The SIGSPATIAL Special

The SIGSPATIAL Special is the newsletter of the Association for Computing Machinery (ACM) Special Interest Group on Spatial Information (SIGSPATIAL).

ACM SIGSPATIAL addresses issues related to the acquisition, management, and processing of spatially-related information with a focus on algorithmic, geometric, and visual considerations. The scope includes, but is not limited to, geographic information systems.

Current Elected ACM SIGSPATIAL officers are:

- Chair, Cyrus Shahabi, University of Southern California
- Past Chair, Mohamed Mokbel, University of Minnesota
- Vice-Chair, Goce Trajcevski, Iowa State University
- Secretary, Egemen Tanin, University of Melbourne
- Treasurer, John Krumm, Microsoft Research

Current Appointed ACM SIGSPATIAL officers are:

- Newsletter Editor, Andreas Züfle, George Mason University
- Webmaster, Chrysovalantis (Chrys) Anastasiou, University of Southern California

For more details and membership information for ACM SIGSPATIAL as well as for accessing the newsletters please visit [http://www.sigspatial.org](http://www.sigspatial.org).

The SIGSPATIAL Special serves the community by publishing short contributions such as SIGSPATIAL conferences’ highlights, calls and announcements for conferences and journals that are of interest to the community, as well as short technical notes on current topics. The newsletter has three issues every year, i.e., March, July, and November. For more detailed information regarding the newsletter or suggestions please contact the editor via email at azufle@gmu.edu.

Notice to contributing authors to The SIGSPATIAL Special: By submitting your article for distribution in this publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights:

- to publish in print on condition of acceptance by the editor,
- to digitize and post your article in the electronic version of this publication,
- to include the article in the ACM Digital Library,
- to allow users to copy and distribute the article for noncommercial, educational or research purposes.

However, as a contributing author, you retain copyright to your article and ACM will make every effort to refer requests for commercial use directly to you.

Notice to the readers: Opinions expressed in articles and letters are those of the author(s) and do not necessarily express the opinions of the ACM, SIGSPATIAL or the newsletter.

The SIGSPATIAL Special (ISSN 1946-7729) Volume 10, Number 1, March 2018.
# Table of Contents

Message from the Editor.  
*Andreas Züfle*  
1

## Section 1: Special Issue on Urban Analytics and Mobility (Part 1)

### Introduction to this Special Issue  
*Andreas Züfle*  
2

### A Brief Overview of Machine Learning Methods for Short-term Traffic Forecasting and Future Directions  
*Yaguang Li, Cyrus Shahabi*  
3

### Time-Dependent Route Scheduling on Road Networks  
*Lei Li, Jiwon Kim, Jiajie Xu, Xiaofang Zhou*  
10

### High Definition Maps in Urban Context  
*Andi Zhang, Xin Chen, Goce Trajcevski*  
15

## Section 2: SIGSPATIAL 2017 Event Reports

### Conference Report:  
The 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2017) Redondo Beach, California, USA  
*Goce Trajcevski, Siva Ravada, Roberto Tamassia*  
21

### ACM SIGSPATIAL GIS Cup 2017 - Range Queries under Fréchet Distance  
*Martin Werner, Dev Oliver*  
24
Message from the Editor

Andreas Züfle
Department of Geography and GeoInformation Science, George Mason University, USA
Email: azufle@gmu.edu

Dear SIGSPATIAL Community,

The newsletter serves the community by publishing short contributions such as SIGSPATIAL conferences’ highlights, calls and announcements for conferences and journals that are of interest to the community, as well as short technical notes on current topics.

In the first section of this March 2018 issue, we have a special issue on the topic of “Urban Analytics and Mobility”. The choice for this topic follows the rapid trend of the last years: The UrbanGIS’17 workshop at SIGSPATIAL 2017 has grown to be the largest workshops, a large number of papers and research sessions are focusing on related topics, and the SIGSPATIAL 2017 keynote by Bryan Mistele, Founder & CEO of INRIX, discussed many future challenges in urban environments.

The second section consists of SIGSPATIAL 2017 event reports, including the main conference report and a report of the 2017 SIGSPATIAL CUP.

I want to sincerely thank all authors of for their generous contributions of time and effort that made this issue possible. I hope that you will find the newsletters interesting and informative and that you will enjoy this issue.

You can download all Special issues from:

http://www.sigspatial.org/sigspatial-special

Yours sincerely,
Andreas Züfle
SIGSPATIAL Newsletter Editor
The SIGSPATIAL Special

Section 1: Special Issue on Urban Analytics and Mobility (Part 1)

ACM SIGSPATIAL
http://www.sigspatial.org
Introduction to this Special Issue: 
Urban Analytics and Mobility (Part 1)

Andreas Züfle  
Department of Geography and GeoInformation Science, George Mason University  
Email: azufle@gmu.edu

Mobility in Urban environments is a problem of global scale. According to the latest INRIX Global Traffic Scorecard \[1\], drivers in the United States wasted $305 billion in 2017 alone. This number is derived from more than 11 billion liters of wasted fuel per year, and 6.9 billion of man-hours stuck in traffic per year \[2\]. In addition, the INRIX study shows that this problem is global. Measured per-capita, people in Russia and Thailand spend even more time in traffic, while Brazil, South Africa, the UK, and Germany are only slightly behind the US.

The SIGSPATIAL community has always been working diligently on providing new ideas, algorithms and solutions to alleviate this problem, shown by many publications on traffic prediction, travel time optimization, route planning and related topics\[1\]. At the same time, this research field is changing rapidly. As discussed by Bryan Mistele, Founder & CEO of INRIX, in his keynote at SIGSPATIAL 2017\[2\], “the mega-trends of Autonomous, Connected, Electric and Shared vehicles (the ‘ACES’) are transforming transportation”.

Following this discussion, this special issue of the SIGSPATIAL Special Newsletter contains three articles which present visions, challenges, and solutions to improve transportation issues in urban environments.

- In the first article, Li and Shahabi give a brief overview of machine learning methods for short-term traffic forecasting and discuss future directions,
- in the second article, Li, Kim, Xu and Zhou present an introduction to time-dependent route scheduling, thus using speed profiles of a network for improved routing,
- in the third article, Zang, Chen and Trajcevski discuss the challenge of employing high definition maps in urban context, in order to improve self-localization of vehicles for more efficient navigation.

I would like to thank the authors for their contributions, and I hope the readers will enjoy reading this issue and find it useful in their research work.

References


\[1\] For example, see the SIGSPATIAL 2017 Program: [https://sigspatial2017.sigspatial.org/program/](https://sigspatial2017.sigspatial.org/program/)
\[2\] [https://sigspatial2017.sigspatial.org/keynotes/#bryan](https://sigspatial2017.sigspatial.org/keynotes/#bryan)
Abstract

Short-term traffic forecasting is a vital part of intelligent transportation systems. Recently, the combination of unprecedented data availability and the repaid development of machine learning techniques have brought on immense advancement in this field. In this paper, we aim to provide a brief overview of machine learning approaches for short-term traffic forecasting to facilitate research in related fields. We first introduce traffic forecasting and the challenges, and then introduce different approaches for modeling the temporal and/or spatial dependencies. Finally, we discuss several important directions for the future research.

1 Introduction

Traffic forecasting is the core component of intelligent transportation systems (ITS). The goal of traffic forecasting is to estimate future traffic conditions of a transportation network based on historical observations. Based on the forecasting horizon, traffic forecasting can be categorized as short-term forecasting and long-term forecasting. In this paper, we will focus on short-term traffic forecasting, whose forecasting horizon is usually less than or equal to one hour. Short-term traffic forecasting is important for various applications, including route planning [17], traffic control [10], car dispatching [1], etc.

Figure 1: Complex spatial dependency among different traffic time series. Reprinted from [19] with permission.
This problem is challenging mainly due to the complex spatial and temporal dependencies [19]. On the one hand, traffic time series demonstrate strong temporal dynamics. Recurring incidents such as rush hours or accidents can result in formation of non-stationary time series, rendering forecast challenging. On the other hand, sensors on the road network contain complex yet unique spatial correlations. Figure 1 illustrates an example of spatial dependencies among different traffic time series. Suppose there are traffic sensors on three roads, i.e., sensor 1, 2 and 3. The traffic time series of sensor 1 and 2 are correlated, while those of sensor 1 and sensor 3 are not. Though road 1 and road 3 are close in the Euclidean space, they demonstrate very different behaviors.

Traffic forecasting has been studied in various communities ranging from transportation system [29, 31], through economics [28, 9], and to data mining [21, 20], and its methods mainly fall into two categories: knowledge-driven approach and data-driven approach. In transportation and operational research, knowledge-driven methods usually try to computationally model the transportation network through queuing theory and simulating driver behaviors in traffic [8, 4]. With the availability of increasing amount of traffic data, data-driven machine learning approaches for traffic forecasting have received considerable attention. In this paper, we will give a brief overview of different data-driven machine learning approaches for short-term traffic forecasting and describe potentially future directions in this field.

2 Overview of Short-term Traffic Prediction Approaches

In this section, we give a brief overview of different traffic forecasting approaches. These approaches are categorized into two types based on whether they model the spatial correlation among different traffic time series.

2.1 Traffic Forecasting without Modeling Spatial Dependency

Traffic forecasting can be modeled as a time series regression problem and thus various time series analysis approaches have been applied to this problem.

Historical Average models the traffic flow as a seasonal process, and uses the weighted average of previous seasons as the prediction. For example, suppose the season is 1 week, then the prediction for this Wednesday is the averaged traffic speeds from last four Wednesdays. As the historical average method does not depend on short-term data, its performance is invariant to the small increases in the forecasting horizon. Auto-regressive integrated moving average (ARIMA) is a popular model for time series analysis and has been successfully applied to traffic forecasting [16]. ARIMA consists of three parts: 1) the Auto-regressive (AR) part indicates that the evolving variable of interest can be approximated using a linear combination of its own historical values, 2) the Moving average (MA) part is used to model the residual from the AR part using a weighted combination of random noises at various previous time steps, and 3) the Integrate (I) part models the difference between adjacent values rather than raw values. In [34], the authors use Seasonal ARIMA to capture the periodicity in the traffic flow, while in [26] ARIMA is augmented with historical average to better model the rush hour traffic behavior.

Other popular time series methods for traffic forecasting include K-nearest Neighbor (KNN) [42, 3], Support Vector Regression (SVR) [30], particle filter, Hidden Markov Model [27], Gaussian Process [36], etc. However, these time series models usually rely on the stationary assumption, which is often violated by the real-world traffic data.

To model the non-linear temporal dependency, neural network based approaches have also been applied to traffic forecasting. In [22, 12], the authors propose to use stacked denoising encoder and deep belief networks to model the temporal behavior. In [24, 40, 15], the authors model the temporal dependency using Recurrent

---

1For this and all subsequent reprinted figures, the original authors retain their copyrights, and permission was obtained from the corresponding author.
Neural Networks (RNN), which is a type of neural network with self-connection, and is able to perform non-linear auto-regression. However, the majority of the above-mentioned approaches model each traffic time series separately, failing to capture the spatial dependency among them.

2.2 Traffic prediction with Modeling Spatial Dependency

To capture the spatial dependency among traffic time series, researchers have extended existing approaches to process multivariate time series. The resulted models include Vector Auto-regressive [11], Vector ARIMA [13], Spatiotemporal ARIMA [25], Spatiotemporal HMM [14, 37]. In [6], the authors further propose to first group similar sensors and then perform multi-task learning on each group. An alternative way to model the relationship among different time series is the latent space model which first transforms the raw traffic time series into the latent space and then learns the spatiotemporal dependency. In [39], the authors propose a temporal regularized matrix factorization based approach which performs vector auto-regression in the latent space. While in [7], the authors model the road network as a graph, and propose to learn the attributes of vertices in latent spaces which captures both topological and temporal properties.

However, existing machine learning models either impose strong stationary assumptions on the data (e.g., auto-regressive model) or fail to account for highly non-linear temporal dependency (e.g., latent space model [39, 7]). Deep learning models deliver new promise for time series forecasting problem. To capture the spatial dependency of the traffic, recent studies [35, 23, 41] propose to model the transportation network as an image and use Convolutional Neural Networks (CNN) to extract spatial features. One drawback of these methods is that they ignore the topology of the underlying transportation network, e.g., in Figure 1, two roads in different directions of a highway, though close in Euclidean distance, can have significantly different traffic pattern because of the network topology.

To resolve this issue, Li et al. [19] model the underlying road network as a directed weighted graph and propose diffusion convolutional recurrent neural network (DCRNN) which systematically captures the topological dependency using diffusion convolution. Diffusion convolution is a new form of convolutional operation defined on the graph based on the diffusion nature of traffic. Later, in [38], the authors speed up this model by replacing RNN with CNN to model the temporal dependency. In [5], the authors introduce DeepTransport, which models the spatial dependency by explicitly collecting upstream and downstream neighborhood roads for each individual road and conduct convolution on these neighborhoods. The methods discussed above are not exhaustive, and more related work and details can be found in a recent survey paper [32] and references therein.
3 Discussion and Future Directions

In this section, we present several future directions for traffic forecasting.

Traffic prediction in extreme cases While traffic patterns in normal conditions are easy to predict, a more interesting question in traffic forecasting is to forecast traffic for extreme conditions, which include both peak hours and post-accident traffic forecasting. In [40], the authors propose to learn a representation of accident features with auto-encoder and then combine with recurrent neural network for post-accident traffic forecasting. While improved performance is observed, the proposed model fails to consider the correlation among different sensors and the results can still be improved further.

Fuse traffic prediction with other applications Many important applications in transportation are strongly related to traffic prediction. One example is travel time estimation (ETA) [18]. Currently, traffic prediction and travel time estimation are usually performed independently. It is desirable to have a model that jointly models these two problems and achieves improved results for either task.

Long-term temporal dependency modeling Very long-term temporal dependency usually exists in traffic data, e.g., the current traffic situation can be strongly correlated with a day, a week or even several months ago. Currently, the most popular approaches to model the non-linear temporal dependency is recurrent neural networks (RNN). However, due to the sequential nature of RNN, it is hard to model very long term dependencies [33]. Besides, RNN is not efficient to train as it is hard to parallelize. Thus efficient approaches that are able to capture long-term non-linear temporal dependencies are much needed.

Evaluation metric design Popular metrics to evaluate traffic forecasting include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) which are calculated by averaging across all sensors. These metrics put an equal importance on all sensors and time slots. However, we argue not all sensors and time slots are equally informative w.r.t. evaluating the performance. Figure 2 shows each location’s standard deviation from its historical average. Generally, the larger the standard deviation is the harder to predict the traffic at that location. Arguably, locations and times with higher error, e.g., busy intersections during peak-hours, are more important to predict. Alternatively, predicting the average speed on
all freeways from midnight to 5am is not very difficult. Thus, it might be beneficial to have a metric that give more rewards to predicting traffic at tougher locations and times.

**Interpretable traffic prediction**  Many machine learning models are used for traffic prediction. Though, good performance is achieved, the prediction made by the model are usually not interpretable. As shown in Figure 3, it is desirable to identify which spatial and temporal components affect the model’s prediction. Besides, rather than a single prediction, it is more informative to predict a distribution, e.g., the mean and the variance of the Gaussian distribution, which would help decision making as well as other related applications, e.g., travel time estimation [2].

**Acknowledgments**

This research has been funded in part by NSF grant CNS-1461963, Caltrans-65A0533, the USC Integrated Media Systems Center (IMSC), and the USC METRANS Transportation Center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the sponsors such as NSF.

**References**


Time-Dependent Route Scheduling on Road Networks

Lei Li∗, Jiwon Kim∗, Jiajie Xu#, Xiaofang Zhou∗#

∗ School of Information Technology and Electrical Engineering, The University of Queensland
# School of Computer Science and Technology, Soochow University
{l.li3, jiwon.kim, zxf}@uq.edu.au

Abstract

Navigation has been an important tool for human civilization for thousands of years, and the latest technologies like online map services and GPS satellites have brought it up to a new level. Now people can easily identify where we are on earth, find any places they want to go, and retrieve best routes to get there. Although there are plenty of tools that are convenient and fast enough for basic uses, it is still far from optimal. For example, most systems only consider various type of distances as the optimization goals, while the traveling time, which needs to consider traffic conditions, is a more appropriate one. However, it is both hard to acquire the traffic condition information and to compute time-dependent fastest paths. Therefore, in this article, we present an introduction to the time-dependent route scheduling, from speed profile generation to route scheduling and query answering.

1 Introduction

With the prevalence of GPS enabled devices and mobile network, various of navigation systems have been widely adopted by public transportation systems, logistics companies, private vehicles and a broad range of location-based services. These systems first find where we are on planet earth, then compute a reasonable path to our destinations, most of which are based on shortest path algorithms [4, 9, 15]. During the trip, they provide turn-by-turn navigation using real-time map-matching and real-time path computation. Some of them even keep users’ trajectories, like O2O taxi service providers Uber and DiDi. In fact, they are becoming more and more popular around the world and have obtained a tremendous amount of trajectories generated by their taxi drivers every day. However, although these trajectories can reveal the traffic conditions of different parts of a city at different time periods, they are mostly used for behavior analysis and customer support.

In spite of their popularity, there are still some untreated shortcomings. The most obvious one among them is the lack of considering the traffic condition. The reason for it is two-fold. Firstly, it is hard to describe traffic condition because unlike the distance, it changes over time and is not an inherent property of the road. Secondly, it is hard to compute the fastest path as the query is actually more complicated. Therefore, in this article, we present some techniques to solve these problems.

A road network is usually modeled as a time-dependent graph, where each edge is associated with a function that returns the time cost of traveling from one vertex to another at a given departure time. The set of these functions is called a speed profile. Obviously, with the help of traffic sensors and cameras, we can always get an accurate speed profile. But they are expensive and unrealistic to cover the entire road network. On the other hand, with the vast amount of historical and real-time trajectory data at hand, we are able to derive a speed profile at a much larger scale with little cost [12]. However, it is not straightforward to achieve this. We first need to attach the GPS points of the trajectory data to the roads by map matching [16]. After that, these attached
GPS points are converted to speed data by collecting them into different time slots. Because many roads and time slots may not have any speed data at all, various techniques are proposed to estimate the missing values. Finally, compression is appreciated due to the large size of a raw speed profile.

With the speed profile at hand, the shortest path problem can be extended to the time-dependent scenario and categorized according to the weight function type and defining whether waiting on the vertices or not. The shortest path problem is at one extreme that the weight functions are static and no waiting is allowed. If the weight functions are time-dependent, it is the Single Starting-Time Fastest Path (SSFP) problem. It can be solved in \( O(\|V\| \log \|V\| + \|E\|) \) time if first-in first-out property holds [6]. Earliest Arrival Path and Latest Departure Path are two variations of it. If waiting is allowed on the starting vertex, it is the Interval Starting-Time Fastest Path (ISFP) [10, 5]. It can be viewed as finding an optimal departure time during a given time period that has the shortest total travel time. The problem complexity is \( \Omega(T(\|V\| \log \|V\| + \|E\|)) \) [8] and \( T \) is number of turning points in the destination vertex’s travel cost function. If waiting is further allowed on a set of vertices, the problem is generalized to the Minimal On-Road Time (MORT) problem [11, 13]. In this case, the total travel time is decomposed into waiting time and on-road time, and waiting on some vertices can reduce the on-road time, like avoiding a traffic jam. Logistics companies use it to reduce their fuel consumption and tourists use it to plan their trips. The categories are shown in Table 1.

However, due to the inherent complexity of \( \Omega(T(\|V\| \log \|V\| + \|E\|)) \), the fastest path is slow to compute. Hence, various indexes on paths are extended to time-dependent scenario to speed up query answering. Figure 1 illustrates the categories. On one extreme is the fastest path directly with no index, and on the other extreme is the All Pair pre-computation. Most of the existing approaches fall into the online speed-up category, which adds various precomputed structures like shortcuts [1] and landmarks [3] to prune the searching space. Nevertheless, an expensive searching is still required. Another approach is to extend the 2-Hop labeling [2], which is widely used in the static graph, to the time-dependent environment. However, although it is much faster than the online search approaches, its label size is already big on the static graph, it would be much larger on the time-dependent road network.

The rest of this article is organized as follow: we first present how to derive a speed profile from the trajectory date in Section 2. Then we introduce a general form of a fast path problem MORT in Section 3. In Section 4, we present the time-dependent 2-hop labeling for fast query answering. Finally, we conclude in Section 5.

---

<table>
<thead>
<tr>
<th>Graph Type</th>
<th>Path Problem</th>
<th>Objective</th>
<th>Waiting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Graph</td>
<td>Shortest Path</td>
<td>( \sum_{i=1}^{|E|} \text{Weight}(v_i, v_{i+1}) )</td>
<td>No</td>
</tr>
<tr>
<td>Time-Dependent Graph</td>
<td>SSFP</td>
<td>General</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Earliest Arrival</td>
<td>( \text{Min} (\text{Arrival}(v_i)) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Latest Departure</td>
<td>( \text{Max} (\text{Departure}(v_i)) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISFP</td>
<td>( \text{Min} (\text{Arrival}(v_d) - \text{Departure}(v_s)) )</td>
<td>Starting Vertex</td>
</tr>
<tr>
<td></td>
<td>MORT</td>
<td>( \text{Min} (\sum_{i=1}^{|E|} \text{Time}(v_i, v_{i+1}, \beta(v_i))) )</td>
<td>A set of vertices</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fastest Path</th>
<th>Online Speed-up</th>
<th>2-Hop Labeling</th>
<th>All Pair</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(1) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>( O(</td>
<td>|V|</td>
<td>|V| +</td>
<td>|E|</td>
</tr>
</tbody>
</table>

---

Table 1: Time-Dependent Path Problems

*Weight* is a static function, *Time* is the time-dependent function and \( \beta(v_i) \) is the departure time from \( v_i \). \( v_s \) is the starting vertex and \( v_d \) is the destination vertex.
2 Speed Profile Generation

2.1 Speed Data Collection

A trajectory is a series of GPS points. After matching it on the map, each point is attached to a road. With the timestamp and road length information, we can compute the road speeds at different time points. Next we convert it into a speed profile. The most straightforward method is to use these speed data directly, which would result in a set of linear piecewise speed functions. However, it is not practical because it would be too zigzag and hard to use. Thus, we use a histogram-based approach to collect the speed data. Specifically, we divide one day’s time into T slots with the same length. Then the speed data that fall into the same slot will be added up together to get an average speed. Thus, the influence of the outliers is reduced dramatically. T= 5 minutes has the best performance according to our experiment.

2.2 Missing Value Estimation

Even though the GPS-based trajectory data has a higher coverage of the road network than other approaches, it is still hardly possible to cover every edge. So it also faces the sparsity problem. There are several approaches to solve this problem. Firstly, the speed profile of each road can be viewed as a vector. Then by finding the most similar ones to fill each other, we can estimate the missing values. Moreover, the speed profile of the whole road network is essentially a matrix. Thus, we can take advantage of matrix-factorization based collaborative-filtering, together with other road features to fill the voids [14]. Last but not the least, hidden Markov can also be used to estimate the missing speeds [14].

2.3 Compression

The histogram-based speed profile can be viewed as a type of Time Series Data [7], and compressing a speed profile falls into the category of Time Series Segmentation. Sliding window, top-down and bottom-up are three popular approaches to compress it. The Sliding window algorithm has a linear running time, but it has a bad compression rate, while the bottom-up algorithm takes the longest time to compute but has the best compression rate. After compression, the histogram-based speed profile is changed into a set of linear piecewise functions.

3 Minimal On-Road Time Route Scheduling

3.1 MORT Problem

As described in Section 1, the MORT is the general form of all the other path problems. It is more complicated because it is hard to know if we should wait on a vertex or not, and how long should we wait on it. The problem is solve by computing a minimum cost function $C_i(t)$, which records the minimum on-road travel time from $v_s$ to $v_i$ that arrives $v_i$ on time $t$, for each vertex $v_i$. When the destination’s $C_d(t)$ is computed, the result is found. The minimum cost function is from the perspective of arrival time rather than the departure time for the following reasons: if we depart at the same time, we can have different waiting schedules and different arrival time, which would further result in different on-road time. Therefore, the departure time perspective cannot handle the waiting. If we use the arrival time, we can ignore when we departed, where and how long we waited, because there is always a minimal value at each arrival time. There are two ways to construct the minimum cost functions: we can build the minimum cost functions over the whole query time interval iteratively in a Dijkstra way, or constructs it sub-time-interval by sub-time-interval until the end of the time interval. The key observation of this problem is that the minimum cost function on the waiting vertices is non-increasing because
waiting on a vertex would not increase the on-road time, and we can wait on it until the traffic condition becomes better. By configuring the size of waiting vertex set and departure time interval, we can use it to solve any kind of path problem.

3.2 Approximation Algorithm

The time complexity of the <i>MORT</i> algorithm is significantly affected by the number of turning points in <i>C_d(t)</i>. What is worse, it grows larger as the expansion grows, which makes the computation slower and slower. So the key to speed up is decreasing the number of turning points, especially the useless ones. However, we cannot determine if one turning point will end up with the optimal result until the final <i>C_d(t)</i> is constructed. Therefore, we design an approximation approach that can guarantee the final result is no less than <i>αC_d(t)</i>, <i>α</i> ∈ (0, 1].

Suppose a route is made up of a series of consecutive edges <i>Est = v_1, v_2, ..., v_n</i> and <i>||Est||</i> is the length of <i>Est</i>. If we apply an approximation factor <i>α1</i> on <i>v_1</i>, <i>α2</i> on <i>v_2</i> and so on, the error of the final result does not grow linearly: ||Est’|| = (((<i>e_1α_1 + e_2α_2</i>) + <i>e_3α_3</i> + ... <i>e_nα_n</i>) = <i>α1α2α3...α_n e_1 + α2α3...α_ne_2 + ... + α_ne_n</i> = <i>Π^j=1 α_j e_j</i>. To achieve ||Est’|| ≥ <i>α</i>||Est||, we have to concentrate the pruning power to the latter vertices by setting a global turning point number threshold: Only those vertices whose turning point numbers are larger than <i>ρ</i> will be pruned. Such approximation approach can benefit all the related time-dependent algorithms.

4 Time-Dependent 2-Hop Labeling

Like the distance query on static graph, fastest travel time query can also use the 2-hop labeling approach. For each vertex <i>v</i> ∈ <i>V</i>, we pre-compute two sets of labels: out-labels <i>L_{out}(v_i) = \{v_j, f_{v_i, v_j}(t)\}</i> and in-labels <i>L_{in}(v_i) = \{v_j, f_{v_j, v_i}(t)\}</i>, where <i>v_j</i> is a hop vertex and <i>f_{v_i, v_j}(t)</i> returns the minimal cost from <i>v_i</i> to <i>v_j</i> at different departure time <i>t</i>. We can answer a query only using the labels: <i>Q_f(v_s, v_d, L) = Min(f_{v_s, v_d}(t))</i> = <i>Min(f_{v_s, v_d}(t) \oplus f_{v_i, v_d}(t)) = Min(f_{v_i, v_d}(f_{v_s, v_i}(t)))</i> = <i>f_{v_s, v_d}(t), t ∈ T, v_i ∈ H_{v_s, v_d} = L_{out}(v_s) \cap L_{in}(v_d)</i>, where <i>⊕</i> is the concatenation of two linear piecewise functions and <i>Min(\_)</i> takes the smaller parts of two functions. Although it is fast to answer a query, the index size is huge, which limits its power to small graphs. In order to apply it on an ordinary road network, we first partition the road network into edge-disjoint grids. Then the 2-hop labeling is constructed within each grid and also among the boundary vertices of the grids. The query can be answered by concatenating three partial results: <i>v_s</i> to its boundary vertices <i>G^B_s</i>, <i>G^B_t</i> to <i>G^B_j</i>, and <i>G^B_j</i> to <i>v_d</i>. Our time-dependent 2-hop labeling can boost the query answering time by hundreds of times.

5 Conclusions

In this article, we briefly discuss and introduce the techniques used to answer a time-dependent path query on a road network. The speed profile generation can produce the time-dependent cost functions from trajectory data through map-matching, speed data collection, missing value estimation and compression. Then we present our <i>MORT</i> algorithm, which can answer all the existing path problems. Finally, we describe the time-dependent 2-hop labeling for fast query answering.

Acknowledgement. The work reported in this article is partially supported by the National Nature Science Foundation of China (61472263, 61772356, 61572335, 61402312) and the Australia Research Council (LP130100164, DP170101172).
References


High Definition Maps in Urban Context

Andi Zang\textsuperscript{1}, Xin Chen\textsuperscript{2}, and Goce Trajcevski\textsuperscript{3}

\textsuperscript{1}Department of Electrical Engineering and Computer Science, Northwestern University
\textsuperscript{2}HERE North America
\textsuperscript{3}Department of Electrical and Computer Engineering, Iowa State University

Abstract

Part of the challenges in the quest for smart cities is to enable effective navigation for different types of mobile users: from pedestrians, through drivers, to autonomous vehicles. While the data sources to facilitate such tasks abound, one of the pressing problems is how to enable efficient management and download of the data needed to populate the screens of devices with the appropriate visualization.

In this paper, we present an overview of the different issues and the main features of the existing technologies that, in one way or another, could be used as foundations for effective solution for generating quality maps. We also discuss the possible approaches for addressing such issues in the context of accurate self-localization of vehicles.

1 Introduction and Motivation

Highly accurate, precise, and detailed lane-level maps – also known as High Definition (HD) Maps, as described in Open Lane Model by the Navigation Data Standard (NDS) \cite{1} – are critical for enabling safe automated driving. Lane-level maps augment vehicle sensor information for contextual analysis of the environment, assist the vehicle in executing controlled maneuvers beyond its sensing range, and provide precise vehicle positioning and orientation in map coordinates \cite{2}. These maps often include localization objects such as signs, barriers, poles, and surface markings to provide the vehicle with a more comprehensive and accurate knowledge of the environment. Given that there are millions of kilometers of roads in the world, it is cost-prohibitive and time-consuming to manually create and maintain such lane information at a centimeter-level precision. Currently, and for the foreseeable future, most automobile manufacturers are focused on autonomous driving on large, controlled interstate highways.

The road networks pertaining to urban settings are often represented by nodes and links (equivalently, vertices and edges) and are mainly classified in two categories, which can further be refined to sub-categories: \textit{interstates} (highway grade and above, which have most length of links) and \textit{local roads} (urban region and sub-urban region, which contains most number of nodes). Compared to local roads, interstates typically have higher quality (and standardized) construction, clearer traffic patterns, and fewer possible hazards (pedestrians, etc.). Moreover, these roads support the majority of the transportation industry and as such, autonomous trucks are among the first vehicles to actually apply autonomous driving techniques \cite{3}. The lengths of the three largest highway networks in the world, U.S., China, and India, are 103, 446, and 79 thousand kilometers \cite{4} respectively. Considering currently reported HD Maps manual modeling efficiency, it could take years to map the entire high level road networks even with thousands of workers. Hence, many HD map automated/semi-automated modeling algorithms are proposed to tackle this problem in highway scenario. At the same time, in
urban (city) scenario, road construction is less standardized, has weaker featured objects – e.g., bad lane marking paint, flatten curb, tree occlusion, etc. – and has more uncertainties of non-road objects such as construction zones, nearby vehicles, trees, etc. These factors make HD map modeling an extremely hard problem to be fully (or even semi) automated at present. The amount of human labor required to model a city, by far exceeds the amount required for modeling hundreds of miles of highway network(s).

Many works in spatial and spatio-temporal databases, as well as Geospatial Information Systems (GIS), have generated mature algorithms that were designed to build complete/refined map-based applications. Whether it is the mature GIS systems or research prototypes, there are two basic facets of the problem: how to generate the data, and how to store/retrieve it.

In the rest of this paper, we overview the popular (Tile-based) data structure for representing maps in Section 2. Section 3 follows with a discussion of popular techniques for modeling HD maps/data. Section 4 presents a more focused discussion on autonomous driving and HD maps and gives concluding remarks.

Figure 1: Overlay of hierarchical tile sizes in real world at level 18 (cyan), 19 (green), 20 (yellow) and 21 (red) on satellite image in downtown Chicago, near 41.890853, −87.628045. Tile size at level 21 (red) is 14.21 meters (tile size is not fixed and subject to latitude and longitude).

2 Tile Based Map Data Structure

Tile-based data structure is being widely used in GIS largely due to its inherent advantages such as: (1) ease of maintenance (the data warehouse, in other words, is easy to read, edit, and not too complicated for updates and insertion of new data); (2) enabling discrete addressing scheme (also amenable for parallelization of the computations); and (3) flexibility in terms of multi-resolution information representation, when one needs to sacrifice space to improve efficiency. Typically, they are hierarchically structured over some base-level, which offers compatibility with most of the GIS data warehouses and applications such as HD maps, web-based GISs (e.g. map viewers, satellite image viewers, etc.), as well as Location-Based Services and Digital Terrain Model (for geology and meteorology).

\[1\] We note that the Navigation Data Standard (NDS) relies on a tile-based database, and is being widely used among auto makers for navigation purpose, see https://www.nds-association.org/#thestandard
Hierarchical tiled-based systems recursively partition the map data into four quadrants (similar to Quadtrees), and save the subdivided data of each level – essentially, pre-storing it at different resolutions. The advantage is that one can now have an added flexibility to query different level detail of data (resolution) on demand – thus accommodating to diverse requests’ configurations and requirements.

Figure 1 illustrates the tile-based system in real world, from level 18 to level 21.

3 HD Maps Modeling

Highly accurate, precise, and detailed lane-level maps, also known as High Definition (HD) Maps, are critical to enable safe automated driving [6]. Lane-level maps augment vehicle’s sensors information for contextual analysis of the environment. This, in turn, assists the vehicles in executing controlled maneuvers beyond its sensing range, and provides more precise vehicle positioning and orientation in map coordinates. HD maps often include localization objects such as signs, barriers, poles, and surface markings to provide the vehicle a more comprehensive and accurate knowledge of the environment. The three most frequently used types of such objects are: lane boundary, occupancy grid and text information (cf. [2]). An illustration is shown in Figure 2.

3.1 Lane Boundary Geometry

Ground based data, mainly from imagery and LiDAR, is the primary data source used to automatically extract lane information [7] in academia and industry. It has distinct advantages such as high precision, rich information (e.g. color and geometry), and is usually not affected by to top-down occlusion (e.g. trees, buildings, overpasses and tunnels). Some researchers [8,9,10] have proposed to detect road surfaces and lane markings from LiDAR using the highly accurate and precise 3D measurements in a LiDAR point cloud. Moreover, a point cloud aligned with perspective imagery can be used to generate training data [11] to assist lane-marking detection in perspective imagery. On the other hand, ground based data sources have many limitations such as object occlusion, infrequent updates, prohibitive cost (i.e., data storage, computation and acquisition) and limited coverage.

State of the art HD maps modeling techniques extract lane marking from different kinds of sources. One such kind is ground based data mainly from imagery and LiDAR [7,9,10,11]. Another kind of source is the 3D aerial based data, for example, from satellite or drones [12,13,14,15].

3.2 Occupancy Grid and Terrain Model

In many applications, the occupancy grids are extracted mainly from ground based LiDAR point cloud [16,17]. We note that the terrain models (more specifically, Digital Terrain Models) also originate from ground based LiDAR [18]. These can be “perceived” as the occupancy grids underneath the vehicle, and be categorized as a type of occupancy grid. Thinking about the size of each type of HD map components, the number of occupancy grids is significantly larger than the number of control points from line boundary geometry [2]. This, in turn, often results in ignoring the data size of lane boundaries – especially in urban areas, where buildings, poles, trees and other stationary objects all over the place.
Figure 2: Illustration of HD map components: (a) one sample segment of road from perspective view; (b) three key components of a HD map: lane boundary geometry (green lines), occupancy grid (yellow voxels) in 25-cm resolution and road sign (red bounding box) within 36 meters from view point; and (c) overlay of HD map and street view image.

4 HD Map Application: Accurate Vehicle Self-Localization

Vehicle self-localization (ego-localization) is a key component of autonomous driving and often depends on a combination of sensor-based and HD-Map-based location data. It demands high precision, real-time, and robust data management and algorithmic techniques that can handle very harsh conditions such as GPS denial/imprecision, traffic occlusion, and low lighting.

The standards of definition precision in the context of vehicle self-localization are mainly based on, and used in approaches relying upon, Global Navigation Satellite System (GNSS) [19, 20]. With the trends of miniaturization and commercialization of LiDAR devices, many Simultaneous Localization And Mapping (SLAM) systems have been proposed. LiDAR techniques are known for their high precision as 3D information is captured directly, compared to reconstructing 3D information from a 2D stereo camera. These two approaches are typically used to solve point cloud based localization. SLAM can be performed in a known environment (i.e. occupancy grid map) [21, 22, 23].

Alternatively, lane-level objects such as lane markings [24], pole-like objects [25], curbs [26], and even occupancy grids [27] can be detected and used for self-localization. Using additional information such as HD Maps, features can be used to estimate vehicle/camera position using triangulation. While LiDAR-based solutions are superior in terms of effectiveness, their shortcomings are: (1) the actual cost; and (2) weather dependency [28]. These limitations have a significant impact on limit the use of LiDAR-based solutions for performing point-cloud based self-localization.

No matter which approach or combination of approaches are used to localize the vehicle, an HD map with real-time object detection and recognition algorithms are one of the key components towards enabling the use of self-driving vehicles in urban settings.

State of the art HD map modeling techniques are fairly well adopted for use in highway scenarios - which directly benefits autonomous driving for interstate logistic. This enables significant savings in labor cost and efficiency, with the expectation to even further decrease such costs in the near future. However, solving this problem in urban scenarios – which is more related to citizens daily commute – has still a lot of challenges. Existing techniques for objects detection and recognition, based on the traditional Machine Learning (ML) approaches are not effective enough. This, in turn, also spurs the need for different ML methodologies – namely,
the training data of urban HD map (especially lane markings and boundaries) is costly, and even inaccurate since it involves worker’s personal perspective.

Complementary to this – even if one assumes that the data warehouse (tile-based structure) and data (HD maps) are readily available, enhancing systems performance is another challenging topic, and of course, more challenges are raising. The size of HD map is too big to be expected to be stored in vehicles on-board devices. Thus, novel efficient map data retrieval algorithms and transmission paradigms are needed for improving the adoption of autonomous vehicles in urban settings.

References


The SIGSPATIAL Special

Section 2:
SIGSPATIAL 2017 Event Reports
Conference Report:
The 25th ACM SIGSPATIAL International Conference on
Advances in Geographic Information Systems
(ACM SIGSPATIAL 2017)
Redondo Beach, California, USA
November 7—10, 2017

Goce Trajcevski
Iowa State University, USA
gocet25@iastate.edu

Siva Ravada
Oracle Corporation
siva.ravada@oracle.com

Roberto Tamassia
Brown University, USA
rt@cs.brown.edu

This report describes the development and finalization of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2017), held in Redondo Beach, California, USA, November 7-10, 2017.

Historically, what is now ACM SIGSPATIAL conference started as a series of workshops and symposia in 1993. Its aim was to promote the interdisciplinary discussions among researchers, developers, users, and practitioners and fostering research in all aspects of Geographic Information Systems – hence the initial acronym ACM GIS – and the focus was on novel systems based on geospatial data and knowledge. It continued its mission of providing a forum for original research results, addressing conceptual, design, and implementation aspects of geospatial data ranging from applications, user interfaces and visualization, to data storage, query processing, indexing and data mining. The conference is now the premier annual event of the ACM Special Interest Group on Spatial Information (ACM SIGSPATIAL).

The technical program of the conference was decided in a two-stage process:

1. each submitted paper was first reviewed by at least three members of a carefully chosen program committee (PC) consisting of experts in the relevant fields. Our PC had a total of 107 volunteers from academia and industry, plus an additional 22 members who were designated as the Senior PC. The assignment of papers to reviewers followed a bidding stage, during which PC members were allowed to express ranked preferences regarding their willingness to review a particular submission. In addition to three reviewers from the PC, each paper was also assigned a designated Senior PC member who studied the reviews, discussed the merits of the submission with the reviewers, wrote a metareview, and formulated an accept/reject recommendation.

2. For the first time in 2017, we implemented a rebuttal phase where the authors received preliminary versions of the reviews and metareview and were offered the opportunity to address the concerns raised therein by submitting a response. The reviews, metareviews, and accept/reject recommendations were then finalized taking into account the responses and the selection of papers to include in the conference program was ultimately made by the PC Chairs. Certain papers that were not accepted for the conference, with the permission of the authors, were forwarded to the conferences Workshop Chairs to be considered for inclusion in relevant workshops co-located with SIGSPATIAL.

Papers were submitted and accepted in different categories. We received a total of 180 research submissions and 13 industrial experience and systems submissions. We accepted 39 of those as full 10-page research papers
for oral presentation, resulting in an acceptance rate of 19.7%. We accepted an additional 43 submissions as poster presentations (22.2% acceptance rate), to be published as 4-page papers. We also received 24 demonstration submissions, of which we accepted 14 for live demonstrations, to be published as 4-page papers (acceptance rate of 58%). Finally, once again we encouraged the submission of papers describing visionary ideas. Of the 14 vision papers submitted, 6 were accepted for oral presentation (42.8% acceptance rate) and publication as 4-page papers. Our reviewers put in a significant amount of effort in reviewing the papers and our hope is that the reviews were beneficial even to those authors whose papers were not accepted.

Continuing the tradition, ACM SIGSPATIAL 2017 had a Cup programming contest, which focused on similarity search in databases of moving object trajectories under Frechet distance. The competition received 28 submissions and the teams totaled 68 members submitting formal entries. Three entries were selected as winners, and were additionally qualified for an invited paper, an oral presentation and award prizes during the banquet.

For the second time, after its debut in 2016, the conference had a Student Research Competition that aimed at providing a forum for undergraduate and graduate students to share their research results and exchange ideas with other students, judges, and conference attendees. This year, 4 papers (co)authored by graduate students and one paper authored by undergraduate students were selected to enter the final round of the competition.

ACM SIGSPATIAL 2017 had two distinguished speakers: Markus Gross (Disney Research and ETH Zurich) with a keynote presentation Disney Research - Technology to Create the Magic, and Bryan Mistele (INRIX) whose keynote addressed Building Smarter Cars & Cities from Spatial Data.

The conference was expertly chaired by Shawn Newsam (University of California, Merced, USA) and Erik Hoel (Esri, USA). It was preceded by 11 associated workshops managed by the Workshop Co-Chairs were John Krum (Microsoft, USA) and Mohamed Sarwat (Arizona State University, USA), in addition to the respective individual workshops organizers.

Recreating and organizing such a vibrant forum from year to year takes tremendous efforts and collaboration among many dedicated individual. We are especially grateful to our PC, Senior PC and external reviewers, who generously and carefully reviewed the submissions and produced valuable feedback for both us and the authors. To produce the proceedings, we had the pleasure to work closely with two individuals who took a lot of the burden and did a great job. Proceedings Co-Chairs Jie Bao (Microsoft Research Asia, China) and Yao-Yi Chiang (University of Southern California, USA). We thank Ruby Tahboub and Ibrahim Sabek (University of Minnesota, USA) who were extremely responsive as our Webmasters and we are also very thankful to the Publicity Co-Chairs: Ahmed Aly (Google, USA), Muhammad Aamir Cheema (Monash University, Australia), Sangho Kim (Esri, USA) and Yanhua Li (Worcester Polytechnic Institute, USA). Furthermore we thank Ahmed Lbath (University of Grenoble 1 - Joseph Fourier, France) and and Xun Zhou (University of Iowa, USA) who served as Poster Co-Chairs and extend our special thanks to Dev Oliver (Esri, USA) and Martin Werner (Leibniz-University Hanover, Germany) who organized the SIGSPATIAL Cup programming contest this year.

Many other fine individuals were involved and did a great job for the technical organization of the event and were in charge of many related activities. We thank Jing (David) Dai (Google, USA) and Wei-Shinn Ku (Auburn University, USA) who served as Treasurer Co-Chairs, along with our special thanks to Fusheng Wang (Stony Brook University, USA) and Moustaffa Yousef (Egypt- Japan University of Science and Technology, Egypt) who were in charge of organizing the SRC (Student Research Competition). We are indebted to Sarana Nutanong (City University of Hong Kong, China) and Zhenhui Li (Penn State University, USA) who served as Registrations Co-Chairs. Local Arrangements co-Chairs Petko Bakalov (Esri, USA) and Ahmed Eldawy (University of California, Riverside, USA) are the one to whom we are all indebted for the hard work that they put in ensuring that everything ran smoothly at the venue.

We are also thankful to the two ACM SIGSPATIAL ECs (Executive Committees) the one that provided the initial support (whose term finished in June 2017) and the newly-elected EC for their expert, sustaining guidance of the conference: MohamedMokbel (Chair, University of Minnesota), Shawn Newsam (Vice-Chair, University of California at Merced), Roger Zimmermann (Secretary, National University of Singapore), and
Egemen Tanin (Treasurer, University of Melbourne); followed by Cyrus Shahabi (Chair, University of Southern California), Goce Trajcevski (Vice-Chair, Iowa State University), John Krum (Treasurer, Microsoft) and Egemen Tanin (Secretary, University of Melbourne)

A distinguished token of gratitude is due to Hanan Samet, Cyrus Shahabi, and Kentaro Toyama for bringing this conference to the forefront in 2007 and starting ACM SIGSPATIAL.

Very special thanks and recognitions are in order for our generous corporate sponsors Google (Gold Sponsor), Esri, Lyft, Facebook (Silver Sponsors), Oracle, NVIDIA (Bronze Sponsors), Microsoft, IBM, UK Ordnance Survey many of whom have supported this conference for multiple years; and it is in order to recognize the appreciation of the work of Mark McKenney (Southern Illinois University Edwardsville, USA) and Farnoush Banaei-Kashani (University of Colorado, Denver, USA) who were instrumental in getting these companies as our sponsors. We are also grateful for the publishing sponsorship by both Morgan & Claypool (Silver Publisher Sponsor) and Springer Publishers (Bronze Publisher Sponsor).

A very distinct token of gratitude goes to the US National Science Foundation (NSF) for its institutional sponsorship enabling travel-grants for students, along with NVIDIA who provided a sponsorship for the SIGSPATIAL Cup and the awards for the winners, both financial ones as well as high-end NVidia cards. Last, but not the least, the top three papers out of the 6 accepted vision papers received an award from the Computing Research Associations Computing Community Consortium (CCC).

Every year, the conference highlights the most important advances in GIS and provides a forum for lively exchange of ideas among leading researchers and practitioners in the field. We are confident that you will find a similar value in this record of the conference. In conclusion, we would like to express once again our gratitude to all the authors who submitted papers, the members of the PC and senior PC, the conference officers, and all the other individuals who contributed their expertise and time to make the conference possible.
ACM SIGSPATIAL GIS Cup 2017 - Range Queries under Fréchet Distance

Martin Werner¹, Dev Oliver²
¹German Aerospace Center (DLR), Remote Sensing Technology Institute (IMF),
²Environmental Systems Research Institute (Esri)
martin.werner@dlr.de, doliver@esri.com

Abstract

The 25th ACM SIGSPATIAL GIS Conference on Advances in Geographic Information Systems was held in November 2017. In conjunction with the main conference, we organized the 6th GIS-focused algorithm competition called the ACM SIGSPATIAL GIS Cup 2017. The contest was about calculating range queries using the Fréchet distance of trajectories in mobility datasets.

1 Introduction

The ACM SIGSPATIAL GIS Conference addresses all sorts of topics in the fields of geographic information systems since 25 years. In addition to scientific result presentations in the form of papers, short papers, posters, and demos, the conference started to acknowledge the art of algorithm design and implementation very much from a practical perspective through the ACM SIGSPATIAL GIS Cup. Each year, a challenging yet well-researched computational topic is chosen and participants shall solve the given problem in high accuracy, quality, and performance.

For this year’s conference, the organizers decided to highlight the Fréchet distance of trajectories. Geospatial trajectories have become an invaluable source of information in spatial analysis, urban computing, transport network research, and map construction. However, comparing trajectories is – in general – quite hard. In this context, a trajectory represents the continuous movement of an object through space. For practical reasons, however, trajectories are usually represented as a sequence of time-stamped spatial locations \( T = [(p_1, t_1), (p_2, t_2), \ldots , (p_m, t_m)] \) together with the assumption that the linear interpolation between subsequent samples is a sufficiently accurate reconstruction of the movement of the object between two consecutive location samples. In the language of GIS, therefore, a trajectory is represented as a LINESTRING feature together with an attribute representing time.

When comparing trajectories, several aspects might get different treatment: we can use the temporal information in an absolute manner looking for similar trajectories happening at the same time (e.g., detecting when two people are following the same trajectory at the same time, enabling for example real-time ride sharing) or in a relative manner (e.g., identifying trajectories with a similar speed pattern). In contrast, we can also completely ignore the temporal information, for example, in applications such as map reconstruction in which the spatial and topological aspects of trajectories are more important.

From a formal point of view, trajectories can be represented as continuous maps \( T \) from the unit interval \([0, 1]\) to space. In this setting, the unit interval represents the time along the trajectory, so given a trajectory \( T : [0, 1] \to \mathbb{R}^2 \), the time 0 is mapped to the first point \( t(0) \) of the trajectory and \( t(1) \) points to the last point. From this perspective, we can see that the set of all continuous trajectories forms an infinite-dimensional
vector space and - therefore - infinitely many sensible distance measures do exist. This is realized in a number of trajectory distances that have been defined and used throughout literature including, but not limited to, Euclidean distance, Hausdorff distance, closest points, dynamic time warping, edit distance on real trajectories (EDR), or edit distance with real penalties (ERP). See [5] for a nice introduction into various trajectory distances. Still, one of the most classical definition from mathematics has been introduced by Fréchet [7] and can be seen as an extension to the Hausdorff distance of sets.

The Fréchet distance is best introduced informally: imagine a dog and an owner each moving along his own trajectory. They start at the beginning, have to travel through all of the trajectory both ultimately reaching the end of their trajectories and are not allowed to go backwards at any time. With these rules in place, the Fréchet distance is the minimum length of a leash connecting the dog and his owner, where the minimization has to be taken over all possible movements. More formally,

\[ d_F(T_1, T_2) = \inf_{\alpha, \beta} \sup_{t \in [0,1]} ||T_1(\alpha(t)) - T_2(\beta(t))|| \]

defines the Fréchet distance of \( T_1 \) and \( T_2 \), where \( \alpha \) and \( \beta \) range over all possible continuous and non-decreasing functions \( \alpha, \beta : [0, 1] \rightarrow [0, 1] \). First note that this distance measure is very good in that it fulfills all properties of a metric, especially, the triangle inequality and that it is exploiting the temporal evolution of the trajectory without fixing it too much. In the real world, for example, two trajectories from the same road will have a very small Fréchet distance even in different traffic conditions. One of the main drawbacks of Fréchet distance is that it is difficult to compute both from a computational complexity perspective as well as from an implementation point of view in that it is not easy to get all details right and efficient on real computers. In this context, the aim of the challenge is twofold: First, we want to show that even though the worst-case complexity is quadratic, it is possible to exploit spatial indexing structures in case of realistic mobility trajectories in order two answer hundreds of queries per second on large datasets. Second, we wanted the community to provide highly performant reference implementations which can be used in GIS research making the application of Fréchet distance easier in practice.

2 Sponsors and Supporters

This year, the challenge was sponsored by NVIDIA and IBM and it is a great help for the SIGSPATIAL community. NVIDIA sponsored the challenge by donating a current NVIDIA Titan X GPU, which we hope will trigger some research on GPU computing in the spatial domain. IBM supported our activities by donating $300 in cash for the second place and $200 in cash for the third place. Additionally, the authors of the best three submissions were invited to write a short paper about the key ideas of their approaches and to give an oral presentation of their work in a conference special session.

3 Problem Description

The participants of the challenge were given a large set of map-matched trajectories in the San Francisco area. These trajectories consist of shortest paths and fastest paths according to travel time each with and without some random restrictions in the street network. A sample of this dataset is depicted in Figure 1. This dataset contains 20,199 trajectories of varying length ranging from 10 points per trajectory to 768 points with a mean length of 247 points. The dataset contains roughly five million points in total.

The background for generating a dataset of this type is the fact that the structure of quite short and efficient paths is dominant in real-world trajectory datasets and greatly simplifies the practical complexity of calculating the Fréchet distance. Similarly, we created a set of 30,000 random queries with distances drawn from a uniform random distribution bound by the dataset diameter.
Given this data, the authors were invited to provide a program that reads in a dataset and a set of queries and computes a multitude of Fréchet range queries at once. As we expect that no strongly subquadratic algorithm for computing the Fréchet distance exists [3], we thought that the challenge would be solved through smart spatial indexing with only a minor influence of the quality of the Fréchet distance decision algorithm. Given that the number of distance calculations for a naive solution calculating the distance matrix of all trajectories would need 605,970,000 trajectory comparisons, we expected sophisticated spatial pruning to be the key to this problem.

4 Submissions and Winners

The challenge received considerable attention with 28 submissions from 69 individual authors and contributors from all over the world. The spatial distribution of authors as depicted in Figure 2 is quite similar to the distribution of participants in ACM SIGSPATIAL GIS and shows, therefore, that the challenge has attracted a representative subset of our community. Interestingly, there is a strong cluster in central Europe including Germany and the Netherlands. This is possibly due to the fact that there is a long tradition in studying the Fréchet distance in this area following up on seminal work by Helmut Alt, who introduced the free-space diagram for computing the Fréchet distance in a joint work with Michael Godau [1].

The best three submissions were written in C++ and used some spatial indexing based on the first and last point of the candidate trajectory exploiting that the leash will need to connect these two pairs of points in any
valid reparametrization. Then, they differ on how to further reduce the set of computations by identifying true positives and true negatives with simple computations. Finally, the free-space diagram is being solved for the remaining candidates with different implementations. For more details, the reader is referred to the excellent papers invited for the three best submissions. In addition, it is great to see that all source code has been published, too – the papers contain links to the implementations.

The submission of Bringman and Baldus wins the cup for being fastest in the evaluation. They first uses some coarse spatial index, then some heuristics to prune further, and finally a recursive free space decision algorithm for validating the remaining candidates [2]. The second place was won by the submission of Buchin, Diez, van Diggelen and Meulemans [4]. They first uses a grid index to manage the trajectory endpoints and creates up to four simplifications of the trajectories. The problem is then first solved on the simplifications proceeding from coarse to fine finally solving the full Fréchet decision problem on the original pair of trajectories. The third place is given by the submission of Fabian Dütsch and Jan Vahrenhold [6]. They introduce a concept called the annulus of a trajectory and apply some other heuristics for efficiently pruning candidates. Finally, the Fréchet decision problem is being solved. This submission was only slightly slower than the preceding two submissions and notably stable even with malformed datasets.

Acknowledgments

We want to acknowledge the support from our sponsor NVIDIA for donating an NVIDIA Titan X GPU to the winner of this challenge and our sponsor IBM for donating $300 to the second place and $200 to the third place. Furthermore, we want to express our thanks to the organizing committee of the ACM SIGSPATIAL Conference on Advances in Geographic Information Systems for hosting this competition.

References


The ACM Special Interest Group on Spatial Information (SIGSPATIAL) addresses issues related to the acquisition, management, and processing of spatially-related information with a focus on algorithmic, geometric, and visual considerations. The scope includes, but is not limited to, geographic information systems (GIS).

The Association for Computing Machinery (ACM) is an educational and scientific computing society which works to advance computing as a science and a profession. Benefits include subscriptions to Communications of the ACM, MemberNet, TechNews and CareerNews, full and unlimited access to online courses and books, discounts on conferences and the option to subscribe to the ACM Digital Library.

- SIGSPATIAL (ACM Member) ................................................................. $ 15
- SIGSPATIAL (ACM Student Member & Non-ACM Student Member) .................................................. $ 6
- SIGSPATIAL (Non-ACM Member) ........................................................................................................... $ 15
- ACM Professional Membership ($99) & SIGSPATIAL ($15) ............................................................... $114
- ACM Professional Membership ($99) & SIGSPATIAL ($15) & ACM Digital Library ($99) ............... $213
- ACM Student Membership ($19) & SIGSPATIAL ($6) ........................................................................... $ 25

payment information

Name ________________________________________________________
ACM Member # _____________________________________________
Mailing Address _____________________________________________
___________________________________________________________
City/State/Province _________________________________________
ZIP/Postal Code/Country ________________________________
Email ______________________________________________________
Mobile Phone ______________________________________________
Fax ________________________________________________________

Credit Card Type: ☐ AMEX ☐ VISA ☐ MC
Credit Card #: ______________________________________________
Exp. Date: __________________________________________________
Signature: __________________________________________________
Make check or money order payable to ACM, Inc.
ACM accepts U.S. dollars or equivalent in foreign currency. Prices include surface delivery charge. Expedited Air Service, which is a partial air freight delivery service, is available outside North America. Contact ACM for more information.

Questions? Contact: ACM Headquarters
2 Penn Plaza, Suite 701
New York, NY 10121-0701
voice: 212-626-0500
fax: 212-944-1318
e-mail: acmhelp@acm.org

Remit to: ACM
General Post Office
P.O. Box 30777
New York, NY 10087-0777

Advancing Computing as a Science & Profession