# TTI Prediction in Urban Road Network Using Computer Vision Techniques

LI Xiahan

# Table of Content

# Overview

This project has 3 phases:

Phase I:

**Input**: Get the textile format data from Didi company.

**Procedure**: Analyze and pre-process the data, then transform the textile data to image data, during which characteristics of transportation must be preserved.

**Output**: Transformed image data includes transportation characteristics from input.

Phase II:

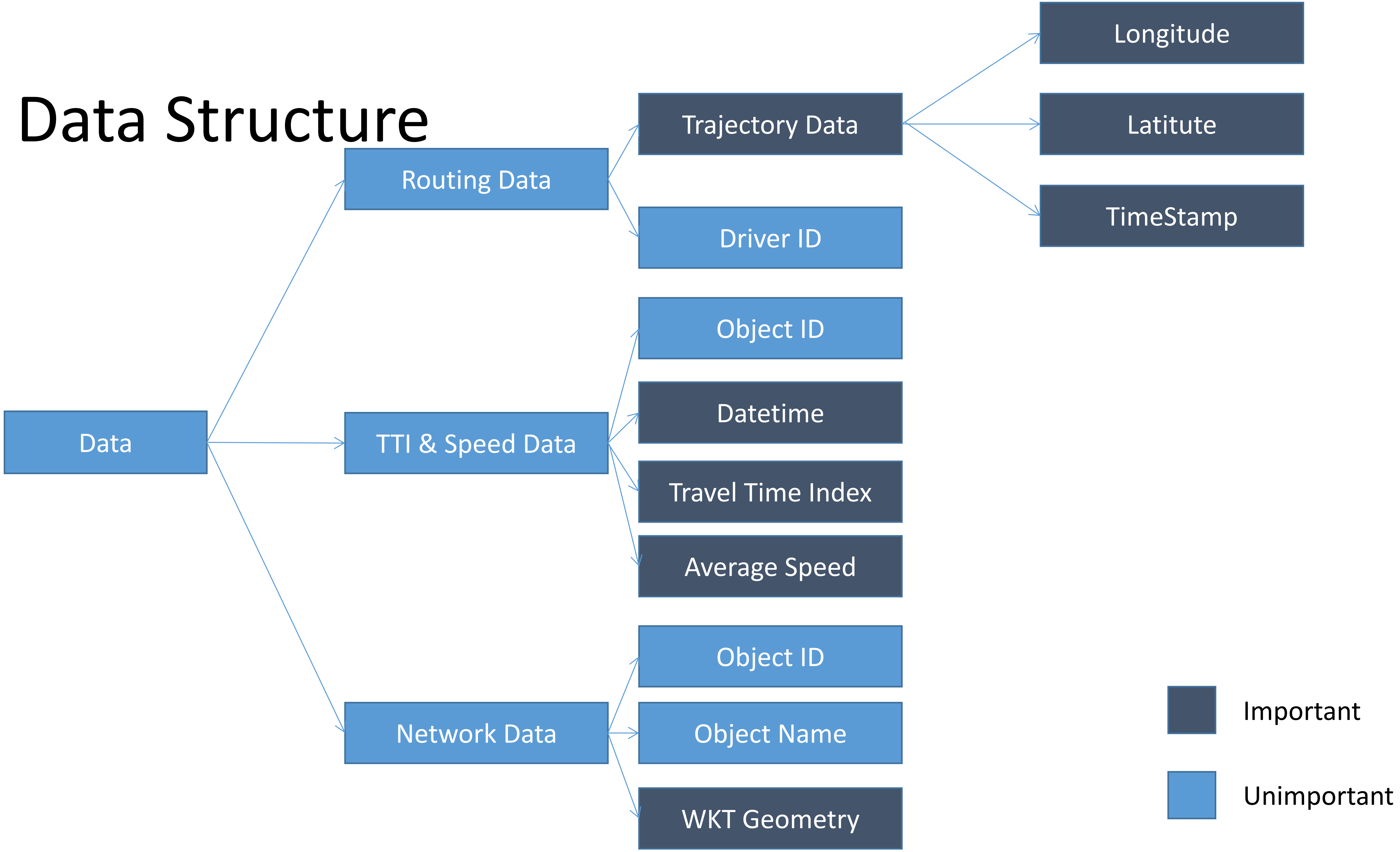
**Input**: Image data from output of phase I.

**Procedure**: Use deep learning method to do computer vision prediction

**Output**: Measurable deep learning related results, e.g. accuracy, loss.

Phase III: Discussion and try for other datasets using this method.

# Data Structure



# Routing Data

## Routing Data is consist of Driver ID and Trajectory Data.

- **Driver ID** is the desensitization of drivers' personal information. Example: 4ecea6b8473789e1fdbfea71adfb2451
- **Trajectory Data** is consist of 3 parts: **longitude**, **latitude** and **timestamp**. Example: 104.12226 30.67012 1539952150, 104.12226 30.67012 1539952160

1. **Longitude**: longitude data based on GCJ-02 coordinate system. Example: 104.12226
2. **Latitude**: latitude data based on GCJ-02 coordinate system. Example: 30.67012
3. **Timestamp**: the time when the longitude and latitude data is recorded.  Example: 1539952150. Which can be transform to datetime, like 1539952150 -> 2018-10-19 20:29:10 (UTC +8, Singapore)

| | | |
|---|---|---|
| 4ecea6b8473789e1fdbfea71adfb2451 | 8ac962c8b85724f4602dbc9e50c3ff46 | [104.12226 30.67012 1539952150, 104.12226 30.67012 1539952160, |
| 3018c0c10f32ec1aab592168bb6f6787 | 7535711890e34f9476ad616fba2648e4 | [104.05973 30.68469 1539949303, 104.05973 30.68469 1539949304, |
| 876853121ec96be317b065c286081976 | cf9d01b5bf5de4415c3a56f0c6815b53 | [104.09237 30.67939 1539949359, 104.09237 30.67939 1539949362, |
| 414b5e790bdc74bc23ce18694ce04909 | 065bc24b5194fa06b7d87c592c4d9582 | [104.09797 30.65295 1539949558, 104.09807 30.65311 1539949561, |
| bb53ed3292cbccc4faca7a20effd221b | f8f1a4a9f0ccc3af0c7a5db7b91e4fc5 | [104.04503 30.70138 1539949951, 104.04502 30.70139 1539949953, |
| d284a1b9d8b2a718fe64d78a6e8cb955 | 9423200264957ffa8cbad6f45936e46a | [104.05008 30.65319 1539949988, 104.05001 30.6532 1539949989, |

# TTI & Speed Data

TTI & Speed Data is consist of Object ID, Datetime, Travel Time Index and Average Speed

- **Object ID** is the identification data of the road. Example: 281931
- **Datetime** is the date and time when the specific road has the indicated travel time index and average speed. Example: 2018/10/19 20:00:00
- **Travel Time Index(TTI)** is a link-weighted ratio of actual travel time and free flow time. High TTI indicates the severe congestion situation of a specific road. Example: 2.45013
- **Average Speed** is the average speed of all the vehicles in the specific road at the specific time. Example: 14.7419

| | | | |
|---|---|---|---|
| 282659 | 2018/10/19 20:00 | 1.89428 | 21.0006 |
| 282964 | 2018/10/19 20:00 | 1.47032 | 24.84 |
| 281952 | 2018/10/19 20:00 | 1.35722 | 43.2719 |
| 281933 | 2018/10/19 20:00 | 1.50704 | 29.3833 |
| 282289 | 2018/10/19 20:00 | 2.61735 | 14.4739 |
| 282860 | 2018/10/19 20:00 | 1.38475 | 29.7054 |
| 282966 | 2018/10/19 20:00 | 1.08128 | 96.1934 |

# Network Data

## Network data a is consist of  Object ID, Object Name, and WKT Geometry

- **Object ID** is the identification data of the road. Example: 281931. Which is corresponding to the Object ID in TTI & Speed Data.
- **Object Name** is the Chinese name of the road, whose detailed information can be further investigated on the database like OpenStreet Map and Google Satellite Map. Example: 二环路西段:广福路,金牛大道(West Section of the Second Ring Road: Guangfu Road, Jinniu Avenue)
- **WKT Geometry** is “Well-known text representation of geometry”, which can use coordinates to represent varies of geomatric objects, like points, linestrings, polygons, networks and geometry collections, etc. Example: LINESTRING(104.10583 30.67989,104.10649 30.67871)

| | | |
|---|---|---|
| 181581 | 双流区 | POLYGON((104.11863 30.22707,104.11661 30.22718,104.11644 30.22719,104.11491 3 |
| 181582 | 简阳市 | POLYGON((104.42813 30.08827,104.42665 30.08851,104.42629 30.08857,104.42363 3 |
| 281863 | 八里桥路:北站东二路,三环路 | MULTILINESTRING((104.07437 30.71442,104.0742 30.71483,104 |
| 281864 | 八里桥路:三环路,北站东二路 | MULTILINESTRING((104.07452 30.71324,104.0745 30.71338,104 |
| 281865 | 二环路西段:广福路,金牛大道 | MULTILINESTRING((104.02746 30.64001,104.02707 30.64039),( |
| 281866 | 二环路西段:金牛大道,广福路 | MULTILINESTRING((104.02536 30.64168,104.02488 30.64216,10 |

# Methodology

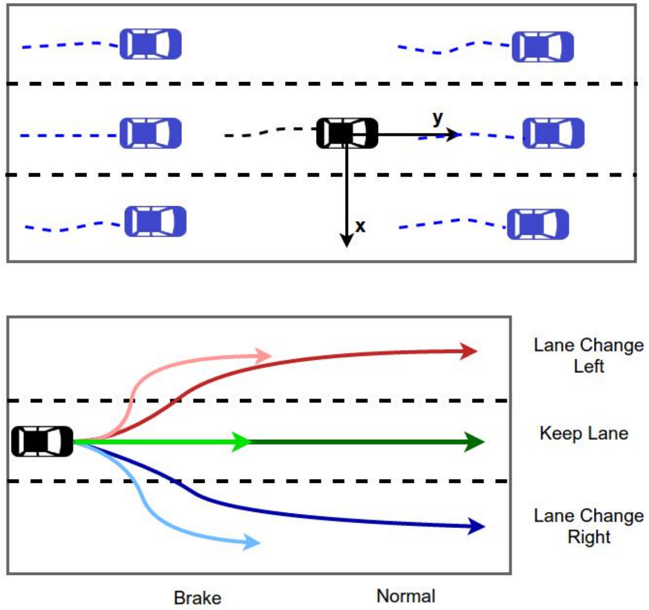
**Computer Vision Feasibility**: Discuss the feasibility of using computer vision to do urban road transportation prediction.

**RGB 3 phase construction**: Discuss the way to construct RGB 3 phases of pictures which contain trajectory, network and geometrical information.

**Design of input, output and constraints**: Discuss input, output and some constraints of this method.

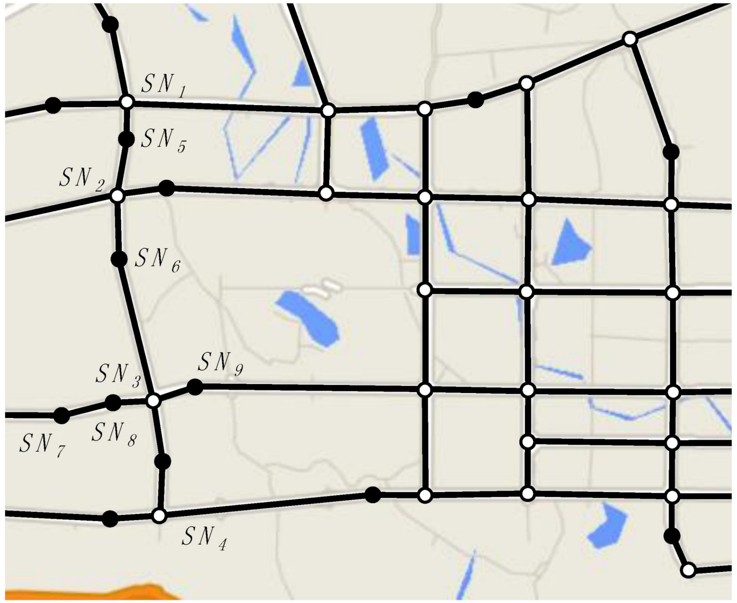
# Computer Vision Feasibility

Although most of the deep learning network used in transportation prediction, like RNN, LSTM, etc. are non-computer vision network.
Trajectory Data, Network Geometry Data and Geographical Data can be visualized.




Trajectory Data

https://www.arxiv-vanity.com/papers/1805.05499/




Network Geometry Data

https://www.mdpi.com/2220-9964/5/9/163


Geographical Data

libanciel.com/uploaded/VwAutocad.jpg

# Computer Vision Feasibility -- Similar Projects

*Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction*

Written by researchers from Microsoft Research, Beijing, China etc.

In the thesis, researchers fomulate the crowd flow into graphical matrix and use ST-ResNet to do computer vision prediction.

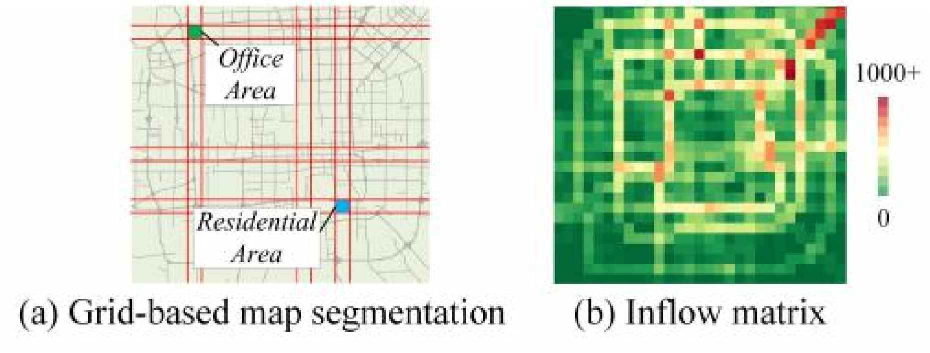



Figure 2: Regions in Beijing: (a) Grid-based map segmentation; (b) inflows in every region of Beijing

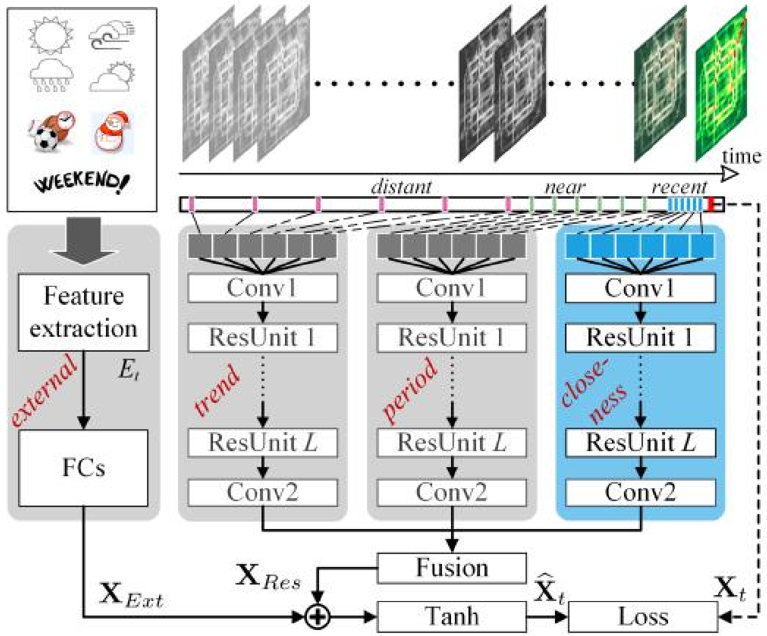



Figure 3: ST-ResNet architecture. Conv: Convolution; ResUnit: Residual Unit; FC: Fully-connected.

# Computer Vision Feasibility -- Similar Projects

Kaggle Data Science Competition: Google Doodle Recognition.
Predict the classification of figures(bird, mouse, etc.) using trajectory data of handdrawing.
Overall, accuracy and precision of using computer vision method is better than those of non-computer vision method.

| | countrycode | drawing | key_id | recognized | timestamp | word |
|---|---|---|---|---|---|---|
| | ES | [[[2, 0], [17, 74]], [[8, 32, 38], [21, 49, 68... | 5786031599124480 | True | 2017-01-01 00:50:54.745600 | owl |
| | CH | [[[36, 40, 47, 53, 74], [99, 120, 135, 134, 89... | 6291395033694208 | True | 2017-01-01 00:50:02.055500 | owl |
| 6 | FI | [[[92, 80, 49, 35, 21, 6, 0, 0, 7, 27, 67, 106... | 5746607960096768 | True | 2017-01-01 00:49:30.136290 | owl |
| | | [[[58, | | | | |

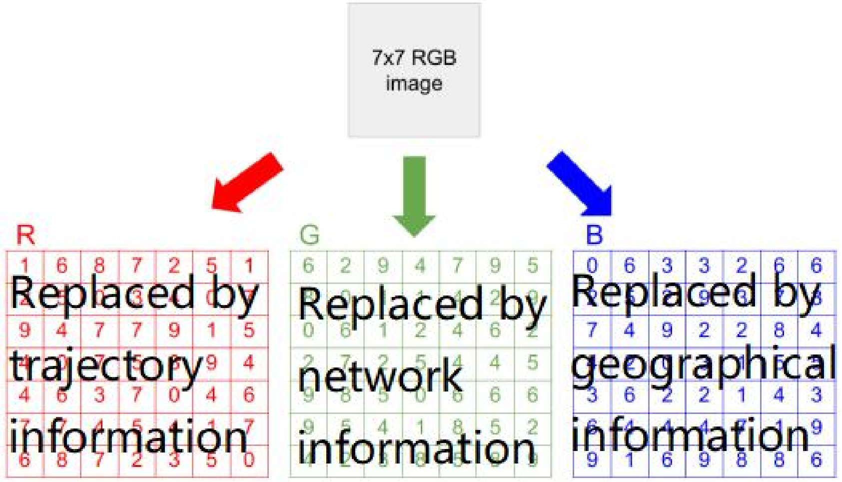
Trajectory Data

Trajectory Data Visualization

# RGB 3 phase construction

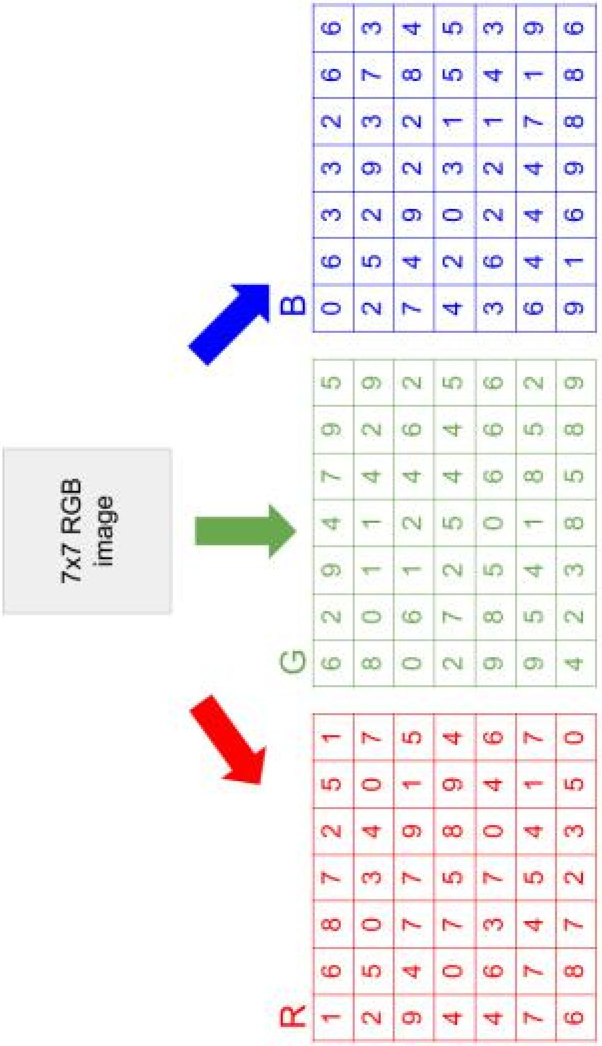
In computer vision, one pixel can be represented by 3 subpixels: value of Red(R), value of Green(G) and value of Blue(B). The value of subpixels are mutual-independent.

In 8 bpp(bits per pixel), the value is an interger ranging from 0 to 255.




1. Extract the Trajectory data,Network Geometry data and geographical data from the original dataset.
2. Construct the layout of the data distribution.
3. Put those values and layout into Red, Green and Blue phase.

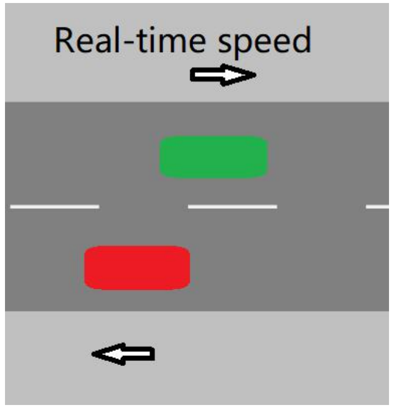
# RGB 3 phase construction




Accessibility from a given origin
Geographical identity

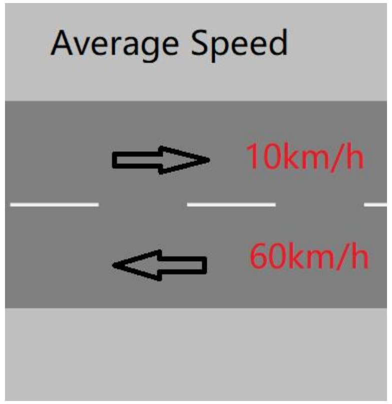

| | | | |
|---|---|---|---|
| 78 | 77 | 77 | 78 |
| 77 | 76 | 76 | 77 |
| 76 | 75 | 75 | 76 |
| 75 | 74 | 74 | 75 |

Trajectory of drivers
Real-time speed of drivers



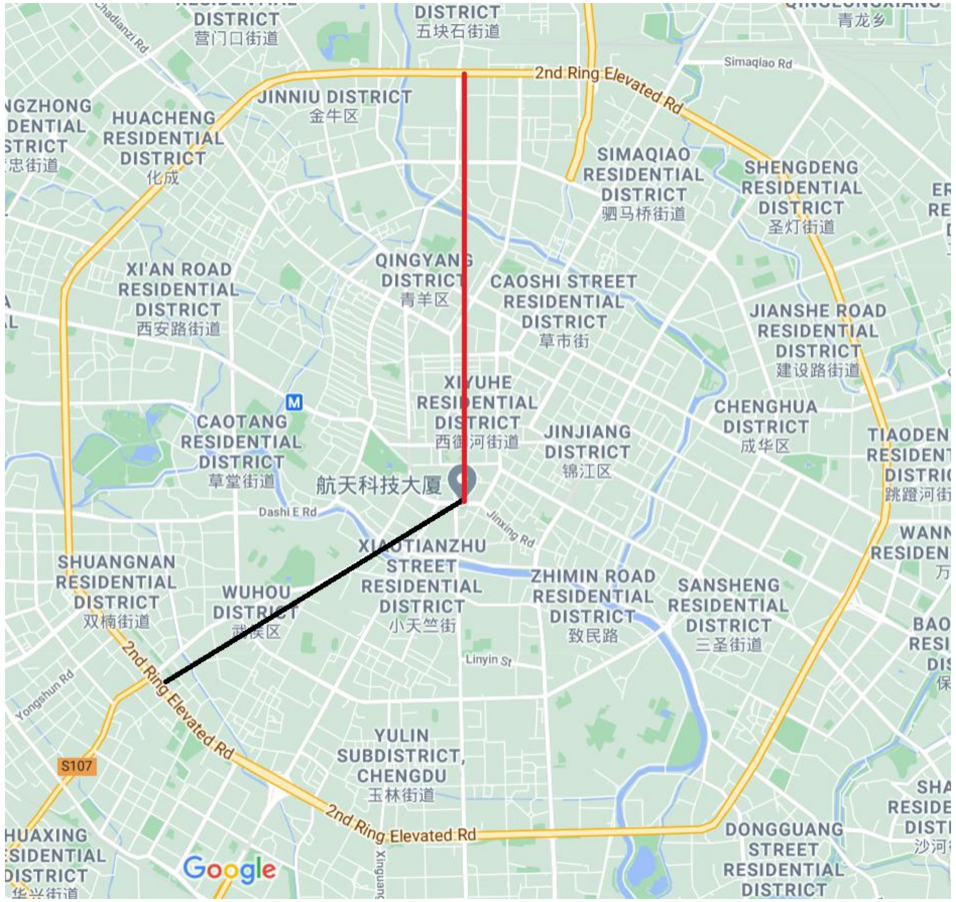

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 32 | 33 | 35 |
| 54 | 60 | 64 | 0 |
| 0 | 0 | 0 | 0 |

Geometric shape of roads
Average speed on roads

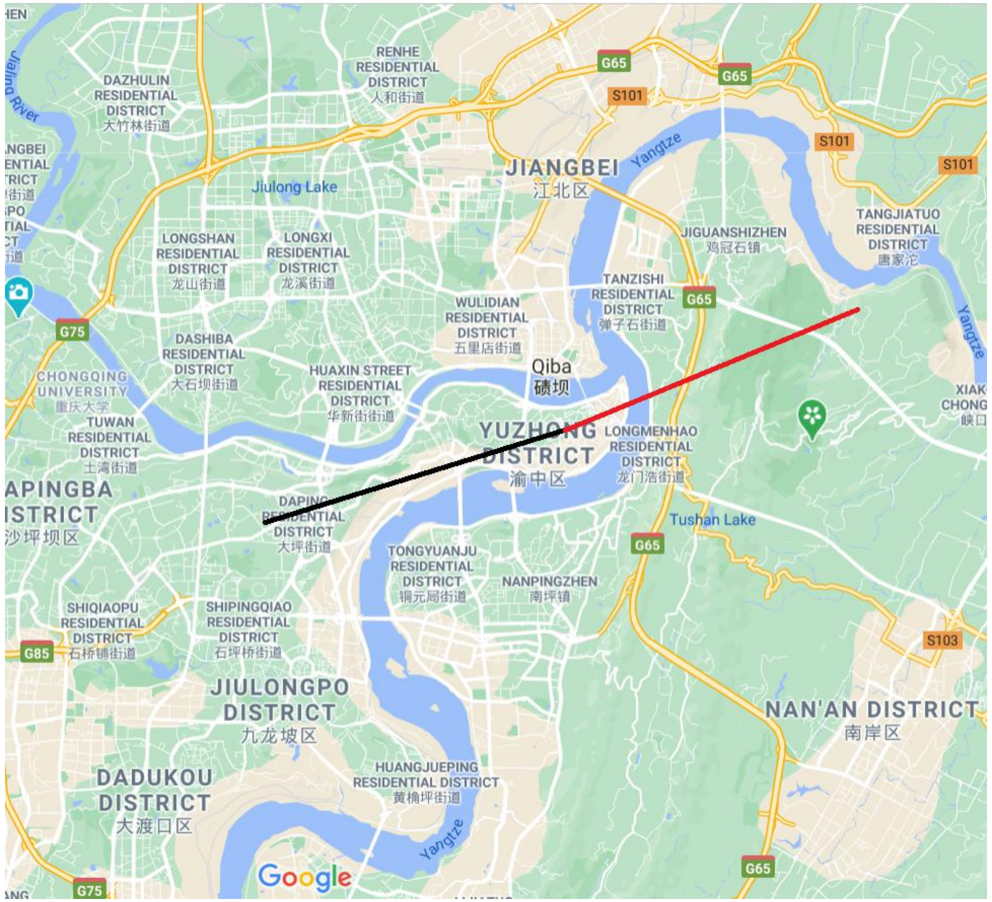



| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 10 | 10 | 10 | 10 |
| 60 | 60 | 60 | 60 |
| 0 | 0 | 0 | 0 |

# Geographical Identity




Accessibility is proportion to distance




Accessibility is not proportion to distance

# Design of input, output and constraints

Based on previous information, back to the original data, the inputs are:

- Red: WKT Geometry of road, average speed on the road.
- Green: Trajectory of driver, Real-time Speed of driver.
- Blue: Euclidean Distance from the center.

The remains are possible outputs: Travel Time Index, Datetime. Among which Travel Time Index is more reasonable.

Constrains: Every pictures represent a scheme in a <u>certain time interval</u> at a <u>certain grid location</u>.(To be continued)

# Data Pre-Process

**Data Sampling**: The original data has high memory use(>10GB), we need to sampling the data in 1 hour (E.g. Oct. 19th 20:00 - 21:00) from whole two months.

**Precision Reduction**: Reduce the precision(from 0.00001 to 0.0001) to reduce the complexity of computation and ignore the road width.

**Area Restriction**: When constructing grids, only let original layout consist dense network in urban area. Recognize the suburb and rural district and eliminate them to reduce the trivial data.

**Prospective Output**: Plan to get an about 20 million pixel picture(represent 1 hour) with acceptable memory use.

# Data Sampling

Sampling the data in 1 hour (E.g. Oct. 19th 20:00 - 21:00) from whole two months.

TTI and Avg speed data: 516KB data extracted from 880 MB zip data
Trajectory data: 156 MB data extracted from 3.84GB zip data

# Sampling Detail

The 'road.zip' is TTI and Avg speed data.

The data is time-ranked, so we just use 'skiprows' to locate the sample data.

In [2]:
```
import pandas as pd
new_data = pd.read_csv('../Data/road.zip', nrows = 2000,  header = None, compression = 'zip')
new_data
```

Out[2]:

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **0** | 283404 | 2018-01-01 00:00:00 | 1.39778 | 39.3890 |
| **1** | 283442 | 2018-01-01 00:00:00 | 1.07396 | 82.3449 |
| **2** | 283002 | 2018-01-01 00:00:00 | 1.09723 | 27.7525 |
| **3** | 282048 | 2018-01-01 00:00:00 | 1.74019 | 15.2246 |
| **4** | 283424 | 2018-01-01 00:00:00 | 1.32598 | 35.5202 |
| **...** | ... | ... | ... | ... |
| **1995** | 282042 | 2018-01-01 00:10:00 | 1.11330 | 30.3001 |
| **1996** | 282372 | 2018-01-01 00:10:00 | 1.24875 | 38.0083 |
| **1997** | 282958 | 2018-01-01 00:10:00 | 1.21440 | 41.4962 |
| **1998** | 283198 | 2018-01-01 00:10:00 | 1.19730 | 35.4399 |
| **1999** | 282392 | 2018-01-01 00:10:00 | 1.30356 | 29.8333 |

2000 rows × 4 columns

In [3]:
```
new_raw_data = pd.read_csv('../Data/road.zip', skiprows = 67385000, nrows = 15000, header = None, compression = 'zip')
new_raw_data.head()
#Totally less than 84 million rows
#67500000 10-20-08
#67400000 10-19-21
```

Out[3]:

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **0** | 282364 | 2018-10-19 19:30:00 | 1.97917 | 19.3380 |
| **1** | 281966 | 2018-10-19 19:30:00 | 1.45141 | 21.2024 |
| **2** | 282456 | 2018-10-19 19:30:00 | 1.35912 | 27.0706 |

# Sampling Detail

The 'chengdushi_1010_1020.zip' is the trajectory data, whose timestamp is randomly ranked and mixed with coordinate data.

```
[104.12226 30.67012 1539952150, 104.12226 30.67012 1539952160,
[104.05973 30.68469 1539949303, 104.05973 30.68469 1539949304,
[104.09237 30.67939 1539949359, 104.09237 30.67939 1539949362,
[104.09797 30.65295 1539949558, 104.09807 30.65311 1539949561,
[104.04503 30.70138 1539949951, 104.04502 30.70139 1539949953,
[104.05008 30.65319 1539949988, 104.05001 30.6532 1539949989,
```

```
##Extract 1 hour data for experiment, the whole data is about 2.8 million rows
import pandas as pd
Oct10_Oct20_data=pd.read_csv('../Data/chengdushi_1010_1020.zip',skiprows = 0,
                             header = None,usecols=[0,1,2],compression = 'zip',iterator = True)
loop = True
chunkSize = 10000
chunks = []
while loop:
    try:
        Oct1020_df = pd.DataFrame(Oct10_Oct20_data.get_chunk(chunkSize))
        for n, ele in enumerate(Oct1020_df[2]):
            element = ele.strip('[]').split(',')
          #  print("length: ",len(element))
            for p in range(len(element)):
                elem = element[p].split(' ')
       # print(elem[-1])
                if int(elem[-1]) > 1539950400 and int(elem[-1]) < 1539954000:
            #1539950400:2018-10-19 20:00:00; 1539954000:2018-10-19 21:00:00
                   # print(Oct1020_df.iloc[n])
                    chunks.append(Oct1020_df.iloc[n])
                    break
                else:
                    pass

    except StopIteration:
        loop = False
        print("Iteration is stopped.")
print("begin concat")
cd = pd.DataFrame(chunks)
cd.to_csv('2018-10-19-20_2018-10-19-21.csv')
print('saved')
```

```
Iteration is stopped.
begin concat
saved
```

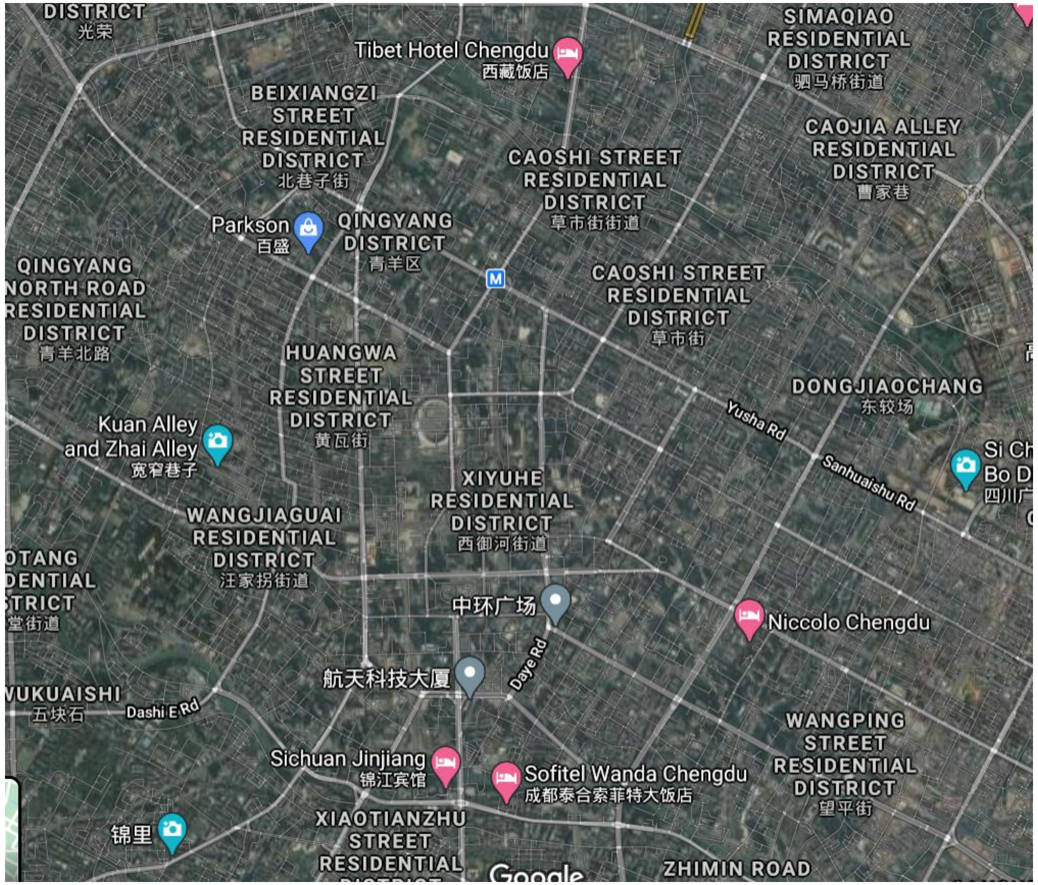
# Precision Reduction

Reduce the precision from 0.00001(Original data) to 0.0001 to reduce the 100 times complexity of computation and ignore the road width.

According to math computation:

- Width of car lane is designed to be about 3.5 m, 8 lane road need 28 m.
- 0.00001 degree of longitude/latitute = 11.12m, cannot ignore width. We need to mark the number of lanes on the road one by one, using google map.
- 0.0001 degree of longitude/latitute = 111.2m, can ignore width.

# Precision Reduction




Low precision, 1D lines, ignore width

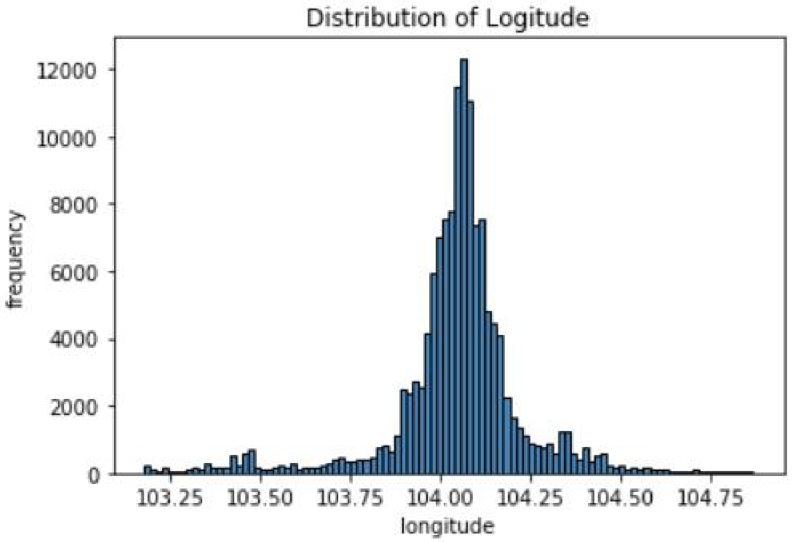

High precision, 2D strips, cannot ignore width

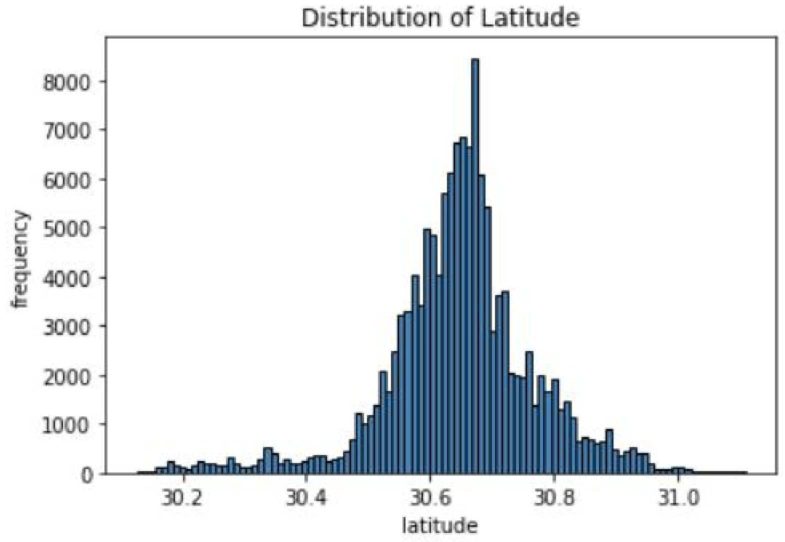
# Area Restriction

During grid construction, reduce the trivial transportation data in suburb and rural area.
In the network data, Chengdu is an Administrative District includes urban, suburb and rural area, whose logitute ranges from 103.08 to 104.87 (nearly 160km)
Restrict the center area whose transportation is about 80%-90% of the total Chengdu District.

# Area Restriction

The result of area restriction will be in grid of 4800 * 3200, with resolution 0.0001, whose coverage is 80% of the total road transportation.




```
print("80% longitude upperbound: ",ctotalmean - 1.285*ctotalstd)
print("80% longitude lowerbound: ",ctotalmean + 1.285*ctotalstd)
print("80% longitude resolution: ",int(2*1.285*ctotalstd*10000))
print("Decision: 80% -> 4800 resolution")
```

```
80% longitude upperbound:  103.80757612576159
80% longitude lowerbound:  104.28703584611256
80% longitude resolution:  4794
Decision: 80% -> 4800 resolution
```

```
print("80% latitude upperbound: ",latitotalmean - 1.285*latitotalstd)
print("80% latitude lowerbound: ",latitotalmean + 1.285*latitotalstd)
print("80% latitude resolution: ",int(2*1.285*latitotalstd*10000))
print("Decision: 80% -> 3200 resolution")
```

```
80% latitude upperbound:  30.490722038689967
80% latitude lowerbound:  30.805258330891498
80% latitude resolution:  3145
Decision: 80% -> 3200 resolution
```

# Prospective Output

After Area Restriction and Precision Reduction, we plan to get three matrixs whose dimension is at least 4800*3200. The three matrixs represent for R, G and B values of the picture.

We sample the data by one hour, out of 2 months. If we do the pre-processing for all data, we will get about 60000 pictures.

The size of one picture will be over 1GB

# Further Plan

**Picture Segmentation**: because the size of pictures in prospective outputs is large, we make small segments, like 1024*1024 or 512*512 segments sampling from original 4800*3200. Because of the geographical information stored in B phase, the relationship of positions between each segments still exists.

**Deep Learning**: Apply deep learning computer vision and do prediction. We can say all process before here is Phase I, and deep learning will be phase II.