space and time. For each event, users can check the original videos. Both focus on traffic events, but none of them on the interactions of these events. Our work studies these interactions, and we use trajectory data, which requires different event detection algorithm.

2.3 Propagation Graph Visualization

Propagation graphs may be visualized by animations and small multiples. However, these techniques have limitations [39]. Therefore, people also designed other visual metaphors. The spatial, temporal and topological aspects of the propagation graph can be visualized by separate techniques, like FlowMap [48], Massive Sequence View [47] and graph layout [44]. It remains challenging to visualize all aspects in one view. In our work, we have applied the animation, flow map and graph layout techniques.

3 Overview

In this section, we first present the design requirements. After that we describe the input data, and define the traffic jam data model. Finally we present the system workflow.

3.1 Design Requirement

To study traffic jams, we need a data model, according to which we extract and structure the traffic jam data. It should satisfy the following three requirements:

R1: Complete We require that basic traffic jam information is available, including location and time. Besides, speed information should always be there, even when there is no traffic jam. It helps users to understand how traffic condition changes, and to check whether the traffic jam detection is appropriate.

R2: Structured We require that the traffic jams in the model are interrelated: we are not only interested in individual traffic jams at separate locations and time, but also how these traffic jams are related, and how they propagate from one location to another.

R3: Road bound We require the traffic jams to be defined on roads, and to propagate along the road network, as they are actually happening. This help users to associate the traffic jam data with their real world knowledge during visual exploration.

A visual interface to explore and analyze the data model, should satisfy the following requirements.

R4: Informative We require that the system shows all critical information of the traffic jams, including location, time, propagation path, size of the propagation, and the road speed.

R5: Multi-level We require that the traffic jams can be explored at multiple levels. The lowest level should be the congestion behavior on a single road segment. Above that we require to analyze the traffic jam propagation among different road segments, and to compare different propagations. On the highest level, we require to study the congestion status of the whole city.

R6: Filterable We require to filter traffic jams according to spatial, temporal properties, and size of propagation. In this way, we can focus on specific types of traffic jams, and make deeper analysis of them.

3.2 Description of Input Data

We use GPS trajectory data and road network data as input, to calculate and analyze traffic jams. GPS trajectory data contains many *trajectories*. Each trajectory consists of a list of *sampling points*. Each *sampling point* has a position record ($\langle longitude, latitude \rangle$ for 2D data), time stamp *time*, speed magnitude *velocity*, moving direction *vangle*, and optionally a set of attributes $\langle a_0, a_1, ...a_{n-1} \rangle$. These *sampling points* are sorted in time ascending order. Each part between two consecutive *sampling points* is called a *trajectory segment*.

A *road network* consists of *nodes* and *ways*. Each node has a spatial position. It can be either an intersection or a shape point. Each way contains an ordered list of nodes that defines the spatial position and shape of the way. A way can be a one-way street or a two-way street.

Our GPS dataset is a real taxi dataset recorded in the city of Beijing, which is prone to traffic jams. The dataset contains the GPS trajectories of 28,519 taxis. Estimated from a government report [5], they include 43% of all licensed taxis in Beijing, and account for 7% of the traffic flow volume within Beijing's 4th Ring. The dataset spans 24 days, from March 2nd to 25th, 2009. It contains 379,107,927 sampling points, and the data size is 34.5GB. The only attribute is the boolean *passengerState*, indicating whether there are passengers in the taxi. The sampling rate is one point per 30 seconds. However, 60% of the sampling points are missing, so, two consecutive points frequently have a time difference of over 3 minutes.

Our road network dataset comes from a query from OpenStreetMap's jXAPI [17]. We extract all roads in the spatial range from 116.109E to 116.673E and from 39.743N to 40.119N. This gives 40.9MB of data, containing 169,171 nodes, and 35,422 ways.

3.3 Traffic Jam Data Model

Our model structures three types of information: the road speed, the traffic jams, and the relationships between traffic jams. In our model, the time is discretized into *time bins*. The two directions on a way are treated separately, each as a directed way (abbrev. as *dWay*). A dWay and a time bin are the smallest spatial and temporal unit.

The road speed information gives a basic description of the road condition. For each dWay, at each time bin, there will be a *speed* record. The speed value can be empty if it can not be estimated.

The traffic jam information summarizes all the detected traffic jams. It consists of a list of traffic jam events (abbrev. as *events*). An event is defined as a triple $\langle d, t_0, t_1 \rangle$, where the dWay d is the location of the event and the integers t_0 and t_1 with $t_0 \leq t_1$ are the start and end time bin of the event, respectively. So, the whole event takes place in the interval $[t_0..t_1]$ that spans $t_1 - t_0 + 1$ time bins.

The relationships between traffic jams are characterized by traffic jam propagation graphs (abbrev. as *graphs*). A graph is a directed network of events, defined as $\langle V, E \rangle$, where V is a set of events, and E is a set of directed links between events. It is both acyclic and connected. A directed link is notated as $e_1 \rightarrow e_2$, meaning that event e_1 leads to e_2, or equivalently, e_2 is caused by e_1. An event can be caused by 0 or more events and can also lead to 0 or more events. For each graph, a spatial propagation path (abbrev. as *path*) can be derived, which is a directed network of dWays. It can have cycles. A link $d_1 \rightarrow d_2$ in a path means that the corresponding traffic jam propagates from dWay d_1 to d_2.

3.4 Work Flow

Our visual analysis work consists of two phases. The first phase is preprocessing, in which we start from the input data, and extract traffic jam data that fits our model. The second phase is visual exploration, in which we explore the preprocessed data. Figure 2 gives an overview of our system. We will explain the preprocessing phase in Section 4, and the visual exploration phase in Section 5.

4 Preprocessing

Our preprocessing phase consists of six steps. The first two steps improve the quality of the input data. In step 1 *Road Network Processing*, we improve the road network quality by filtering out irrelevant data, merging and splitting ways, and correcting errors. In step 2 *GPS Data Cleaning*, the trajectories are cleaned. One obvious thing is to remove GPS errors. To accurately estimate road speed later on, we also filter out stops that do not reflect traffic conditions, such as parking.

To estimate road speed we perform another two steps. In step 3 *Map Matching*, we match GPS trajectories to the road network to correlate the trajectory speed with the road speed. After this step, each trajectory sampling point is mapped to one position on a *dWay* (not a lane), and each trajectory segment is mapped to a path on the road network. In step 4 *Road Speed Calculation*, we estimate the speed of a *dWay* at a time bin, based on the speed of the trajectories that map to it. This can be performed by averaging the trajectory speed. This estimation can be inaccurate due to, for instance, insufficient number of mapped trajectories, and incomplete filtering of parking cases in step 2.

In the last two steps, propagation graphs are constructed on traffic jam events. In step 5 *Traffic Jam Detection*, traffic jam events are detected based on speed. For each dWay, an abnormally low speed for consecutive time bins, is considered as a traffic jam event. Finally, in step 6 *Propagation Graph Construction*, we predict the causal relationships between the detected traffic jam events, based on their spatial temporal relationship.

In the rest of this section, we discuss the preprocessing steps in more detail. Further details on their parameter setting are in the appendices.

4.1 Road Network Processing

The road network data downloaded from OpenStreetMap not only contains highways, but also waterways, buildings, etc. Therefore, we first extract all drivable ways from the data. Then we filter out the tiny road pieces that are not connected to the major network, and ensure all roads connected together. After that, we hope that the heading relation between two dWay is clear and unidirectional. Therefore, we reconstruct the ways in the road network data, such that two ways can only intersect at their end points. In the reconstruction, we require that the length of each way is less than 1km, which ensures the spatial resolution.

4.2 GPS Data Cleaning

For the GPS data, we remove five kinds of records: the irrelevant data, the erroneous data, the low sampling data, the non-jam stop data and the tiny trajectory data. We use a set of filters to achieve this.

Data out of the spatial range of the road network is irrelevant and removed by filter F1. Problems in erroneous data with respect to time or position are removed by filters F2 and F3. They typically manifest as two records with identical time stamps or segments with high speed.

F1: Unrealistic Coordinates We remove sampling points outside the range [116.109E, 116.673E] x [39.743, 40.119N].

F2: Duplicated Time Stamp If in a trajectory there are points with the same time stamp, we only keep the first occurrence and remove the other points with the same time stamp.

F3: High Speed We consider a trajectory segment speed higher than 90km/h unrealistic. In such cases we remove the trajectory segment and split the trajectory into two parts.

A low sampling rate results in trajectories with long segments or long time intervals. Our speed calculation is based on trajectory segments, so we require realistic speed change between the start and end

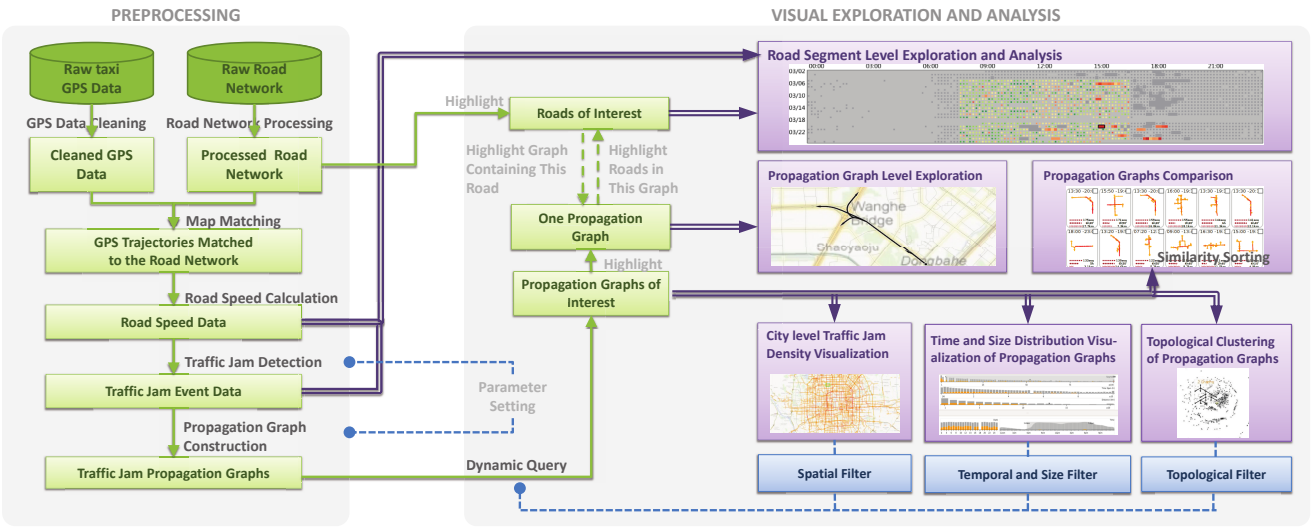

Fig. 2. The work flow of our system. In the preprocessing step, we extract traffic jam data from GPS trajectories and a road network. In the visual exploration step, we analyze the extracted traffic jams and their propagation.

points of a segment for interpolating accurately. This is not possible in such cases. Therefore, filter F4 and F5 remove them.

F4: Long Distance We remove segments with length over 2km.

F5: Long Time We remove segments with time interval over 10min.

Non-jam stop data are due to parking, passengers getting in or out of the car, and stopping to wait for passengers. This does not include waiting for green lights, because long time waiting for green light implies congestion. Filter F6 removes the first parking case and F7 removes the passenger cases.

F6: Parking We assume taxis staying within a 50m radius during 30min are actually parking, and thus remove these points.

F7: Waiting for Passenger We remove segments where the passengerState attribute changes. This splits trajectories into ones with constant passengerState. Then we remove stops at the beginning and end of the shorter trajectories, assuming that taxi drivers usually wait for new passengers immediately after dropping old ones, or that they wait until they have a new one. For stops at the beginning or at the end, we assume either a few points with identical positions, or a point with velocity equal to zero.

F6 is implemented using a stop detection algorithm [36]. We do not plan to identify interesting spots as in the original paper, but parking stops, including the cases when GPS position seriously oscillates (Figure 3(Right)). Therefore, we just use the Euclidean distance in their algorithm, not the distance along the path.

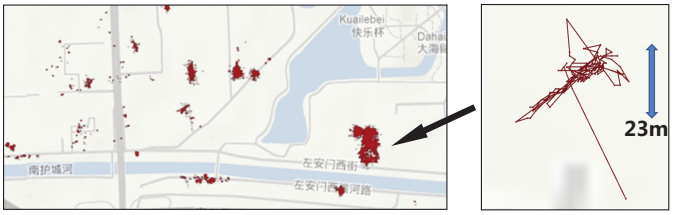

Fig. 3. (left) Stops removed by F6, with each sampling point represented by a red dot. (right) One stop with the sampling points connected by red lines. It spans 97min, and seems to oscillate due to GPS drift.

Tiny trajectories are mostly small fragments generated by filters. By rendering them on the screen, we find that they can hardly be used. We remove them by filter F8.

F8: Tiny Trajectory We remove all trajectories with at most 5 sampling points or less than 500m long.

The filters are applied in the order: F1, F2, F3, F4, F5, F6, F7, where filter F8 is applied directly after each filter to remove tiny trajectories.

4.3 Map Matching

We adopt the ST-matching algorithm [30] for map matching, since it is suitable for data with low sampling rate. However, the algorithm can not be directly used in our work, and we adapt it at three points. First of all, as most of the ways in our road network data do not have speed limit records, T-matching is impossible. Therefore, we only do S-matching. Analysis in the original paper [30] shows that the accuracy then drops by 2%. We consider that acceptable. Secondly, as our road network data has a few errors, such as wrong road directions and missing roads, we allow trajectory sampling points and trajectory segments to be unmatched. Otherwise, there would be many errors, as shown in Figure 4. We assume a missing match is better than a wrong match, in terms of accurately estimating the road speed. Sampling points without candidate match points are considered unmatched. Trajectory segments with transmission probability V less than a threshold Δ are considered unmatched. Finally, we would match each trajectory sampling point to one position on one dWay, therefore we need to know the driving direction of the taxi at each sampling point. This is achieved in a post processing step by simply looking at the matched position of neighbouring sampling points.

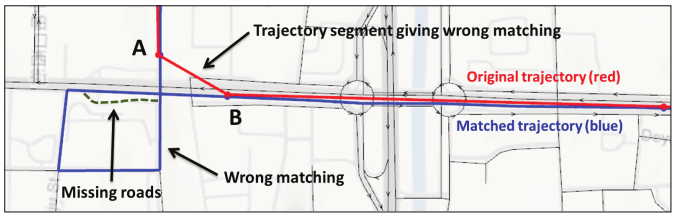

Fig. 4. The map matching produces many errors, if we do not allow unmatch. One example is the red trajectory segment from sampling point A to B matching to a long blue path. This is due to missing roads.

4.4 Road Speed Calculation

After mappping the trajectories to the road network, we can use trajectory speed to calculate road speed. In this step, we only use the matched parts of the trajectories. We choose a time bin size of 10min. For each dWay and for each time bin, we extract all taxi trajectories that pass the dWay within this time bin. We reconstruct the movement of the taxis assuming they follow the map matching result, and move at constant speed between two consecutive sampling points. Therefore, we can calculate an average travel speed for each taxi. After removing the taxis with exceptionally high speed (detected by an outlier detection algorithm [1]), we make an average of the average speeds on the remaining taxis and get the road speed. The speed averaging is per

trajectory, not per sampling point. We also record *support*, which is the number of remaining taxis. The higher the *support*, the higher the accuracy of the road speed calculation. We define that a speed estimation is *valid* when $support \geq min\_support$. The default value is $min\_support = 5$.

4.5 Traffic Jam Detection

After calculating the road speed, we do a traffic jam event detection on each dWay. Our idea is to use a speed threshold per dWay based on an estimation of the free-flow speed of the dWay. A speed limit may be a good estimation. Unfortunately we do not have it in our data. Krogh et al. [25] estimate free-flow speed from non-peak hour speed records. However, in Beijing different dWays may have different non-peak hours. Instead, we sort all valid speeds for a dWay in ascending order, and pick the speed value at the percentage $F\%$ position. Then each time bin on this dWay, with a valid speed less than percentage $C\%$ of the free flow speed, is said to have a low speed. The default parameter values, $F = 85$ and $C = 45$, give us 400,985 events.

4.6 Propagation Graph Construction

Now we have extracted events for all dWays, we build the propagation graphs by defining directed links among events. We use a rule based method. We assume a directed link $e_1 \rightarrow e_2$ exists if and only if $e_1.t_0 \leq e_2.t_0 \leq e_1.t_1$, and $e_1.d$ is immediately ahead of $e_2.d$. The former statement is a temporal constraint, saying that when e_2 starts, e_1 is still happening. The latter statement is a spatial constraint, saying that the two events are spatially connected, and the traffic jams propagate backward. The backward propagation is our assumption, which means the traffic jam will propagate in a reverse direction to the direction of traffic flow. Although it is not firmly validated, many observations and experiments [14, 45] support this. We have this constraint because our temporal resolution is not high enough. When we observe two adjacent roads congest at the same time bin, it is not clear from the data which leads to which. In our road network, it is usually the case that one dWay is ahead of another. One exception is for the two directions on the same two-way street. We do not make any link between them, because such propagation is associated with a u-turn traffic flow. By experience, such u-turn traffic flow is usually not dominant in the total traffic flow volume, and not likely to propagate traffic jams. Besides, our test shows adding such links it will add considerable noise in the constructed graphs.

We construct the graphs with a modified version of the STOTree algorithm [29] and end up with 226,227 graphs of which 162,429 contain only one event. We calculate the spatial propagation path and three size measures for each graph: number of events, time span, and total distance. The latter is the sum of the length in kilometers of all traffic jam events in the graph.

5 Visualization Design

According to the design requirements in Section 3.1, we provide our system with five views (in four windows). We design a pixel-based road speed view (embedded in Figure 1(b)) to show the speeds and events of one *dWay*. We design a graph list view (Figure 1(c)) to show the propagation graphs, and the graph projection view (Figure 1(e)) to show their topological relationships. We design a spatial view (Figure 1(a)) to show the traffic jam density on each *dWay*, and the propagation path of one highlighted graph. We also design a multi-faceted filter view (Figure 1(d)), to filter the propagation graphs.

5.1 Pixel Based Road Speed View

In our system, the road speeds and traffic jam events carry the low level traffic jam information. In designing a visualization for them, we have two concerns. Firstly, we need a compact visualization to be able to present multiple roads side by side for comparison. Secondly, according to our experience, road speed variation has strong daily and weekly patterns. It is important to present them in the analysis.

With these concerns in mind, we design a table-like pixel based visualization for a dWay, as illustrated in Figure 5(c). Each row represents a day, each column represents a 10 minutes time interval, and each cell represents a time bin. Optionally, the table can be divided into weekly blocks, by the black horizontal lines. We use cell color to represent the road speed on a dWay at the corresponding time bin. The color scale is given in Figure 5(b): red represents low, and green high speed. For cells without a valid speed estimation, we use gray. Mouse hovering over a cell reveals detailed speed information, including the time of the cell, the speed value and its *support* between brackets.

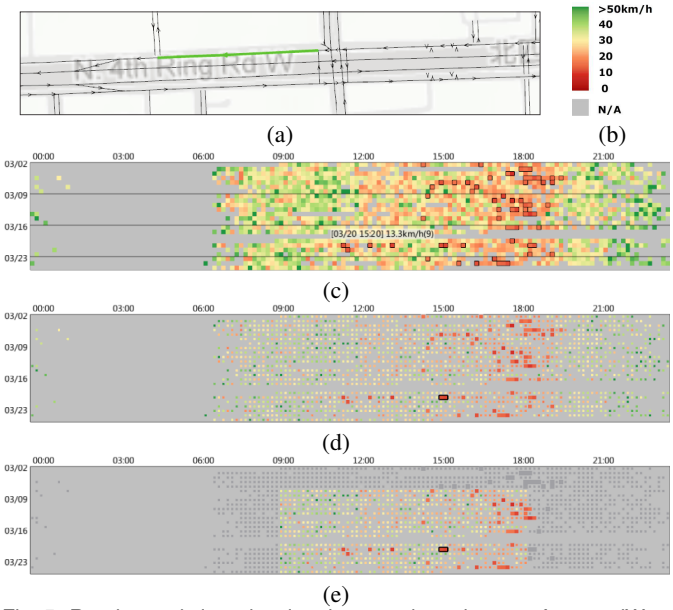

Fig. 5. Road speed view showing the speeds and events for one dWay. For the green road in (a) the speed variation is shown in (c). Each row represents one day, and each column represents 10min in a day, so each cell is a time bin. Cell color represents the calculated speed, with the color scale in (b). We mark the extracted events by black boxes in (c). Instead of using black boxes, we can use cell size to mark the events, as shown in (d). The events involved in the currently highlighted propagation graph are highlighted in a thick black box. When filtering is applied, all irrelevant cells turn gray, as shown in (e).

In order to show the events on this dWay, we draw black boxes on the road speed view. Figure 5(c) illustrates this. The cells covered in the box correspond to the time bins in traffic jams. Specifically, the left/right boundary represents the start/end time of the event. If we are more interested in the events, than in the details of speed, we can use cell size to mark events, as shown in Figure 5(d). We make the cells in traffic jam events, which we call *event cells*, bigger than the non-event cells. Events pop out in this style, and no black boxes are required to mark events. This is especially useful when we embed the road speed view in the spatial view, as shown in Figure 1(b). Then, due to limited screen space, we have to compromise speed for event information. Using black boxes would seriously hide the cell color.

In the road speed view, we can highlight a traffic jam propagation graph by clicking on an event, which then will be marked by a thick black box (Figure 5(c),(d)). The propagation graph containing this event will be highlighted, and shown in the spatial view.

We only show information for cells satisfying the filter, other cells turn gray, including non-event cells outside of the time range (defined by the temporal filters), and event cells not belonging to the selected propagation graphs. It is possible that an event outside the time range is not gray, as long as its corresponding propagation graph intersects with the time range. A filtered road speed view is shown in Figure 5(e).

5.2 Graph List View

After showing the speeds and events on individual roads, we consider showing the propagation graphs. This is information on a higher level, and reveals the interactions of traffic jams on different roads. In designing the visualization to show the propagation graphs, we have two concerns. Firstly, there are many propagation graphs, but we can only show a few simultaneously on the screen. Secondly, we need to com-

pare propagation graphs. We design the graph list view to fulfil the above requirements.

To reduce the number of propagation graphs to show, we use filtering and sorting. We have a topology filter in the graph projection view (Section 5.3), a spatial filter in the spatial view (Section 5.1), and five histogram filters for time and size in the attribute filter view (Section 5.5). We only show propagation graphs satisfying these filters. When there are still too many graphs, we sort them and only show the top N graphs, usually with $N \in [10, 50]$. The sorting can be based on the number of events, the time span, or the total distance.

To compare propagation graphs, we make a small multiple interface, as illustrated in Figure 1(c). It shows the propagation graphs as icons. These icons are arranged in a matrix style. Each icon represents one graph, and shows its spatial propagation path, temporal information, and the three size measurements. Color of the propagation path shows the congestion time at each location, with red being 4 hours and orange being 10 min. When users highlight a graph icon, the path of this graph will be shown in detail in the spatial view. A more detailed design of the graph icon is illustrated in Figure 6. The icon design and the matrix layout together allow side by side comparison of propagation graphs. However, in the matrix layout, two graphs that are far apart, are hard to compare. Therefore, we allow user to pin interesting graphs. Pinned graphs are listed separately as icons below the original icon matrix, which facilitates comparison, and allows reviewing at any time. Besides pinning, we allow users to select one graph and highlight all graphs that are spatially similar. The search for these similar graphs is limited to the visible top N graphs. The selected graph is put at the front of the graph list, while similar graphs are put immediately behind it, in decreasing similarity order. Icons for the selected graphs and its similar graphs have a red frame. We define the spatial similarity of two graphs, as the Jaccard coefficient [2] of their set of dWays.

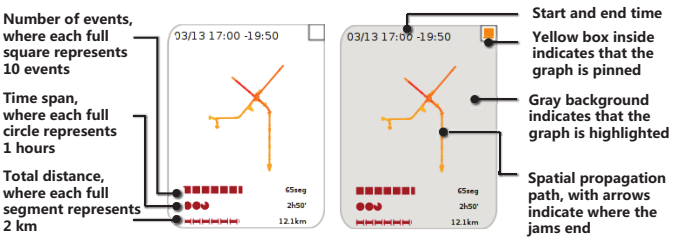

Fig. 6. The graph icon shows concise information of a propagation graph, including the start/end time, the spatial propagation path, the size in terms of the number of events, the time span, and the total distance. It also indicates the highlight state and pin state of the graph.

5.3 Graph Projection View

Also focusing on the propagation graph level, the graph projection view summarizes the topological information of the path of all graphs. However, the large number of graphs makes projection difficult. Our basic idea is to first divide graphs in topological clusters, then to project the clusters. To make clusters, we perform a topology preserving simplification on the path of each graph, by removing all mid-dWays, i.e. the dWays with both in-degree and out-degree being one. Then we calculate a feature vector for each simplified path $\langle I_0, O_0, I_1, O_1, ..., I_n, O_n \rangle$, where I_i is the number of nodes with in-degree i, O_i is the number of nodes with out-degree i, and n is the maximum of all in-degrees and out-degrees. Each dimension of the feature vector is normalized separately. Graphs with identical feature vectors are put in the same cluster. In our case, $n = 4$, and we get 212 clusters. For cluster projection, we use MDS [52]. The distance between two clusters is defined as the Euclidean distance between the feature vectors. The interface is shown in Figure 1(e). Each cluster is rendered as a point, with color indicating the number of graphs it contains. Darker means more graphs. Mouse hover shows the exact number of graphs, with a rendering of the graph using the Sugiyama layout [44] of the OGDF library [3]. The propagation direction is from top to bottom. Users can also draw a lasso to filter graphs in this view.

5.4 Spatial View

We require an overview of the traffic jams on a city level and a detailed inspection of propagation graphs. These tasks are achieved in the spatial view. See Figure 1(a).

The spatial density of a traffic jam is used to give a city-level overview. This density is defined on each dWay as the total congestion time on that dWay. We use color to encode the density: A dark red color means the dWay is most congested; a red color means less congested, followed by orange. A gray color represents no congestion.

The path of the highlighted propagation graph, is rendered as a flow map in black. We are especially concerned about the topology of the path, e.g. the start/end points and merging/branching points. The start points are the dWays that "creates" the jams, while the end points are the dWays that "absorb" the jams. In branching points, congestion in one dWay propagates to at least two dWays. In merging points, congestion in one dWay results from at least two dWays. To highlight such features, we use black circles for the start points, and black arrows for the end points. The branching/merging points can be simply discovered by looking at the path. Besides watching the static path, users can also play an animation of the propagation.

We allow users to check the speed and event information of multiple dWays, by embedding mini road speed views inside the spatial view. A green stippled line segment connects the mini road speed view and the dWay it represents. Users can create, move and delete the mini road speed views. Aligning them side by side allows a user to compare the speed patterns of roads.

Users can draw a rubber-band rectangle to set a spatial filter for the graphs. This is indicated by a green rectangle in Figure 1(a). They can also play trajectory animations. In this way, they can roughly validate the detected traffic jams, and make detailed observations.

5.5 Multi-faceted Filter View

We provide five interactive histograms to make a dynamic query on the propagation graphs. This is illustrated in Figure 1(d). The two histograms on the bottom show the temporal distribution of traffic jams. One by dates, and one by day times. On the top left corner of the date histogram, there are two buttons, allowing the users to observe data only in weekdays or weekends. The three histograms on top show the size distributions in terms of number of events, time span, and total distance. On each of the histograms, users can select a range. The five range queries on the histograms, plus the spatial query in the spatial view, and the topological filter in the graph projection view, form the whole filtering of our system. This filtering mechanism is a simplification of cross filtering [50]. Only propagation graphs satisfying all filters will be selected, and listed in the graph list view.

We provide two visual modes for the histograms. In absolute mode (Figure 7(a)), we only show the statistics of data passing the filter, in orange. The height of the bins is in a linear scale. In relative mode (Figure 7(b)), we also show the statistics of all the data as a gray background. As the scale of all data and data passing the filter may differ a lot, the height of bins is in logarithmic scale. We mark the information of the highlighted propagation graph on the histograms. Figure 7 shows the mark in the date histogram (as a black triangle) and in the time histogram (as a time range covered by stipple lines).

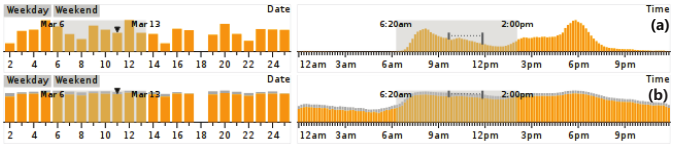

Fig. 7. Date and time histogram in absolute mode (a) and relative mode (b). The temporal information of the highlighted propagation graph is marked.

6 Visualization Results and Case Study

Our system provides users with visual insights on complex traffic data from multiple perspectives. The following cases demonstrate the capabilities and effectiveness of our visual analysis system.

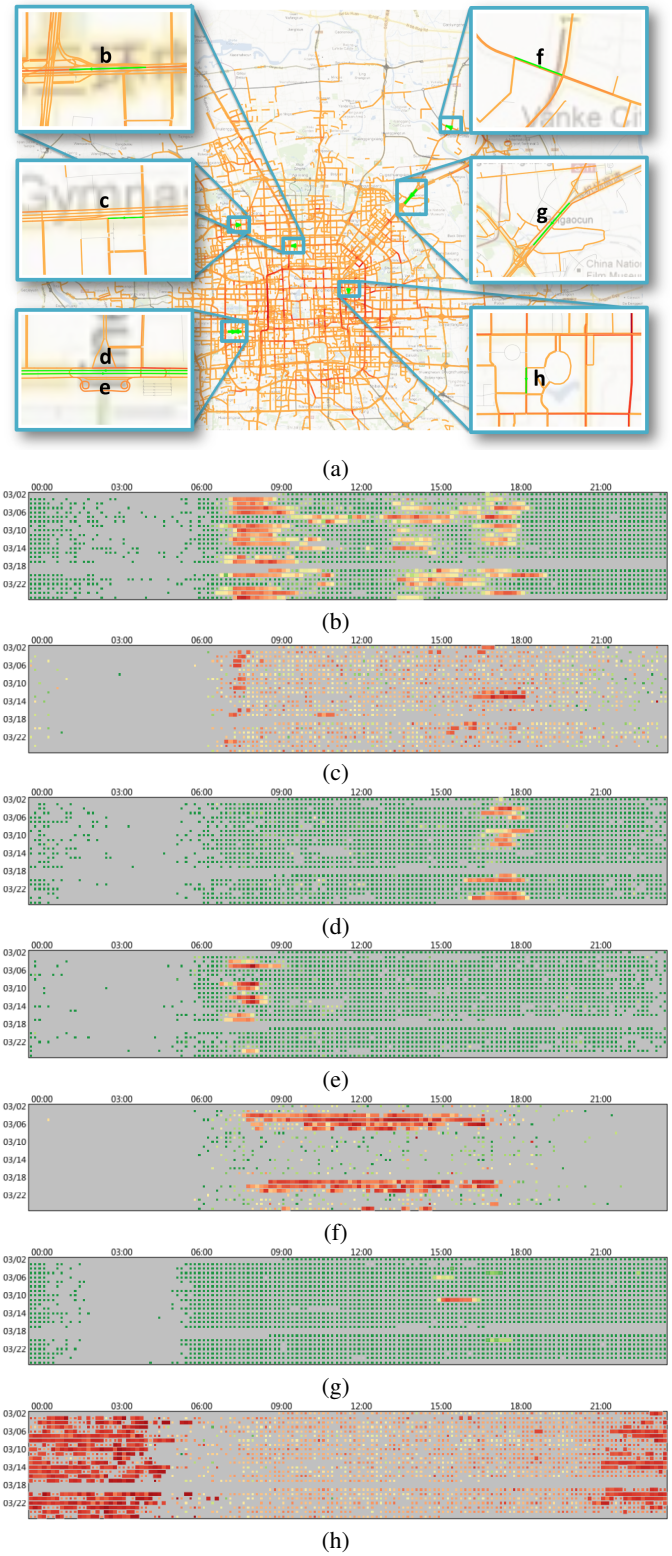

Fig. 8. (a) In Beijing, different roads have different traffic patterns. (b) The main road in the North 3rd Ring is regularly congested at weekdays in the morning and afternoon. (c) This road is beside two primary schools, it is also congested at weekdays, but usually before 7:30am, when parents send their children to school. (d,e) The two directions of the tunnel just outside Beijing West Station congest at different times, one only in the morning, one only in the afternoon. (f) The road besides the new National Exhibition Center at Shunyi is congested when there are exhibitions. (g) The Airport Express is occasionally congested by unpredictable incidents. (h) The road to the east of Beijing Worker's Stadium is regularly congested at the night of Friday and Saturday.

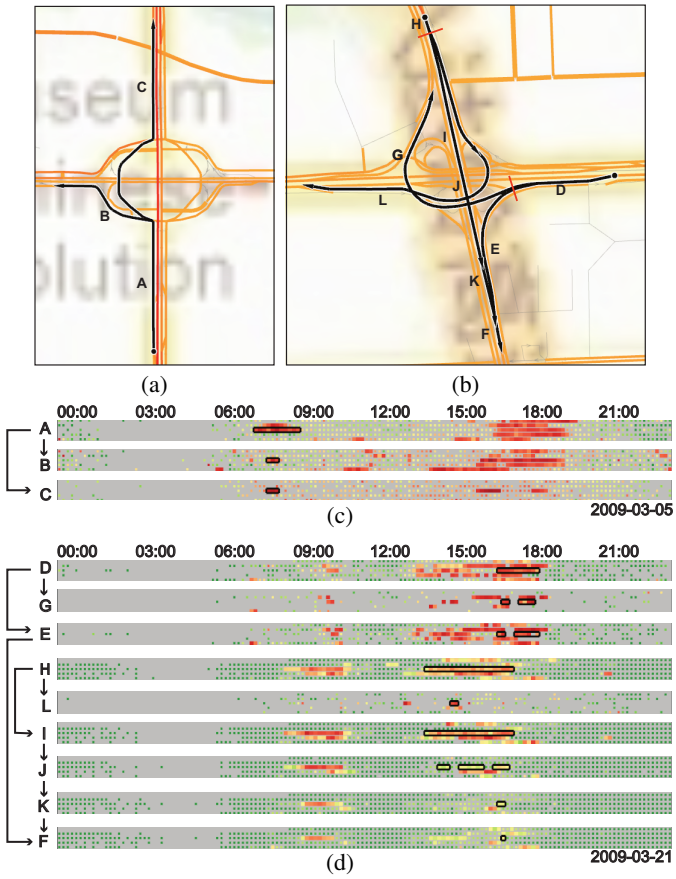

Fig. 9. Traffic congestion propagation and speed of road segments. (a) Congestion propagation in Lianhua Bridge on the West 3rd Ring of Beijing. (b) Congestion propagation in the Badaling highway intersection on the North 5th Ring of Beijing, where the red lines indicates the connection points of road segments. (c) Speed of road segments in (a). (d) Speed of road segments in (b).

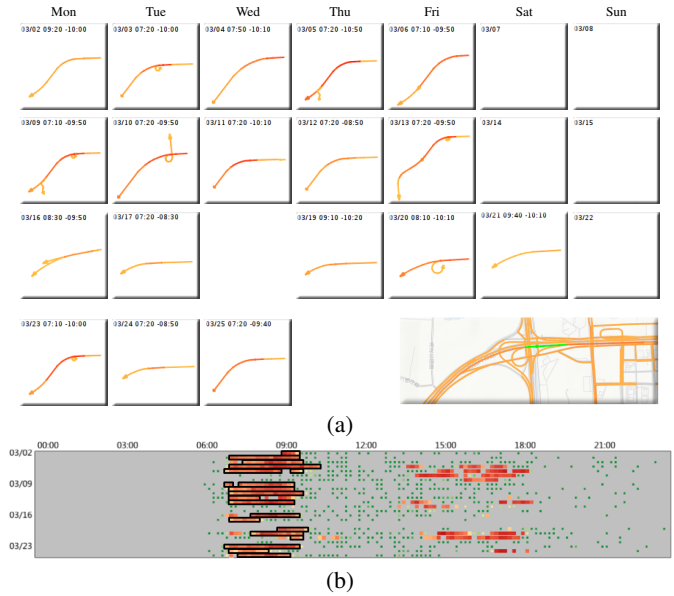

Fig. 10. Traffic congestion propagation graph pattern in Wanquanhe bridge. (a) Propagations in the morning of each day: blank glyph for no congestion propagation on that day, no glyph for missing data. (a inset) Road network around Wangquanhe bridge. (b) Speed view of the green road segment on the bridge.

6.1 Case 1. Road Segment Level Exploration and Analysis

With the road speed view, our system provides users with an aggregated visualization of the traffic speed on a road segment. Figure 8 shows seven road segments (dWays) in Beijing. Each road speed view gives a clear visual summary of the traffic congestion patterns of that segment. We can observe that for most roads, the traffic starts to be visible around early morning 6:00 am, and most weekdays suffer morning and afternoon traffic peaks, while the traffic on weekends is much lighter. Figure 8(b) shows a road segment of the North 3rd Ring with clearly a morning peak, when people go to work, and an evening peak, when people go back home. For roads close to a primary school, the morning peak comes earlier, as parents bring their children to school before they go to work (Figure 8(c)). Figures 8(d) and (e), show the two opposite directions of a road with very different behaviors. The differences can be contributed to the directional traffic from home to work. Figure 8(f) shows another pattern that is heavily influenced by local activities. The road is south of a large Exhibition Center, and congests only at days with exhibitions. While most of the congestions mentioned above have some regularity and predictability, some traffic jams occur more randomly. The road segment on the Airport Express (Figure 8(g)) usually has smooth traffic, but can occasionally be congested by incidents. The road shown in Figure 8(h) is east of Beijing Worker's Stadium, and hosts many bars. It has more traffic load throughout the night. Friday and Saturday have earlier heavy traffic in the evening, since people visit the bars earlier during weekends.

6.2 Case 2. Visual Propagation Graph Analysis

By filtering on temporal, spatial and size properties of traffic congestion propagation, users can explore different traffic jam propagation graphs with our visual interface. The detailed propagation can be examined in a road speed view. Figures 9(a) and (b) show two different traffic congestion propagations. Figures 9(c) and (d) show their road speed. Congestion propagation in Figures 9(a) happened on a road named Lianhua Bridge on the west 3rd Ring of Beijing. Place A on the road was struck by a traffic congestion first. With a clear time delay segments B and C become, almost simultaneously, congested too. Figure 9(b) presents a more complex traffic congestion propagation. It happened at the Badaling highway intersection on the north 5th Ring of Beijing. This propagation was caused by two sources: D and H. Firstly H was congested. Then congestions in I, J, and K occurred gradually with some delay. When D became congested, E, a branch of F was affected badly. At the same time, F became congested. When H was free from congestion, I to K were all free from congestion, too. E continued to be congested until D was relieved from traffic jams.

6.3 Case 3. Congestion Propagation Pattern Exploration

We enables users to compare traffic congestion propagation graphs in an area at different times. For example, Wanquanhe Bridge is located at the north-west corner of the 4th Ring in Beijing; see the right-bottom of Figure 10(a). Figure 10(b) shows the speed of a road segment on this bridge. Except for missing data on March 18, a strong periodicity is presented throughout the whole data set. On every weekday, traffic congestion occurs from about 7 a.m. till 10 a.m. In weekends, the traffic jam in the morning was replaced by one in the afternoon. To study propagations in the morning of weekdays in detail, we list them day by day, as shown in Figure 10(a). We found that all these congestions originated from the road around Zhongguancun Science Park, an area with many high-tech enterprises. Although the traffic congestion propagation graphs differ by some branches, the main bodies of these graphs are the same, firstly from east to west and then to the south.

7 Discussion

The preprocessing steps in our system require many parameters. They are important to produce usable traffic jam data for further visual exploration. However, finding proper settings for these parameters is not trivial. Currently, we do so based on analysis of distributions, on experience, and on comparison to manually labelled data. We also perform sensitivity analysis on these parameters. Additionally, our visual interface gives users visual feedback of the extracted traffic jam data. When users are not satisfied with the results, they can redo the last two steps of the preprocessing with different parameters. This takes approximately one minute for the Beijing taxi data on our workstation.

Our work focuses on event analysis, and uses animation of moving objects to visually evaluate the detected traffic jams. Further analysis in space and time is possible. For instance, by clustering the road segments based on their speed change over time, or by clustering the time bins based on the spatial distribution of speed at that time. The space-in-time and time-in-space SOM [8] provides such techniques. We will consider it in the future.

Our work studies the causal relationship, the propagation of traffic jams. However, studying the correlation of traffic conditions is also important. This can be done using PGMs. PGMs provide the possibility to predict the traffic condition in different roads at different times, based on current observations. It is also able to model negative correlations, which is obviously interesting. We consider incorporate the PGMs in our visual analysis.

In intelligent transportation systems, pre-warning before accidents is more important than post-accident response, since it might help to save lives and costs. As for traffic jams, a traveller may expect that a route plan become more intelligent and adaptive. This obviously heavily depends on the prediction performance for traffic jams. Our current work is more on post-jam analysis. However, we plan to extend our system to include real-time prediction.

Finally, it is challenging to summarize large number of propagation graphs. It is especially difficult to clearly put them in semantic clusters, simultaneously considering the spatial, temporal, size and topology aspects. Our system treats these aspects separately, and gives an overview in terms of each of them.

8 Conclusion and Future Work

In this work, we have presented an interactive visual analysis system to analyze traffic jams in a realistic large scale road network. We use 24 days of taxi GPS trajectories in Beijing and a corresponding street network from OpenStreetMap. In a data driven approach we clean the GPS trajectories from sensor errors and fix apparent errors in the road network. With the cleaned data we can accurately map the driving trajectories to the road network and subsequently, compute road speeds. After estimating free flow speed on each road segment, we automatically detect traffic jam events at roads based on relative low-road-speed detection. The concatenation of these events in propagation graphs shows how a traffic jam propagates both in space to adjacent roads and in time. Based on the automatic computing results, we then build a visual interface for interactive exploration of the detected traffic jam information both in detail on a road segment as well as on a higher level in a spatial view on a map and in a small multiples view with propagation graphs. We support the analysis by efficient filtering of space, time, size and topology, and providing structured visualizations of the graphs through sorting by size and similarity. Finally, we provide a number of case studies that demonstrate the effectiveness of our system. Our system can provide users with insights from multiple levels and perspectives.

Our future work includes improving the traffic jam model, support more analysis tasks, and enable real-time traffic prediction. We will also try better visual encodings for the propagation graphs. We consider to make a formal evaluation of our system.

Acknowledgments

The authors thank Datatang and OpenStreetMap for providing the data, and the anonymous reviewers for their valuable comments and suggestions. This work is supported by National NSFC Project (No. 61170204) and National NSFC Key Project (No. 61232012).

Appendices

Our preprocessing steps have many parameters. Correctly setting them is crucial, but not trivial. We discuss the parameter settings below.

Table 1. Parameter settings for GPS data cleaning filters. The remaining filters F1, F2, and F7, have R values 17%, 0.08%, and 13%, respectively.

Filter	R%	Value	Sensitivity	Explanation
F3	0.42%	90km/h	0.0008% per 1km/h	Speed limits are between 30 to 120km/h. However, splitting high speed segments does not affect traffic jam extraction.
F4	3.0%	2km	0.2% per 0.1km	It corresponds to at least 3 dWays when matched to the road network.
F5	3.3%	10min	0.6% per 1min	It corresponds to 20 missing sampling points.
F6	62%	50m	0.3% per 10m	It covers 98.7% of the GPS drift errors, assuming it obeys a Gaussian distribution with standard deviation = 20m [40].
		30min	0.2% per 1min	We are not aware of traffic jams in Beijing, in which vehicles are completely stuck for over 30min.
F8	20%	5 points	1% per point	By rendering trajectories with less than 5 points, we find that they can not be convincingly matched to the road network.
		500m	0.1% per 100m	Same as above.

A GPS Data Cleaning

Our GPS data cleaning relies on a set of filters. All of them try to remove erroneous or unusable data. An appropriate parameter setting aims to strike a balance between the percentage R of points removed and the noise level. We perform sensitivity analysis on most of them, showing how much more/less data will be removed, if the parameters change slightly. We use the One-at-a-time strategy [4], and calculate the partial derivative of R for each parameter. We summarize the filter parameters in Table 1. The current parameter setting removes 74.5% of the data, most of which are stops removed by F6 and F7.

B Map Matching

The map matching result is analysed by comparison to manually labelled data, with Mao et al.'s technique [32]. We randomly choose 500 trajectories from our dataset, containing 14,632 sampling points and match them automatically. After manual correction with Unmatch being allowed, we consider the result as the "ground truth". The map matching accuracy based on a "ground truth" is defined as: $Accuracy = \sum_{i=1}^{n} |M_i \cap T_i| / \sum_{i=1}^{n} |M_i \cup T_i|$, where n is the number of trajectories, and M_i and T_i are the sets of directed ways for the i-th trajectory in the map matching result and the ground truth, respectively.

We choose 400 of the 500 trajectories to estimate the best parameters: candidate search radius r, candidate number k, normal distribution standard deviation σ (we assume the mean μ is zero), and our minimum transmission probability Δ. The original ST-matching paper [30] recommended $r = 50m$, $k = 5$, $\sigma = 20m$, and no Δ which is equivalent to $\Delta = 0.0$. Testing the combinations with $r \in \{20m, 50m, 100m\}$, $k \in \{3, 5, 10\}$, $\sigma \in \{10m, 20m, 50m\}$ and $\Delta \in \{0.0, 0.2, 0.4, 0.6\}$, gave accuracy ranging from 81.6% to 91.7%. The best of these combinations ($r = 50m$, $k = 5$, $\sigma = 20m$, $\Delta = 0.2$) gives a 92.6% accuracy on the remaining 100 trajectories, and is used by us to do the map matching. As a result, 5% of the sampling points and 7% of trajectory segments are unmatched.

C Road Speed Calculation

The setting of parameter *min_support* is a balance between the confidence level of the prediction and the number of speed estimates (See Figure 11). If users want accurate data, then they choose a high value. If they want more data, a low value. Our default setting is $min\_support = 5$. Under this setting, given a $\pm 4mph$ error bound, the theoretical confidence level [33] of calculated speed in freeways are guaranteed to be above 58% under all conditions, and 85% with normal traffic volume. However, the confidence level in arteries are quite low (44% and 72%). Users may change this setting. For each dWay, we define a *Coverage* value, which is the ratio of time bins with valid speed, i.e. $support \geq min\_support$. Figure 12 shows the result.

D Traffic Jam detection

The traffic jam detection result is also analysed by comparing with manually labelled data. We randomly select 50 freeway road segments (dWays) from the road network, within the 4th Ring of Beijing. We play an animation of one day traffic data, and manually label the congested time bins. This labelling is a bit subjective, since sometimes the road condition is between congestion and free flow, and sometimes the number of trajectories on that road is insufficient. In these situations, the results depend on human judgment. Still, we assume the labelled data is a "ground truth", based on which we calculate an accuracy: $Accuracy = \sum_{i=1}^{n} |S_i \cap L_i| / \sum_{i=1}^{n} |S_i \cup L_i|$, where n is the number of dWays, S_i is the set of time bins detected as congested for the i-th dWay, and L_i is the set of time bins labelled as congested in the "ground truth" for the i-th dWay.

We use the labelled data to estimate the best parameters, including the speed percentage of free flow speed F, and of congestion speed C. We have tested F from 100 to 70 and C from 60 to 30, both at an interval of 5. The results are shown in Figure 13. Although the combination of $F = 75, C = 50$ gives the highest accuracy, namely 79.7%, it is not stable. We choose a stable combination $F = 85, C = 45$, which gives 76.3% accuracy.

The labelling provides a means to incorporate human preference in the traffic jam detection. For example, when users want only definite and serious traffic jams, they can relabel the 50 freeways accordingly, then re-estimate the parameters, and redo the traffic jam detection.

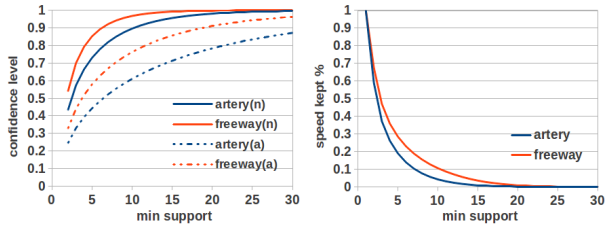

Fig. 11. Trade off of *min_support* selection. (left) The confidence level of speed calculation with different *min_support* values, under different road conditions: freeway(n) and artery(n) with normal traffic volume, freeway(a) and artery(a) with very low/high traffic volume. (right) The relative number of speed estimation with different *min_support*.

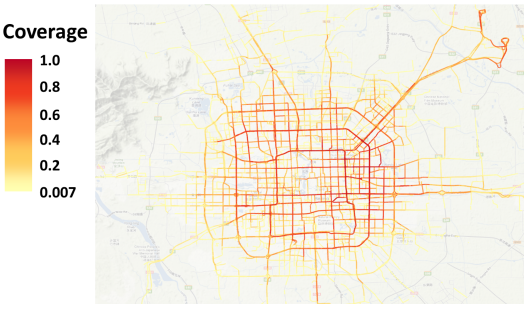

Fig. 12. The coverage of dWays in our data.

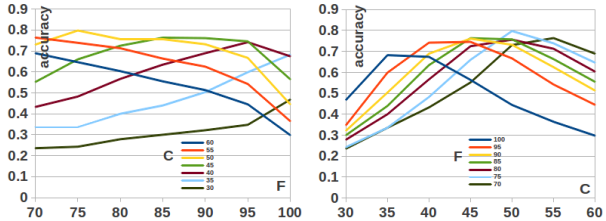

Fig. 13. Accuracy of event detection under different parameter settings. (left) Accuracy vs. F, under different C. (right) Accuracy vs. C, under different F.

REFERENCES

[1] Chauvenet's criterion. http://en.wikipedia.org/wiki/Chauvenet's_criterion.

[2] Jaccard index. http://en.wikipedia.org/wiki/Jaccard_index.

[3] Open graph drawing framework. http://www.ogdf.net/ogdf.php.

[4] Sensitivity analysis. http://en.wikipedia.org/wiki/Sensitivity_analysis.

[5] Beijing transportation research center: Annual report of beijing transportation development, 2010.

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

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

[8] G. Andrienko, N. Andrienko, S. Bremm, T. Schreck, T. Von Landesberger, P. Bak, and D. Keim. Space-in-time and time-in-space self-organizing maps for exploring spatiotemporal patterns. *Comput. Graph. Forum*, 29(3):913–922, 2010.

[9] G. Andrienko, N. Andrienko, and M. Heurich. An event-based conceptual model for context-aware movement analysis. *Int. J. Geogr. Inf. Sci.*, 25(9):1347–1370, 2011.

[10] G. Andrienko, N. Andrienko, C. Hurter, S. Rinzivillo, and S. Wrobel. From movement tracks through events to places: Extracting and characterizing significant places from mobility data. In *Proc. IEEE VAST*, pages 161–170, 2011.

[11] G. Andrienko, N. Andrienko, M. Mladenov, M. Mock, and C. Poelitz. Extracting events from spatial time series. In *Information Visualisation (IV)*, pages 48 –53, 2010.

[12] N. Andrienko and G. Andrienko. Spatial generalization and aggregation of massive movement data. *IEEE Trans. Vis. Comput. Graph.*, 17(2):205 –219, 2011.

[13] R. Bauza, J. Gozalvez, and J. Sanchez-Soriano. Road traffic congestion detection through cooperative vehicle-to-vehicle communications. In *Proc. IEEE Conf. on Local Computer Networks*, pages 606–612, 2010.

[14] W. Beaty. Traffic waves, sometimes one driver can vastly improve traffic. http://trafficwaves.org/, 1998.

[15] S. Chawla, Y. Zheng, and J. Hu. Inferring the root cause in road traffic anomalies. In *Proc. IEEE ICDM*, pages 141–150, 2012.

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

[17] I. Dees. Openstreetmap jxapi. http://wiki.openstreetmap.org/wiki/Xapi.

[18] W. Dong and A. Pentland. A network analysis of road traffic with vehicle tracking data. In *AAAI Spring Symp.: HBM*, pages 7–12, 2009.

[19] 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 *Proc. IEEE PacificVis*, pages 163 –170, 2011.

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

[21] C. Hurter, B. Tissoires, and S. Conversy. Fromdady: Spreading aircraft trajectories across views to support iterative queries. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1017–1024, 2009.

[22] V. Jain, A. Sharma, and L. Subramanian. Road traffic congestion in the developing world. In *Proc. ACM Symposium on Computing for Development*, pages 11:1–11:10, 2012.

[23] P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. In *Proc. intl. conf. on Advances in Spatial and Temporal Databases*, pages 364–381, 2005.

[24] T. Kapler and W. Wright. Geotime information visualization. In *Proc. IEEE InfoVis*, pages 25–32, 2004.

[25] B. Krogh, O. Andersen, and K. Torp. Trajectories for novel and detailed traffic information. In *Proc. ACM SIGSPATIAL International Workshop on GeoStreaming*, pages 32–39, 2012.

[26] P. Laube, S. Imfeld, and R. Weibel. Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science*, 19(6):639–668, 2005.

[27] T. Liebig, C. Korner, and M. May. Scalable sparse bayesian network learning for spatial applications. In *Proc. IEEE ICDM Workshops*, pages 420–425, 2008.

[28] H. Liu, Y. Gao, L. Lu, S. Liu, H. Qu, and L. M. Ni. Visual analysis of route diversity. In *Proc. IEEE VAST*, pages 171–180, 2011.

[29] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing. Discovering spatio-temporal causal interactions in traffic data streams. In *Proceedings of ACM SIGKDD*, pages 1010–1018, 2011.

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

[31] J. Machay and D. Media. How does google detect traffic congestion?, 2013. http://smallbusiness.chron.com/google-detect-traffic-congestion-49523.html.

[32] H. Mao, W. Luo, H. Tan, L. M. Ni, and N. Xiao. Exploration of ground truth from raw gps data. In *Proc. ACM SIGKDD International Workshop on Urban Computing*, pages 118–125, 2012.

[33] J. Nezamuddin, L. Crunkleton, P. J. Tarnoff, and S. E. Young. Statistical patterns of traffic data and sample size estimation. http://www.catt.umd.edu/documents/variance_analysis_v1.pdf, 2009.

[34] R. Ong, F. Pinelli, R. Trasarti, M. Nanni, C. Renso, S. Rinzivillo, and F. Giannotti. Traffic jams detection using flock mining. In *Machine Learning and Knowledge Discovery in Databases*, volume 6913 of *LNCS*, pages 650–653. Springer Berlin Heidelberg, 2011.

[35] M. Pack, K. Wongsuphasawat, M. VanDaniker, and D. Filippova. Ice–visual analytics for transportation incident datasets. In *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, pages 200 –205, 2009.

[36] A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proc. ACM symposium on Applied computing*, pages 863–868, 2008.

[37] N. Piatkowski, S. Lee, and K. Morik. Spatio-temporal models for sustainability. In *Proc. SustKDD Workshop within ACM KDD*, 2012.

[38] H. Piringer, M. Buchetics, and R. Benedik. Alvis: Situation awareness in the surveillance of road tunnels. In *Proc. IEEE VAST*, pages 153–162, 2012.

[39] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko. Effectiveness of animation in trend visualization. *IEEE Trans. Vis. Comput. Graph.*, 14:1325–1332, 2008.

[40] D. Rodriguez, E. Shay, and D. A. Cohen. Using gps and accelerometers in neighborhood research. http://www.activelivingresearch.org/files/GPS-Accelerometers_Workshop.pdf, 2008.

[41] R. Scheepens, N. Willems, H. van de Wetering, G. Andrienko, N. Andrienko, and J. van Wijk. Composite density maps for multivariate trajectories. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2518–2527, 2011.

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

[43] A. Slingsby, J. Wood, and J. Dykes. Treemap cartography for showing spatial and temporal traffic patterns. *J of Maps*, pages 135–146, 2010.

[44] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst., Man, Cybern.*, 11(2):109 –125, 1981.

[45] Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S. ichi Tadaki, and S. Yukawa. Traffic jams without bottlenecksexperimental evidence for the physical mechanism of the formation of a jam. *New Journal of Physics*, 10(3):033001, 2008.

[46] 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.

[47] S. van den Elzen, D. Holten, J. Blaas, and J. van Wijk. Reordering massive sequence views: Enabling temporal and structural analysis of dynamic networks. In *Proc. IEEE PacificVis*, pages 33–40, 2013.

[48] K. Verbeek, K. Buchin, and B. Speckmann. Flow map layout via spiral trees. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2536–2544, 2011.

[49] T. von Landesberger, S. Bremm, N. Andrienko, G. Andrienko, and M. Tekusova. Visual analytics methods for categoric spatio-temporal data. In *Proc. IEEE VAST*, pages 183–192, 2012.

[50] C. Weaver. Cross-filtered views for multidimensional visual analysis. *IEEE Trans. Vis. Comput. Graph.*, 16(2):192 –204, 2010.

[51] N. Willems, H. van de Wetering, and J. van Wijk. Visualization of vessel movements. *Comput. Graph. Forum*, 28(3):959–966, 2009.

[52] P. C. Wong and R. D. Bergeron. Multivariate visualization using metric scaling. In *Proc. IEEE Visualization*, pages 111–118, 1997.

[53] Y. Yang, Z. Cui, J. Wu, G. Zhang, and X. Xian. Fuzzy c-means clustering and opposition-based reinforcement learning for traffic congestion identification. *Journal of Information and Computer Science*, 9:2441– 2450, 2012.

[54] Y. Zheng and X. Zhou, editors. *Computing with spatial trajectories*. Springer, 2011.