Multi-scale Graph Convolutional Network for Intersection Detection from GPS Trajectories

Yifang Yin*, Abhinav Sunderrajan#, Xiaocheng Huang#, Jagannadan Varadarajan#, Guanfeng Wang#, Dhruva Sahrawat*, Ying Zhang*, Roger Zimmermann*, and See-Kiong Ng*
*National University of Singapore, Singapore
#GrabTaxi Holdings, Singapore
*{idsyin, seekiong}@nus.edu.sg, {dhruva, zhangyin, rogerz}@comp.nus.edu.sg
#{abhinav.sunderrajan, xiaocheng.huang, jagan.varadarajan, guanfeng.wang}@grab.com

ABSTRACT

To facilitate the reconstruction of high-quality road networks, intersections as the key locations provide valuable information about the network topology. However, only a few efforts have been made on the data-driven automatic detection of intersections from, *e.g.*, large-scale GPS trajectories. To bridge the gap, we propose a machine learning based intersection detection approach based on large-scale real-world GPS trajectories of drivers from the Grab ride-hailing service. Instead of representing locations with vector descriptors, we innovatively propose a graph representation that models a location together with its local surroundings to improve the descriptiveness of the location descriptors. Moreover, we present a multi-scale graph convolutional network (GCN) to generate robust graph-level descriptors, followed by logistic regression to discriminate intersections from non-intersections. The experimental results show that our proposed multi-scale graph model outperforms the conventional multi-scale vector representation by 8.5%. Appealingly, the proposed graph representation can be considered as a general location descriptor, which can be used in a variety of geo-based applications other than intersection detection for location modeling.

CCS CONCEPTS

• **Information systems → Geographic information systems**;
• **Computing methodologies → Neural networks**.

KEYWORDS

Intersection detection; graph convolutional network; multi-scale feature fusion; large-scale real-world GPS data

1 INTRODUCTION

Grab [13] as a smartphone-enabled ride-hailing service provider similar to Uber and Lyft, has been seen exponential growth in recent years [6]. With the smartphone app, Grab is able to collect the location of its drivers with rich sensor data such as bearing and speed to provide high-quality ride hailing services. Recent research has made great efforts to reconstruct this underlying road network from GPS trajectories automatically, in order to aid in the construction of maps [1]. A road network is usually denoted as a directed graph, where the vertices represent the key points and the edges represent the road segments between them [12]. Recently, a few efforts have been made to first detect road junctions, such that next the road network can be constructed more conveniently by connecting the detected intersections based on GPS trajectories [8]. However, the effectiveness of earlier work can be limited due to issues including threshold selection, model simplicity, lack of large-scale training data, *etc.*

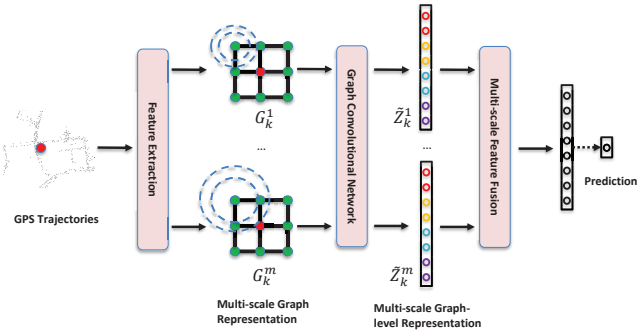

Figure 1: System overview of our proposed intersection detection method.

The service of ride-hailing providers relies significantly on the quality of the digital map. Though OpenStreetMap (OSM) provides the community user-generated maps of the world, its data completeness and accuracy vary significantly in different cities. Considering the high cost of manual corrections of the map data, Grab would like to develop data-driven approaches that can learn from cities with high quality map data (*e.g.*, Singapore) to automatically discover missing roads and intersections in cities with poor map data (*e.g.*, Jakarta). To this end, we present in this paper a novel multi-scale graph convolutional network [9] that can effectively detect intersections from large-scale GPS trajectories. The system overview is illustrated in Figure 1. We formulate the intersection detection as a two-class classification problem. To extract descriptive features, we improve existing shape descriptors in the following two aspects: 1) we extract multi-scale features using shape descriptors with different sizes, and 2) we propose a graph representation that models a location together with the local environment

GeoAI'19, November 5, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6957-2/19/11…$15.00
https://doi.org/10.1145/3356471.3365234

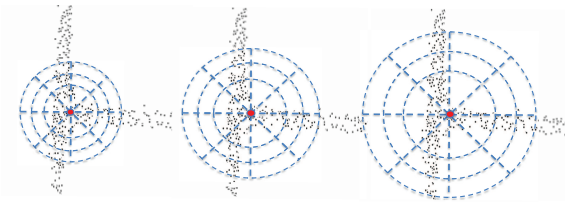

Figure 2: Use multiple shape descriptors of different sizes at the target location.

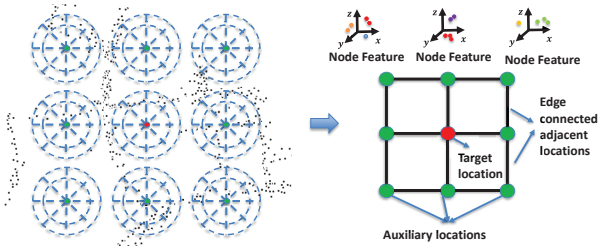

Figure 3: Model the target location and the corresponding local environment as a graph.

in the nearby regions. With the multi-scale graph representation as input, denoted as $\mathbf{G_k^j}$, we next perform graph convolutions to effectively encode not only node features but also graph substructures to generate robust graph-level representations, denoted as $\tilde{\mathbf{Z}}_\mathbf{k}^\mathbf{j}$, for each input location where k represents the k-th location and j represents the j-th scale. Finally, we fuse the multi-scale graph-level representations based on feature concatenation and apply logistic regression to discriminate intersections from non-intersections.

2 INTERSECTION DETECTION

To train an intersection detector, we first extract feature descriptors for locations by processing the nearby GPS traces. Next, we train a multi-scale graph convolutional network, which is capable of detecting intersections with various sizes and orientations.

2.1 Location Modeling

To extract location feature vectors, we adopt the shape descriptor proposed by Fathi and Krumm [4]. At a given location, the shape descriptor divides the nearby regions into multiple histogram bins. We process the trajectories and only keep the ones passing through the inner circle of the shape descriptor to reduce noise. For every kept GPS trajectory, we add one point to each of the bins that the trajectory passes through. We map the bins to a vector and normalize it using the L1 norm. Similarly, we also quantize the moving direction of the GPS points into bins, map it to a vector, and normalize it using the L1 norm. The feature vectors that were extracted based on vehicles' locations and moving directions are next concatenated into a single feature vector at each location as the location descriptor. Let $\mathbf{L}$ denote a set of locations with n samples, and $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$ denote the corresponding normalized feature descriptors extracted at locations in $\mathbf{L}$. The classifier directly trained on $\mathbf{X}$ may suffer from low accuracy due to the sensitivity of the shape descriptor to the scale and orientation of intersections. To solve these issues, we modify the shape descriptor in the following two aspects to make it more descriptive and robust.

Using descriptors of different sizes. Probably the most straightforward solution is to use multiple shape descriptors of different sizes at each location to exact features at different scales, and let the classifier learn from the most representative features at each location. As shown in Figure 2, instead of using a global shape descriptor, we use, *e.g.*, three different size descriptors with the radius of the inner circle set to 10, 20, and 30 meters, respectively. Thereby, three feature vectors are extracted to capture the local distribution of GPS points around a location at three different scales, which are next passed to a neural network where improved classification results can be obtained by applying feature fusion techniques.

Modeling locations as graphs. As pointed out by Chen *et al.* [3], road junctions do not occur in isolation and their characteristics such as size and orientation can be closely related to the location environment. Based on this observation, we propose to use a graph to represent a location and model the intersection detection as a multi-graph classification problem. As illustrated in Figure 3, in addition to the target location (*i.e.*, the location for which we want to detect if it is an intersection or not), we also sample eight auxiliary locations in the nearby regions. We add edges (with weight set to one) between vertical and horizontal neighbors to represent the relationship of locations in the geospatial domain. We extract features using the shape descriptor at each of the nine locations and associate them with the corresponding node as the node feature.

The above two feature modeling methods focus on different aspects, and therefore can be combined to generate a multi-scale graph representation. Formally, let $\mathbf{L}$ denote a set of locations with n samples, and $\tilde{\mathbf{G}} = \{\mathbf{G_1}, \mathbf{G_2}, ..., \mathbf{G_n}\}$ denote the corresponding multi-scale graph representation of locations in $\mathbf{L}$. For each $\mathbf{G_k} \in \tilde{\mathbf{G}}$, $\mathbf{G_k} = \{\mathbf{G_k^1}, \mathbf{G_k^2}, ..., \mathbf{G_k^m}\}$ is a set of graphs with the same structure but different node features, and m is the number of different size shape descriptors we use for node feature extraction. Based on this representation, we next present a multi-scale graph convolutional network for intersection detection.

2.2 Multi-scale Graph Convolutional Network

Formally, an input graph is represented as $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{A})$ (we omit the super- and sub-scripts of $\mathbf{G_k^j}$ for presentation simplicity) where $\mathbf{V}$ is the set of nodes, $\mathbf{E}$ is the set of edges, and $\mathbf{A}$ is the adjacency matrix. Let $v_i \in \mathbf{V}$ denote a node and $e_{ij} = (v_i, v_j) \in \mathbf{E}$ denote an edge. The adjacency matrix $\mathbf{A}$ is a 9×9 matrix with $\mathbf{A}_{ij} = 1$ if $e_{ij} \in \mathbf{E}$.

With the graph structure and node features as inputs, we pass $\mathbf{G_k} = \{\mathbf{G_k^1}, \mathbf{G_k^2}, ..., \mathbf{G_k^m}\}$ to a shared-weight two-layer GCN with 50 hidden units followed by ReLU activation and Dropout to obtain a graph-level representation by concatenating node representations. Let $\tilde{\mathbf{X}}$ represent the node information matrix, the i-th row of which is the feature vector of node v_i. The graph convolution layer takes the following form [7]:

$$\mathbf{Z} = f(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{X}}\mathbf{W}) \quad (1)$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix of the graph with added self-loops, $\tilde{\mathbf{D}}$ is the diagonal degree matrix with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $\mathbf{W}$ is a matrix of trainable parameters of the graph convolution layer. f is a nonlinear activation function, for which we choose ReLU activation in our architecture. $\mathbf{Z}$ is the output matrix of the graph convolution layer.

The graph convolution can effectively encode both graph substructures and node features, which is very helpful for multi-graph classification. We generate a compact representation for each input graph by concatenating the node features outputted by the two-layer graph convolutional network into a graph-level feature vector, denoted as $\tilde{\mathbf{Z}}^{\mathbf{j}}_{\mathbf{k}}$ corresponding to input graph $\mathbf{G}^{\mathbf{j}}_{\mathbf{k}}$.

Next, we fuse the multi-scale graph representations $\{\tilde{\mathbf{Z}}^{\mathbf{1}}_{\mathbf{k}}, \tilde{\mathbf{Z}}^{\mathbf{2}}_{\mathbf{k}}, ..., \tilde{\mathbf{Z}}^{\mathbf{m}}_{\mathbf{k}}\}$ encoded at each location $l_k \in \mathbf{L}$. We first pass $\tilde{\mathbf{Z}}^{\mathbf{j}}_{\mathbf{k}}$ to one shared-weight fully-connected layer with 50 hidden units followed by ReLU activation, and then aggregate the output features based on concatenation. The output layer of our proposed graph convolutional network contains only one unit indicating the probability of the input location being an intersection or not. As this is a two-class classification problem, we apply Sigmoid activation at the output layer and adopt the binary cross-entropy as our loss function.

3 EVALUATION

3.1 Experimental Dataset and Settings

The evaluation dataset is collected in Singapore by Grab and is used for evaluating the classification accuracy of our proposed intersection detection algorithm. It contains tracks of 7,461 drivers and 4,687,072 GPS points in total, with a sampling rate of one GPS point in every five seconds. The ground-truth locations of intersections are derived using the python library OSMnx [2] from OpenStreetMap, which is an open project that provides user-generated maps of the world [5]. The locations of non-intersections are randomly sampled on and off roads that are at least 100 meters away from any intersection. The locations without any GPS points in the nearby regions for feature extraction are removed. We further divide the dataset into 80%-20% splits for training and evaluation. For the shape descriptor, we used 6 circles, 16 angular slices, and 36 moving direction bins. Followed the settings suggested by Fathi and Krumm [4], we set the ratio between radii of consecutive circles to be 1.3.

3.2 Classification Accuracy Comparison

We compare the following four intersection detection methods to justify the effectiveness of our proposed approach.

- **Vector**: This method uses the vector descriptor $\mathbf{x_k}$ generated based on a global shape descriptor as the input feature to train a simple neural network with two hidden layers as the classifier.
- **Multi-scale Vector**: This method concatenates multi-scale vector descriptors generated based on different size shape descriptors as the input feature to train a simple neural network with two hidden layers as the classifier.
- **Graph**: This method uses the single-scale graph representation we proposed as the input feature to train a two-layer graph convolutional network, followed by one output layer as the classifier.
- **Multi-scale Graph**: This method uses the multi-scale graph representation we proposed as the input feature to train a two-layer graph convolutional network, followed by multi-scale feature fusion sub-network and one output layer as the classifier.

Table 1: Intersection classification comparison of different methods on the Singapore dataset: precision, recall, and F1 measure on locations of intersections.

Methods	Scale	Precision	Recall	F1
Vector	10	0.71	0.56	0.62
	20	0.70	0.65	0.67
	30	0.73	0.68	0.70
Multi-scale Vector	-	0.73	0.68	0.70
Graph	10	0.71	0.71	0.71
	20	0.73	0.75	0.74
	30	0.75	0.78	0.77
Multi-scale Graph	-	0.75	0.81	0.78

Table 2: Intersection classification comparison of different methods on the Singapore dataset: precision, recall, and F1 measure on locations of non-intersections.

Methods	Scale	Precision	Recall	F1
Vector	10	0.62	0.76	0.68
	20	0.66	0.71	0.68
	30	0.66	0.73	0.69
Multi-scale Vector	-	0.68	0.73	0.71
Graph	10	0.70	0.69	0.69
	20	0.73	0.70	0.71
	30	0.76	0.72	0.74
Multi-scale Graph	-	0.78	0.72	0.75

Tables 1 and 2 reports the precision, recall, and F1 measure obtained by different intersection detection methods. Column scale in the tables refers to the radius of the inner circle of the shape descriptor we used for feature extraction. As can be seen, method **Vector** performed unstable with different scale settings. This is because this method only used one global shape descriptor to extract features from GPS trajectories, which is not sufficient to capture the correct scale at different locations. Methods **Multi-scale Vector** and **Graph** improved method **Vector** in two different aspects as introduced in Section 2 and therefore they were able to obtain better classification results at both intersections and non-intersections. Comparatively, method **Graph** is more effective as it models a location with additional features of the local environment. It outperformed method **Vector** at all scales in terms of the F1 measure. Finally, method **Multi-scale Graph** combines the advantages of **Multi-scale Vector** and **Graph**, which obtained the best classification results: F1 measure of 0.78 and 0.75 at locations of intersections and non-intersections, respectively.

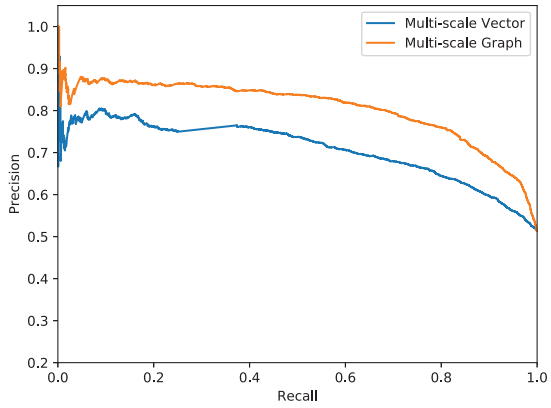

Figure 4: Comparison of the precision-recall curve of the multi-scale vector model and the proposed multi-scale graph model.

One advantage of machine learning based methods for intersection detection compared to traditional heuristic methods is that we can easily draw the precision-recall curve and choose an appropriate threshold to balance the tradeoff between precision and recall based on the system requirements. Figure 4 shows the precision-recall curves obtained by the multi-scale vector model and our proposed multi-scale graph model. From the figure we can see that our proposed multi-scale graph model outperforms the multi-scale vector model by a large margin. Our model is able to obtain a high precision (*e.g.*, larger than 0.8) with a recall up to 0.7. The precision only goes down quickly when the recall is larger than 0.8. Comparatively, the multi-scale vector method is struggling to obtain a precision larger than 0.8, which indicates the effectiveness of our proposed graph representation for classification.

4 RELATED WORK

Automatic intersection detection has been a critical research problem for a variety of AI and LBS applications. Heuristic methods define intersections as locations where the drivers change their moving directions or locations that connect more than two road segments. For instance, Wu *et al.* presented a clustering-based intersection detection algorithm from coarse-gained GPS data [10]. Xie *et al.* proposed to detect the longest common subsequences between pairs of GPS trajectories [11]. However, the aforementioned methods mostly suffer from thresholding, resulting in undetected or falsely detected intersections. For machine learning based methods, Fathi and Krumm presented the first work on training a classifier based on a shape descriptor extracted from GPS traces to detect road intersections [4]. They adopted a 2D circular window as the local shape descriptor, which described the distribution of GPS points at each location on the map. Next, a classifier that used the Adaboost algorithm was trained based on the shape descriptors. Chen *et al.* followed this path and further proposed a scale- and rotation-invariant descriptor for intersection detection [3]. Considering that the road junctions can have different sizes depending on the local environments, they proposed to estimate the scale and canonical orientation at each location before feature extraction. Finally, an SVM was trained to recognize intersections and the corresponding types. However, due to the lack of public large-scale and high-sampling rate GPS trajectory datasets, the development of machine learning techniques for automatic intersection detection is still in its early stage where only simple feature descriptors and classifiers were adopted and evaluated in previous work.

5 CONCLUSION AND FUTURE WORK

We innovatively propose to model a location using a graph representation, and subsequently propose a multi-scale graph convolutional network to perform classification for intersection detection from GPS trajectories. Our proposed graph representation is more descriptive and robust compared to traditional feature vector representations as it not only models the target location but also selects several auxiliary locations in the neighborhood to model the corresponding local environment. In the future, we plan to exploit more variations in terms of the graph structure, *e.g.*, the number and position of the nodes, the weight of the edges, *etc.* for the graph representation extraction.

ACKNOWLEDGMENT

This work was funded by the Grab-NUS AI Lab, a joint collaboration between GrabTaxi Holdings Pte. Ltd. and National University of Singapore.

REFERENCES

[1] Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, and Carola Wenk. 2015. A Comparison and Evaluation of Map Construction Algorithms using Vehicle Tracking Data. *Geoinformatica* (2015), 601—-632.
[2] Geoff Boeing. 2017. OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks. *Computers, Environment and Urban Systems* (2017), 126–139.
[3] Chen Chen, Cewu Lu, Qixing Huang, Qiang Yang, Dimitrios Gunopulos, and Leonidas Guibas. 2016. City-Scale Map Creation and Updating Using GPS Collections. In *International Conference on Knowledge Discovery and Data Mining*. 1465–1474.
[4] Alireza Fathi and John Krumm. 2010. Detecting Road Intersections from GPS Traces. *International Conference on Geographic Information Science* (2010).
[5] M. Haklay and P. Weber. 2008. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing* (2008), 12–18.
[6] David King, Deborah Salon, and Matthew Conway. 2018. Trends in Taxi Use and the Advent of Ridehailing, 1995–2017: Evidence from the US National Household Travel Survey. *Urban Science* 2 (2018).
[7] Thomas N Kipf and Max Welling. 2016. Semi-supervised Classification with Graph Convolutional Networks. *CoRR* (2016). https://arxiv.org/abs/1609.02907
[8] Radu Mariescu-Istodor and Pasi Fränti. 2018. CellNet: Inferring Road Networks from GPS Trajectories. *ACM Transactions on Spatial Algorithms and Systems* (2018), 8:1–8:22.
[9] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* (2009), 61–80.
[10] Junwei Wu, Yunlong Zhu, Tao Ku, and Liang Wang. 2013. Detecting Road Intersections from Coarse-gained GPS Traces Based on Clustering. *Journal of Computers* (2013), 2959–2965.
[11] Xingzhe Xie, Wenzhi Liao, Hamid Aghajan, Peter Veelaert, and Wilfried Philips. 2016. A Novel Approach for Detecting Intersections from GPS Traces. In *IEEE International Geoscience and Remote Sensing Symposium*. 1816–1819.
[12] Yifang Yin, Rajiv Ratn Shah, and Roger Zimmermann. 2016. A General Feature-based Map Matching Framework with Trajectory Simplification. In *ACM SIGSPATIAL International Workshop on GeoStreaming*. 7:1–7:10.
[13] Zhiyin Zhang, Xiaocheng Huang, Chaotang Sun, Shaolin Zheng, Bo Hu, Jagan Varadarajan, Yifang Yin, Roger Zimmermann, and Guanfeng WANG. 2019. Sextant: Grab's Scalable In-Memory Spatial Data Store for Real-Time K-Nearest Neighbour Search. In *International Conference on Mobile Data Management*.