Monitoring Manhattan's traffic from 5 cameras?

Siheng Chen
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
sihengc@andrew.cmu.edu

Yaoqing Yang
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
yyaoqing@andrew.cmu.edu

Jelena Kovačević
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
jelenak@cmu.edu

Christos Faloutsos
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
christos@cs.cmu.edu

ABSTRACT

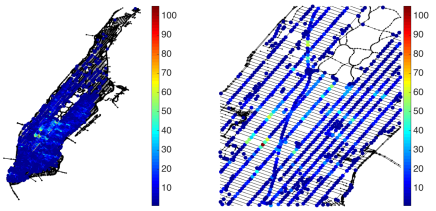
Is it possible to monitor the entire traffic in Manhattan at a few road intersections? In this paper, we propose a series of novel graph data processing techniques to handle complex and nonsmooth traffic data. Then, we validate our proposed techniques on Manhattan's taxi pickups during the years of 2014 and 2015. We are able to approximately recover the taxi-pickup activities in Manhattan by taking samples at only 5 selected intersections. We believe that the same techniques can be applied to recover other types of traffic data. The advantages of our methods are (a) quality: we are able to recover the taxi-pickup activities in entire Manhattan with small error from only 5 selected intersections; (b) scalability: we use a tree structure and principle component analysis to make this method efficient for large- scale graphs.

Keywords

Traffic monitoring, taxi pickups, graph data processing

1. INTRODUCTION

Urban data record the behavior of urban ecosystems. Analyzing those urban data potentially leads to improvements of the urban lives [7, 8]. As one of the most critical components of urban data, traffic data is a key to understand the mobility pattern and to make cities more efficient; however, traffic data is usually sparse because usually only a few sensors are installed to cover a limited number of intersections [10]. Some previous work monitored traffic by using side information, such as GPS data [9]. In this paper, we aim to monitor traffic in the entire city from camera installed at a few intersections, which can count the number of passing vehicles. We model traffic data as data supported on a city street graph where the value at a node of the graph reflects the traffic activity at the corresponding intersection. In this setting, monitoring traffic data is nothing but a task of graph data sampling and recovery [2,3]. Previous works along this line consider sampling and recovering smooth graph data, which means that any two adjacent nodes have similar values; however, real traffic data lie in a high-dimensional space and are not smooth on a city street graph; see Figure 1. We handle nonsmooth graph data by a series of novel graph data processing techniques, including adaptive pieceiwise-constant approximation, super-graph Fourier basis construction and graph data sampling. We validate our proposed method on Manhattan's taxi pickups in the years of 2014 and 2015. We are able to approximately recover the taxi-pickup activities in the entire Manhattan by taking samples at only 5 selected intersections. Here we focus on taxi-pickup activities, but we believe the same techniques can be applied to recovering many other types of traffic data. The propose method is also efficient and scalable to large-scale graphs.



(a) Global view. (b) Zoom-in plot.

Figure 1: Taxi-pickup distribution at 6 pm on January 1st, 2015. The taxi pickups at two adjacent intersections can be a lot different. Thus, the traffic data are *not smooth* on the Manhattan street graph.

2. PROBLEM FORMULATION

We consider a city street graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of nodes (intersections), $\mathcal{E} = \{e_1, \ldots, e_M\}$ is the set of edges (streets). In this paper, we use a *graph data* to model traffic activities in a city by assigning the number of nearby taxi pickups $x_n \in \mathbb{R}$ to the node v_n. The graph data can be written as a vector $\mathbf{x} = \left[x_1, x_2, \ldots, x_N\right]^T \in \mathbb{R}^N$. Let $C \subseteq \mathcal{V}$ be a set of nodes (an area in a city). We can represent this set by using an indicator function, $\mathbf{1}_C \in \mathbb{R}^N$, where $(\mathbf{1}_C)_i = 1$ when $v_i \in C$, and 0 otherwise. That is,

UrbComp'16, August 14, 2016, San Francisco, USA.

the values are ones in the node set C and zeros in the complement node set $\bar{C} = \mathcal{V}/C$. When the node set C forms a connected subgraph, we call C a *piece* and $\mathbf{1}_C$ a *one-piece graph data*. A piecewise-constant graph data is a linear combination of several one-piece graph data. That is, $\mathbf{x} = \sum_{i=1}^{K} \mu_i \mathbf{1}_{C_i}$, where each C_i is a piece, μ_i is a constant and K is the number of pieces.

Sampling & Recovery. We consider sampling the number of taxi pickups at several selected intersections and then recovering the number of taxi pickups at the rest intersections. Mathematically, we sample M values at selected indices (intersections) in a graph data $\mathbf{x} \in \mathbb{R}^N$ to produce a sampled version $\mathbf{y} = \Psi\,\mathbf{x}$, where the sampling operator Ψ is a linear mapping from $\mathbb{R}^N$ to $\mathbb{R}^M$ with $\Psi_{i,j} = 1$ when we sample the jth node in the ith measurement, and 0, otherwise. Here we consider experimentally designed sampling, which chooses the sampled indices beforehand. We then interpolate $\mathbf{y}$ to get a recovery $\widehat{\mathbf{x}} = \Phi\mathbf{y} \in \mathbb{R}^N$, where Φ is the interpolation operator designed based on the sampling operator Ψ.

3. PROPOSED METHOD

The proposed method involves two phases: the learning phase and the real-time phase. In the learning phase, we prepare for all the operators needed in the real-time phase using the traffic data. In the real-time phase, we sample the traffic activities at a few selected intersections and recover the rest by using the operators learned in the learning phase.

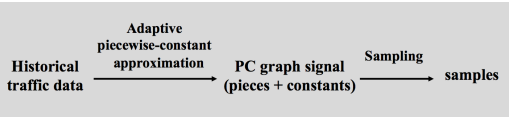


Figure 2: Learning phase includes two main modules: adaptive piecewise-constant approximation implemented by adaptively pruning a decomposition tree and sampling implemented by sampling bandlimited graph data. In the learning phase, we decide which intersection to sample.

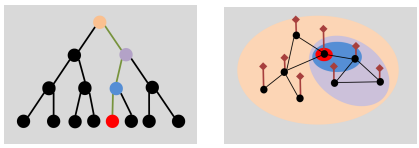
3.1 Learning Phase

The purpose of the learning phase is to learn important patterns from historical traffic data and then decide which intersections we need to sample. An initial idea is to construct a graph that promotes smoothness for historical traffic data and then use graph sampling techniques to design samples. Does the original Manhattan street graph promote smoothness for traffic data? Figure 1 shows the taxi-pickup distribution at 6 pm on January 1st, 2015. We see that many intersections have many more number of taxi pickups than their adjacent intersections and the overall taxi-pickup distribution is barely smooth. We thus need to learn a graph from traffic data. However, a city street graph is usually huge and historical traffic data is limited. For example, Manhattan has 13,670 intersections. It is thus inefficient and not robust to construct a huge graph from limited historical traffic data. To overcome this difficulty, we should reduce the size of the graph. The idea is to explore local information and do graph coarsening, which groups those neighboring intersections as one super-node. This is equivalent to approximate a graph data by using a piecewise-constant graph data, which captures both local smoothness (the number of taxi pickups at two adjacent intersections are similar). and fast transition (the number of taxi pickups at two adjacent intersections have a large difference).

After piecewise-constant approximation, the dimension reduces from the number of intersections to the number of pieces. We then construct a super-graph whose nodes are pieces and edges are the dependencies between pieces. Then, we can use graph sampling techniques to design samples. Figure 2 overviews the procedure of the learning phase. The two main modules are adaptive piecewise-constant approximation and graph sampling. We now elaborate the implementation of these two modules.

Adaptive piecewise-constant approximation. The goal is to adaptively search for a piecewise-constant graph data to approximate traffic data. The key of the proposed approximation technique is to design a series of nonoverlapping pieces that captures the variation of an graph data. There are usually two approaches to design such a series of pieces: predesigned approach and learning approach. In a predesigned approach, we design pieces before accessing any traffic data. We can simply use physical partitions, such as zipcodes and census blocks, but these partitions may not be flexible enough to capture complex variations in traffic data; on the other hand, in a learning approach, we learn a series of pieces to fit traffic data. However, there are multiple restrictions in the optimization: those pieces are connected, nonoverlapping and cover the vertex domain. It is inefficient and unrobust to solve a nonconvex optimization problem.

We consider combining the advantages of these two approaches. We first design a set of redundant pieces before accessing any data. Because of redundancy, this set is able to capture various shapes and sizes. We then select the best series of nonoverlapping pieces by pruning this set according to historical traffic data. This approach is both adaptive and efficient. Since the set of redundant pieces is constructed beforehand, the bottleneck of the computational complexity is the pruning stage. We will show that by taking advantage of a binary-tree-based pruning technique, the computational complexity of the pruning stage is merely $O(N)$.



(a) Global view. (b) Zoom-in plot.

Figure 3: Growing a binary tree in (a) is equivalent to decomposing a graph in (b). The green path in (a) is a decomposition in (b), where same color indicate the one-to-one mapping from a node in the decomposition tree to a piece in a graph.

A set of redundant pieces can be constructed via a binary tree decomposition as shown in [4]. This set is provably useful to represent arbitrary piecewise-constant graph data. The main idea is to recursively partition a piece into two disjoint pieces until all the pieces are individual nodes. Figure 3 shows an example. A node in Plot (a) represents a piece in Plot (b) and an edge represents a kinship where a parent node partitions into two children nodes. The top node (in orange) represents the entire vertex domain and the bottom nodes represents all the individual nodes. The green path in Plot (a) is a decomposition in Plot (b), where the same color indicates the one-to-one mapping from a node in a decom-

position tree to a piece in a graph. We use the 2-means clustering to implement graph partitioning [4]. For each piece, we select two nodes with the longest geodesic distance as the community centers and assign all the other nodes to their nearest community center based on the geodesic distance. Then, we recompute the community center for each community by minimizing the summation of the geodesic distances to all the other nodes in the community and assign each node to its nearest community center. We repeat the above procedures for several iterations until the community centers converge. Please find more details in [4].

By using the binary tree decomposition, we obtain $(2N-1)$ pieces, where each piece corresponds to a node in the binary tree. The binary tree decomposition is redundant and captures various sizes and shapes of pieces. We then prune the obtained set of pieces and select the best series of nonoverlapping pieces according to historical traffic data. Let $\mathcal{C} = \{\mathbf{1}_{C_i}\}_{i=1}^{2N-1}$ represent the set of constructed pieces. We aim to select a subset of pieces that minimizes the following objective function,

$$\begin{aligned} \mathrm{D}^*, \mathrm{Z}^* &= \arg\min_{\mathrm{D}_i \in \mathcal{C}, \mathrm{Z}} \|\mathrm{X} - \mathrm{D}\,\mathrm{Z}\|_F^2 + \lambda\, \dim(\mathrm{Z}), \qquad (1) \\ &\quad \text{subject to } \mathrm{D}\,\mathbf{1} = \mathbf{1}, \end{aligned}$$

where $\mathrm{X} \in \mathbb{R}^{N \times L}$ is the matrix representation of historical traffic data, $\mathrm{D} \in \mathbb{R}^{N \times K}$ is the matrix representation of the selected pieces with D_i being the ith column, $\mathrm{Z} \in \mathbb{R}^{K \times L}$ is the matrix that stores the constants of all the pieces and $\dim(\mathrm{Z}) = KL$. Note that the number of pieces K is a variable to be optimized during the optimization because we do not know how many pieces in advance.

The first term in the objective function pushes the piecewise-constant approximation to fit the given data. The second term punishes a large size of the constant matrix Z and avoids overfitting; that is, when λ is large, we tend to select fewer pieces from $\mathcal{C}$ to fit data and when λ is small, we tend to select all the pieces in $\mathcal{C}$ to fit data. The constrain requires that all the selected pieces are nonoverlapping and covers the entire vertex domain. Since each column in D is a one-piece graph data, the optimization problem (1) finds the best piecewise-constant approximation for the given traffic data. Since the constructed pieces in $\mathcal{C}$ have a nice tree structure, we can easily obtain the global optimum of (1) by pruning the tree, which follows the paradigm in [5, 11]. The main idea is to compare the representation based on a parent piece to the representation based on its two children pieces and see which representation minimizes the objective function (1). For example, C_1 is a parent piece and C_2, C_3 are its children pieces. Since the parent piece and two children pieces represent the same vertex domain ($C_1 = C_2 \cup C_3$), to satisfy the constraint, we either choose the parent piece or its two children pieces. The cost of using the parent piece is $\min_{\mathbf{z}} \left\|\mathrm{X} - \mathbf{1}_{C_1}\mathbf{z}^T\right\|_F^2 + \lambda L$, with optimum $\left\|\mathrm{X} - \mathbf{1}_{C_1}\mathbf{1}_{C_1}^T\,\mathrm{X}\right\|_F^2 + \lambda L$, and the cost of using the child pieces is $\min_{\mathrm{Z} \in \mathbb{R}^{2\times L}} \left\|\mathrm{X} - \begin{bmatrix}\mathbf{1}_{C_2} & \mathbf{1}_{C_3}\end{bmatrix}\mathrm{Z}\right\|_F^2 + 2\lambda L$, with optimum $\left\|\mathrm{X} - \begin{bmatrix}\mathbf{1}_{C_2} & \mathbf{1}_{C_3}\end{bmatrix}\begin{bmatrix}\mathbf{1}_{C_2} & \mathbf{1}_{C_3}\end{bmatrix}^T\mathrm{X}\right\|_F^2 + 2\lambda L$. Each time, we compare their costs and choose the one with a smaller cost to update the representation at the parent piece. The pruning process starts from the bottom level of the decomposition tree and move to an upper level iteratively until we reach the top level. Through the pruning process, we obtain the global optimum of (1).

Super-graph construction and sampling. We next model each selected piece as a super-node and construct a super-graph. Since the selected pieces already capture the local similarities, the connections among super-nodes are not relevant to the geodesic distance any more. We need to learn a super-graph to promote smoothness for historical traffic data and then design which super-nodes to sample. In graph sampling, we usually model a smooth graph data as a bandlimited graph data [3] whose sampling strategy is designed based the corresponding graph Fourier basis. Thus, instead of constructing a full super-graph, we directly construct a graph Fourier basis. Recall that the bandlimited assumption requires that most energy of a graph data is concentrated in the low-pass band; that is, we need to find a graph Fourier basis that pushes the energy to the subspace spanned by its first few basis vectors. Thus, all we need is the first few columns in the graph Fourier basis, which can be simply obtained by principal component analysis. Principal component analysis uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables [1], which exactly fits our requirement. Mathematically, let the constant matrix $\mathrm{Z}^* \in \mathbb{R}^{K \times L}$ be a matrix of graph data on the super-graph, we obtain the first M graph Fourier basis vectors (principal components) by solving the following optimization problem,

$$\mathrm{V}^* = \arg\min_{\mathrm{V} \in \mathbb{R}^{K \times M}} \left\|\mathrm{Z}^* - \mathrm{V}\,\mathrm{V}^T\,\mathrm{Z}^*\right\|_F^2, \text{ subject to } \mathrm{V}^T\,\mathrm{V} = \mathrm{I}.$$

It is true that we can obtain a truncated graph Fourier basis directly from X, which is equivalent to set λ to be zero in (1); however, the computation is less efficient and the obtained principal components are learned from noisy and limited historical data and do not take advantage of the local grouping[1], which is explored by (1). Next, we design a sampling operator by using graph sampling techniques. For example, we solve $\Psi^* = \arg\max_{\Psi} \sigma_{\min}(\Psi\,\mathrm{V}^*) \in \mathbb{R}^{M \times K}$ by using a greedy method, which is shown in [3]. Note that we sample super-nodes, instead of individual intersections. To directly operate on individual intersections, the sampling operator is $\Psi^*(\mathrm{D}^*)^T \in \mathbb{R}^{M \times N}$, which means that Ψ^* selects some pieces from D. We then need to sample all the nodes in the selected pieces, or sample several nodes in selected pieces and estimate the average values. In the experiments, we find that the selected pieces happen to be single nodes, then we only need to sample at the selected intersections.

3.2 Real-time Phase

In the learning phase, we obtain three important operators: selected pieces D^*, truncated graph Fourier basis V^*, sampling operator Ψ^*. Given a real-time traffic data $\mathbf{x} \in \mathbb{R}^N$, we first take samples at the selected intersections, $\mathbf{y} = \Psi^*(\mathrm{D}^*)^T\,\mathbf{x}$. Then we use the interpolation operator to recover all the constants, $\mathbf{z} = \mathrm{V}^*(\Psi^*\,\mathrm{V}^*)^\dagger\mathbf{y}$. Finally, we obtain a piecewise-constant approximation to the real taxi pickups by $\widehat{\mathbf{x}} = \mathrm{D}^*\,\mathbf{z} = \mathrm{D}^*\,\mathrm{V}^*(\Psi^*\,\mathrm{V}^*)^\dagger\Psi^*(\mathrm{D}^*)^T\,\mathbf{x}$, where the interpolation operator is $\Phi = \mathrm{D}^*\,\mathrm{V}^*(\Psi^*\,\mathrm{V}^*)^\dagger$. Figure 4 illustrates the procedure in real-time phase.

[1]Piecewise-constant approximation can be regarded as a denoising module. Many experiments indicate that reducing the dimension to $N/2$ provides the best recovery performance in the end, which is better and faster than directly working with X.

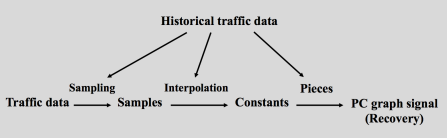
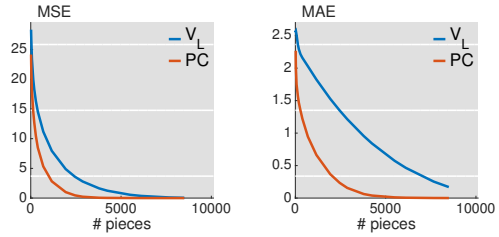


Figure 4: In real-time phase, we sample the selected nodes, recover all the constants, and finally obtain the piecewise-constant estimation to the real-time traffic data.



(a) MSE. (b) MAE.

Figure 5: Approximation error as a function of the number of pieces. Piecewise-constant approximation (red) significantly outperforms smooth approximation (blue).

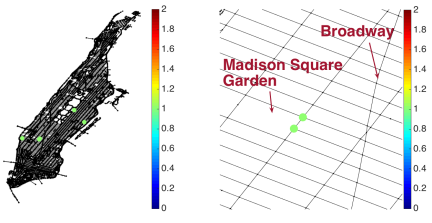
4. VALIDATIONS

We validate the proposed method on a dataset of Manhattan's taxi pickups. We sample the number of taxi pickups at several intersections and recover the taxi pickups at the rest intersections.

Dataset. We consider a public dataset about taxi pickups in Manhattan[2], which is particularly interesting because taxis are valuable sensors of city life [6–8]. Here we use the dataset in the year of 2014 and 2015. We focus on rush hours during workdays (6 pm from Monday to Friday). We accumulate the taxi-pickup activities within a hour and project each taxi pickup to its closest intersection, obtaining 261 graph data in the year of 2014 for learning and 261 graph data in the year of 2015 for real-time processing.

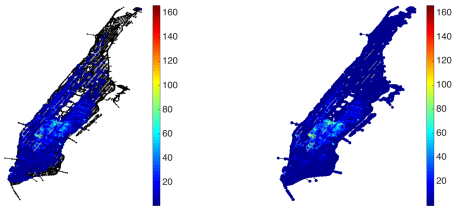
Results. We first validate the proposed adaptive piecewise-constant approximation. We solve (1) based on 261 graph data in 2014 by varying the regularization parameter λ. Two metrics are used to quantify the performance, including mean square error (MSE $= \frac{1}{261N}\sum_{i=1}^{261}\|\widehat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2$) and mean absolute error (MAE $= \frac{1}{261N}\sum_{i=1}^{261}\|\widehat{\mathbf{x}}_i - \mathbf{x}_i\|_1$), where $\widehat{\mathbf{x}}_i$ is the recovered taxi pickups in the ith day and $\mathbf{x}_i$ is the real taxi pickups in the ith day. Figure 5 compares the approximation errors between the graph Fourier basis based on the Laplacian matrix (V_L, in blue) and adaptive piecewise-constant approximation (PC, in red). We see that PC significantly outperforms V_L in terms of both metrics. We then set $\lambda = 1$ (corresponds to 3788 pieces) and obtain 5 samples provided by the optimal sampling operator, as shown in Figure 6. As discussed before, these 5 pieces happen to be individual nodes. Two adjacency intersections around Penn Station are sampled, indicating that Penn Station is the weathercock of Manhattan's traffic.

[2]Data is downloaded from http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml



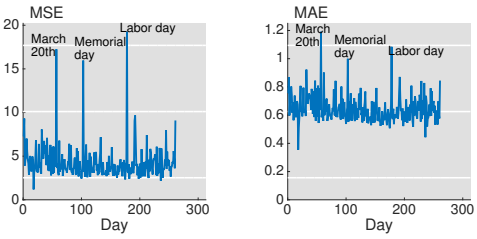
(a) Global view. (b) Zoom-in plot.

Figure 6: Selected 5 intersections. Two adjacency intersections around Penn Station are sampled.



(a) Real distribution. (b) Recovered distribution.

Figure 7: Recovered taxi pickups at 6 pm, Jan. 6th, 2015.



(a) MSE. (b) MAE.

Figure 8: Daily recovery error in the year of 2015. Plot (b) shows that the average error at each intersection is merely 0.6 taxi pickups during the rush hour.

We next validate those learned operators to the graph data in 2015. Figure 7 shows the recovery of taxi-pickup activity at 6 pm, Jan. 6th, 2015 by only using 5 samples. Even we just use 5 samples, the recovered taxi-pickup distribution is very close to the real taxi-pickup distribution. Finally, we show the daily recovery errors in Figure 8. We see that the recovery errors are particularly large at three days: March 20th, Memorial day and Labor day, which means that the proposed method fails to model abnormal days. But in general, the recovery error is small. For example, Figure 8 (b) shows that the average error at each intersection is merely 0.6 taxi pickups during the rush hour every weekday.

5. CONCLUSIONS

We set a goal to monitor Manhattan's traffic from a few intersections. Finally, we are able to obtain a decent recovery of the taxi-pickup distribution by taking samples at only 5 selected intersections. The main techniques involves adaptive pieceiwise-constant approximation via decomposition tree pruning, super-graph Fourier basis construction via principal component analysis and sampling of graph data. Other than monitoring traffic, the proposed method can be used to anomaly detection and traffic prediction in the future.

6. REFERENCES

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[2] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovačević. Signal recovery on graphs: Variation minimization. *IEEE Trans. Signal Process.*, 63(17):4609–4624, Sept. 2015.

[3] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević. Discrete signal processing on graphs: Sampling theory. *IEEE Trans. Signal Process.*, 63:6510 – 6523, Aug. 2015.

[4] S. Chen, R. Varma, A. Singh, and J. Kovačević. Signal representations on graphs: Tools and applications. *http://arXiv:1512.05406*, 2015.

[5] R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Trans. Inf. Theory, sp. iss. Wavelet Transforms Multiresolution Signal Anal.*, 38(2):713–718, Mar. 1992.

[6] J. A. Deri and J. M. F. Moura. Taxi data in New York City: A network perspective. In *Proc. Asilomar Conf. Signal, Syst. Comput.*, pages 1829–1833, Pacific Grove, CA, Nov. 2015.

[7] H. Doraiswamy, N. Ferreira, T. Damoulas, J. Freire, and C. T. Silva. Using topological analysis to support event-guided exploration in urban data. *IEEE Trans. on Visualization and Computer Graphics*, 20(12):2634–2643, 2014.

[8] 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 Trans. on Visualization and Computer Graphics*, 19(12):2149–2158, Dec. 2013.

[9] Q.-J. Kong, Q. Zhao, C. Wei, and Y. Liu. Efficient traffic state estimation for large-scale urban road networks. *IEEE Trans. Intelligent Transportation Systems*, 14(1):398–407, 2013.

[10] J. Poco, H. Doraiswamy, H. T. Vo, J. L. D. Comba, J. Freire, and C. T. Silva. Exploring traffic dynaics in urban environments using vector-valued functions. *Computer Graphics Forum (in proceedings of EuroVis 2015)*, 34(3):161–170, 2015.

[11] K. Ramchandran and M. Vetterli. Best wavelet packet bases in a rate-distortion sense. *IEEE Trans. Image Process.*, 2(2):160–175, Apr. 1993.