DOI: 10.1111/cgf.12628
Eurographics Conference on Visualization (EuroVis) 2015
H. Carr, K.-L. Ma, and G. Santucci
(Guest Editors)

Volume 34 (2015), Number 3

Exploring Traffic Dynamics in Urban Environments Using Vector-Valued Functions

Jorge Poco[1], Harish Doraiswamy[1], Huy. T. Vo[1], João L. D. Comba[2], Juliana Freire[1], and Cláudio. T. Silva[1]

[1] New York University, USA [2] Instituto de Informática, UFRGS, Brazil

Abstract

The traffic infrastructure greatly impacts the quality of life in urban environments. To optimize this infrastructure, engineers and decision makers need to explore traffic data. In doing so, they face two important challenges: the sparseness of speed sensors that cover only a limited number of road segments, and the complexity of traffic patterns they need to analyze. In this paper we take a first step at addressing these challenges. We use New York City (NYC) taxi trips as sensors to capture traffic information. While taxis provide substantial coverage of the city, the data captured about taxi trips contain neither the location of taxis at frequent intervals nor their routes. We propose an efficient traffic model to derive speed and direction information from these data, and show that it provides reliable estimates. Using these estimates, we define a time-varying vector-valued function on a directed graph representing the road network, and adapt techniques used for vector fields to visualize the traffic dynamics. We demonstrate the utility of our technique in several case studies that reveal interesting mobility patterns in NYC's traffic. These patterns were validated by experts from NYC's Department of Transportation and the NYC Taxi & Limousine Commission, who also provided interesting insights into these results.

1. Introduction

Data captured in urban environments provide valuable information about the behavior of many components of a city. The analysis of such data has the potential to derive knowledge that can be used to make cities more efficient, as well as inform policies and planning decisions. Traffic is a key component of an urban ecosystem.

To understand and optimize the traffic infrastructure, urban planners need to explore and analyze traffic patterns from historic data over different periods of time and in different parts of the city. Questions pertaining to traffic patterns in a city can be broadly categorized as *scalar-based* and *mobility-based* tasks. Scalar-based questions involve a fixed property of the traffic such as speed and density of traffic. Tasks of interest from this category include exploring how traffic speeds vary throughout a city during different times over different days. Mobility-based tasks, on the other hand, involve studying the flow of traffic along various streets of the city. These include exploring the flow of slow-moving traffic, free-flowing traffic, and direction of traffic. Additionally, in order to ensure that a proposed change to this infrastructure does not have adverse affects, they should also be able to simulate traffic dynamics under various constraints. But doing so is challenging for many reasons, in particular, the sparseness of traffic data that is captured and the complexity of the analyses that need to be carried out.

Traffic data is often obtained from traffic cameras or fixed readers (e.g., EZ-pass). However, only a small number of these devices are deployed in practice. GPS-tracked vehicles are another potential source of traffic information. A subset of these sensors are already being used by popular map services such as Google maps and Apple maps to provide real-time traffic information to users. However, their coverage is incomplete and limited to segments of major roads, and hinders the analysis as well as the accuracy of derived models.

While tracking all vehicles is not feasible, it is possible to track an important subset: taxis. Taxi fleets in many cities are equipped with GPS. Consider, for example, New York City (NYC): 13,000 taxis make, on average, 500,000 trips and carry over 1 million passengers every single day; totaling roughly 170 million trips per year. Given this high penetration rate of taxis in large cities, it is therefore reasonable to assume that the taxis can be used as probe vehicles, and taxi movement and travel times are representative of the overall traffic and provides a broad coverage of the city in space and time [ZHUK13]. Unfortunately, taxi data captured by the NYC Taxi & Limousine Commission contains neither the location of the taxis at regular intervals nor

the route taken—they only contain information about locations and times for pickups and dropoffs. Thus, an important question is whether it is possible to derive accurate traffic information from these data.

Exploring traffic mobility dynamics through visualization is also a challenging task. Previous works have predominantly focused on the global movement of objects, and proposed techniques for visualizing trajectories either directly on a map, or through the use of specialized metaphors like the space-time cube and flow map [AA13]. While these techniques are effective for analyzing object movement, they do not capture the local mobility dynamics resultant from their collective motion. Furthermore, in the presence of a large number of objects, such visualizations quickly get cluttered.

Vector field visualization techniques [MLP*10a], on the other hand, are capable of visualizing localized flow and have been effectively used in various applications in scientific visualization and computer graphics [BZK09,CEW*08, ZMT06]. Given that the traffic movement can be considered as a "flow" over the road network, we propose to adapt techniques from vector field visualization to help answer the above mentioned mobility-based tasks. However, they require the underlying domain on which the flow is computed to be continuous (the domain is usually a subset of an Euclidean space). But the road network of a city on which the traffic flows is a graph, which is a discrete domain. Traditional vector field techniques therefore cannot be directly applied to visualize traffic dynamics.

Contributions. We propose a new approach to support interactive analysis and visualization of traffic mobility dynamics. First, we overcome the sparseness of sensors with a scalable model that takes as input data about taxi trips and derives the required traffic information. The model relies on an efficient data-driven closest path algorithm to compute plausible routes for trips, which are subsequently used to infer traffic speed. We validated our model against speed values obtained from EZ-pass tag readers, and the results show that our model derives accurate predictions, outperforming existing models.

To support interactive exploration of the traffic data, the predicted traffic speed for all roads is then used to derive a traffic function, which corresponds to a time-varying probabilistic vector-valued function defined on the graph representing the road network. We adapt two well-known vector field visualization techniques, particle advection techniques and global techniques, to visualize local traffic flow and to provide global traffic overviews, respectively. We evaluate our approach through a study of traffic dynamics in NYC. We present use cases that show how our technique can be used to accomplish various analysis tasks, including traffic mobility visualization, identification of traffic bottlenecks, and simulation of traffic flow. As part of this evaluation, we sought feedback from experts from the NYC Department of Transportation (DoT) and the Taxi & Limousine Commission (TLC), who also used our technique to study specific cases that were of interest to them.

2. Related Work

In this section we discuss related work in two categories: methods that infer traffic information from movement data, and those that visualize these data. We also review vector field techniques commonly used in scientific visualization.

Traffic Data. Traffic data can be acquired in different ways, for example, from road sensors, video captured at roadside locations, or floating sensors such as GPS-equiped vehicles. The use of video images captured from specific roadside locations allows the use of image processing techniques for tracking vehicle movements in different frames, as well as estimation of mean traffic speed [DCPun]. Several papers use data captured from GPS devices to estimate routes and compute traffic speed. Traces can be collected from GPS-enabled taxis at different sampling speeds, ranging from short time intervals (seconds to minutes) to those captured only at the source and destination, like the NYC taxi data. The usual procedure to identify routes taken by a taxi (or any other vehicle) is to use a map-matching approach, which maps the best possible route taken by the taxi using the locations of the taxi during the trip. We refer the reader to the survey by Quddus et al. [QON07] for a detailed discussion on various map-matching algorithms. Zou et al. [ZmXZ05] presented a technique for studying speed in arterial roads using GPS locations and speeds recorded at regular sampling periods during rides. Lou et al. [LZZ*09] proposed a low-sampling-rate map matching algorithm, which only considers temporal and spatial constraints on the trajectories. They defined the sampling rate to be low if there is only one sample for every two minutes. Since the NYC taxi data does not have any samples from a trip except for its start and end locations, these methods cannot be directly applied. Recently, Zhan et al. [ZHUK13] formulated the problem of identifying traffic speeds on different streets using the NYC taxi data as a non-linear optimization problem. Their approach takes 15 minutes to solve this optimization on a region involving a small region in Midtown Manhattan (5% of the roads in Manhattan). Moreover, they only compute the average hourly speeds of the streets using trips corresponding to a week. This approach is clearly not tractable to be able to use all the trips over entire Manhattan. Santi et al. [SRS*13] assumed that taxis choose the shortest path, and used Dijkstra's algorithm to compute their paths. As we show later, since taxis seldom take the shortest path, such an assumption is likely to result in inaccurate estimation of the traffic speeds. Note that neither of these models was validated against actual traffic speeds.

Density-map based visualizations [WVDWVW09, SWvdW*11], which are used for visualizing object movement, can be used to study patterns of taxi trips. However, they will not be useful for studying the localized traffic flow patterns resulting form their combined movement. Visualization of traffic data is often accomplished through the use of information visualization techniques. Tominski et al. [TSAA12] use stacked views to display the attributes associated with known trajectories. More recently,

Ferreira et al. [FPV*13] and Wang et al. [WLY*13] proposed visualization tools for the exploration of transportation-related data. Other techniques used in the visual analytics of movement data along with their attributes are described in the survey by Andrienko et al. [AA13]. All of these techniques focus on the global movement of objects. To the best of our knowledge, our work is the first to use vector field based techniques to address this problem.

Vector Field Visualization. Vector field and flow visualization has been an active field of research for over two decades. We refer the readers to the following surveys [LHD*04, MLP*10a, PVH*03] for detailed literature reviews. More recently, GPUs have been extensively used for interactive visualization of particle flows [KKKW05, KKW05, BSK*07, BKKW08]. Topology-based methods have also been used to identify features and visualize vector fields [PPF*11].

We are interested in the flow of traffic over the road network. Existing vector field methods use an Euclidean geometric domain, and compute flows over a continuous domain. Since traffic data is modeled as a vector-valued function over a discrete directed graph, these methods cannot be directly applied. As far as we know, there has been no work that applied vector field methods over discrete domains.

3. Background

We now discuss definitions and visualization techniques related to vector fields. More detailed discussions on these concepts can be found in [Baj99, Tel08, MLP*10b].

3.1. Vector Field

An n-dimensional vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ is a n-tuple where $v_i \in \mathbb{R}$. An n-dimensional vector field is a vector-valued function $f : D \rightarrow \mathbb{R}^n$ that maps a geometric domain to an n-dimensional vector. The geometric domain D is usually a subset of the Euclidean space, $D \subseteq \mathbb{R}^n$. Fig. 1 shows two examples of a 2-dimensional vector field. Fig. 1(a) shows an example of a stationary vector field, i.e., a vector field that doesn't change over time. Fig. 1(b) shows an example of a time-varying vector field.

In this work, we are interested in modeling the flow of traffic in a city. To do this, we represent the geometric domain as a directed graph, and define a vector-valued function on the edges of the graph. The direction of the vector is the same as the direction of an edge. Also, each directed edge is also associated with a real value denoting the number of taxis that ply on the road corresponding to that edge.

3.2. Visualization of vector fields

Particle advection techniques used to visualize 2D vector fields introduce an abstract weightless particle into the vector field and visualize the effects of the vector field on this particle. Consider a particle at location $\mathbf{l}_t = (x, y)$ at time t. Let its velocity at this location and time be $\mathbf{v}_t$. After a time dt, the location of the particle is given by $\mathbf{l}_{t+dt} = \mathbf{l}_t + dt \cdot \mathbf{v}_t$. Integrating the above differential equation from current time to a later time t' gives us the location of the point at time t'.

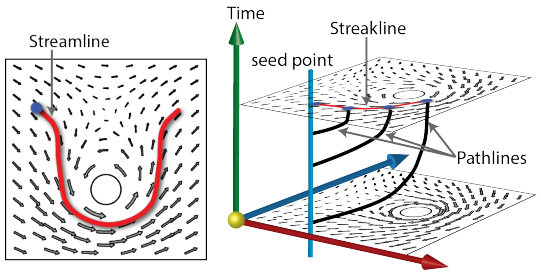

Figure 1: *Particle advection techniques.* ***(a)*** *Stationary vector field. The red streamline is computed starting from the blue seed point.* ***(b)*** *Time-varying vector field. The three black lines correspond to pathlines starting at different time steps. The red streakline is the locus of a stream of particles injected into the vector field.*

The aspect of the particle that is tracked defines the type of a field line. Field lines are typically used to visualize and analyze the history of flow from a given location(s) of interest. We are mainly interested in three types of field lines:

Path lines. These track the path of a weightless particle in a vector field. The three black lines in Fig. 1(b) are pathlines that track a particle starting at different time steps.

Stream lines. These lines track the path of the weightless particle in a stationary vector field. When used for a given time instant of a time-varying vector field, they essentially show the direction of the particle movement for that instant. The red line in Fig. 1(a) shows a streamline that tracks a particle starting at the blue seed point.

Streak lines. Consider a continuous release of particles into the vector field. The locus of these particles forms a streak line. The red line in Fig. 1(b) represents one such streakline.

Later in Section 5, we formally define the traffic function, and describe how we adapt these techniques to support its visualization.

4. Traffic Model

The NYC taxi data gathered by TLC consists of GPS-derived start and stop times and locations, distance traveled, trip duration, as well as fare and tip. The data set we obtained contains information for roughly 520 million taxi trips during 2009, 2011 and 2012. The road network of NYC is represented as a directed graph $G(V, E)$. Each directed edge $e \in E$ represents a segment of a road, and each node $v \in V$ represents the intersection point of two or more roads. When a road allows traffic flow in both directions, there are two directed edges corresponding to that road. The length of each road segment e_i is given by d_i. Let t_i denote the time taken for a vehicle to go from one intersection point to another along the road segment represented by e_i. We assume that the origin and destination of all taxi trips correspond to nodes of this graph. In case a trip begins or ends in the middle of a street, we approximate the corresponding location to the nearest intersection node.

Given the actual distance $dist_T$, and time $time_T$ for each trip T, we compute the average road speed of traffic in two stages: (1) Identify the plausible routes for every trip; and (2) Use the identified routes to infer the speed of traffic along various roads. In this section, we first describe the closest path algorithm used to identify plausible routes for a taxi trip. Next, we explain the procedure to infer the speed of traffic using the computed routes. Finally, we validate our model with the available albeit partial traffic data.

4.1. Computing Closest Paths

Problem motivation. The path taken by a taxi could be approximated using the shortest path between the origin and destination. However in practice, the taxi driver does not consider the shortest path for a majority of the trips. This is reflected in the input data, where we observe that the length of the shortest path does not correspond to the actual distance of a given trip. For example, in a sample 1 week period, 56% of trips had a total length greater than that of the shortest path between source and destination. So, rather than using the shortest path to identify the route taken by a taxi, we are interested in identifying paths whose lengths are closest to the actual distance of a given trip.

Formally, given a source v_{st} and destination v_{en} of a trip T, our goal is to find k paths between v_{st} and v_{en} having length *closest* to $dist_T$. A brute force method of going through all possible paths between v_{st} and v_{en} is not practical given the size of the search space, which is exponential in the size of the road network. We therefore need to use a heuristic to prune and traverse through the search space. One possible method is to find a set of k' shortest paths, for a large enough k' such that the length of the k'^{th} path is closest to $dist_T$. This can be accomplished using the Yen's ranking loopless shortest paths algorithm [Yen71] which inductively computes the i^{th} shortest path between two nodes using the common subpaths of the $(i-1)$-shortest paths. The algorithm uses the result of the shortest path as starting point, followed by a relaxation procedure until the distance constraint is met.

Unfortunately, due to a grid-like structure of the road network in most part of NYC, Yen's approach is not practical, since it has to compute and discard a large number of suboptimal paths. In our experiments, this number was as high as $10,000$ in many cases (see Appendix A).

Closest path algorithm. We propose an alternate heuristic-based solution, which can efficiently compute a set of top-k closest paths by breaking the problem into a set of shortest-path queries. Our intuition is that when a taxi driver chooses to take a path that is not the shortest path due to traffic conditions, passenger requests, etc., he would still have to pass through a set of way-points along this route having the following property: the path taken between any two consecutive way-points is the shortest path between the two points. We use the number of such way-points n as a heuristic to prune the search space. We now redefine the closest paths problem as follows. Given a road network $G(V,E)$, a pair of source-destination nodes (v_{st}, v_{en}), and a distance $dist_T$, find k sets of way-points $W_n = \{w_1, w_2, ..., w_n | w_i \in V\}$ such that the length of the shortest paths from v_{st} to v_{en} that passes through all nodes in W_n are closest to $dist_T$:

$$\arg\min_{W_n} |l_{v_{st},w_1} + \sum_{i=1}^{n-1} l_{w_i,w_{i+1}} + l_{w_n,v_{en}} - dist_T|$$

where $l_{a,b}$ is the length of the shortest path from a to b. This formulation reduces the size of the search space to $O(|V|^n)$ paths. In our application, we use $n = 1$ in order to reconstruct the taxi paths. Thus, our problem is reduced to find node w that minimizes the following:

$$\arg\min_{w} |l_{v_{st},w} + l_{w,v_{en}} - dist_T|$$

Implementation and efficiency. To efficiently execute a closest path query, we first pre-compute the shortest paths between all pairs of vertices of the road network. Since the road network is a planar graph, $|E| = O(|V|)$, this pre-computation can be accomplished in $O(|V|^2 \log |V|)$ time (one Dijkstra computation is done for each vertex of the network). Using this pre-computed data, the closest path query can be answered in $O(|V|)$ time for each pair (v_{st}, v_{en}).

Note that Yen's ranking algorithm has quadratic time complexity of $O(k|V|(E+|V|\log|V|))$. Comparing to an implementation of the Yen's ranking loopless paths algorithm by Martins and Pascoal [ksp, MP03], our technique was over 600x faster. We were able to compute top-k closest paths for all taxi trips in a month in under 15 minutes while the same task would take the other method over a week to complete. This test was performed on an 8-core machine.

4.2. Closest Path Traffic Model

In this model, we first select k paths whose distance is closest to the actual distance of the trip. Unless otherwise mentioned, we use $k = 20$ throughout the paper. Let $\{T_1, T_2, \ldots, T_k\}$ denote the k-closest paths corresponding to a trip T. Consider one such possible path T_i, and let the corresponding trip T take time $time_T$. Let the start and end positions of this trip be v_{st} and v_{en}, respectively. This path can be represented as a set of nodes of the graph G as $T = \{e_{i_1}, e_{i_2}, \ldots, e_{i_l}\}$. We now have the relation: $t_{i_1} + t_{i_2} + \ldots + t_{i_l} = time_T$. We assume that the time taken to travel a road segment is proportional to its length, that is, we use the average speed of a trip to approximate the speed of each road segment that is part of that trip. Then, the time taken for each road segment for this trip is given by

$$t_{i_k} = \frac{d_{i_k}}{\sum_{j=1}^{l} d_{i_j}} \times time_T \qquad (1)$$

Let $closeness_i = |dist_T - dist_{T_i}|, \forall i \in [1,k]$. We then define a weight for each possible path $w_i = \frac{1}{closeness_i + \varepsilon}$, where ε is a small constant used to avoid division by zero. That is, the weight of a possible path is inversely proportional to its closeness to the actual distance.

Using the above formulation for the trips that happen within the time period of interest, we get a set of equations corresponding to each of the trips. Let n be the number of

trips that passes through road segment e_i in a given time interval. Solving Equation 1 for the n trips, we get a set of values for $t_i = \{t_i^1, t_i^2, \ldots, t_i^n\}$.

The possible speeds $s_i = \{(s_i^1, w_i^1), (s_i^2, w_i^2) \ldots, (s_i^n, w_i^n)\}$ of each road segment e_i are then computed, where $s_i^x = \frac{d_i}{t_i^x}$. Note that each speed value in this set is also associated with a weight equal to the weight of the path that was used to obtain that speed value. The weighted mean $\bar{s}_i$ and variance σ_i^2 is computed as follows [Wes79]:

$$\bar{s}_i = \frac{\sum_{j=1}^{n} s_i^j \times w_i^j}{\sum_{j=1}^{n} w_i^j} \qquad \sigma_i^2 = \frac{\sum_{j=1}^{n} (s_i^j - \bar{s}_i)^2 \times w_i^j}{\frac{n-1}{n} \sum_{j=1}^{n} w_i^j}$$

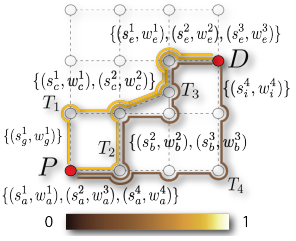

Figure 2: *The closest path model chooses the k-closest paths for every trip. This example uses $k = 4$.*

Fig. 2 illustrates this procedure. For a trip T from P to D with distance $dist_T$, we find 4 paths $\{T_1, T_2, T_3, T_4\}$ whose lengths are closest to $dist_T$. Here, paths T_1 and T_2 have the same length and are closer to $dist_T$ than paths T_3 and T_4. Therefore, the weights w_1 and w_2 are greater than the weights w_3 and w_4. Also note that the set $s_e = \{s_e^1, s_e^2, s_e^3\}$ is of size 3 since the road segment e is part of 3 of the 4 paths.

Validation. To validate the closest path traffic model, we compare it with the shortest-path model [SRS*13], as well as with the actual data obtained using EZ-pass tag readers. NYC has a set of such readers placed at strategic points in order to collect traffic information. We had access to data for the month of November, 2011, corresponding to Madison Ave. and Lexington Ave. between 49th and 57th streets. We use the Kullback-Leibler (KL) divergence metric [KL51] to compare the two traffic models with the actual data. For two distributions P and Q, the KL divergence measure $D_{KL}(P||Q)$ computes the amount of information lost when approximating P using Q. Given the traffic derived from the closest path model $M_{closest}$, we first compute a set of divergence measures $D_{closest} = D_{KL}(tag||M_{closest})$ for different time periods. Here, tag represents the distribution obtained using the EZ-pass tag readers. A lower divergence value reveals a better approximation of the observed traffic distribution. We repeat this process using the shortest path to infer traffic speeds to obtain $D_{shortest}$.

Fig. 3 plots the histogram of $D_{closest}$ and $D_{shortest}$. Note that most of the divergence values of the closest path model is close to zero, implying that this model is a good approximation of the observed speed distribution. Moreover, the closest path model has more divergence values close to zero than the shortest path model, implying that this model better approximates the observed speed distribution when compared to the shortest path model. As mentioned earlier, this

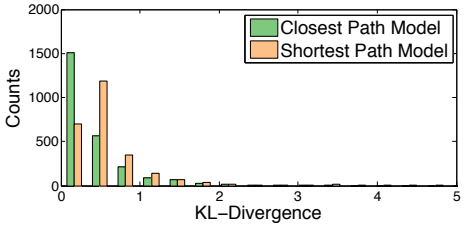

Figure 3: *Histogram of the KL divergence measures obtained when comparing the closest path and shortest path models with the distributions computed using data from EZ pass tag readers. Note that values of $D_{closest}$ from more time periods are closer to zero than the values of $D_{shortest}$ indicating that the closest path model better approximates the observed traffic speeds.*

is due to the fact that many of the taxi trips do not take the shortest path from its pick-up location to its drop-off location, and therefore, the closest path model does a better job of identifying the most probable routes. More details as well as further experiments can be found in Appendix C.

5. The Traffic Function

Visualization of the traffic function is essential to understand the traffic dynamics of a city. In this section, we first formally define the traffic function, and describe its computation. Next, we discuss different visualization techniques and the design choices made to support the traffic function.

5.1. Vector-Valued Function

Recall that the road network is modeled as a directed graph $G(V,E)$. Given the discrete nature of the domain, the traffic function at a given time instant is defined as a vector-valued function $f : E \to \mathbb{R}^3$, $E \subseteq V \times V$, which maps each edge of the graph to $\mathbf{v} = (v_x, v_y, d)$. Since the direction of (v_x, v_y) is the same as the direction of an edge, the vector $\mathbf{v}$ can be implicitly represented as $\tilde{\mathbf{v}} = (s, d)$. Here, s denotes the speed of traffic along a road (magnitude of the vector), and d denotes the density of taxis plying on that road.

Using the above formulation, for a given time interval, the speed at each edge e_i is equal to the mean speed $\bar{s}_i$ of the corresponding road (Section 4.2). The density of taxis along that road is computed as $d_i = \sum_{j=1}^{n} w_i^j$.

Given the large number of samples (greater than 20000 per hour), the density directly translates to the probability that a taxi takes a particular road. It is used to compute both the local probability that a taxi uses a particular road, as well as a global distribution, which denotes the probable locations of taxis at any given time. As we will show later, this probability is used to visualize the taxi movement patterns.

The traffic speed varies depending on the time of day, as well as the day of the week. Therefore, we compute the traffic function for short 5 minute intervals for different days of the week. Fig. 4 shows an example traffic function for one time step. The directed edges denotes the streets, and the vectors associated for four of those edges are shown.

5.2. Visualizing the Traffic Function

We now describe our adaptation of vector field visualization techniques to the traffic function defined above.

5.2.1. Flow Lines

In traditional vector fields, given a time instant, the direction of the flow at any point in space is constant. Unlike such functions, there are multiple possible directions a particle can take at each node of the underlying directed graph of the traffic function. Each of the different alternatives available for choosing the direction results in visualizing different interesting phenomena in the traffic flow. We now discuss these alternatives and their significance.

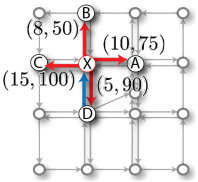

Figure 4: *A sample traffic function.*

Direction of lowest speed. When a particle is at a node, then the adjacent edge having the lowest speed is chosen by the particle. This traces the path of slowest moving traffic, and is useful to identify bottlenecks in the traffic flow. For example, consider a particle that reaches node X in Fig. 4 along the edge (D,X) while tracing the flow line. Using this option, the particle will continue moving forward along edge (X,B). While tracing the particle, we do not allow a flow line to choose the reverse edge. Fig. 5(c) shows an example set of pathlines computed from a random set of seed points using the edge of lowest speed to select a direction.

Direction of highest speed. The particle chooses the edge with highest speed at a node. The path traced in this case helps identify possible routes with relatively free flowing traffic. For example, a particle at node X in Fig. 4 will choose edge (X,C). Fig. 5(b) shows pathlines computed using the edge of highest speed to select a direction.

Direction of highest probability. The particle chooses the edge having highest density of taxis. This helps identify the most probable paths taken by taxis from a given position. Using this option, a particle at node X in Fig. 4 will again choose edge (X,C), since it has the highest probability. Fig. 5(d) shows streamlines computed using the edge with highest probability to select a direction.

Random direction. In order to simulate different scenarios, depending on whether to mimic the movement of taxis, or general traffic, the direction is chosen either uniformly at random, or using the taxi probability respectively.

Streamlines should be used for visualizing the traffic flow when the user is interested in analyzing traffic at a given instant of time as it provides an "instantaneous" view of the vector function. Pathlines, on the other hand, should be used when the user intends to look at the traffic flow over a larger time interval. This is because, in an abstract sense, the pathlines provide a temporal history of the flow over the said time interval. Using any of the above choices for computing the streamline or pathline, will result in a smooth trajectory. However, in case of streaklines, it is possible for particles injected at different times into the flow to take different paths,

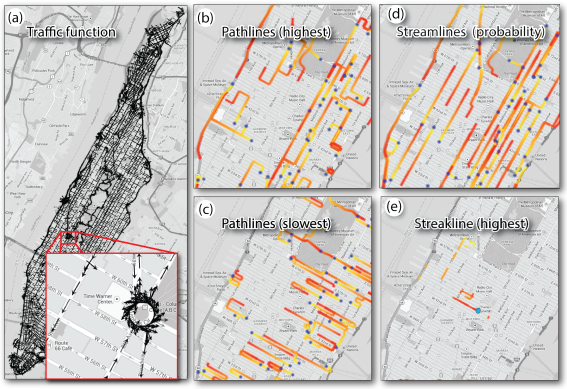

Figure 5: *The traffic function of Manhattan is displayed using glyphs (a). Given the multiple directions a particle can take at a node of the graph, we propose different direction choices for computing flow lines depending on the aspect of the traffic the user wants to visualize (b,c,d,e).*

depending on the dynamics of the traffic. Fig. 5(e) shows a streakline computed from a single seed location, using the edge of highest speed to select the direction. Note that there are different components for this streakline, indicating frequent changes in the traffic dynamics.

5.2.2. Global Visualizations

Having a "big picture" of the traffic is essential to study possible global trends in the traffic. We accomplish this through the use of two global techniques.

Color map. In this visualization, the user can view one of the variables of the traffic function, the speed distribution or the taxi distribution, using a color map. Multiple roads allow for two-way traffic. In such a case, it would be possible to visualize the quantity only across one direction. When using speed distribution, we choose the direction of lowest speed, while for taxi distribution, we choose the direction of largest distribution. Alternatively, the user can also view the distributions along a single direction. The user can interactively change the color map that is used.

Glyphs. We use arrow glyphs to visualize the traffic function. The size of the arrow is mapped to the speed, and we map the taxi density to opacity. While this visualization is useful when viewing a small region, such a visualization gets cluttered when glyphs are used to visualize a large region. This can be observed in Fig. 5(a).

5.2.3. Visualization Choices and Filters

In addition to the visualizations described above, we support various filters and options to help user visualize different aspects of the traffic function. When displaying flow lines, the user can either color them based on direction, or based on time. In the former case, two different colors are used to visualize the stream lines moving northward and southward respectively. In the latter case, a color map is used to indicate the time. Again, the user can choose and modify the color map that is used. The user can also specify the start time and

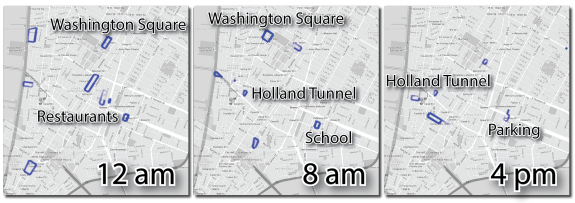

Figure 6: *Orbits found at different times of the day. Note that a few orbits like the ones around a school and the parking lot occur only at certain times, while the ones near Holland tunnel and Washington Square occur more frequently.*

duration for which the flow line is computed. The user can then filter them based on average speed of flow lines.

Choosing good seed points is important to obtain effective visualizations. In addition to allowing users to choose seed regions, we also automatically generate seed points. In case the user is interested in flow of regular traffic, the seed points are selected uniformly at random. If the user is interested in taxi movement, then the seed points are generated using the global probability distribution obtained using the density parameter of the traffic function. This ensures that the concentration of taxis are reflected in a realistic manner.

5.2.4. Animation

Animating movement of multiple particles for short durations helps observe the traffic dynamics of a city. It is particularly useful for simulating probable traffic flows. During the animation, we explicitly distinguish between two types of vehicular traffic – taxis and non-taxis. All the filters and options discussed above applies to animations as well.

6. Traffic Dynamics in Manhattan

Our visualizations were demonstrated to experts from NYC DoT and TLC. In addition to providing feedback on our results, they also used our visual interface to study specific cases to test a series of hypotheses. Below, we discuss case studies where we show how the techniques we propose can be used to explore mobility patterns in Manhattan, and report on insights obtained from the experts.

The traffic patterns typically remain the same during a given season, and varies across seasons. When not targeting a specific day or event, users can explore traffic mobility patterns for a given season. Unless otherwise mentioned, we use the function corresponding to the summer months of 2011 in this section.

6.1. Identifying Traffic Bottlenecks

Locally slow moving traffic can be identified by tracking flow lines computed using the *direction of slowest speed*. Traffic jams typically start within a small region and propagate to neighboring regions [JSS12]. Experts from the DoT are interested in identifying such regions so that they can design and employ policies to ease congestion and improve traffic flow. By looking at just the traffic speeds, they can get a big picture of the traffic patterns, but looking at local traffic

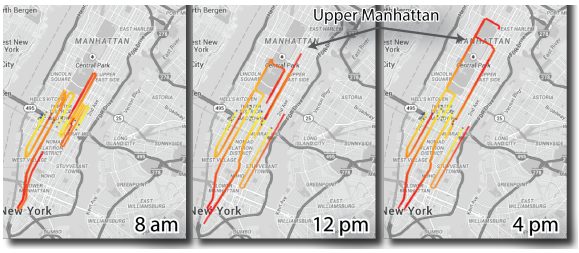

Figure 7: *Pathlines at different time periods showing the most probable flow of taxis. Note that the pathlines tend to cross into Upper Manhattan only in the latter part of the day. The direction is color encoded from yellow to red.*

flow allows them to pinpoint specific regions to focus on in order to analyze different bottlenecks.

Since we are interested in traffic bottleneck at a given instant over time, we decided to explore streamlines at different time instants, computed using direction of slowest speed. During this exploration, we observed that streamlines from seed points in certain areas converged into orbits. A subset of these orbits, especially those that span small regions (such as a neighborhood block), corresponds to a closed loop of roads where the speed of traffic flow is slower than in neighboring roads, potentially indicating the presence of congestion.

Fig. 6 shows the different orbits found during three different time steps in the lower Manhattan area. The experts were particularly interested in a few specific orbits highlighted in this figure. There was an orbit at the entrance of Holland tunnel that was present during most of the day. This is due to the traffic moving towards Jersey City which concentrates in this area. There was an interesting orbit surrounding a block that hosts a school. We found this orbit only at 8am – the dropoff time. When looking at the set of orbits at 4 pm, the experts immediately identified the one around a parking lot. They mentioned that it was very difficult to find parking in that area, as a result, people often double park for short periods, causing the traffic congestion.

In addition to the illustrated orbits, we also found larger orbits (surrounding multiple blocks) in different parts of the city at various times. While it is not immediately clear as to what these signify, we intend to further explore such phenomena in the future together with the experts. Congestion in general can surface in various patterns, and we believe that orbits is just one of them. In our current implementation, we do not automatically detect orbits, so it is possible that many such orbits might be missed out (depending on the randomly selected seed points). However, given the interesting results, we plan to explore topology-based techniques on the traffic function in the future to identify traffic patterns.

6.2. Taxi Patterns

Experts from TLC were interested in identifying common roads taken by taxis. This will help them identify prominent locations to place data capturing sensors in order to collect data from taxis at regular intervals. We accomplish this through the use pathlines to visualize the general flow

Figure 8: *Traffic trends in Manhattan. This sequence of maps shows the visualization of the mean speed (in MPH) of traffic on all Fridays from 8 am until midnight. Note the change in traffic trends during different times of the day.*

patterns of taxis in Manhattan. Fig. 7 illustrates the pathlines computed using the most probable direction for four different time steps. The seed points were selected to be in Lower Manhattan area, and the pathlines were computed for a time period of 20 minutes. While the pathlines confirm that the flow of taxis are usually over the avenues, the shape of the lines also indicates the general direction of flow. For example, at 8 am we observe that the probability of taxis moving beyond Midtown is low, and therefore the pathlines do not extend beyond Central park. As the day progresses, we notice that taxis tend to move more towards Upper Manhattan.

6.3. Traffic Patterns

Traffic trends on Fridays. In order to get a global view of the general trend in Manhattan's traffic on Fridays, we visualize the speed of traffic using a color map. For this experiment, we divided the day into hourly time intervals, and used all the taxi trips that happened on all Fridays during Summer (June, July, and August) to compute the traffic function. Fig. 8 illustrates the various trends at different times of the day. We observe that the speed decreases during the day starting from 8 am and remains almost the same during the afternoon, indicating an increase in traffic movement. This is especially true in lower and middle Manhattan, where the business district and many tourist spots are located. An interesting point to note is that the traffic remains slow even at 8 pm in parts of lower Manhattan, which hosts a lot of restaurants. We also observe that the speed of traffic in and around Time Square (green rectangle) is slow at this time.

High speed traffic flow. Path lines of short time duration can be used to indicate the general direction of traffic flow. In this example, we are interested in identifying the variation in speeds between north-bound and south-bound traffic between Upper Manhattan and Lower Manhattan, respectively. We accomplish this by visualizing short path lines, where the choice of direction was based on highest speed. Fig. 9

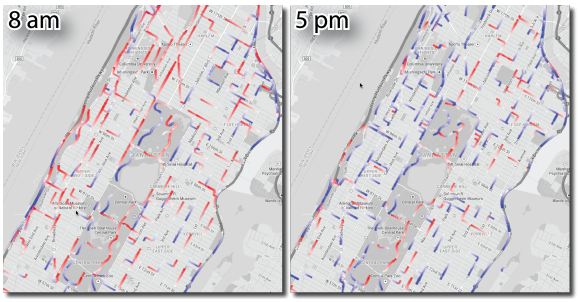

Figure 9: *Short 1 minute pathlines are used to identify direction of high speed traffic flow. Red pathlines indicate northward direction, while blue pathlines indicate southward movement. Note that the direction of high speed is upward during the day, and reverses at 5 pm.*

shows high speed traffic flow at different times of the day. In particular, direction of high speed is usually northwards during the day. This can be seen by the number of red pathlines at 8 am. Interestingly, the direction of high speed traffic flow reverses at 5 pm. This is the time people usually return home from work, thus resulting in slower northward moving traffic. This reverse in direction was found only during the evening time period. Later during the night, the direction of high speed was evenly spread towards both directions.

We also observe that the direction of high speed traffic is usually along the avenues (vertical), and that of low speed traffic is along the streets (horizontal). This is illustrated by the pathlines in Fig. 5(b) and 5(c). This is because the traffic signal is longer along the streets than avenues.

6.4. Simulating Road Blocks

Analysts at the DoT are also interested in understanding traffic patterns when roads are blocked. In particular, how the speeds vary, and how they effect the flow. Such road blocks are common in Manhattan, for example during parades. To support the study of what-if scenarios and help decision makers plan for upcoming events, we can simulate road blockage. Policies can then be designed to better handle such road blocks in the future.

Since we focus on historical data, we know that the trips that occurred during a road block event would not pass through the corresponding roads. Computing routes for such trips is accomplished by setting the distance of the corresponding edges of the road network to infinity during the duration of the event. This ensures that no taxi trip during a road block event takes a route involving the blocked roads. The traffic function for the time period of the blockage is then computed using this modified graph. The set of blocked road segments were obtained by looking at roads where the density of the taxis was zero [DFD*14]. For this experiment, we chose to simulate the road block that happened on 24 November 2011 between 34th and 42nd Street due to the Macy's Thanksgiving Day parade. In particular, we are interested in studying the transition of traffic flow when the

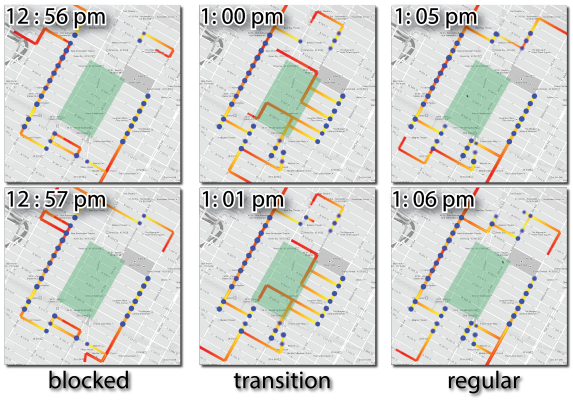

Figure 10: *Simulating traffic flow during road blocks. The streamlines are computed from the set of blue seed points. The direction of highest speed was used for computing the path taken. The changes in the streamlines indicates that the speeds of the roads in the blockage region initially increases during the unblocking period, before stabilizing to normal flow. The direction is color encoded from yellow to red.*

road block is removed. We therefore choose streamlines to visualize the phenomena at different time instants.

Fig. 10 shows a set of streamlines, where the seed points are chosen to be close to region of the road block. We are interested in observing the change of high speed flow patterns, and therefore the choice of direction while computing the streamlines was direction of highest speed. The variation in streamlines are shown at different periods – when the roads were blocked, the instant when the block was removed, and once the traffic flow stabilizes after the event. Note that when the roads open up to traffic, the streamlines which was otherwise moving along avenues, now passes through the roads that were blocked. This is because when those roads were unblocked, the speed along these roads become greater than the neighboring roads. This is due to lack of vehicles on these roads at this time. However, the traffic stabilizes back to normal flow, which is indicated by the streamlines again following the normal pattern.

7. Discussions and Future Work

Closest path algorithm. The identification of correct routes taken by a taxi is dependent on the number of way-points n used for computing the closest paths. While each taxi trip could involve different number of way-points, it is computationally expensive (exponential) to traverse through this entire search space. In practice, at least for the sample of the taxi trips for which we had the actual route, we found that using $n = 1$ resulted in close to accurate predictions of the taxi routes (see Appendix B). Additionally, using $n = 1$ also results in an efficient linear time computation which is an advantage especially due to the large number of trips that has to be processed. Further, we also assume that there are no loops in the trip routes given the large number of small length trips that happen (Appendix E). In future work, we plan to explore techniques which can efficiently provide better correctness guarantees.

Traffic function computation. The number of closest paths, k, is an important parameter for computing the traffic function. Using a small value for k could potentially miss identifying the actual routes taken. In order to identify a good value for k, we picked a time period, and computed the traffic function for varying values of k. As k increases, more paths in the resulting set have lengths significantly different from the actual length thus decreasing the weight assigned to these paths. Such paths therefore have very little influence in the traffic function. We found $k = 20$ provided a stable traffic function (see Appendix D).

While we can vary the size of k and / or n based on the length of a trip, we believe this will not provide a significant improvement in the accuracy since most of the taxi trips that happen in Manhattan are of short lengths (Appendix E).

Our model assumes that in a given trip, each road will be traveled at the same speed. While it is true that the average speed of a trip is not necessarily equal to the speed of a particular road segment, given the large number of diverse trips on each road segment, we expect the averaging to converge to the actual speed. This property was confirmed during validation of the traffic model. Additionally, this assumption makes computation of the traffic model possible in linear time, and thus handle the large amount of trips (approximately 170 million trips per year). In future, we plan to use additional information such as the number of lanes and vehicle capacity of different roads to improve our model.

Visualization using color map. The computed traffic information includes more information than what is visualized using color maps. In particular, there is data on traffic speeds and densities along both directions on any given two-way street, as well as information on the uncertainty of the computed data. While these quantities are important, it is not clear how these can be represented in a single image without adding clutter. We intend to investigate this in future.

Expert feedback. The focus of this work was twofold – obtain an efficient scalable traffic model; and the novel adaptation of vector field techniques for visualizing traffic flow and its advantages. The developed system was a proof-of-concept prototype using which we could demonstrate our techniques to experts and obtain feedback on the usefulness of our techniques for their analyses. As a next step, we intend to focus on the usability and user interaction of our system so that it can be deployed at the different NYC agencies.

Acknowledgments. The authors thank the TLC and DoT for providing the data used in this paper and feedback on our results. This work was supported in part by a Google Faculty Award, an IBM Faculty Award, the Moore-Sloan Data Science Environment at NYU, the NYU School of Engineering, the NYU Center for Urban Science and Progress, AT&T, NSF award CNS-1229185, and CNPq Processes 476685/2012-5 and 309483/2011-5.

References

[AA13] ANDRIENKO N., ANDRIENKO G.: Visual analytics of movement: An overview of methods, tools and procedures. *Information Visualization 12*, 1 (2013), 3–24. 2, 3

[Baj99] BAJAJ C. (Ed.): *Data Visualization Techniques*. Wiley, 1999. 3

[BKKW08] BURGER K., KONDRATIEVA P., KRUGER J., WESTERMANN R.: Importance-driven particle techniques for flow visualization. In *Proc. PacificVIS* (March 2008), pp. 71–78. 3

[BSK*07] BÜRGER K., SCHNEIDER J., KONDRATIEVA P., KRÜGER J., WESTERMANN R.: Interactive visual exploration of unsteady 3d flows. In *Proc. EuroVIS* (2007), pp. 251–258. 3

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (2009), 77:1–77:10. 2

[CEW*08] CHEN G., ESCH G., WONKA P., MUELLER P., ZHANG E.: Interactive procedural street modeling. *ACM Trans. Graph. 27*, 3 (2008), Article 103: 1–10. 2

[DCPun] DAILEY D., CATHEY F., PUMRIN S.: An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE ITS 1*, 2 (Jun), 98–107. 2

[DFD*14] DORAISWAMY H., FERREIRA N., DAMOULAS T., FREIRE J., SILVA C.: Using topological analysis to support event-guided exploration in urban data. *IEEE TVCG 20*, 12 (2014), 2634–2643. 8

[FPV*13] FERREIRA N., POCO J., VO H. T., FREIRE J., SILVA C. T.: Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. *IEEE TVCG 19*, 12 (2013), 2149–2158. 2

[JSS12] JAIN V., SHARMA A., SUBRAMANIAN L.: Road traffic congestion in the developing world. In *Proc. ACM DEV* (2012), pp. 11:1–11:10. 7

[KKKW05] KRUGER J., KIPFER P., KONDRATIEVA P., WESTERMANN R.: A particle system for interactive visualization of 3d flows. *IEEE TVCG 11*, 6 (2005), 744–756. 3

[KKW05] KONDRATIEVA P., KRUGER J., WESTERMANN R.: The application of gpu particle tracing to diffusion tensor field visualization. In *Visualization, 2005. VIS 05. IEEE* (2005), pp. 73–78. 3

[KL51] KULLBACK S., LEIBLER R. A.: On information and sufficiency. *Ann. Math. Statistics 22* (1951), 79–86. 5

[ksp] An implementation of k-shortest path algorithm. URL: https://code.google.com/p/k-shortest-paths/. 4

[LHD*04] LARAMEE R. S., HAUSER H., DOLEISCH H., VROLIJK B., POST F. H., WEISKOPF D.: The state of the art in flow visualization: Dense and texture-based techniques. *CGF 23*, 2 (2004), 203–221. 3

[LZZ*09] LOU Y., ZHANG C., ZHENG Y., XIE X., WANG W., HUANG Y.: Map-matching for low-sampling-rate gps trajectories. In *Proc. ACM SIGSPATIAL GIS* (New York, NY, USA, 2009), ACM, pp. 352–361. 2

[MLP*10a] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. *CGF 29*, 6 (2010), 1807–1829. 2, 3

[MLP*10b] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. *CGF 29*, 6 (2010), 1807–1829. 3

[MP03] MARTINS E., PASCOAL M.: A new implementation of yen's ranking loopless paths algorithm. *4OR 1* (2003), 121–133. 4

[PPF*11] POBITZER A., PEIKERT R., FUCHS R., SCHINDLER B., KUHN A., THEISEL H., MATKOVIĆ K., HAUSER H.: The state of the art in topology-based visualization of unsteady flow. *CGF 30*, 6 (2011). 3

[PVH*03] POST F. H., VROLIJK B., HAUSER H., LARAMEE R. S., DOLEISCH H.: The state of the art in flow visualisation: Feature extraction and tracking. *Computer Graphics Forum 22*, 4 (2003), 775–792. 3

[QON07] QUDDUS M. A., OCHIENG W. Y., NOLAND R. B.: Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies 15*, 5 (2007), 312 – 328. 2

[SRS*13] SANTI P., RESTA G., SZELL M., SOBOLEVSKY S., STROGATZ S. H., RATTI C.: Taxi pooling in new york city: a network-based approach to social sharing problems. *CoRR abs/1310.2963* (2013). 2, 5

[SWvdW*11] SCHEEPENS R., WILLEMS N., VAN DE WETERING H., ANDRIENKO G., ANDRIENKO N., VAN WIJK J.: Composite density maps for multivariate trajectories. *IEEE TVCG 17*, 12 (2011), 2518–2527. 2

[Tel08] TELEA A.: *Data visualization - principles and practice.* A K Peters, 2008. 3

[TSAA12] TOMINSKI C., SCHUMANN H., ANDRIENKO G., ANDRIENKO N.: Stacking-based visualization of trajectory attribute data. *IEEE TVCG 18*, 12 (2012), 2565–2574. 2

[Wes79] WEST D. H. D.: Updating mean and variance estimates: An improved method. *Commun. ACM 22*, 9 (1979), 532–535. 5

[WLY*13] WANG Z., LU M., YUAN X., ZHANG J., WETERING H. V. D.: Visual traffic jam analysis based on trajectory data. *IEEE TVCG 19*, 12 (2013), 2159–2168. 2

[WVDWVW09] WILLEMS N., VAN DE WETERING H., VAN WIJK J. J.: Visualization of vessel movements. *CGF 28*, 3 (2009), 959–966. 2

[Yen71] YEN J. Y.: Finding the k shortest loopless paths in a network. *Management Science 17*, 11 (1971), pp. 712–716. 4

[ZHUK13] ZHAN X., HASAN S., UKKUSURI S. V., KAMGA C.: Urban link travel time estimation using large-scale taxi data with partial information. *Transportation Research Part C: Emerging Technologies 33*, 0 (2013), 37 – 49. 1, 2

[ZMT06] ZHANG E., MISCHAIKOW K., TURK G.: Vector field design on surfaces. *ACM Trans. Graph. 25*, 4 (2006), 1294–1326. 2

[ZmXZ05] ZOU L., MIN XU J., ZHU L.-X.: Arterial speed studies with taxi equipped with global positioning receivers as probe vehicle. In *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on* (sept. 2005), vol. 2, pp. 1343 – 1347. 2