Using Topological Analysis to Support Event-Guided Exploration in Urban Data

Harish Doraiswamy *Member, IEEE*, Nivan Ferreira *Student Member, IEEE*, Theodoros Damoulas, Juliana Freire *Member, IEEE* and Cláudio T. Silva *Fellow, IEEE*

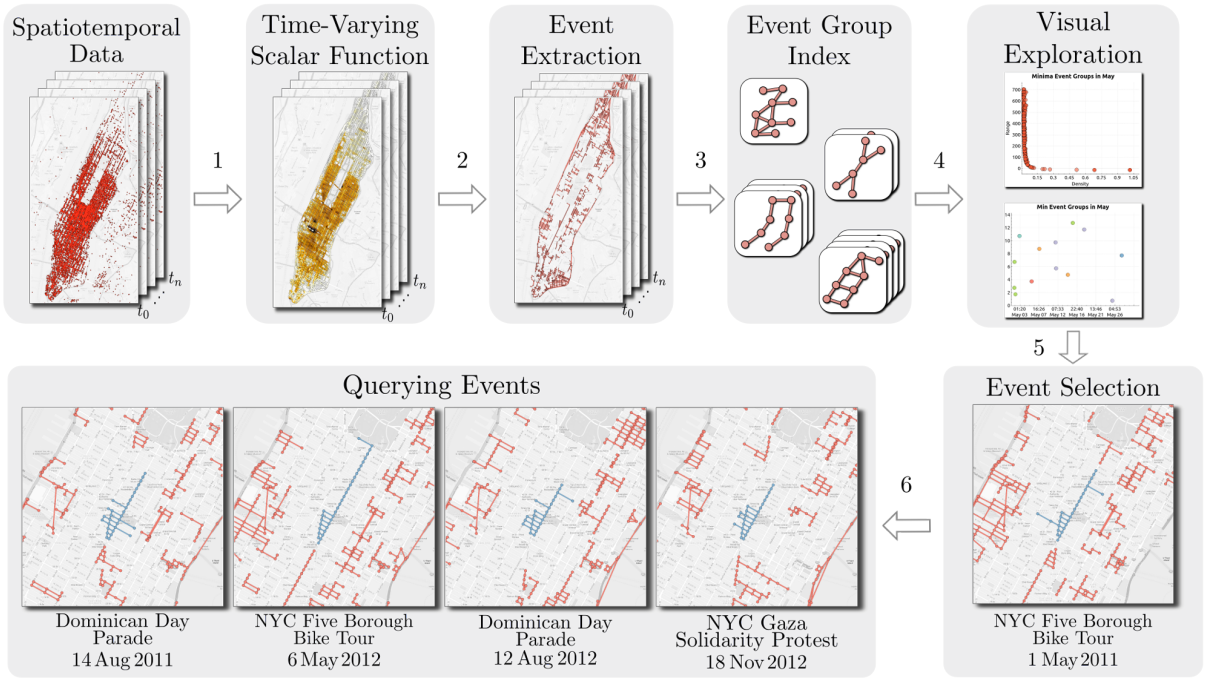


Fig. 1. Overview of the event guided exploration technique. First, (1) the input data is transformed into a time-varying scalar function. (2) Topological features are computed from the scalar functions to identify the set of events. (3) An event group index is then created from the identified events to support efficient querying over a large number of events. (4) A visual interface guides the user towards interesting events (5) in the data, allowing them to select an event and (6) interactively search for similar events.

Abstract— The explosion in the volume of data about urban environments has opened up opportunities to inform both policy and administration and thereby help governments improve the lives of their citizens, increase the efficiency of public services, and reduce the environmental harms of development. However, cities are complex systems and exploring the data they generate is challenging. The interaction between the various components in a city creates complex dynamics where interesting facts occur at multiple scales, requiring users to inspect a large number of data slices over time and space. Manual exploration of these slices is ineffective, time consuming, and in many cases impractical. In this paper, we propose a technique that supports event-guided exploration of large, spatio-temporal urban data. We model the data as time-varying scalar functions and use computational topology to automatically identify events in different data slices. To handle a potentially large number of events, we develop an algorithm to group and index them, thus allowing users to interactively explore and query event patterns on the fly. A visual exploration interface helps guide users towards data slices that display interesting events and trends. We demonstrate the effectiveness of our technique on two different data sets from New York City (NYC): data about taxi trips and subway service. We also report on the feedback we received from analysts at different NYC agencies.

Index Terms—Computational topology, event detection, spatio-temporal index, urban data, visual exploration.

1 Introduction

Recent technological innovations have enabled the collection of enormous amounts of data pertaining to cities, from conventional sensors, such as power consumption [30] and noise [56], to more "unconventional" means of capturing city dynamics such GPS in vehicles [5, 22, 54], mobile devices [26], and social media [12, 29]. Cities all over the world are not only collecting these data, but they are also making the data available (see e.g., [43, 44, 45]). If properly analyzed, urban data can take us beyond today's imperfect and often anecdotal understanding of cities to enable better operations, informed planning, and improved policy. These data are also a rich source for social scientists who aim to better understand cities and their populations.

However, there are many challenges involved in enabling the effective analysis of urban data. They stem not only from the volume of data, but also from the inherent spatio-temporal complexity of the underlying processes in a city. The usual approach to analyze this kind of data is to use different types of aggregation and produce visual summaries [2, 35]. These lead to a trade-off between the level of aggrega-

• *H. Doraiswamy, N. Ferreira, T. Damoulas, J. Freire, and C. Silva are with New York University E-mail: {harishd,nivan.ferreira,damoulas,juliana.freire,csilva}@nyu.edu.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014; date of publication xx xxx 2014; date of current version xx xxx 2014. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

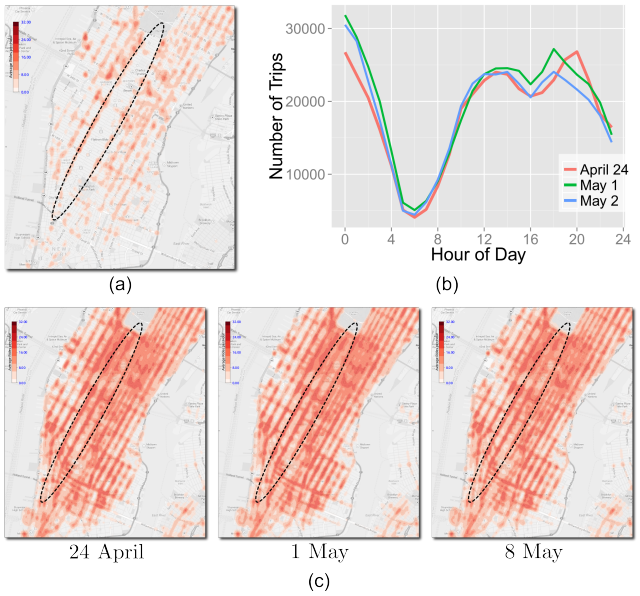


Fig. 2. It is difficult to identify short irregular patterns when the data is aggregated over either space or time. **(a)** A heatmap of taxi locations in Manhattan on 1 May 2011 between 8 am and 9 am. Note that the path of the bike tour contains no taxis. **(b)** The time series plots compare the number of trips that occurred in Manhattan on three Sundays in 2011: 24 April, 1 May, 8 May. It is difficult to distinguish between the three Sundays using just the number of trips, even though an entire stretch of streets are blocked to traffic on May 1st. **(c)** The trips are aggregated over time and displayed as a heat map for the three Sundays. Note that the path of the bike tour (highlighted) looks similar in all the heat maps.

tion and the number of data slices to be explored. The use of a coarse (spatial or temporal) aggregation reduces the number of data slices, but it may result in loss of information. Consider the example illustrated in Fig. 2(a), which shows a heatmap of pickups and drop-offs of yellow taxi cabs in New York City on Sunday 1 May 2011 between 8 and 9 am. Note that there are no taxi pickups and drop-offs along 6^{th} avenue. This avenue was closed to traffic at this time for the annual NYC 5 Boro Bike Tour. However, as shown in Figures 2(b) and (c), a coarse level of aggregation makes it difficult to identify small or local events. While a finer level of aggregation would avoid these problems, it requires the exploration of a large number of data slices. This problem is more prominent in an urban environment, where important patterns/events can happen at multiple scales [14, 22]. Manual (exhaustive) exploration in such cases is not only time consuming, but for large data sets, it becomes impractical. For example, temporal aggregation of a year's worth of data into a discrete set of hourly intervals results in over 8000 data slices to be explored per year.

In recent years, techniques and systems have been proposed that attempt to streamline and better support exploratory analyses of spatio-temporal data. These include sophisticated visualization and interaction techniques that allow users to freely explore the data at various levels of aggregation [3, 14, 22, 51, 54]. However, effective interaction with spatio-temporal visualizations remains a challenge [25, 49] and even using these techniques domain experts may still need to examine a prohibitively large number of spatio-temporal slices to discover interesting patterns that represent both regular and irregular behavior.

Contributions. As a step towards addressing this challenge, we propose an efficient and scalable technique that automatically discovers events and guides users towards potentially interesting data slices. Event detection is accomplished through the application of topological analysis on a time-varying scalar function derived from the urban data. We use the minima and maxima of a given function to represent the events in the data. Intuitively, a minimum (maximum) captures a feature corresponding to a valley (peak) of the data. For example, the lack of taxis along 6^{th} avenue during the bike tour event forms a local minimum and is therefore captured using our technique. The use of topology also allows for events having an arbitrary spatial structure. In order to support a potentially large number of events, we design an indexing scheme that groups similar patterns across time slices, thereby allowing for identification of not only periodic events (hourly, daily, and weekly events), but also of events with varying frequency (regular and irregular). Thus, unlike previous approaches that impose a rigid definition of what constitutes an event [5], our technique is flexible and able to capture a wide range of spatio-temporal events. The index further allows users to efficiently search for occurrences of similar patterns. Compared to techniques based on statistical analysis that support different kinds of events, our approach is computationally efficient and scales to large data sets. We also describe a visual interface designed to aid in event-guided exploration of urban data that integrates the event detection and indexing techniques. The interface allows users to interactively query and visualize interesting patterns in the data. Finally, we show the effectiveness of our approach on two urban data sets: information about taxi trips, collected by the NYC Taxi & Limousine Commission (TLC), and subway service, published by the Metropolitan Transit Authority (MTA). We presented results of preliminary analyses to experts at both the TLC and MTA. While they offered insights for some of the events we found, they were surprised by others which indicated potential problems they had to investigate (see Section 7). This initial feedback suggests our technique is effective and has the potential to help in exploratory analyses of large, spatio-temporal urban data.

2 Related Work

The problem of event detection in the context of spatio-temporal data has been recognized and addressed in previous works. We discuss approaches to event detection in three categories: those that use computational topology for feature tracking, those related to visual analytics, and techniques for event detection from statistics and machine learning. Note that there is no universal definition of an event [5]. It is thus difficult to quantitatively compare different techniques. Here, we present a qualitative comparison where we consider flexibility to support different event types, efficiency, and scalability.

Computational topology. Computational topology has been used to identify and track features of spatio-temporal data. Laney et al. [34] and Bremer et al. [8] used the Morse decomposition of a scalar field to identify features of the input and track these features across time using geometric properties of the features. Pascucci et al. [48] identified features, which correspond to burning cells during turbulent combustion, using merge trees and tracked them by computing the overlap of the features. Widanagamaachchi et al. [55] extended this technique and designed a framework to explore time-varying data. Kasten et al. [31] mapped critical points of the input scalar function across time steps and created a merge graph that is used to track unsteady flow fields. Doraiswamy et al. [17] identified cloud systems in each time step using the join and split tree, and tracked them across time using optical flow. These methods are only interested in movement of features across consecutive time steps, which is accomplished by looking at adjacent time-slices. Such tracking cannot be applied to our problem, since we need to identify features that have similar behavior but are spread across non-adjacent time slices.

Visual Analytics for Spatio-Temporal Data. Scholz et al. [50] proposed a technique to analyze hotspots using taxi data in San Francisco. They pre-defined regions of interest, modeled the taxi activity in each census tract in these regions, and used the model to predict the life cycle of hotspots. By pre-defining regions of interest and using artificial boundaries such as census tracts, this approach is likely to miss events which are of arbitrary shapes and happen at different granularities (see Fig. 2). Maciejewski et al. [37] used kernel density estimation to model hotspots in spatial distributions along with time series analysis to detect anomalous hotspots. Andrienko et al. [3, 4, 5, 6] proposed visual analytics procedures to determine places of interest based on high-frequency events that also have high density of occurrence. The user first applies a set of filters to define features of potentially interesting events. Then, the points with those features are clustered to find interesting recurring locations. Unlike these techniques, which have a

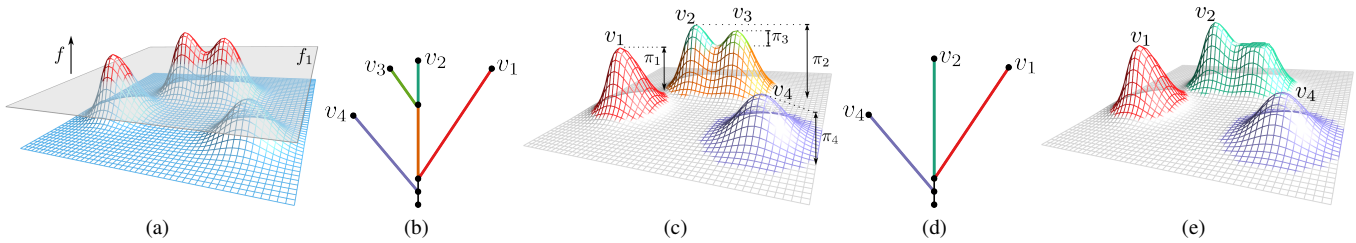


Fig. 3. Topology of scalar functions. **(a)** The height function f defined on a graph. The super-level set at f_1 is the set of all points above the highlighted plane and consists of *two* components (colored red). **(b)** Join tree of f. **(c)** The features of the input are defined based on the edges of the join tree. The labeled peaks denote the set of maxima. The features are colored the same as the corresponding edges in (b). π_i denotes the persistence of maximum v_i. Intuitively, the persistence of each maximum is equal to the height the corresponding peak. **(d)** The simplified join tree obtained from removing the maximum v_3. **(e)** The resulting smoothed function.

rigid definition of events (e.g., high density and recurrent occurrence), our technique is able to capture a wide spectrum of events, both based on density (low and high) and frequency of occurrence at different time scales.

Statistical Event Detection in Spatio-Temporal Data. The problem of event detection has also been studied by the statistics and machine learning communities [28, 33, 38, 41, 42, 53]. The area is closely related to spatial scan statistics [32] and anomaly detection [13], albeit exploiting the spatio-temporal nature of the domain and focusing on the discovery of "interesting" contiguous regions in space and time. Previous work examined multiple overlapping spatio-temporal subsets of data and identified significant deviations from a *baseline*, e.g., an expectation over time, via a frequentist likelihood ratio test or a Bayesian posterior probability distribution over events [42]. However, the majority of the literature has focused on either purely spatial data or has accounted for temporal variations and effects via simplistic approaches such as exponentially weighted linear regression or data partitioning based on day-of-week or season. Furthermore, the time complexity for these approaches is exponential $O(2^N)$ in the number of *pre-defined* space-time partitions, with polynomial approximations (non-exhaustive search) available only for the frequentist hypothesis tests that require extensive randomization [38] for p-value estimation. In contrast, our technique allows for detection of events that can have arbitrary spatial geometry, different time intervals, scales up with polynomial time complexity of $O(n^2)$ in the number of events, and enables user exploration of urban data sets via efficient event querying. The latter ensures flexibility of the technique across applications and domains, where users can define and query interesting events based on prior knowledge and different spatio-temporal properties of the data.

3 Background

The topological representation of large data sets provides an abstract and compact global view that captures different features and leads to enhanced and easier analysis across applications [23, 47]. In this section, we briefly introduce concepts from computational topology that serve as the basis of the proposed technique. Comprehensive discussions on this topic can be found in [19, 27, 39].

Scalar functions. A *scalar function* maps points in a spatial domain to real values. The spatial domain of interest in this work is a graph G representing a particular aspect of an urban environment like the road network. The scalar function is represented using the graph G, together with a piecewise linear (PL) function $f : G \to \mathbb{R}$. The function is defined on the vertices of the graph and linearly interpolated within each edge. Fig. 3(a) shows an example of a scalar function defined on a graph representing a terrain. The function value at each point on this graph is equal to the point's y-coordinate. A *super-level set* of a real value a is defined as the pre-image of the interval $[a, +\infty)$, the set of all points having function value greater than or equal to a. Similarly, the *sub-level set* of a is the pre-image of the interval $(-\infty, a]$. Fig. 3(a) highlights the super-level set at function value f_1.

Critical points. The *critical points* of a smooth real-valued function are exactly where the gradient becomes zero. Points that are not critical are *regular*. We are interested in the evolution of super-level sets (sub-level sets) against decreasing (increasing) function value. Topological changes occur at critical points, whereas topology of the super-level set (sub-level set) is preserved across regular points [39].

The critical points of a PL function are always located at vertices of the mesh [7, 20]. Consider a sweep of the function f in decreasing order of function value. The nature of topological change to the super-level sets of f when the sweep passes a vertex determines the type of that vertex. A new super-level set component is created at a *maximum*, while two super-level set components merge into one at a *join saddle*. Similarly, during the sweep of the input in increasing order of function value, a new sub-level set component is created at a *minimum*, while two sub-level set components merge into one at a *split saddle*. The scalar function shown in Fig. 3(a) has 4 maxima (see Fig. 3(c)).

Different types of critical points of a scalar function capture different types of features. In particular, a maximum captures a peak of the function, where the function value is higher than its neighborhood. Similarly, a minimum captures a valley of the function. The set of peaks and valleys are the natural features of a given function, and are therefore of interest in this work. We use the set of minima and maxima to represent features (events) of the given data.

Topological persistence. Consider the sweep of the input function f in decreasing order of function value. As mentioned above, the topology of the super-level sets change when this sweep encounters a critical point. A critical point is called a creator if a new component is created, and a destroyer otherwise. It turns out that one can pair up each creator v_c uniquely with a destroyer v_d that destroys the component created at v_c. The persistence value of v_c is defined as $\pi_c = f(v_c) - f(v_d)$, which is intuitively the lifetime of the feature created at v_c, and is thus a measure of the importance of v_c. The traditional persistence of the global maximum is equal to ∞ since there is no pairing destroyer for that maximum. In this paper, we use the notion of extended persistence [1] which pairs the global maximum with the global minimum. For the height function shown in Fig. 3(a), the persistence of each feature corresponds to the height of the corresponding peak, highlighted in Fig. 3(c). Given an input domain of size n, the persistence of the set of minima and maxima can be computed efficiently in $O(n \log n)$ time [18, 21].

Join tree and split tree. The join tree and split tree abstracts the topology of a scalar function f, and are useful for extracting and representing features of f (the regions corresponding to maxima and minima). The *join tree* tracks the changes in the connectivity of super-level sets of an input function f with decreasing function value. Fig. 3(b) shows the join tree corresponding to the function shown in Fig. 3(a). The *split tree* of f is defined similarly, and tracks the connectivity of the sub-level sets of f with increasing function value. Nodes of the join tree and split tree correspond to the set of critical points of f.

Regular points are often inserted into the join/split tree as degree-2 nodes to obtain an *augmented join tree/augmented split tree*. We use the subgraph of the input mesh induced by the regular vertices that are part of an edge in the augmented join/split tree to represent the feature corresponding to the maximum/minimum. The colors of the different features of the function in Fig. 3(c) correspond to the colors of the edges of the join tree shown in Fig. 3(b). Optimal algorithms exists to compute join and split trees of a PL function [10, 15, 36, 46].

Simplification. The input is often simplified to remove noise. This is accomplished by removing low persistent features. The join and split trees provide an efficient mechanism to perform this simplification [11]. Removing an edge in the join/split tree corresponds to smoothing the corresponding region of the function. For example, consider the feature represented by v_3 in Fig. 3 which has low persistence. Simplifying this feature corresponds to smoothing the function in order to remove the maximum v_3. The simplified join tree is shown in Fig. 3(d), while function resulting from this simplification is illustrated in Fig. 3(e). Features can also be simplified based on geometric measures like hyper-volume [11].

4 Urban Data and Scalar Functions

We model urban data as a time-varying scalar function f defined on a graph G, where the temporal dimension is represented as a set of discrete time steps. In this section, we describe two urban data sets that we use in the paper and the scalar functions derived from them.

4.1 NYC Taxi Data

The taxi data set gathered by the TLC is composed of historical data of taxi trips in NYC. Each trip consists of pickup and drop-off locations and times, along with other relevant data such as the fare and tip. There are on average 500 thousand trips each day. In our experiments, we use the taxi data for trips that took place in Manhattan during 2011 and 2012. The data set is first divided into a set of hourly intervals. Note that this time interval is not fixed, and can be changed depending on the application. Manhattan is represented using the graph corresponding to its road network. Each node of this graph represents an intersection of two or more streets, and each edge corresponds to a street segment.

Analysts at the TLC and at the Department of Transportation (DoT) are interested in identifying traffic-related events that have led to road closures as well as taxi hot spots (see Section 7.1). To capture these events, we define the scalar function for an hourly interval at each node of this graph as the density of taxis within a small circular region surrounding the corresponding location. The radius of the circular region is approximately equal to half the distance between two avenues in Manhattan. The density is then computed as the Gaussian weighted sum of the trips within this neighborhood, where the weights correspond to the trip's distance from the node. This ensures that trips closer to a node have a higher contribution to the density compared to a trip that is farther away. Recall that the set of minima and maxima are used to represent events in the data. Given a single time step, a minimum of the above function represents a region where the density of taxis is lower than its local neighborhood, implying a relative scarcity of taxis in that region. Similarly, a maximum represents a region where the density of taxis is higher than that of its local neighborhood, implying a relatively high concentration of taxis.

Such a density function can also be used on many other urban data sets such as twitter feeds [52] and GPS traces from mobile devices [26] which have a representation similar to that of the taxi data.

4.2 MTA Subway Data

The MTA provides real-time information for the numbered lines of the NYC subway system [40]. These data consist of the time stamps of all the stops for all the trips that happen each day. Engineers at MTA are interested in analyzing data from these feeds to improve operations and scheduling of the subway system. In particular, they are interested in delays in the schedule of the different trains. In order to capture this for a given train, we compute the scalar function at each station as the average delay of the train at that station. The delay at a station for a given train is the difference between the time the train was scheduled to arrive and the actual time at which it arrived. The underlying graph used to define the scalar function is essentially a simple path representing the route of the train. The nodes of this path corresponds to the different stations along its route. As with the previous data set, the temporal dimension is divided into a set of hourly intervals. The above scalar function is computed for each of the train lines.

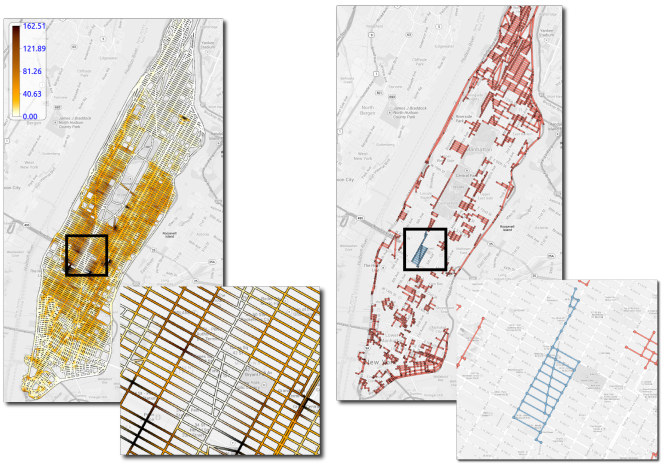


Fig. 4. Identifying minima events. *Left:* The scalar function corresponding to the time step 10 am-11 am on 24 November 2011 is shown using a heat map. *Right:* The set of minima events identified using the split tree. Each connected component of the colored subgraph corresponds to an event. The event corresponding to Macy's Parade is highlighted.

5 Identifying and Managing Events

Our framework consists of two main steps. First, a set of potential events is computed from the input scalar function – these constitute all features from each time step of f. Then, similar single time-step events are grouped and an index is built that supports efficient queries over a possibly large number of events. The process is illustrated in Fig. 1 and the details are presented below. We use the NYC taxi data as a running example to illustrate our technique.

5.1 Computing Events

First, the split tree of the scalar function f is computed. The set of minima and the regions corresponding to them constitute the set of *minima events*. Since we are mainly interested in the set of "significant" events, simplification of the split tree is performed to prune uninteresting (noisy) minima. We use a small threshold (close to zero) during this simplification process. While persistence captures the importance of a feature only in terms of the scalar function, for the taxi data, we are also interested in capturing the geometric size of a feature. We therefore use hyper-volume as the importance measure. The hyper-volume [11] of a topological feature is defined as the integral of the input scalar function over the corresponding region. This allows features that occupy a large area but have low persistence (depth) to also be considered important. Thus, this simplification retains "deep valleys" as well as "shallow, but large" valleys. Note that using persistence instead of hyper-volume could potentially remove the large shallow valleys during the simplification process. The top-k from the set of minima that remain after simplification and their corresponding regions constitute the set of minima events. Here, k is a user defined parameter. In our experiments, we found that setting $k = 50$ provides a good threshold that is large enough to ensure no significant event is lost. Fig. 4 shows the scalar function and the associated set of minima events identified for the time step 10-11am on 24 November 2011. This was one of the time steps during which the Macy's Thanksgiving Parade (highlighted in the figure) occurred. The set of *maxima events* are computed similarly using the join tree of the scalar function.

5.2 Event Group Index

Each of the events computed in the previous stage corresponds to a single time step. Multiple such events can be part of a larger *macro event* that spans multiple time steps. For example, the Macy's Parade consisted of a set of events that spanned several hours when the roads were blocked for the parade. To group such events, we first define a notion of similarity between events based on their geometric and topological properties. Similar events within a user-defined time interval are then grouped together to obtain the set of *event groups*. There can

potentially be a large number of event groups across different time intervals. To support efficient search over event groups, we define a *key* that is used to index these groups.

5.2.1 Similarity Between Events

An event E is formally represented as a pair (R, τ), where R is a subgraph of G denoting the spatial region of E, and τ is a real number that represents the topological importance of E. τ is the same measure that is used to simplify the join and split trees in the previous step, which for the taxi data is the hyper-volume of E. Consider two events $E_1(R_1, \tau_1)$ and $E_2(R_2, \tau_2)$. We use the *graph distance metric* [9], δ, to measure the *geometric similarity* between R_1 and R_2:

$$\delta(E_1, E_2) = 1 - \frac{|R_1 \bigcap R_2|}{\max(|R_1|, |R_2|)},$$

where $R_1 \bigcap R_2$ denotes the maximum common subgraph between R_1 and R_2, and $|R|$ denotes the number of nodes in R. The *topological similarity* between two events is defined as:

$$T(E_1, E_2) = |\tau_1 - \tau_2|$$

The geometric similarity measures the amount of overlap between two regions, ensuring that similar regions have a significant overlap. The topological similarity on the other hand ensures that the two events are topologically close with respect to the topological importance measure used. Two events E_1 and E_2 are *similar* if $\delta(E_1, E_2) \leq \varepsilon_\delta$ and $T(E_1, E_2) \leq \varepsilon_\tau$, where, ε_δ and ε_τ are user-defined thresholds.

5.2.2 Event Group and Event Group Key

An *event group* comprises a set of similar events that occur within a given time interval. A brute-force approach to compute event groups from a set of n events would require the computation of similarity between all pairs of events ($\binom{n}{2}$ comparisons). For example, even considering 50 events per hourly time step results in a total of 1200 events per day. Since urban data sets typically contain data for multiple years, computing similarity between all event pairs is not practical.

To avoid a combinatorial explosion, we propose to group events for fixed time intervals. In this paper, we use a time period equal to one month. We choose this time interval since it provides a good trade-off between efficiency and number of events: there is a sufficient number of events so as to not to miss periodic events, but the number of events in this interval is small enough and does not create a computational bottleneck. Moreover, a time frame of a month provides a natural and easily understandable abstraction for the user to explore event groups.

Given an event group $\Sigma = \{E_1, E_2, \ldots, E_k\}$, we define the *event group key* of Σ as (R_Σ, τ_Σ), where

$$R_\Sigma = \bigcap_{i \in [1,k]} R_i \text{ and } \tau_\Sigma = \sum_{i=1}^{k} \tau_i / k$$

The above definition of the event group key follows directly from the definition of geometric and topological similarity measures. R_Σ is the maximum common subgraph of the geometric regions of all the events in Σ. Since the events in Σ are similar, we can conclude that there is considerable overlap among them due to the similarity condition. Thus, R_Σ provides a good representation for the region where events in Σ occur. τ_Σ captures the topological importance of the Σ as the average of the topological importance of the events in Σ. The definition of event group key also helps in using a consistent definition for the similarity between event groups. Two event groups are similar if their keys satisfy the similarity constraints described earlier.

5.2.3 Computing Event Groups

Even when restricting the events to be within a time interval, comparison between all pairs of events is a costly operation. To decrease the number of such comparisons, we relax the condition of similarity between two events: two events are similar if their event groups are similar. Using the relaxed condition, event groups for each time interval are computed as follows. Initially, each event is its own group. The algorithm iterates through the set of event groups to identify similar group pairs. Events are processed in increasing order of their time step. When two similar event groups are found, they are merged into a single group. The algorithm continues until no event groups are similar. Fig. 5 shows an event group corresponding to the Macy's Thanksgiving Parade along with its event group key.

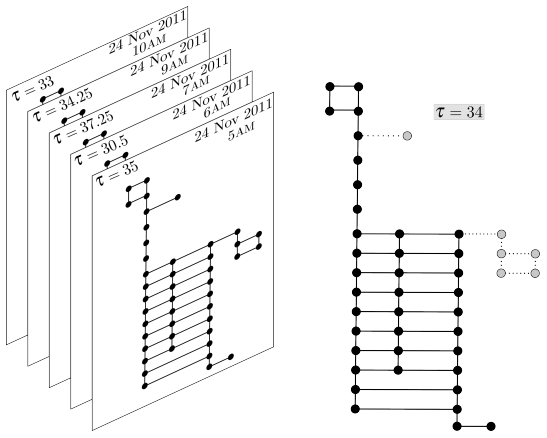


Fig. 5. Event group index. The event group key of an event group corresponding to Macy's Parade. The black nodes and edges correspond to the maximum common subgraph of regions of all the events in the event group, and is used to represent the region of the event group.

The quadratic number of comparisons to be performed among the events present in a given time interval is still a computationally expensive operation (approximately 650 million comparisons in the worst case are required per month assuming 50 events per hour). However, utilizing the spatial information of the events, it is possible to drastically reduce the number of comparisons. The spatial region of the input graph is first divided into a set of smaller subregions. An event intersecting a subregion is assigned to that subregion. It is then sufficient to group only events present in each subregion. Note that an event can be assigned to multiple subregions. These multi-region events can be efficiently handled by maintaining the event groups using the union-find data structure [16].

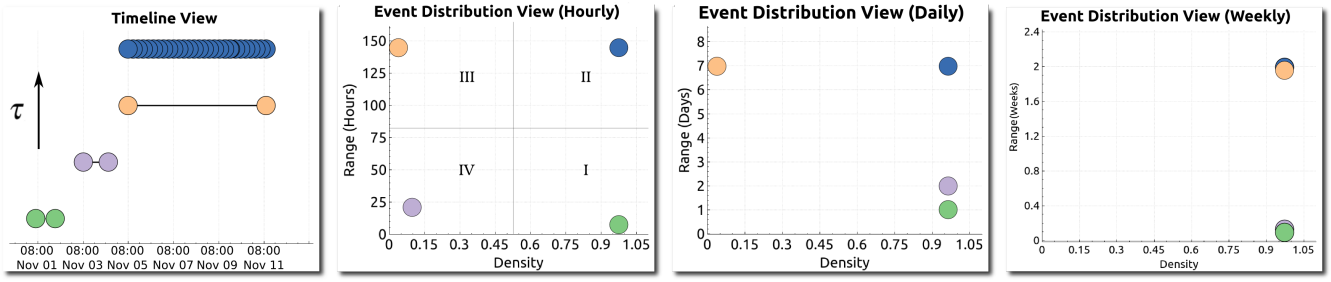
5.2.4 Analysis

Time complexity. Let the graph G have N nodes. Computing the join and split trees of a scalar function defined on G takes $O(N \log N + N\alpha(N))$ time [10], where α is the inverse Ackermann function. Given n events per time interval, computing event groups for that period requires $O(n^2)$ time in the worst case. However, we note that in practice the constant associated with the above bound is small, thus allowing for fast computation. For example, when using 50 events per hour on the taxi data, the average number of events assigned to a subregion of Manhattan was about 1000. Manhattan was divided into a total of 50 subregions, thus amounting to a maximum of 25 million comparisons as opposed to 650 million that is required by a brute-force technique.

Scalability. The set of event groups along with the corresponding keys are stored separately on disk for each time interval. As and when data is obtained for newer time steps, it is easy to update the event group index. Computing the scalar function followed by identifying the events is independent for each time step. If the newer events correspond to an already existing time interval, then they are grouped into the event groups of that interval. If the time interval does not exist, then a new set of event groups are formed for this time interval.

5.3 Querying Events

Once an event is discovered, it is often useful to identify *similar* events that may have occurred at different times. Manual search is impractical due to the large number of data slices. For example, consider a case where a particular road block had unintended effects on the traffic in Manhattan. Experts at the DoT are interested in identifying how frequently such road blocks occur, if they are periodic, and if they have the same effects. This will help them design preventive measures and improve decision making.



(a) (b) (c) (d)

Fig. 6. The visual exploration interface. **(a)** The time line view showing 4 event groups with different densities and ranges. In this view, event groups are sorted according to their topological importance τ. **(b)** The event group distribution view where the time resolution is an hour. **(c)** The event group distribution view when the time resolution is changed to a day. The daily event (purple) moves to Region I for this time resolution. **(d)** The event group distribution view when the time resolution is changed to a week. The weekly event (orange) moves to Region II for this time resolution.

The event group key enables the efficient evaluation of such similarity queries. Given a query consisting of a geometric pattern (a subgraph) together with a real value, we perform a linear search over the set of event groups to identify the set of events similar to the given query. Since the set of event groups for each time interval are stored separately on disk, queries can be executed in an out-of-core manner by sequentially loading event groups from each interval.

6 Visual Exploration Interface

In this section we describe an interactive visual exploration interface that uses coordinated views and allows users to browse through the data based on the detected events. Given the input data set, the event group index is first created in a pre-processing step. The data together with the created index is then loaded by the interface and used to support interactive exploration and querying of events in the data. The rest of this section describes the different views and options available in this interface for data exploration.

6.1 Map View and the Query Interface

The map view, as the name suggests, provides geographical context to the user. For a given time step, the geometry of the different events in that time step is visualized in the context of a map of the city of interest (see Fig. 4). This view also allows users to select events of interest to search for similar events.

6.2 Event Group Distribution View and Timeline View

The event detection technique can generate a large number of event groups, many of which may be uninteresting. It is thus important to allow users to explore the set of event groups and *guide* them towards potentially interesting events. Events can be broadly classified into two categories – recurring events and sporadic (or one-off) events. To capture these categories, we define two attributes – range and density. The *range* of an event group is defined as the amount of time between the first and the last event in that group based on their time steps. Its *density* is defined as the number of events of that group that happen per time unit. It measures the time frequency of the events within the group. The resolution of time determines the time unit. Our tool supports three units – hours, days, and weeks.

A combination of these two attributes enables the classification of an event group. We provide an event group distribution view that uses a scatter plot to visualize the event groups, where the axes correspond to the two attributes. Fig. 6(b) illustrates an event group distribution view consisting of 4 event groups. As illustrated in the figure, the different combinations of the two attribute values roughly divides the event group distribution space into four regions:

Region I Event groups in this region have a low range, but high density. This indicates rare occurrence of such events, and can be used to identify irregular patterns of the data. The green event group shows one such example – it consists of events that happen over two consecutive hours.

Region II Event groups in this region have high range and high density, thus implying that such events occur over frequent periods throughout the given time interval. Event groups in this region can be used to identify trends in the data. The blue event group consists of similar events that occur every hour during a 7 day period, and therefore has both high range as well as density.

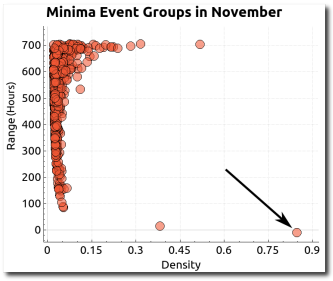


Fig. 7. The event group distribution view for November 2011 after filtering. Note that the event group corresponding to Macy's Parade stands out in this view, thus helping the user identify this pattern.

Region III Event groups in this region consists of a small number of events that span a large range. Such events could move to Region II at a lower time resolution, and could potentially represent patterns that are regular over a large time interval, but irregular with respect to the range of the input data. The orange event group contains events that occur over two weeks, but only for one hour per week. Note that this event group moves to Region II (Fig. 6(d)) when the time resolution is changed to a week.

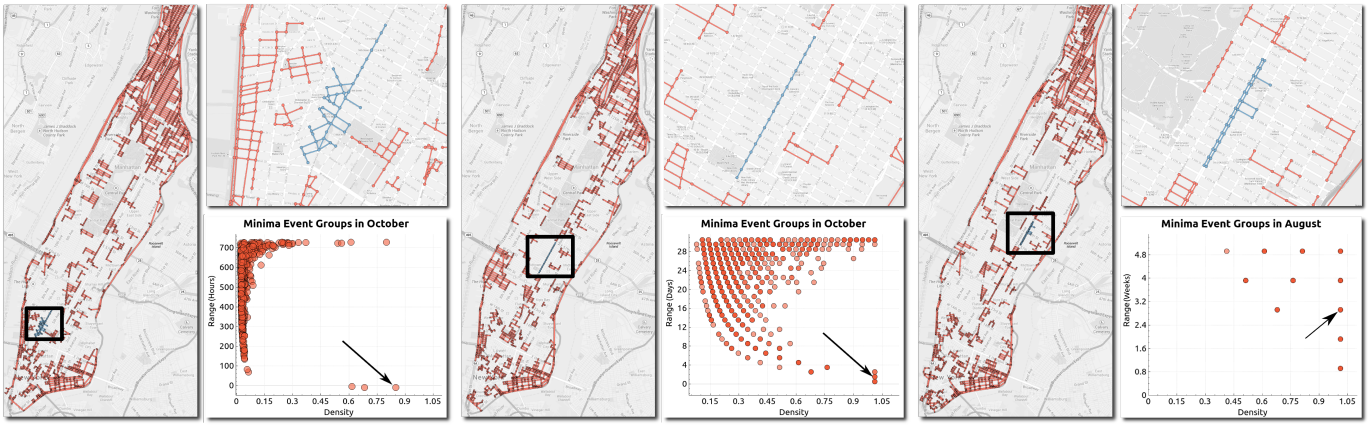
Region IV Event groups in this region have low range as well as low density. However, such events could move to Region I at a lower time resolution. Fig. 6(c) shows an example where the purple event group, having events over two days, moves from Region IV to Region I when the time resolution is changed to a day.

As we show later in the paper, this view acts as a powerful device to help identify many interesting patterns. Using it in conjunction with the query interface simplifies the exploration of large data sets.

While the distribution view gives an overview of an event group, its exact periodicity cannot be inferred accurately. This is instead accomplished using the timeline view which visualizes the individual events in an event group over time (see Fig. 6(a)). This view is inspired by the Gantt chart visualization [24] which is commonly used to represent activities (events) over time. Event groups are represented by a horizontal sequence of points, each point representing individual events. The x-axis in this view represents time. In the timeline view, the event groups are sorted based on their topological importance in order to help the user in identifying significant event groups (the most important event group is on the top). Fig. 6(a) shows the timeline view containing the four event groups described above.

6.3 Filtering Interface

Given the possibly large number of event groups, the ability to filter them not only helps to remove spurious events, but it also allows users to focus on specific types of events. Our visual exploration interface



(a) (b) (c)

Fig. 8. Minima events in NYC. **(a)** Selecting the lone high density event group from Region I of October's event distribution view identifies the event corresponding to the Halloween Parade in 2011. **(b)** Changing the time unit to daily, and selecting high density event groups with range = 2 identifies events corresponding to Hispanic Day Parade and Columbus Day Parade that occurred on consecutive days. **(c)** Changing the time unit to weekly for the month of August, and selecting high density event groups with range = 3 identifies events corresponding to the NYC Summer streets that occurred on three consecutive Saturdays.

supports multiple property-based filters:

Event group size allows users to filter event groups based on the number of events constituting that group. The user can specify both the maximum and minimum size using this filter. For example, if the user is interested in patterns that happen for at least 4 hours, then minimum size should be set to 4.

Event size allows the users to filter event groups based on the geometric size of the events in the group. The geometric size of an event is defined as the size of the subgraph representing that event. For example, the user might be interested in viewing only events that span at least 10 nodes.

Event time allows the user to filter events for a particular time period. For example, the user can search for events that occur only at night.

Spatial region allows the user to select regions on the map and filter out events that occur outside this region.

Fig. 7 shows the event group distribution view for November 2011 after the application of the first two filters. The minimum event group size was set to 4, and the minimum event size was set to 10. The lone highlighted event group in Region I corresponds to Macy's Parade.

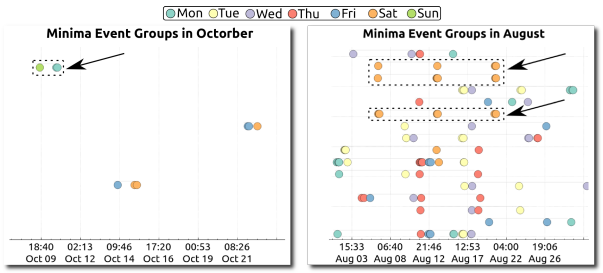
Multiple event groups can overlap in the distribution view. To help identify event groups that are occluded in the distribution view, we also allow users to filter event groups using the distribution view, and visualize the selected events in the time line view.

7 Case Studies

In this section, we present scenarios that illustrate the features and benefits of event-guided exploration for two data sets: NYC taxi trips and subway data (Section 4). The supplemental video demonstrates the use of the visual interface in the exploration of some of the results presented in this section.

7.1 NYC Taxi data

We applied our technique on the density scalar function derived from the NYC taxi data. In what follows, we describe a use case where we explored events at different time granularities and queried for similar events. Motivated by a problem posed by the TLC, we also looked for trends in the data to help them identify areas with high concentrations of taxis. We have been collaborating with experts from the DoT and the TLC who are currently using TaxiVis [22] to analyze the taxi data. We demonstrated the event-guided exploration framework to them. Their feedback was very positive and they expressed interest in using the framework together with TaxiVis to improve policy decisions in their respective organizations.



(a) (b)

Fig. 9. Using the timeline view to isolate events. **(a)** Identifying daily events. The most important event group (the topmost event group) in the timeline view corresponds to the daily event corresponding to the Hispanic Day Parade and Columbus Day Parade. **(b)** Identifying weekly events. The timeline view is used to select a periodic event group, which corresponds to the NYC summer streets that happened on Park avenue for 3 consecutive Saturdays.

7.1.1 Minima events in NYC

Minima events are of interest for the taxi data, since they provide information about regions where there are comparatively fewer taxis. If such minima events occur in places where there is usually a high density of taxis, then this implies blockage of streets. Policy makers in the DoT are particularly interested in identifying such road blocks. By analyzing its spatial location and frequency of occurrence, they are interested in putting in place policies to help them handle such situations. Since such phenomena occur rarely, we focus on Region I of the distribution view to identify such events. We categorize the events into hourly, daily, and weekly respectively, denoting the periodicity of the event. In this section, we briefly discuss a few examples of events that we found while exploring different months.

Hourly events. Fig. 8(a) shows the event group distribution view for the month of October in 2011. Filtering out event groups having less than 4 events and event size greater than 10, and selecting the highlighted event group reveals an event that occurred along Sixth avenue in Greenwich Village on October 31st. This corresponds to the annual NYC Halloween Parade. As shown in the running example, we also find the Macy's Thanksgiving Parade in November. Fig. 1 illustrates the process used to identify the NYC Five Boro Bike Tour that happened on 1 May 2011. Using similar a similar process, we were able to find many other events such as the New year's eve ball drop, St. Patrick's Day Parade in March, etc.

Date	Event
17 March 2011	St. Patrick's Day Parade
7 October 2011	Pulaski Day Parade
10 September 2011	Labor Day Parade
8 September 2012	Labor Day Parade
8 October 2012	Columbus Day Parade
14 October 2012	Hispanic Day Parade
11 November 2012	Veterans Day Parade

Table 1. Event groups similar to Hispanic Day Parade.

Daily events. Fig. 8(b) shows the event group distribution view for October when the time unit is changed to days. The highlighted high density point with range = 2 consists of multiple event groups. However, using the timeline view to choose the most important event group identifies the one that happened on Fifth avenue on October 9th and 10th, 2011 (see Fig. 9(a)). This corresponds to the Hispanic Day Parade on 9th October and the Columbus Day Parade on October 10th.

This change in the resolution of the time unit essentially helps in boosting the density of periodic events that happen on consecutive days, but for only a few hours per day. Note that it will be difficult to isolate this event group in the distribution view when the time unit is an hour since it is part of the dense cluster of points in Region IV (Fig. 8(a)). Exploring the month of May, we also found events corresponding to the 9th avenue Food Festival that happened on May 14th and 15th, 2011.

Weekly events. Changing the time unit to "week", we were able to isolate the event corresponding to the NYC Summer streets that happens on Park avenue as shown in Fig. 8(c). The Summer streets for the Year 2011 occurred on three consecutive Saturdays, 6th, 13th, and 20th August respectively. Note that selecting high density event groups with range equal to 3 weeks results in multiple event groups as shown in the timeline view in Fig. 9(b). This includes event groups in which the events are not periodic. The events in the timeline view are colored based on the day of the week. This helps in visualizing the periodicity of the events, and one can immediately locate the periodic 3 week event group (highlighted in Fig. 9).

7.1.2 Querying events

Querying for events similar to a given pattern is essential for the analyses performed by experts in DoT. Using the interface described in Section 6, we can search for events similar to a selected event that occurs in other months. We now discuss a few results obtained when querying for events similar to the ones automatically detected.

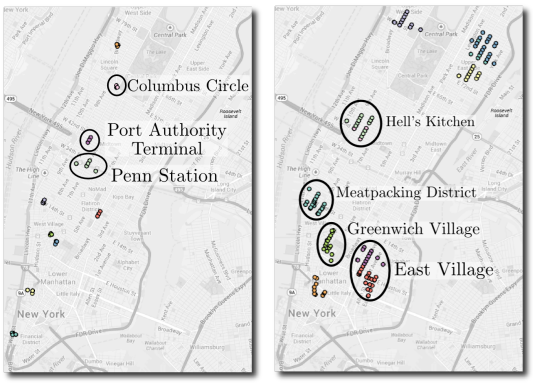
In Fig. 1, searching for events similar to 2011's Five Boro Bike Tour, we find the Five Boro Bike Tour that happened in 2012 together with the Dominican Day Parades that happened in 2011 and 2012. Additionally, we also find that the Gaza solidarity protest was held at the same location on November 18th, 2012.

When querying for patterns similar to Hispanic day parade, we were able to find other parades that also occurred in the same location. Table 1 lists the set of events similar to the Hispanic day parade. Similarly, the query with the New year's ball drop event on December 31st, 2011 returns the same event from January 1st, 2012 and 2011, and 31 December 2012.

7.1.3 Identifying trends

Maxima events show high concentration of taxis. If such concentrations are frequent, then it could imply taxi hot spots. Experts from TLC are particularly interested in identifying such locations. They intend to install data receivers throughout Manhattan to collect data at regular intervals from all the taxis. By placing the receivers at strategic locations such as hotspots they hope to optimize the amount of hardware used. The location of these hot spots can also identify interesting regions in Manhattan.

Selecting highly frequent maxima event groups from Region II of the event distribution view locates the different taxi hot spots in NYC. Fig. 10(a) shows the top 10 hot spots for the month of November 2011. Note that the frequent hot spots include transit locations such as the New York Penn station and the Port Authority Bus terminal, in addition to tourist locations such as Central Park (Columbus circle).



(a) (b)

Fig. 10. Identifying trends. **(a)** Taxi hot spots correspond to areas of high activity in NYC. These include transit locations and tourist spots. **(b)** Taxi hot spots during the night corresponds to areas of active nightlife.

By switching to lower time resolution (weekly) and viewing event groups that consist of only events from 9 pm to 6 am, helps us identify various places of nightlife in Manhattan, as shown in Fig. 10(b). This includes areas popular for restaurants and night clubs in Lower Manhattan such as Greenwich Village, East Village, and Meatpacking district, in addition to the Hell's Kitchen region in Midtown. Note that there is also a high concentration of taxis during this time on both Upper East Side and Upper West Side. Experts from TLC pointed out that this was because a lot of people in that area use taxis to return home late night. It is interesting to note that the frequency of these events was more prominent during weekends compared to weekdays.

7.2 MTA data

To identify events related to delays, we used the average delay of trains at a given station as the scalar function. Since we are only interested in the amount of delay, topological persistence is used as the importance measure for this data set. As we discuss below, we have presented the automatically identified events to engineers at the MTA, who not only provided interesting insights, but were also piqued by some of them.

7.2.1 Identifying trends

Minimum event groups. A minimum event for this function corresponds to a station at which the delay is lower than that of its neighbors. This also signals the station where trains start to get delayed. Therefore, a frequent presence of such events could indicate a problematic situation at a station that needs to be investigated. Such frequent patterns are represented as event groups in Region II of the event group distribution view. We now discuss a few such interesting event groups along with their implications.

Fig. 11 shows the event group distribution view in August 2013 for the southbound *Line 3* trains, which run from Harlem in Manhattan to New Lots Avenue in Brooklyn. A frequent daily minimum event group for this month corresponded to the Wall Street station. Plotting the frequency distribution of events in this group across both hours of the day, as well as days of the week indicates that such an event occurs at this station predominantly during the rush hour period on weekdays. Note that a large number of people use Wall street station, with it being in the middle of the financial district that houses a lot of offices. We noticed this pattern even for other months.

According to MTA engineers, the delay at this station is due to two reasons. Because the station is small, it can be difficult for passengers to board on or off the trains due to the crowds. More importantly, passengers tend to hold doors in order to allow other passengers to board, thus delaying the train from leaving the station. A similar daily minimum event group was also found on the 14^{th} street station. Here the delay is because the 3 train sometimes waits for the 1 train to arrive in order to allow passenger transfer between them.

When considering northbound Line 3 trains, in addition to event groups similar to that of the southbound line, we also found a frequent event group at the Borough Hall station. The MTA engineers

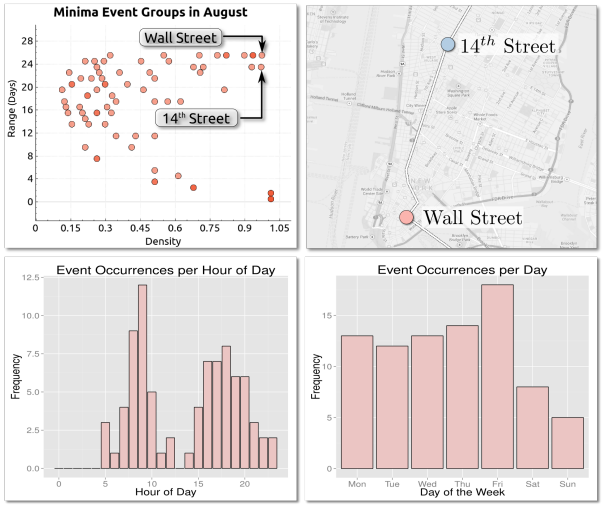


Fig. 11. Frequent minima events for August 2013 for the south bound 3 trains. Two of the event groups in Region II of the event group distribution view (daily events) corresponds to the Wall Street and the 14^{th} Street stations. Frequency distribution of events in the minimum event group corresponding to Wall street station indicates that this event predominantly occurs during rush hours, i.e., between 8 and 10am and between 4 and 8pm. Also note that the frequency of this event is higher on weekdays than on weekends.

did not expect delays to occur at this station, and they were interested in investigating further as to why it occurs. Another unexpected delay was found at the Dyckman Street station along the northbound Line 1 trains. This was interesting for the engineers because this delay occurred predominantly during late nights and early hours of the day.

Maximum event groups. A maximum event for the computed delay function corresponds to a station from which the train makes up on time. One such frequent maximum event group was found on 34^{th} street Penn station for the Line 3 southbound train. This is because the stop after Penn station, which is 14^{th} street, is far from Penn station. Additionally, this stretch of the route is straight allowing the train to speedup and make up on time. Other frequent maximum event groups we found were also along such straight stretches of the route.

8 Discussion and Future Work

Event structure. The structure of topological features depends on the input scalar function. Small changes in the scalar function can change the geometric size of the events found, causing the geometry of the actual event to be split across multiple topological features. For example, the event group found for the NYC 5 Boro bike tour consists of 2 events corresponding to time steps 8 am and 9 am respectively. However when the map view for one of those time steps is viewed, one can find that there exists another event that also corresponds to the bike tour. Fig. 12 illustrates this phenomena, where the blue region is part of the event group identified by exploring the visualization interface. However, by *guiding* the user to this time step, the user can also find the red region. The two regions represents a huge portion of the path taken by bike tour in Manhattan that starts at Battery Park and passes through 6^{th} avenue into Central Park.

As a side effect, this also causes some of the events not to be part of the event group. For example, while the roads are blocked for the Macy's Parade from early hours of the day until 11 am, the event group corresponding to it consists of events from only a subset of these time steps. Again, once the users are assisted towards one of these time steps, they can find the others by exploring close by time steps. It will, however, be interesting to explore techniques to allow combining such split events into a single event.

Similarity computation. The grouping of events depends on the thresholds ε_δ and ε_τ. For the results reported in this paper, we used $\varepsilon_\delta = 0.3$ and $\varepsilon_\tau = 0.2$. While these values gave good results for both the taxi and subway data sets, it would be interesting to explore methods that help identify a good threshold based on the data. The ordering of events when computing event groups can also affect the number of groups found. We plan to explore the effects of event order in the quality of the event groups found.

Fig. 12. Due to the impact of the scalar function on the shape of the events found, the path taken by the bike tour is split into multiple regions.

Currently, we restrict events in a group to be present in the same spatial location. Users might also be interested in the occurrence of events that have a certain shape. For example, using the taxi data, user might be interested to find events where a long stretch of an avenue is blocked irrespective of the location. This is a difficult problem which we intend to explore in future work.

Scalar function computation. While we show two possible transformations of the raw data to scalar functions, we expect many other scalar functions to be useful. The design of the scalar function is dependent on the application. For example, another possible scalar function that can be computed using the MTA data set is the average waiting time of a passenger at each station. MTA engineers are interested in maintaining high frequency between trains especially during peak hours. Hence a frequent maxima event group of such a scalar function would help in identifying stations which have significant waiting times.

Another parameter in designing the time-varying scalar function is the time interval used to convert the temporal component into a set of discrete time steps. We chose an hourly interval since it was small enough to not conceal short events, but was large enough to provide sufficient data to avoid spurious events.

9 Conclusion

In this paper we introduced a topology-based technique for event-guided exploration of urban data, which aims to extend the capabilities of current visual analysis systems by guiding users towards interesting portions of the data. Our technique uses efficient algorithms to capture topological features of spatio-temporal data and enables flexible exploration of events by grouping and indexing events. As demonstrated in a variety of case studies, the proposed approach is efficient, scales to large data sets, and is able to handle a wide range of event types. As discussed in Sec. 8 our work opens many directions for future work, both in exploring possibilities to improve the event detection mechanism and also in user interaction with this novel exploration tool. Finally, we note that while this paper focuses on data from urban environments, we expect our technique to be useful even for other types of spatio-temporal data obtained, such as data produced by scientific simulations and experiments.

Acknowledgments

The authors thank the TLC, DoT, and MTA 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, and NSF award CNS-1229185.

REFERENCES

[1] P. K. Agarwal, H. Edelsbrunner, J. Harer, and Y. Wang. Extreme Elevation on a 2-manifold. *Disc. Comput. Geom.*, 36(4):553–572, 2006.

[2] G. Andrienko and N. Andrienko. Spatio-temporal Aggregation for Visual Analysis of Movements. In *Proc. of IEEE VAST*, pages 51–58, 2008.

[3] G. Andrienko, N. Andrienko, P. Bak, D. Keim, and S. Wrobel. Visual Analytics Focusing on Spatial Events. In *Visual Analytics of Movement*, pages 209–251. Springer Berlin Heidelberg, 2013.

[4] G. Andrienko, N. Andrienko, G. Fuchs, A.-M. O. Raimond, J. Symanzik, and C. Ziemlicki. Extracting Semantics of Individual Places from Movement Data by Analyzing Temporal Patterns of Visits. In *Proceedings of The First ACM SIGSPATIAL International Workshop on Computational Models of Place*, COMP '13, pages 9:9–9:16. ACM, 2013.

[5] G. Andrienko, N. Andrienko, C. Hurter, S. Rinzivillo, and S. Wrobel. From Movement Tracks through Events to Places: Extracting and Characterizing Significant Places from Mobility Data. In *Proc. of IEE VAST 2011*, pages 161–170. IEEE, 2011.

[6] G. Andrienko, N. Andrienko, C. Hurter, S. Rinzivillo, and S. Wrobel. Scalable Analysis of Movement Data for Extracting and Exploring Significant Places. *IEEE TVCG*, 19(7):1078–1094, July 2013.

[7] T. F. Banchoff. Critical Points and Curvature for Embedded Polyhedral Surfaces. *Am. Math. Monthly*, 77:475–485, 1970.

[8] P.-T. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell. Analyzing and Tracking Burning Structures in Lean Premixed Hydrogen Flames. *IEEE TVCG*, 16(2):248–260, Mar. 2010.

[9] H. Bunke and K. Shearer. A Graph Distance Metric Based on the Maximal Common Subgraph. *Pattern Recogn. Lett.*, 19(3):255–259, 1998.

[10] H. Carr, J. Snoeyink, and U. Axen. Computing Contour Trees in All Dimensions. *Comput. Geom. Theory Appl.*, 24(2):75–94, 2003.

[11] H. Carr, J. Snoeyink, and M. van de Panne. Simplifying Flexible Isosurfaces Using Local Geometric Measures. In *Proc. IEEE Visualization*, pages 497–504, 2004.

[12] J. Chae, D. Thom, H. Bosch, Y. Jang, R. Maciejewski, D. S. Ebert, and T. Ertl. Spatiotemporal Social Media Analytics for Abnormal Event Detection and Examination using Seasonal-Trend Decomposition. In *Proc. of IEEE VAST 2012*, pages 143–152. IEEE, 2012.

[13] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[14] R. Chang, G. Wessel, R. Kosara, E. Sauda, and W. Ribarsky. Legible Cities: Focus-Dependent Multi-Resolution Visualization of Urban Relationships. *IEEE TVCG*, 13(6):1169–1175, 2007.

[15] Y.-J. Chiang, T. Lenz, X. Lu, and G. Rote. Simple and Optimal Output-Sensitive Construction of Contour Trees Using Monotone Paths. *Comput. Geom. Theory Appl.*, 30(2):165–195, 2005.

[16] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 2001.

[17] H. Doraiswamy, V. Natarajan, and R. S. Nanjundiah. An Exploration Framework to Identify and Track Movement of Cloud Systems. *IEEE TVCG*, 19(12):2896–2905, 2013.

[18] H. Edelsbrunner and J. Harer. Persistent Homology — A Survey. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Surveys on Discrete and Computational Geometry. Twenty Years Later*, pages 257–282. Amer. Math. Soc., Providence, Rhode Island, 2008. Contemporary Mathematics 453.

[19] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Amer. Math. Soc., Providence, Rhode Island, 2009.

[20] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale Complexes for Piecewise Linear 3-Manifolds. In *Proc. Symp. Comput. Geom.*, pages 361–370, 2003.

[21] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological Persistence and Simplification. *Disc. Comput. Geom.*, 28(4):511–533, 2002.

[22] 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*, 19(12):2149–2158, 2013.

[23] A. T. Fomenko and T. L. Kunii, editors. *Topological Modeling for Visualization*. Springer Verlag, 1997.

[24] H. L. Gantt. *Work, Wages, and Profits*. Engineering Magazine Co., 1913.

[25] Y. Gu and C. Wang. itree: Exploring Time-Varying Data Using Indexable Tree. In *IEEE PacificVis*, pages 137–144, 2013.

[26] J. Gudmundsson, P. Laube, and T. Wolle. Computational Movement Analysis. In *Springer Handbook of Geographic Information*, pages 423–438. Springer, 2012.

[27] A. Hatcher. *Algebraic Topology*. Cambridge U. Press, New York, 2002.

[28] M. Hoai and F. De la Torre. Max-Margin Early Event Detectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2863–2870, 2012.

[29] M. Hu, S. Liu, F. Wei, Y. Wu, J. Stasko, and K.-L. Ma. Breaking News on Twitter. In *Proceedings of the ACM annual conference on Human Factors in Computing Systems*, pages 2751–2754, 2012.

[30] H. Janetzko, F. Stoffel, S. Mittelstädt, and D. A. Keim. Anomaly Detection for Visual Analytics of Power Consumption Data. *Computers & Graphics*, 38:27–37, 2014.

[31] J. Kasten, I. Hotz, B. Noack, and H.-C. Hege. Vortex merge graphs in two-dimensional unsteady flow fields. In *EuroVis - Short Papers*, pages 1–5, 2012.

[32] M. Kulldorff. A Spatial Scan Statistic. *Communications in Statistics-Theory and methods*, 26(6):1481–1496, 1997.

[33] M. Kulldorff, F. Mostashari, L. Duczmal, W. Katherine Yih, K. Kleinman, and R. Platt. Multivariate Scan Statistics for Disease Surveillance. *Statistics in Medicine*, 26(8):1824–1833, 2007.

[34] D. Laney, P. T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the Structure of the Turbulent Mixing Layer in Hydrodynamic Instabilities. *IEEE TVCG*, 12(5):1053–1060, Sept. 2006.

[35] L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for Real-Time Exploration of Spatiotemporal Datasets. *IEEE TVCG*, 19(12):2456–2465, 2013.

[36] S. Maadasamy, H. Doraiswamy, and V. Natarajan. A Hybrid Parallel Algorithm for Computing and Tracking Level Set Topology. In *Proc. Intl. Conf. High Performance Computing*, pages 12.1–12.10, 2012.

[37] R. Maciejewski, S. Rudolph, R. Hafen, A. Abusalah, M. Yakout, M. Ouzzani, W. S. Cleveland, S. J. Grannis, M. Wade, and D. S. Ebert. Understanding Syndromic Hotspots-A Visual Analytics Approach. In *Proc. of IEEE VAST 2008*, pages 35–42. IEEE, 2008.

[38] E. McFowland III, S. Speakman, and D. B. Neill. Fast Generalized Subset Scan for Anomalous Pattern Detection. *Journal of Machine Learning Research*, 14:1533–1561, 2013.

[39] J. Milnor. *Morse Theory*. Princeton Univ. Press, New Jersey, 1963.

[40] NYC MTA API. `http://web.mta.info/developers/`.

[41] D. B. Neill and G. F. Cooper. A Multivariate Bayesian Scan Statistic for Early Event Detection and Characterization. *Machine learning*, 79(3):261–282, 2010.

[42] D. B. Neill, A. W. Moore, and G. F. Cooper. A Bayesian Spatial Scan Statistic. *Adv. Neur. In.*, 18:1003, 2006.

[43] Chicago Open Data. `https://data.cityofchicago.org/`.

[44] NYC Open Data. `http://data.ny.gov`.

[45] Seattle Open Data. `http://data.seattle.gov`.

[46] V. Pascucci and K. Cole-McLaughlin. Parallel Computation of the Topology of Level Sets. *Algorithmica*, 38(1):249–268, 2003.

[47] V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, editors. *Topological Methods in Data Analysis and Visualization*. Springer, 2010.

[48] V. Pascucci, G. Weber, J. Tierny, P.-T. Bremer, M. Day, and J. Bell. Interactive Exploration and Analysis of Large-Scale Simulations Using Topology-Based Data Segmentation. *IEEE TVCG*, 17(9):1307–1324, 2011.

[49] R. E. Roth. An Empirically-Derived Taxonomy of Interaction Primitives for Interactive Cartography and Geovisualization. *IEEE TVCG*, 19(12):2356–2365, 2013.

[50] R. W. Scholz and Y. Lu. Detection of Dynamic Activity Patterns at a Collective Level from Large-Volume Trajectory Data. *International Journal of Geographical Information Science*, (ahead-of-print):1–18, 2014.

[51] G.-D. Sun, Y.-C. Wu, R.-H. Liang, and S.-X. Liu. A Survey of Visual Analytics Techniques and Applications: State-of-the-Art Research and Future Challenges. *J. of Comp. Sci. and Tech.*, 28(5):852–867, 2013.

[52] Twitter API. `https://dev.twitter.com/`.

[53] J. Wakefield and A. Kim. A Bayesian Model for Cluster Detection. *Biostatistics*, 14(4):752–765, 2013.

[54] Z. Wang, M. Lu, X. Yuan, J. Zhang, and H. v. d. Wetering. Visual Traffic Jam Analysis Based on Trajectory Data. *IEEE TVCG*, 19(12):2159–2168, 2013.

[55] W. Widanagamaachchi, C. Christensen, P.-T. Bremer, and V. Pascucci. Interactive Exploration of Large-Scale Time-Varying Data Using Dynamic Tracking Graphs. In *Proc. of IEEE LDAV*, pages 9–17, 2012.

[56] P. H. T. Zannin, M. S. Engel, P. E. K. Fiedler, and F. Bunn. Characterization of Environmental Noise Based on Noise Measurements, Noise Mapping and Interviews: A Case Study at a University Campus in Brazil. *Cities*, 31:317–327, 2013.