Traffic State Prediction Based On Road Correlation Analysis With GPS Data

Mid Report

11812804 董正

Supervisior: 宋轩

Department of Computer Science and Engineering

Mar. 2022

Contents

1 Preliminaries

1.1 Introduction

With the great development of modern cities, the rapid growth of population and the acceleration of urbanization has made transportation systems to an essential infrastructure. And Transportation systems are becoming more and more complex, which causes great pressure on urban traffic management. [1]

Modern transportation systems contain road vehicles, railway transportation and a variety of newly emerged shared travel modes, including online ride-hailing, bike-sharing, etc.. In order to alleviate transportation related problems and manage the expanding transportation systems efficiently, traffic prediction or traffic forecasting is brought up by researchers in recent years.

Traffic forecasting is typically based on consideration of historical traffic state data. In the development of intelligent transportation systems, traffic states are detected by traffic sensors, bus and metro transactions logs, traffic surveillance cameras and GPS device data. However, traffic state data is hard to manage because it involves large data volumes with high dimensionality. Its typical characteristic is that it contains both spatial and temporal domains. The traffic state in a specific location has both spatial dependency, which may not be affected only by nearby areas, and temporal dependency, which may be seasonal. Traditional time series prediction models cannot handle such spatiotemporal prediction problems. Therefore, deep learning methods have been introduced in this area. [2]

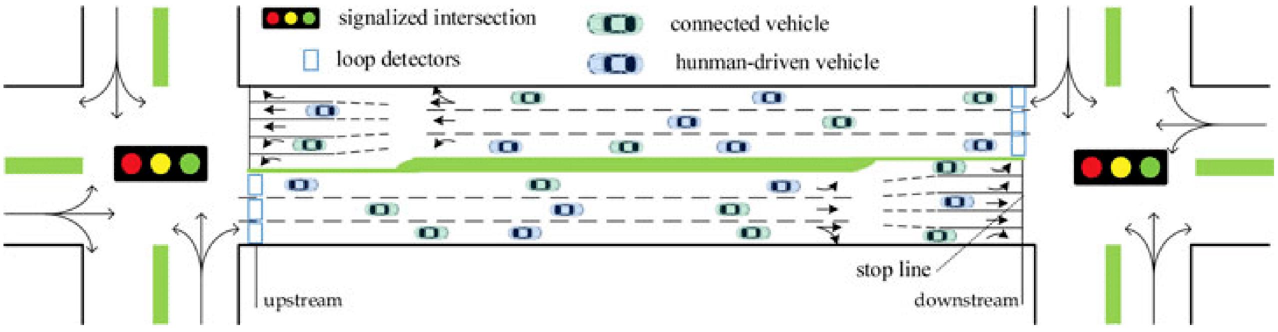


Figure 1: Road Traffic System [3]

1.2 Motivation

Figure 2 shows the workflow of traffic prediction procedure. Then we focus on the data organization of the model's input. As shown in figure 3, we first match a GPS sequence into a trajectory, i.e. a series of roads, then sum the corresponding flow of each road. After that, we will get a flow matrix with time as x-axis and road ID as y-axis. Logically it is just stack the road network's flow vector by time.

Almost every state-of-the-art GNN models use stacked vector [4] as input. However, this kind of stacked vector loses the natural information inside trajectories, which is road transition. Treating a trajectory as discrete points ignores its feature as a whole sequence. This is similar to NLP, where we cannot view a sentence as irrelevant words.

We model the road transition probability as **road correlation** or **road influence**. In summary, our work is to

1. Build a GPS trajectory dataset
2. Extract road correlation via trajectories
3. Embed road correlation into traffic state prediction models

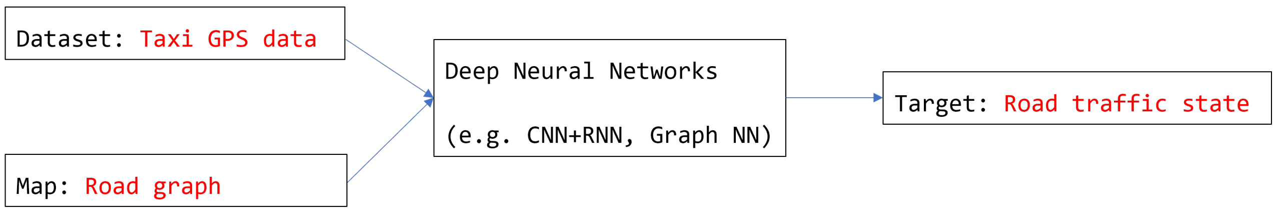


Figure 2: Workflow of traffic prediction

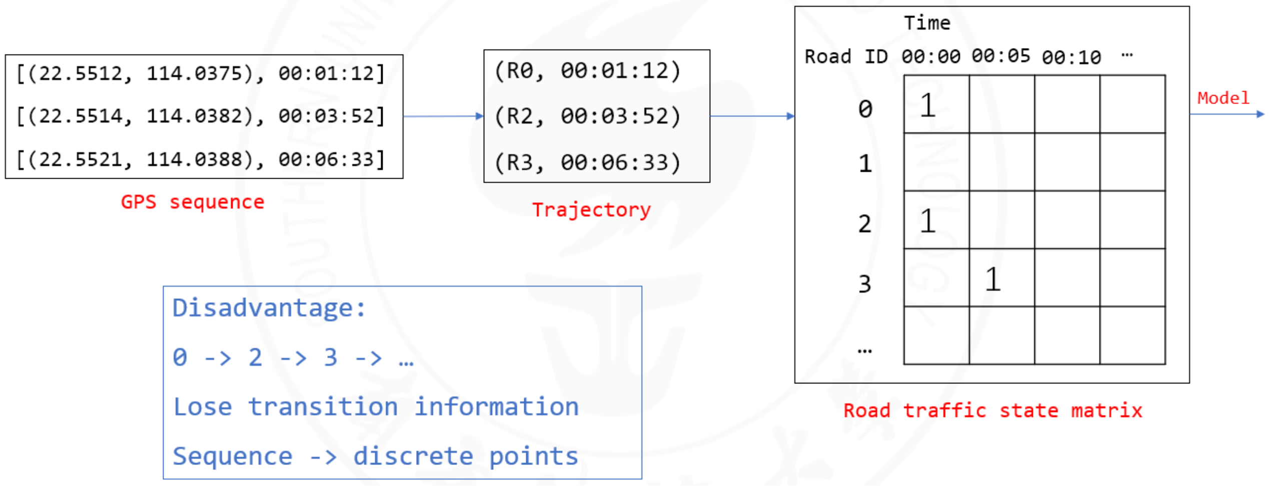


Figure 3: Input Data

1.3 Work Plan

The basic design and plan is shown in the following figure:

1. Process taxi GPS data to get tracks
2. Process road network data to get a basic graph model
3. Match tracks to each road and get the adjacent matrix of the graph
4. Try a simple prediction based on Markov model
5. **Graph embedding**
6. **Influence function design**
7. Combine spatial-temporal models and use them as baseline

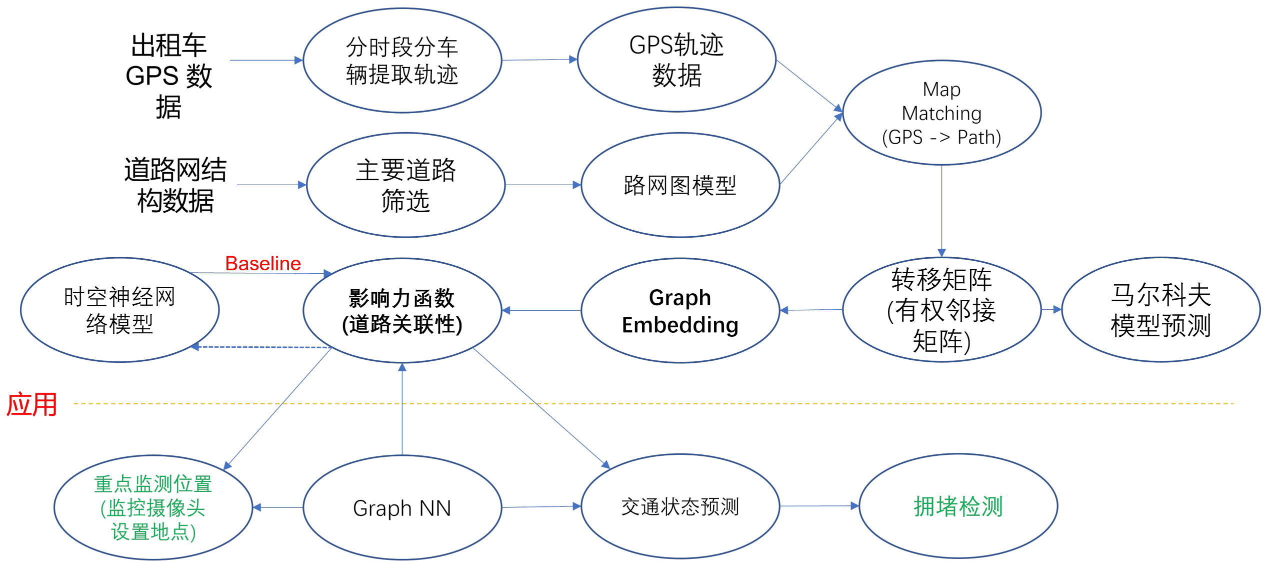


Figure 4: Plan

8. Embed road influence function into graph neural networks to predict traffic states
9. Future Applications: traffic surveillance camera position and traffic jam detection

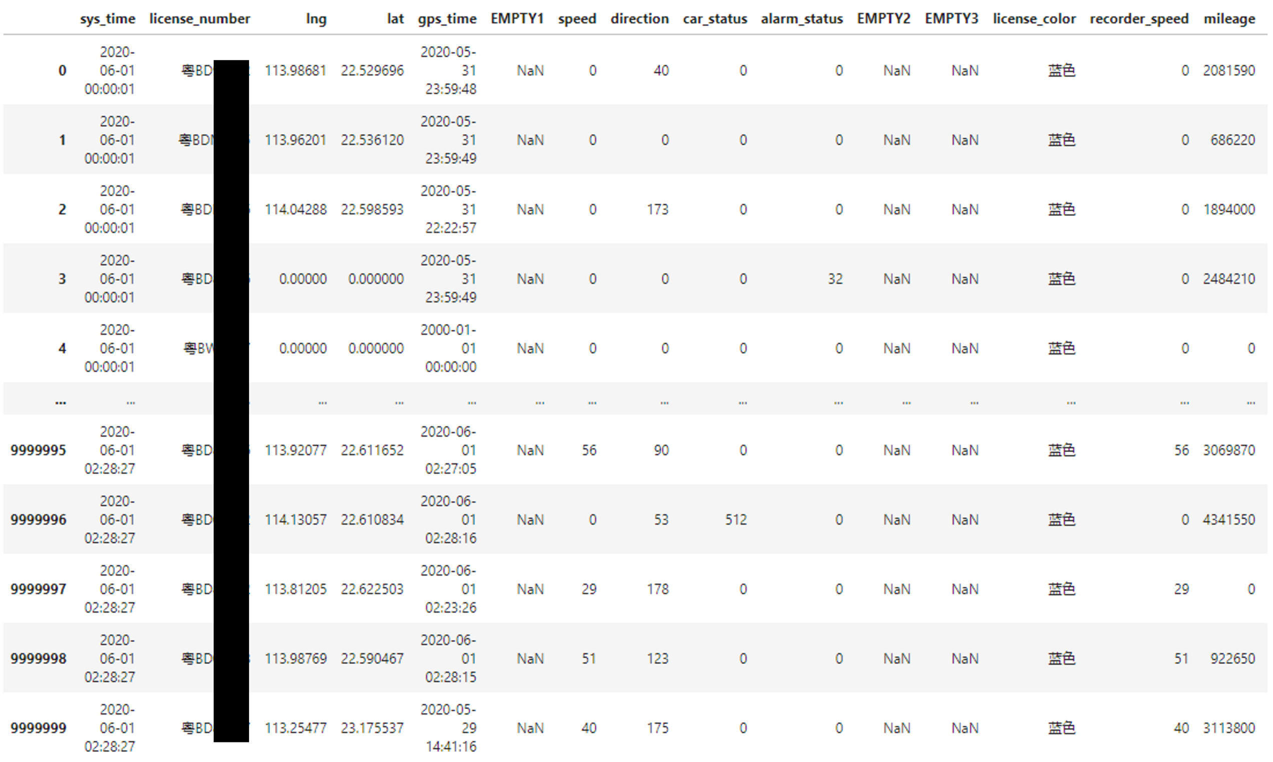
2 Dataset and Data Processing

2.1 Overview

In traffic prediction area, the mostly used datasets are traffic sensor datasets. Traffic sensors can detect road traffic flow and average speed. Therefore, it can be directly entered into deep neural network models. In recent studies, most of the traffic state prediction models use sensor data. On the contrary, GPS data needs a series of data processing methods, which makes it hard to use.

2.2 Data Description

- Region: Shenzhen
- Time: 2020-06
- Content: Taxi trajectory data
  - License number
  - Longitude and latitude
  - Speed
  - Timestamp
- Size: Over 2,700,000,000 rows



	sys_time	license_number	lng	lat	gps_time	EMPTY1	speed	direction	car_status	alarm_status	EMPTY2	EMPTY3	license_color	recorder_speed	mileage
0	2020-06-01 00:00:01	粤BD[redacted]	113.98681	22.529696	2020-05-31 23:59:48	NaN	0	40	0	0	NaN	NaN	蓝色	0	2081590
1	2020-06-01 00:00:01	粤BD[redacted]	113.96201	22.536120	2020-05-31 23:59:49	NaN	0	0	0	0	NaN	NaN	蓝色	0	686220
2	2020-06-01 00:00:01	粤BD[redacted]	114.04288	22.598593	2020-05-31 22:22:57	NaN	0	173	0	0	NaN	NaN	蓝色	0	1894000
3	2020-06-01 00:00:01	粤BD[redacted]	0.00000	0.000000	2020-05-31 23:59:49	NaN	0	0	0	32	NaN	NaN	蓝色	0	2484210
4	2020-06-01 00:00:01	粤BW[redacted]	0.00000	0.000000	2000-01-01 00:00:00	NaN	0	0	0	0	NaN	NaN	蓝色	0	0
...	...	[redacted]	...	...	...	...	...	...	...	...	...	...	...	...	...
9999995	2020-06-01 02:28:27	粤BD[redacted]	113.92077	22.611652	2020-06-01 02:27:05	NaN	56	90	0	0	NaN	NaN	蓝色	56	3069870
9999996	2020-06-01 02:28:27	粤BD[redacted]	114.13057	22.610834	2020-06-01 02:28:16	NaN	0	53	512	0	NaN	NaN	蓝色	0	4341550
9999997	2020-06-01 02:28:27	粤BD[redacted]	113.81205	22.622503	2020-06-01 02:23:26	NaN	29	178	0	0	NaN	NaN	蓝色	29	0
9999998	2020-06-01 02:28:27	粤BD[redacted]	113.98769	22.590467	2020-06-01 02:28:15	NaN	51	123	0	0	NaN	NaN	蓝色	51	922650
9999999	2020-06-01 02:28:27	粤BD[redacted]	113.25477	23.175537	2020-05-29 14:41:16	NaN	40	175	0	0	NaN	NaN	蓝色	40	3113800

Figure 5: Shenzhen Taxi GPS Raw Dataset

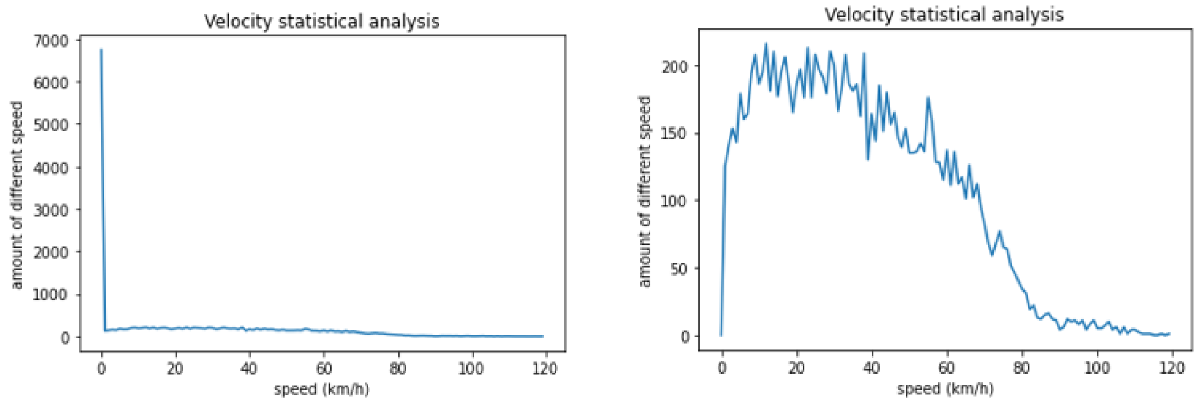
In contrast to those open datasets, we have a completely raw dataset which was taken directly from the raw records in Transportation Bureau of Shenzhen. Unlike the open datasets that can be applied to deep learning models without the need of data cleaning and completion, this raw dataset contains lots of abnormal values, which should be cleaned and re-organized carefully.

2.3 Data Cleaning

Since the raw data contains plenty of noise, the first step is to clean the whole dataset.

1. Drop duplicate
2. Drop time not in 2020-06
3. Drop latitude or longitude is 0
4. Drop speed is 0 or too large because motionless data is useless
5. Drop useless columns

After that, we got 48% rows deleted, around 1.3 billon, which means the raw data is very dirty. For example, the speed distribution before and after data cleaning is shown below:



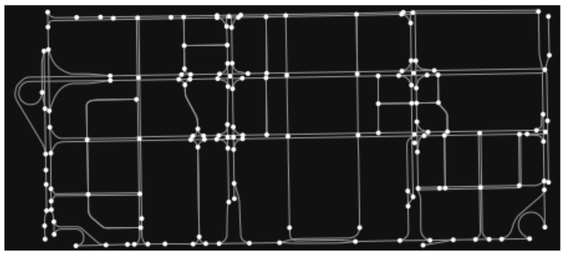
(a) Before Cleaning (b) After Cleaning

Figure 6: Speed Distribution

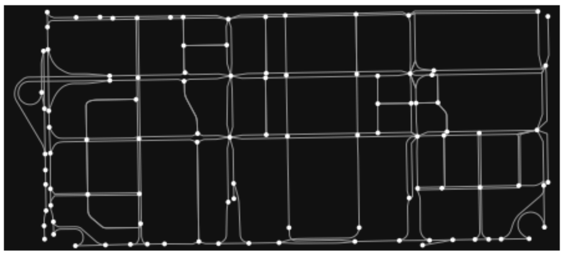
2.4 Data Processing

As shown in figure 8, a GPS point mainly contains five attributes, which are `(Taxi ID, Latitude, Longitude, Speed, Timestamp)`. First, we need to sort these GPS points by timestamp for each taxi ID and get GPS sequences.

To acquire trajectories, we need a road map that contains geometry information of roads. *OpenStreetMap (OSM)* is a collaborative project to create a free editable geographic database of the world. [5] It is easy require map data via its API by *OSMnx* library in *Python*. Since Shenzhen is a big city, we chose a rectangle part of roads in CBD, which is shown in figure 7(a). The return type of the map is `networkx.MultiDiGraph` that can be used and analyzed directly in *Python* code. In addition, as we can



(a) Map acquired via *OSM* API



(b) Map after consolidating nodes

Figure 7: Road Map

see in the figure, the original *OSM* map has lots of nodes that can be consolidated, typically on road intersections. Map simplification is another big research topic in Geographic Information System (GIS) area. A simple way here is to use built-in library functions or just draw manually. The road map after simplification is shown in figure 7(b).

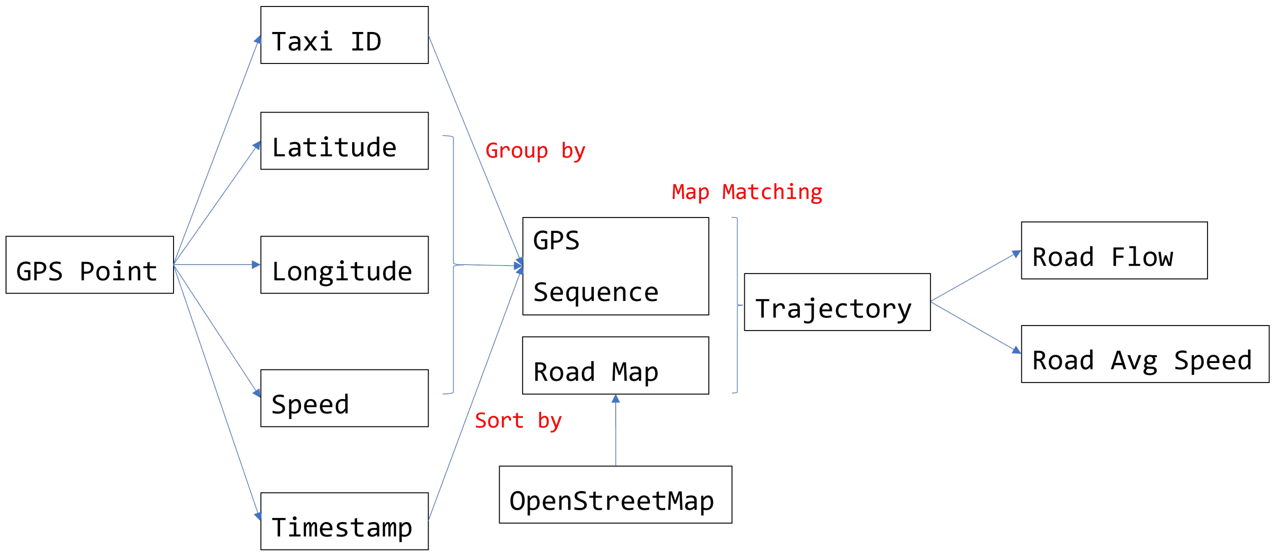


Figure 8: Data Processing Pipeline

Breifly, a trajectory refers to a path in road map, which is a sequence of road IDs in particular. For example, a GPS sequence can be $[(22.551, 114.012), (22.562, 114.023), (22.569, 114.024)]$, and the corresponding trajectory is $[Road_3, Road_{12}, Road_{24}]$. The next step is to combine GPS sequences and road map to get trajectories. Therefore, to decide a GPS point is on which road, we need **Map Matching**. Map matching is the problem of how to match recorded geographic coordinates to a logical model of the real world. The most common approach is to take recorded, serial location points (e.g. from GPS) and relate them to edges in an existing street graph (network), usually in a sorted list representing the travel of a user or vehicle. *Fast Map Matching (FMM)* [6] is an open source map matching framework in *C++* and *Python*. It solves the problem of matching noisy GPS data to a road network. The output example of *FMM* is shown in figure 9.

The last step is counting traffic flow and calculating average speed on each road. We aggregate traffic flow and speed on each 5 minutes. With trajectory data, it is easy to do this by a for-loop and increment. The final representaion of processed data is given in figure 10, and we call the form as “stacked vector” [4]. After that, we can put it into some baseline models to get a basic result.

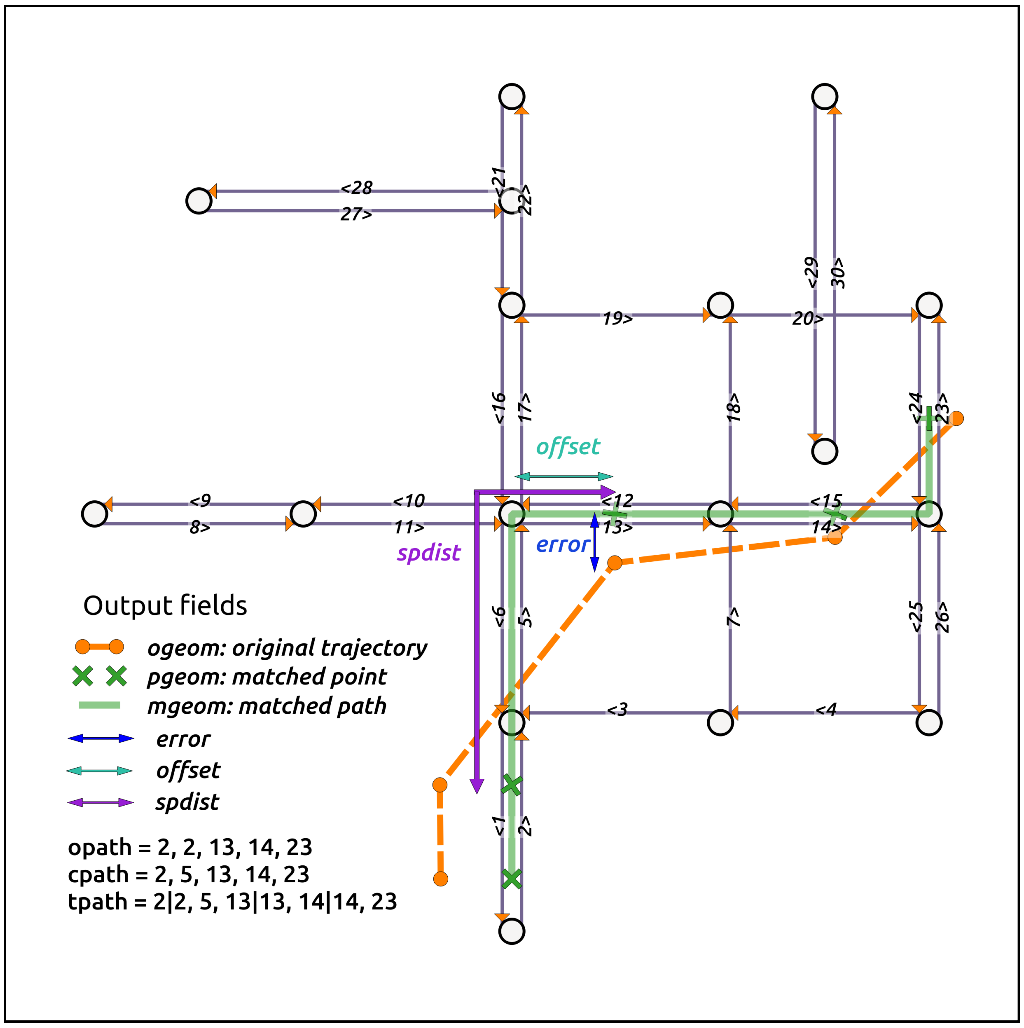
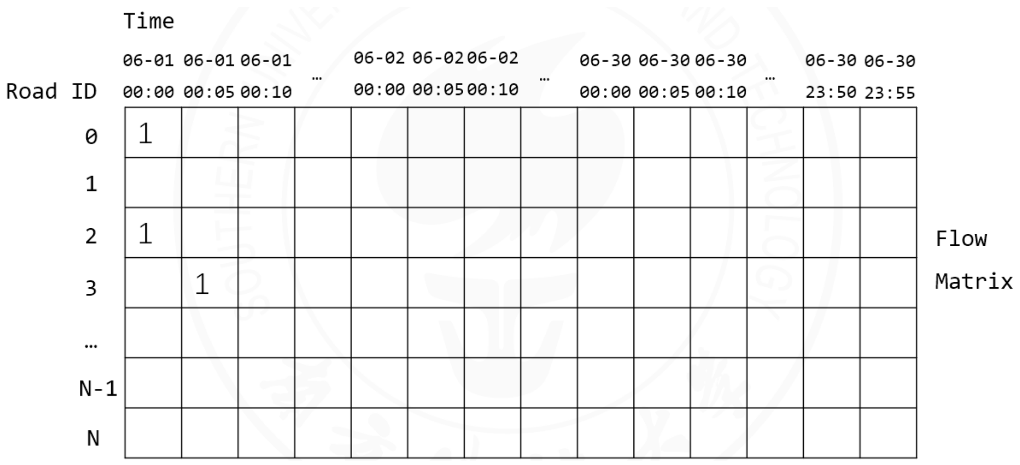


Figure 9: *Fast Map Matching (FMM)*



Time
Road ID
Flow
Matrix

Road ID	06-01 00:00	06-01 00:05	06-01 00:10	…	06-02 00:00	06-02 00:05	06-02 00:10	…	06-30 00:00	06-30 00:05	06-30 00:10	…	06-30 23:50	06-30 23:55
0	1													
1														
2	1													
3		1												
…														
N-1														
N														

Figure 10: Stacked Vector

3 Baseline Experiment

In this section, we tried several SOTA models to test the quality of our dataset.

- AGCRN, NeurIPS 2020 [7]
- Graph WaveNet, IJCAI 2019 [8]
- LSTNet, SIGIR 2018 [9]
- MTGNN, SIGKDD 2020 [10]
- STNorm, SIGKDD 2021 [11]

To evaluate the accuracy of the simulation results, we calculated prediction error with three different metrics, which are mean absolute error (MAE), masked mean absolute percentage error (Masked MAPE), and root-mean-square error (RMSE). They are defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y_i} - y_i|$$

$$Masked\ MAPE = \frac{1}{n}\sum_{i=1}^{n}|\frac{\hat{y_i} - y_i}{y_i}|, y_i \neq 0$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y_i} - y_i)^2}$$

We have compared the performance of the baselines. The left half of table 1 shows the evaluation result of traffic flow prediction. Our dataset has a low MAE and RMSE on these baseline models, but MAPE is a little high compared to traffic sensors datasets. This is because baselines are designed for sensor data, i.e. trajectory data are seldomly used in traffic state forecasting area, which is a challenge to us.

And the right half of table 1 shows the result of road average speed prediction. MAPE is much lower but still higher than traffic sensor datasets (e.g. SOTA models can reach less than 10% MAPE on METR-LA dataset). We consider that it is because our data cleaning process makes the sampling frequency lower than the normal case. And it should be improved in the future.

Model	Flow			Speed		
	MAE	M MAPE	RMSE	MAE	M MAPE	RMSE
AGCRN	3.409	0.332	4.37	4.642	0.238	6.587
GWNET	3.478	0.344	4.791	4.714	0.242	6.666
LSTNet	3.648	0.35	4.819	4.914	0.247	6.81
MTGNN	3.35	0.327	4.279	4.649	0.237	6.593
STNorm	3.303	0.323	4.233	4.637	0.238	6.576

Table 1: Evaluation Results

References

[1] X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi, and B. Yin, "Deep learning on traffic prediction: Methods, analysis and future directions," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[2] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *arXiv preprint arXiv:2101.11174*, 2021.

[3] Z. Yao, Y. Jiang, B. Zhao, X. Luo, and B. Peng, "A dynamic optimization method for adaptive signal control in a connected vehicle environment," *Journal of Intelligent Transportation Systems*, vol. 24, no. 2, pp. 184–200, 2020.

[4] K. Lee, M. Eo, E. Jung, Y. Yoon, and W. Rhee, "Short-term traffic prediction with deep neural networks: A survey," *IEEE Access*, vol. 9, pp. 54739–54756, 2021.

[5] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive computing*, vol. 7, no. 4, pp. 12–18, 2008.

[6] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden markov model with precomputation," *International Journal of Geographical Information Science*, vol. 32, no. 3, pp. 547 – 570, 2018.

[7] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17804–17815, 2020.

[8] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," *arXiv preprint arXiv:1906.00121*, 2019.

[9] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, 2018.

[10] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 753–763, 2020.

[11] J. Deng, X. Chen, R. Jiang, X. Song, and I. W. Tsang, "St-norm: Spatial and temporal normalization for multi-variate time series forecasting," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 269–278, 2021.