Interactive Visualization of Traffic Dynamics Based on Trajectory Data

George A. M. Gomes, Emanuele Santos, Creto A. Vidal
Federal University of Ceará
Fortaleza, Brazil
Email: george@virtual.ufc.br, {emanuele, cvidal}@dc.ufc.br

***Abstract*—Urbanization is accelerating worldwide, giving rise to serious traffic problems. Traffic wave, known as stop-and-go traffic or phantom intersection, is one of the most significant traffic oscillation patterns studied in Traffic Engineering. Usually these studies are numerical experiments that investigate specific places, such as a crossroad or a highway section, and their findings cannot, therefore, be easily applied to sensing device data in a systematic computational approach. In this regard, visual analytics can help by combining automated analysis with interactive visualization for effective understanding, reasoning, and decision-making. In this paper, we present a novel approach for visualizing traffic oscillation patterns by visualizing the objects' movement in space over time, inspired by vector field visualization. We propose an algorithm to control and synchronize the visualization time; a systematic stepwise methodology for exploring sensing device data; and a visualization tool that computes the trajectory data in parallel on the GPU at interactive frame rates. Moreover, our approach is designed to support both batch-processed and streaming data applications. We also present the benefits and limitations of our visualization proposal based on domain expert feedback. Finally, we present performance tests with very encouraging results to support our approach.**

I. Introduction

Urbanization is a process that is accelerating worldwide, giving rise to serious traffic problems. To address this problem, recent research efforts have been focusing on the use of trajectory data, collected from sensing devices such as RFID tags, video cameras, laser scanners, and smartphones [1], [2]. Because of the huge amount of data involved, to analyze trajectory data is not an easy task, and, often, requires exploratory visualization for detecting and describing patterns, trends, and relations within the data [3].

Many researchers have focused on the global aspects of traffic flows [4]–[6]. However, it is also essential to analyze local mobility patterns because the behaviors of individuals affect the traffic flow directly. Frequently, a traffic flow behaves like a supersaturated solution, in which small disturbances, e.g., a car breaking or changing lanes could cause a chain reaction. Those disturbances may cause traffic waves that could become traffic jams.

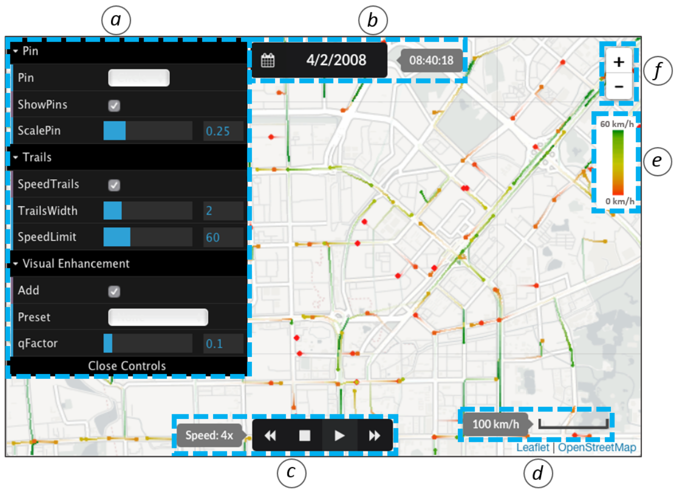
Traffic wave, known as stop-and-go traffic or phantom intersection, is one of the most significant traffic oscillation patterns [7]. This phenomenon is observable through the engagement of vehicles in repeated deceleration-acceleration cycles, and it is likely to incur many adverse impacts on traffic efficiency and sustainability. Researchers have conducted numerous studies investigating traffic oscillation with empirical observations and theoretical models, trying to understand the propagation of traffic waves (see [8] for an overview). In those studies, the main causes of traffic waves were identified: traffic lights, crossings, access ramps, lane changes, lane blocks, uphill gradients, presence of trucks, and accidents. However, most of those studies are numerical experiments that investigate specific places, such as a crossroad or a highway section, and their findings cannot, therefore, be easily applied to sensing device data in a systematic computational approach.

In this regard, visual analytics can help by combining automated analysis with interactive visualization for effective understanding, reasoning, and decision-making on the basis of a very large and complex dataset [9], especially for urban computing [2]. Furthermore, visualization tools are particularly important for analyzing phenomena and processes that are unfolding in geographical space because the heterogeneity of the space and the variety of properties cannot be adequately represented for fully automatic processing [10].

Moreover, most existing visual analysis approaches do not address the time dimension appropriately, because they focus only on the spatial visualization or use some form of temporal dimension simplification. Space and time are inseparable in visual analysis of the dynamics of traffic mobility. The singularities in any of those dimensions must not be discarded because they may reveal implicit relationships [11]. Therefore, to visualize traffic oscillation patterns, the time dimension is essential.

Because trajectories represent moving objects, it is natural to analyze them by visualizing the dynamic characteristics of the objects' movement through animation. That visualization, since it uses the animation paradigm, promotes the real understanding of the objects' collective movement in the city, and the perception of the impacts of individual behaviors on traffic flow.

This paper proposes a new approach that can be used by domain users and city planners to interactively and dynamically visualize traffic oscillation patterns from trajectory data collected by sensing devices. The final visual result of our technique was inspired by animated vector field visualization techniques [12]. Therefore, the movement of an object leaves a trail that persists on the road, which is similar to a path line in vector field visualization. However, in our visualization, this trail is slowly fading out, and its length represents the magnitude of the object's speed. Notice that we do not compute



Pin
Pin
ShowPins ✓
ScalePin 0.25
Trails
SpeedTrails ✓
TrailsWidth 2
SpeedLimit 60
Visual Enhancement
Add ✓
Preset
qFactor 0.1

Fig. 1. Screenshot of a visualization of the traffic flow from 1,811 taxis moving at the same time. The image depicts the main functionalities of our developed tool such as the customization panel (a), where the user can set the marker size, speed limit, trail width, visual enhancement, and other parameters; the date and time display (b); the time control (c); the trail speed's scale (d) and color encoding (e); and zoom control (f). A video showing the animated visualization is available on https://youtu.be/uycpCRSPyLo.

vector fields, i.e., the trajectory data are processed to visualize the objects' movement directly. The main contributions of our approach are:

- We present a method to facilitate the analysis of traffic oscillation patterns by visualizing the objects' movement in the space over time. To this purpose, we use visual aids, such as trails, shapes, and color encoding, to perceive behaviors, such as speed, direction, acceleration, and location (see Figure 1);
- We propose a method to control and synchronize the visualization time that allows an analyst to visualize hours of movement data, in real time, in just a few minutes. Unlike existing approaches that dynamically visualize trajectory data as an animation, our method provides more time control functionalities – pausing, fast forwarding, rewinding, and different speeds. Thus, even when the visualization is accelerated at high speeds, our visual aids allow the analyst to perceive the traffic dynamics in an overview. To support all this, we devised a structured data model based on the idea of animation key frames;
- We present a systematic stepwise methodology for exploring sensing devices data. Our approach allows a non-expert user to visualize traffic dynamics from raw data – latitude, longitude, and time stamp. Unlike many other trajectory data visualization techniques, we do not require map matched data. This work is the first step towards an end-to-end solution;
- We developed a visualization tool that works in parallel on GPU with interactive frame rates, even for comparably large datasets. In our performance tests using an ordinary workstation, we were able to visualize more than 160,000 animated graphic elements at 20fps.

Besides, our approach was designed to support both batch-processed and streaming data applications. In the case of on-line streaming applications, we use the framework proposed by Sacharidis et al. [13] to discover hot motion paths. In their work, the authors present a distributed approach for processing and filtering location updates in the client side. That framework exploits the computational capabilities at the client side, and substantially reduces the processing cost at the server side. In addition, this method minimizes the communication overhead due to fewer location updates. Therefore, the steps of our data preparation phase, see Figure 2, were designed to be performed at interactive frame rates.

Finally, we report an evaluation carried out by an urban planner expert that shows the benefits and limitations of our visualization technique.

The remainder of this paper is organized as follows. In Section II, we review the related work. In Section III, we describe our approach. In Section IV, we report our qualitative evaluation. In Section V, we discuss our implementation. In Section VI, we conclude our paper with an outline of future work.

II. Related Work

Trajectory data of urban centers are being widely studied thanks to their ready availability nowadays. We can easily find surveys in many fields of research such as Urban Computing [2], Data Mining [14], Traffic Engineering [7], and Geographic Information [15]. The visualization community has been helping these research fields by providing effective ways to integrate humans into a data exploration process [16]. Andrienko and his coauthors present an interesting survey of systems and techniques for visual analytics of movement data [10] and a list of challenges in geospatial visual analysis [17]. In this section, we present the related works considering two aspects: global and local mobility patterns.

Global mobility patterns. The majority of the existing research is focused on analyzing global aspects of trajectory data [18]–[20]. Some techniques represent movements between regions as a graph, in which regions are graph nodes and the flows between them are treated as weighted directed edges [5], [6], [21]. Movement occurring between those areas is simplified and visualized using a flow map. Those approaches often aggregate trajectories in space or time to facilitate visual analysis. So, one can no longer see the changes of spatial positions of objects, i.e., the very essence of movement is lost [22]. Unlike these works, our approach was designed to visualize individual behaviors in traffic flow.

Local mobility patterns. There is not much work on the analysis of the local mobility patterns. Kraak [23] proposed the technique space-time cube, which combines time and space in a single 3D display. Movement behavior of an object is shown as a 3D line. The slope of a line segment indicates the speed of movement, e.g., gradual rise means high speed and a steep slope signifies slow movement. Tominsk and coauthors extended this technique to the Stacking-based Visualization [24] and The Great Wall of Space-Time [25]. These methods

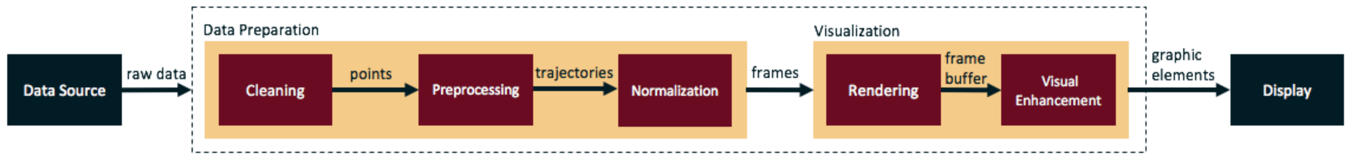


Fig. 2. Our approach's pipeline: in the Data Preparation phase, the system receives raw data from a data source, cleans them and structures them in an animation frame dataset. This frame dataset is rendered in the Visualization phase, which generates graphic elements that are sent to the display.

are quite limited with respect to the number of trajectories they can display. Moreover, the use of 3D visualization may lead to ambiguity due to perceptual problems like potential occlusions [26]. Our method allows the visualization of a great number of objects in 2D space, with minimal cluttering. Guo et al. [27] proposed a visualization method to investigate local traffic patterns and abnormal behaviors. They analyze a local traffic dataset at a road intersection collected through several laser scanners and other auxiliary devices. Despite the fact that this approach allows the visualization of individual moving cars through animation of box-like representations, they do not use visual encoding to depict the object's speed. So, unlike ours, with their approach, it is not possible to visualize traffic oscillation patterns, i.e., traffic waves. Poco et al. [28] proposed a visualization of the traffic dynamics based on the movement of multiple particles in a vector field. They presented a vector-valued function to adapt vector field over road networks. In that method, a particle moves in the field according to a vector's magnitude, computed from the average road speed of traffic, so that individual speed variations are not visualized. Thus, this approach is not suitable for visualizing traffic oscillation patterns. In fact, our approach neither uses vector fields nor computes the closest path for each trajectory. Both are expensive and time-consuming tasks.

Other libraries and tools. Recently, Uber Engineering has released a framework for visual exploratory data analysis of large datasets [29]. Although there is a performance demo using a large taxi trajectory dataset, that framework does not attempt to be a systematic approach for visual analysis of the traffic's dynamics. However, their framework can be used for rendering the visualization. Finally, Treiber [30] presents a microsimulation of traffic flow with six scenarios depicting the most common problems studied in Traffic Engineering. Despite being didactic, Treiber's work is limited with respect to the number of active objects, and uses a naïve visualization technique.

III. Our Approach

A naïve approach to visualize trajectory data as an animation is to interpolate, per time frame, a new point for each trajectory, i.e., to compute the locations of all moving objects at a specific time, see Figure 4. When movement attributes such as speed and acceleration need to be represented using shapes and color encodings, extra computations are required per frame. Furthermore, if trails of the moving objects are needed, the previous locations need to be found for each moving object, which requires a lot of extra computations. In a typical trajectory dataset, it is common to have thousands of objects moving at the same time. Thus, all of those extra computations may impair visualization.

In order to visualize thousands of objects at interactive speed, we need a better solution. So, we propose a systematic stepwise methodology that consists of two phases (represented by the yellow boxes in Figure 2): the Data Preparation phase and the Visualization phase. When the raw data enters the Data Preparation phase, it undergoes three steps in pipeline: the Cleaning step (Section III-A), which removes trajectory inconsistencies; the Preprocessing step (Section III-B), which precomputes space-time attributes, and form the trajectories; and the Normalization step (Section III-C), which structures the trajectories to fit our model. The Visualization phase receives the result of the Data Preparation phase, processes two steps in pipeline: the Rendering step (Section III-D), which receives a structured frame data to be rendered and composes images for the visualization; and the Visual Enhancement step (Section III-E), which enhances the attributes of the previously generated images in order to facilitate visual analysis. The algorithms of the Visualization phase are executed in parallel on the GPU.

A. *Cleaning*

The collection of raw data often comes with inaccuracies caused by the data acquisition devices. So, the Cleaning step mainly removes: the points with bad accuracy; the sampling points outside the analysis range; the duplicated points; and the unrealistic trajectory segments with very high speeds. This step, which was based on Zheng's work [14], produces a point dataset $\mathbf{p}$ for the next steps.

For an on-line application, we adopted the distributed approach for processing and filtering location updates presented by Sacharidis et al. [13]. Each client user executes an algorithm that cleans and compresses its trajectory on-the-fly. For a batch-processed data application, the data cleaning algorithm is $O(n)$ on the number of points in the dataset $\mathbf{p}$, where $n = |\mathbf{p}|$. This step is performed only once for the dataset.

B. *Preprocessing*

To minimize the high computational cost of rendering the large amount of graphic elements in a dynamic visualization, our technique preprocesses the trajectory data so that ordering operations and measures, such as distance and speed, are calculated beforehand and stored in the dataset as attributes.

The first operation in this step consists in traversing the point dataset $\mathbf{p}$ and, for each point in the same trajectory,

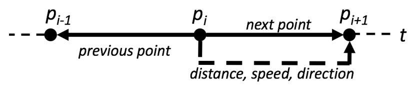


Fig. 3. For each point p_i in a trajectory t, the attributes *distance*, *direction* and *speed* are precomputed, and the references to its neighbors are kept for navigating between points during rendering.

determine the following attributes: current $speed$, $distance$ and $direction$ to the next point, previous neighbor p_{i-1}, next neighbor p_{i+1}, and unique identifier p_{id}; and add them to the point dataset (see Figure 3). The first three attributes are computed from the raw data (latitude, longitude, time stamp), and the last three attributes are used to facilitate the navigation between points. All attributes are stored in a hash map and indexed by p_{id}.

In this preprocessing step, a new trajectory dataset **t** is also created to reduce the complexity of changing the speed used in the visualization. The most important attributes of each element in **t** are the trajectory's identifier t_{id} and its starting time. Next, to know which objects are active at a specific time, the dataset is sorted by starting time and stored in a hash map indexed by t_{id}.

When dealing with batch-processed data, this algorithm has linear complexity on the number of points of the dataset **p** plus the cost to sort the trajectory dataset **t**. So, the total cost of this step is $O(n) + O(m\ \log(m))$, where $n = |\mathbf{p}|$ and $m = |\mathbf{t}|$. This step is also performed only once for the data selection. In the case of on-line applications, we assume that the trajectory points are received already sorted, and the attributes are already computed on the client side.

C. Normalization

To visualize hours or days of moving object data as an animation, a synchronized time control is necessary to view the animation at a higher speed, which provides an overview of the traffic flow and reduces the analysis time significantly. Also, pausing and rewinding the animation is essential to analyze a particular event in detail. For example, with basic map interaction such as zooming and panning, the analyst can focus on the place where the event is happening and view the objects moving at real world speed.

However, this action overloads the rendering process during the animation. For example, suppose an object whose position was captured every second. To visualize an animation that depicts the movement of this object with exhibition speed consistent with the real world speed on the map, it may be necessary to use spatial interpolation to generate intermediate points. Therefore, the slower the animation speed is, the more intermediate points are required, undermining the visualization's performance. On the other hand, the higher the animation speed is, the less points are interpolated, or even original data points can be discarded.

To solve both problems, we use the pose-to-pose principle of animation [31], which defines key frames and determines the animation speed by the number of frames. For this purpose, all trajectories must be normalized, i.e., for each trajectory, a new point corresponding to each key frame time is calculated by interpolation.

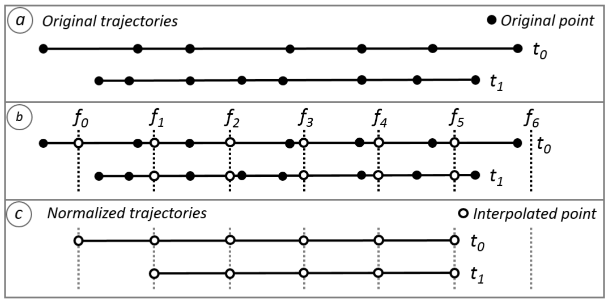


Fig. 4. Example of normalizing two trajectories t_0 and t_1. (a) Original trajectories with their own point frequency. (b) Computing the new points for each trajectory corresponding to key frame times f_i. (c) Normalized trajectories with points corresponding to the key frame times.

In the example of Figure 4, for each animation key frame time f_i, a new corresponding point is interpolated for each trajectory (t_0 and t_1). To simplify the interpolation process, the algorithm uses the point attribute $nextpoint$ to navigate through the trajectory's points. Finally, the frame data is ordered by the point attribute $time$ and stored in a queue **f** to optimize the rendering process explained in the next section.

For a batch-processed data application, this algorithm is executed only once for the dataset. The complexity is linear on the number of points of the dataset **p**. However, it is also necessary to order the frame dataset **f** to speed up the rendering step. Hence, the overall complexity is $O(n) + O(k\ \log(k))$ where $n = |\mathbf{p}|$ and $k = |\mathbf{f}|$. When we process a slow-speed animation, we have $k > n$, because points are interpolated. Otherwise, $k < n$, for high-speed animation.

For an on-line application, the normalization runs on-the-fly. The key frame times are synchronized by a heartbeat network protocol [32]. When necessary, the client side sends its location at a regular interval in the order of seconds. So, it is not necessary to sort the frame dataset. For reducing the communication between client and server, the algorithm uses a dead reckoning approach [33], in which the server side can estimate an object's current position by using previous information: location, speed and direction. Even in an on-line application, the frame data are kept for historical records, enabling visualization rewinding.

D. Rendering

One of the goals of our technique is to create a trail effect for each object so the analyst perceives the dynamics of an object's movement through the magnitude of its speed, through its direction and through its presence. An animated overview with multiple trails promotes the perception of the objects' flow evolution (see Figure 7-d). The final visual result was inspired by vector field pathlines [34], [35].

In this step, we want to render an image with a huge amount of objects and their respective trails at a minimal computational cost to compose a dynamic visualization. First, let us consider the image of a single object crossing a map

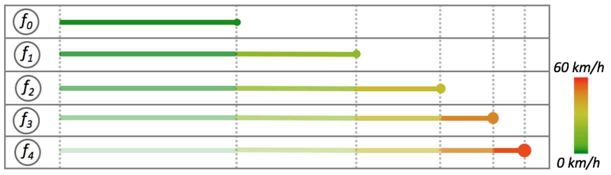


Fig. 5. Rendering an object's trail, frame by frame f_i. In this example, the object was at 60 km/h in frame f_0, then decelerated in frames f_2 to f_4.

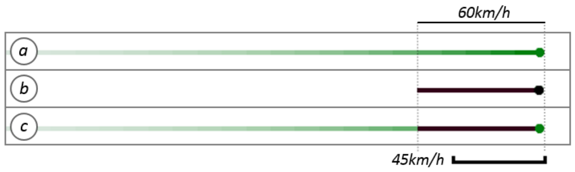


Fig. 6. Visual enhancements applied to an object's trail. (a) Original trail fading according to an exponential function. (b) The trail was cut proportionally to the object's speed and its color was modified. (c) a e b blended.

and leaving a trail that disappears according to an exponential function (Figure 5). In each animation frame f_i, the trail is extended by one line segment, while the opacity of all the lines already drawn is decreased. The opacity value for a trajectory point p_n is given by $\alpha(p_n) = \alpha_0 q^{n-1}$, where the factor q controls how much of the trail disappears per animation time frame. Thus, the greater the value of q is, the larger the length of the trail gets. There is a direct relation between the length of the trail and the corresponding object's speed. In Figure 5, for example, the object's speed in frame f_5 is low, because the trail's length between f_4 and f_5 is short. Besides trail length, we also encode speed magnitude as color and as the size of the marker.

Our technique uses the information in the frame dataset $\mathbf{f}$ and the object attributes in the point dataset $\mathbf{p}$ generated in previous steps to render all active objects in an animation frame at a low computational cost. The rendering process is computed in parallel on the GPU. We exploit the texture mapping capabilities of modern graphics hardware to optimize the process. We have developed texture-based primitives, such as line and circle, that increase the system's performance ten times when compared to common geometric primitives. Thus, we achieve high rendering performance at interactive frame rates. Furthermore, the algorithm has linear complexity on the number of active objects in a time frame. In this step, the same algorithm can be used without any changes in on-line as well as in batch-processed data applications.

E. *Visual Enhancements*

The speed and direction are essential attributes to get a better understanding of the traffic flow dynamics. This step highlights those attributes through a set of image processing operations, such as color correction, alpha cutting, and contrast. The analyst is supported by a preset of visual enhancements to highlight the object's speed's magnitude and direction. First of all, the object's trails are clipped to represent the current speed value of an object. Then, the opacity trail is thresholded by a parameter g, and the trail color is modified (Figure 6-b). The other visual enhancement consists of a combination of the two previous enhancements, so we get the full trail with highlighted speed magnitude (Figure 6-c). Furthermore, a speed reference scale is displayed on the map (Figure 6). The scale is calculated considering: the factor q (see Section III-D), the map's zoom level and the threshold g.

The visual enhancements are computed with shader programs that run directly on the GPU. We receive a texture in a frame buffer from the Rendering step and then the fragments' colors are filtered all at once. In our experiments, the computations to filter a frame buffer impact the graphics rendering performance only by 5%, approximately.

IV. Evaluation by Expert

We developed a visualization tool (Figure 1) to test our approach, and adopted a qualitative test methodology based on field observation [36]. According to this methodology, we presented four simulated scenarios to a specialist, and we asked him to answer two simple questions for each scenario: 1) is there any traffic anomaly? and 2) what was its cause? Also, during the evaluation, we asked the interviewee to describe the traffic flow dynamics.

The trajectory data for those scenarios were generated by a simulator [30] that uses three traffic flow models: the Intelligent-Driver Model to simulate the accelerations and braking decelerations of the drivers [37]; the MOBIL Model to allow vehicles to change lanes according to the safety and incentive criteria [38]; and the Boundary Conditions Model to exploit inflow and outflow conditions [39]. In the cited works, all those models were validated with empirical traffic observations.

We chose simulated scenarios for our evaluation for two reasons: to purposely start disturbances in the traffic flow and to have a controlled environment with no ambiguities. Thus, we could assess with certainty whether the specialist perceives when, where and why the anomaly occurred. For example, in a specific place and time, we can make a vehicle abruptly break or change lanes, causing a chain reaction, i.e., a traffic wave. Those simulations are widely used in Traffic Engineering to investigate the effects of new traffic policies before putting them into operation.

For the evaluation, we selected a specialist who has a master's degree in Transportation Studies and a doctorate in Urban Planning Studies. Nowadays, he is a university associate professor and teaches an Urban Planning course. Besides his academic experience, he also has more than ten years of experience in urban intervention projects, such as tunnels and overpasses, for improving the traffic flow of a big city.

At the beginning of the evaluation, the specialist freely used the visualization tool so that he became familiar with its functionality. At that moment, we used a real trajectory dataset of two big cities captured by taxi GPS sensors. After

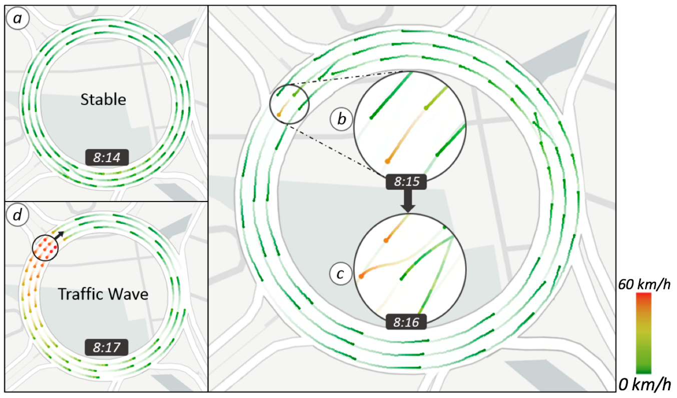


Fig. 7. Circular road scenario. (a) Dense and stable traffic without congestions. The vehicles are moving at a speed close to the limit. (b) A car breaks abruptly. (c) Vehicles following close behind the breaking car change lanes propagating the perturbation to the other lanes. (d) Then, a traffic wave is formed and moves in the opposite direction of the flow.

that, we presented to him four simulated scenarios: circular road, on-ramp, lane closure and uphill gradient.

A. Circular Road

The circular road is a simulation with three-lane traffic in a closed system. This is the simplest scenario since it primarily depends on the speed limit and the road density. Urban planners normally use it to test the optimal speed limit for a given road density [40]. Besides, this scenario is very effective for analyzing the effects and the behavior of traffic waves [7].

Although the simulation prepared for this scenario was a two-hour-long simulation, it could be fast forwarded to be visualized in a few minutes. The first hour depicts a dense and stable traffic without congestions of 20 vehicles/km/lane and a speed limit of 60 km/h (Figure 7-a). At the beginning of the second hour, a car breaks abruptly (Figure 7-b). Then the vehicle just behind has to brake as well to maintain the safety distance. As a consequence, the next vehicle behind has to brake even more and so on. This disorder destabilizes the traffic flow of all lanes causing a traffic wave (Figure 7-d).

In the test, the specialist promptly identified the place and the car that caused the disturbance. His evaluation mainly consisted of observing changes in trail length, color and marker size (Figure 7-b). After identifying the place of disturbance, the specialist fast rewound the visualization to initiate a more detailed exploration by also zooming-in on the map, slowing down the animation and changing the visual enhancement to estimate the car speed before and after breaking abruptly. He said that the disturbance propagation occurs because of the lane changes of vehicles following close behind (Figure 7-c). He also perceived that the disorder was dying out in this simulation, i.e., the traffic flow would return to the initial equilibrium, although he did not see the visualization until that moment.

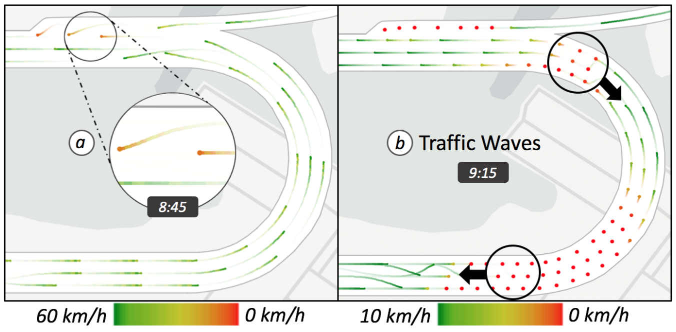


Fig. 8. On-ramp scenario. (a) Vehicles access the main road and cause a traffic wave. (b) The on-ramp access generated two traffic waves. In this visualization, we used a color map to depict a speed range of 0 to 10 km/h.

B. On-Ramp

The on-ramp scenario is an open system that simulates possible traffic bottlenecks caused by an access ramp [41]. In fact, the on-ramp concept poses some situations in which vehicles access an already busy road, e.g., an access exit of a mall or an intersection. This scenario is more tricky than the circular road because one must consider the inflow of the main and of the access roads.

This test was generated from a two-hour-long simulation to evaluate the impacts of vehicles accessing a road in heavy traffic condition, but with constant inflow and outflow of 3,600 vehicles/h with speed limit of 80 km/h. After 30 minutes from the simulation start, we added an on-ramp inflow of 20% of the main inflow, i.e., 720 vehicles/h. Instantaneously traffic waves emerged on the main road close to the on-ramp region. After one hour of simulation, we reduced the main road inflow to less than half of the beginning inflow (1,600 vehicles/h).

The specialist pointed out the place and the moment that the first traffic wave emerged and which cars caused it (Figure 8-a). He affirmed that the dynamics of traffic wave propagation is similar to the circular road scenario, i.e., lane changes are the primary cause of it. However, on-ramp accesses act as a stationary generator of the waves in these cases. For example, Figure 8-b depicts two waves. He also noticed the main road inflow reduction after an hour of simulation, but he said that in these cases the traffic jam takes too long to disappear. In fact, after one hour of simulation, the traffic jam did not dissolve even with a low inflow. Finally, he named the on-ramp effects of traffic frictions and affirmed that one of the biggest challenges for urban planners is estimating the impacts of those effects.

C. Lane Closure

The lane closure is a scenario that reproduces the effects of unusual events occurring on a lane, like road works and accidents or bad street conditions [7]. In this open system, urban planners can simulate the propagation of traffic waves caused by these events.

For this simulation, one of the two lanes of a road is blocked. In the first hour, traffic normally flowed with an inflow of 1,500 vehicles/h and a speed limit of 60 km/h.

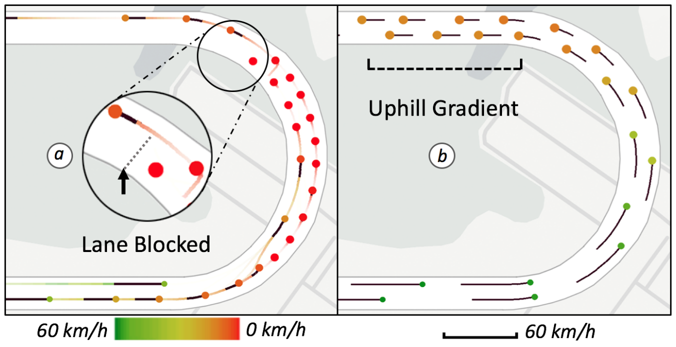


Fig. 9. (a) Lane closure scenario. The inner lane is closed without any block sign on the map. (b) Uphill gradient scenario. The heavy vehicles had their speed reduced significantly impairing the traffic flow in the uphill section.

Then, we increased the speed limit to 80 km/h. After that, the simulation lasted for another hour.

Even without any lane block sign on the map, the specialist noticed the problem naturally, due to the vehicles' behaviors (Figure 9-a). He said that the trails and colors helped to characterize those behaviors. Although he perceived the increase of the cars' speed, he did not associate it with speed limit growth. However, our change of speed limit was the main reason for the flow destabilization, because the higher the speed is, the greater the difficulty for a driver to change lanes becomes. Then, many vehicles failed to change lanes.

D. Uphill Gradient

The uphill gradient scenario simulates the impacts of heavy vehicles such as buses and trucks on the traffic flow. This scenario differs fundamentally from previous ones by flow-conserving bottlenecks [37].

We use a two-lane road with a relatively steep uphill gradient to simulate a very light traffic with an inflow of 1,900 vehicles/h, a speed limit of 80 km/h and 30% of heavy vehicles. In this simulation there were no anomalies, i.e., the traffic flowed normally throughout the simulation. Nevertheless, in the uphill section, the heavy vehicles had their speed reduced significantly impairing the speed of the other vehicles (Figure 9-b).

The specialist did not find out the reason of speed reduction. He supposed it could be a difficulty due to the curve or the bad condition of the road. When we told him that 30% of the vehicles were heavy, he suggested that we highlighted them with a different shape.

E. Specialist Feedback

The specialist recognized the potential of our approach to visualize traffic oscillation patterns from trajectory data. He considered the visualization to be pleasant and that it facilitated the perception of traffic waves. Also, he stated that the trail was an excellent indicator to visualize lane changes, which are one of the leading causes of disturbances in dense traffic. The time control of the visualization, in particular, the rewinding functionality received very positive comments.

He saw that our proposal could help significantly in studies that analyze the impacts of constructing new buildings on the road network. For example, constructing a new mall in a crowded region could cause traffic jams throughout the neighborhood. The majority of the urban planners commonly uses numerical simulators and statistical tools. Thus, he believed that our method could help these planners to make city managers aware of these impacts. For on-line applications, he considered that our method could be useful to predict traffic jam in real time.

Regarding the limitations of our visualization, the specialist noticed the lack of statistical graphs to depict the traffic flow of the roads. In the visualizations of the real trajectory data, i.e., the taxi databases, he considered that it was not possible to visualize oscillations in these databases, only an overview of the traffic flow because taxis are only a sample of the real traffic. He concluded that our approach was more appropriate to visualize data from traffic sensing devices such as RFID tags, video cameras, and laser scanners, that collect data from the entire traffic flow. Maybe that may change, in the near future, when all vehicles would be tracked. Moreover, he believed our visualization would have excellent results in simulations.

V. Our Approach's Performance

In this Section, we describe the implementation of our proposed approach, showing that we can animate the trajectories at interactive rates. The prototypes were implemented using JavaScript and WebGL for graphics on the GPU. For map interaction, we use an API based on OpenStreet Maps. The development platform used by our work is free and Web-based. The pictures and performance tests in this paper have been generated on a workstation with an Intel Core i5 CPU at 3.4 GHz with 16 GB of RAM and a NVIDIA GeForce GTX 1070 with 8 GB of video memory.

In the first prototype, we simultaneously animated 160,000 graphic elements at 20fps. That prototype has been optimized using sprites instead of geometry to minimize memory and processing costs. The final implementation of the technique provides many customization parameters, such as marker size, speed limit, trail width, visual enhancement and threshold factor q (Section III-D).

For performance testing purposes, we use a real-life GPS dataset donated by a taxi company. This dataset was tracked by taxis at the sampling rate of one second. The dataset consists of a day with 3.2 million observations. The raw data came in a 1.3 GB CSV file. The Cleaning step took about 20s, resulting in more than 60% reduction of the number of points in the dataset. This remarkable reduction was because the algorithm discarded several points when the taxis were parked, probably waiting for passengers. The Preprocessing step took about 10s, and the dataset storage size increased 20% with the new attributes. The Normalization step took about 3s and generated almost seven million frames for the real time speed, occupying 980MB of RAM. In this test, we had at most 1,900 taxis moving at the same time. So, the Visualization phase ran without bottlenecks at 30 fps and 60 fps.

VI. Conclusion

In this paper, we presented a novel approach for visualizing traffic oscillation patterns by visualizing the objects movement in space over time. We proposed a method inspired by vector field visualization to visualize the movement of objects dynamically changing over time; an algorithm to control and synchronize the visualization time; a systematic stepwise methodology for exploring sensing devices data; and a visualization tool that computes the trajectory data in parallel on the GPU at interactive frame rates. Moreover, our approach was designed to support both batch-processed and streaming data applications. We also presented the benefits and limitations of our visualization proposal based on domain expert feedback. Finally, we presented performance tests with very encouraging results to support our approach.

This work was the first step towards an end-to-end solution for the visual analysis of mobility dynamics from raw trajectory data. As future work, we plan to integrate our visualization technique into a complete visual analytics tool including support for spatiotemporal queries and statistical graphs. Next, we will test the integrated system in a field study with data from traffic sensing devices, like video cameras. We will also investigate dynamic visualizations for hot motion paths [13] in an on-line application to predict traffic jam in real time. These studies will help us to devise other means to improve the perception of object movement dynamics.

References

[1] Y. Zheng, W. Wu, Y. Chen, H. Qu, and L. M. Ni, "Visual analytics in urban computing: An overview," *IEEE Transactions on Big Data*, vol. 2, no. 3, pp. 276–296, 2016.

[2] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM TIST*, vol. 5, no. 3, p. 38, 2014.

[3] N. Andrienko and G. Andrienko, *Exploratory analysis of spatial and temporal data: a systematic approach*. Springer Science & Business Media, 2006.

[4] D. Guo and X. Zhu, "Origin-destination flow data smoothing and mapping," *IEEE TVCG*, vol. 20, no. 12, pp. 2043–2052, 2014.

[5] C. Hurter, O. Ersoy, S. I. Fabrikant, T. R. Klein, and A. C. Telea, "Bundled visualization of DynamicGraph and trail data," *IEEE TVCG*, vol. 20, no. 8, pp. 1141–1157, 2014.

[6] T. Von Landesberger, F. Brodkorb, P. Roskosch, N. Andrienko, G. Andrienko, and A. Kerren, "Mobilitygraphs: Visual analysis of mass mobility dynamics via spatio-temporal graphs and clustering," *IEEE TVCG*, vol. 22, no. 1, pp. 11–20, 2016.

[7] M. Treiber and A. Kesting, *Traffic Flow Dynamics: Data, Models and Simulation*. Springer, 2013.

[8] X. Li, J. Cui, S. An, and M. Parsafard, "Stop-and-go traffic analysis: Theoretical properties, environmental impacts and oscillation mitigation," *Transportation Research Part B*, vol. 70, pp. 319–339, 2014.

[9] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, *Mastering the information age solving problems with visual analytics*. Eurographics Association, 2010.

[10] N. Andrienko and G. Andrienko, "Visual analytics of movement: An overview of methods, tools and procedures," *Information Visualization*, vol. 12, no. 1, pp. 3–24, 2012.

[11] G. Andrienko, N. Andrienko, U. Demsar, D. Dransch, J. Dykes, S. I. Fabrikant, M. Jern, M. Kraak, H. Schumann, and C. Tominski, "Space, time and visual analytics," *IJGIS*, vol. 24, no. 10, pp. 1577–1600, 2010.

[12] HINT.FM. (2017) Wind map. [Online]. Available: hint.fm/wind

[13] D. Sacharidis, K. Patroumpas, M. Terrovitis, V. Kantere, M. Potamias, K. Mouratidis, and T. Sellis, "On-line discovery of hot motion paths," in *Advances in database technology*. ACM, 2008, pp. 392–403.

[14] Y. Zheng, "Trajectory data mining," *ACM TIST*, vol. 6, no. 3, pp. 1–41, may 2015.

[15] W. Kresse and D. M. Danko, *Springer Handbook of Geographic Information*. Springer, 2012.

[16] W. Chen, F. Guo, and F.-Y. Wang, "A survey of traffic data visualization," *IEEE TITS*, vol. 16, no. 6, pp. 2970–2984, 2015.

[17] G. Andrienko, N. Andrienko, P. Jankowski, D. Keim, M.-J. Kraak, A. M., and S. Wrobel, "Geovisual analytics for spatial decision support: Setting the research agenda," *IJGIS*, vol. 21, no. 8, pp. 839–857, 2007.

[18] G. D. Lorenzo, M. Sbodio, F. Calabrese, M. B., F. Pinelli, and R. Nair, "AllAboard: visual exploration of cellphone mobility data to optimise public transport," *IEEE TVCG*, vol. 22, no. 2, pp. 1036–1050, 2016.

[19] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. T. Silva, "Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips," *IEEE TVCG*, vol. 19, no. 12, pp. 2149–2158, 2013.

[20] R. Scheepens, C. Hurter, H. V. D. Wetering, and J. J. V. Wijk, "Visualization, selection, and analysis of traffic flows," *IEEE TVCG*, vol. 22, no. 1, pp. 379–388, 2016.

[21] Z. Wang, M. Lu, X. Yuan, J. Zhang, and H. Van De Wetering, "Visual traffic jam analysis based on trajectory data," *IEEE TVCG*, vol. 19, no. 12, pp. 2159–2168, 2013.

[22] G. Andrienko, N. Andrienko, and S. Wrobel, "Visual analytics tools for analysis of movement data," *ACM SIGKDD Explorations Newsletter*, vol. 9, no. 2, pp. 38–46, 2007.

[23] M.-J. Kraak, "The space-time cube revisited from a geovisualization perspective," in *21st Cartographic Conference*, 2003, pp. 1988–1996.

[24] C. Tominski, H. Schumann, G. Andrienko, and N. Andrienko, "Stacking-based visualization of trajectory attribute data," *IEEE TVCG*, vol. 18, no. 12, pp. 2565–2574, 2012.

[25] C. Tominski and H.-J. Schulz, "The great wall of space-time," 2012.

[26] T. Munzner, *Visualization analysis and design*. CRC Press, 2014.

[27] H. Guo, Z. Wang, B. Yu, H. Z., and X. Y., "Tripvista: Triple perspective visual trajectory analytics and its application on microscopic traffic data at a road intersection," in *PacificVis*. IEEE, 2011, pp. 163–170.

[28] J. Poco, H. Doraiswamy, H. Vo, J. L. Comba, J. Freire, C. Silva *et al.*, "Exploring traffic dynamics in urban environments using vector-valued functions," in *CGF*, vol. 34, no. 3. Wiley O. Library, 2015, pp. 161–170.

[29] Uber. (2017) Deck.gl. [Online]. Available: uber.github.io/deck.gl

[30] M. Treiber. (2017) Microsimulation of traffic flow. [Online]. Available: www.traffic-simulation.de

[31] F. Thomas, O. Johnston, and F. Thomas, *The illusion of life: Disney animation*. Hyperion New York, 1995.

[32] M. Kawazoe Aguilera, W. Chen, and S. Toueg, "Heartbeat: A timeout-free failure detector for quiescent reliable communication," *Distributed algorithms*, pp. 126–140, 1997.

[33] C. Murphy, "Believable dead reckoning for networked games," *Game engine gems*, vol. 2, 2011.

[34] M. Zockler, D. Stalling, and H.-C. Hege, "Interactive visualization of 3d-vector fields using illuminated stream lines," in *Visualization'96. Proceedings*. IEEE, 1996, pp. 107–113.

[35] C. Beccario. (2017) Earth wind map. [Online]. Available: http://earth.nullschool.net

[36] D. E. Polkinghorne, "Language and meaning: Data collection in qualitative research." *Journal of counseling psychology*, vol. 52, no. 2, p. 137, 2005.

[37] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[38] M. Treiber and D. Helbing, "Realistische mikrosimulation von strassenverkehr mit einem einfachen modell," in *16th Symposium Simulationstechnik ASIM*, vol. 2002, 2002, p. 80.

[39] D. Helbing, A. Hennecke, V. Shvetsov, and M. Treiber, "Micro-and macro-simulation of freeway traffic," *Mathematical and computer modelling*, vol. 35, no. 5-6, pp. 517–547, 2002.

[40] D. Ni, *Traffic Flow Theory: Characteristics, Experimental Methods, and Numerical Techniques*. Butterworth-Heinemann, 2015.

[41] H. Lee, H.-W. Lee, and D. Kim, "Dynamic states of a continuum traffic equation with on-ramp," *Physical Review E*, vol. 59, no. 5, p. 5101, 1999.