Construct a Navigation System using OSMnx's API

Wu Yicheng

April 9, 2023

Abstract

OSMnx is a package used to analyze and visualize road network information in python. In this article, we will use OSMnx API to build a navigation system with basic functions, including Interactive map, Local location, Location query, Route planning and Deviation detection and other basic functions.

Keywords: OSMnx, Navigation System, Location query, Route planning, Deviation detection

Contents

1 Introduction **2**

2 Navigation System **2**

3 Interactive map **2**
3.1 Road information . . . 2
3.2 Building information . . . 3
3.3 Folium Interactive map . . . 3

4 Local location **3**

5 Location query **4**

6 Route planning **4**
6.1 Simple route planning . . . 5
6.2 Travel mode . . . 6
6.3 Public transport . . . 7
6.4 Avoid congestion . . . 8

7 Deviation detection **9**

8 Weekness of the navigation system **10**

1 Introduction

OSMnx is a Python package to retrieve, model, analyze, and visualize street networks from OpenStreetMap. Users can download and model walkable, drivable, or bikeable urban networks with a single line of Python code, and then easily analyze and visualize them. Users can just as easily download and work with amenities and points of interest, building footprints, elevation data, street bearings and orientations, and network routing. Navigation system has become a common tool in People's Daily life. In this article, I try to use OSMnx to build a simple navigation system.

2 Navigation System

A navigation system is a computing system that aids in navigation. A simple navigation system needs to have the following basic functions: Interactive map, Local location, Location query and Route planning. We will use OSMnx to realize these functions separately and combine them together to form a simple navigation system.

3 Interactive map

Take "Piedmont, California, USA" for example, OSMnx uses openstreetmap data as the primary data source for data acquisition, including road information and building information. It also could recognize every junction as a node, and every street as a edge and finally build a networkx graph.

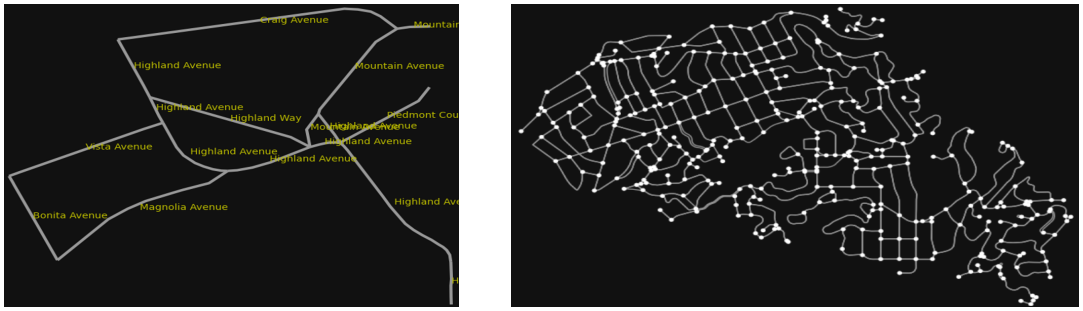
3.1 Road information

Each of these contains a lot of nodes/edges information, including name, length, whether it's a one-way street, type and so on.

u	v	key	osmid	name	highway	oneway	reversed	length	geometry	lanes	maxspeed	bridge	junction
53017091	53064327	0	6345781	Rose Avenue	residential	False	False	230.124	LINESTRING (-122.24760 37.82625, -122.24750 37...	NaN	NaN	NaN	NaN
	53075599	0	6345781	Rose Avenue	residential	False	True	122.235	LINESTRING (-122.24760 37.82625, -122.24771 37...	NaN	NaN	NaN	NaN
53018397	53018399	0	6327298	Lake Avenue	residential	False	False	121.096	LINESTRING (-122.24719 37.82422, -122.24712 37...	NaN	NaN	NaN	NaN
	53018411	0	196739937	Linda Avenue	tertiary	False	False	37.803	LINESTRING (-122.24719 37.82422, -122.24713 37...	NaN	NaN	NaN	NaN
	53097980	0	196739937	Linda Avenue	tertiary	False	True	99.976	LINESTRING (-122.24719 37.82422, -122.24773 37...	NaN	NaN	NaN	NaN

Figure 1: Road information

And we could make the road map by the information:



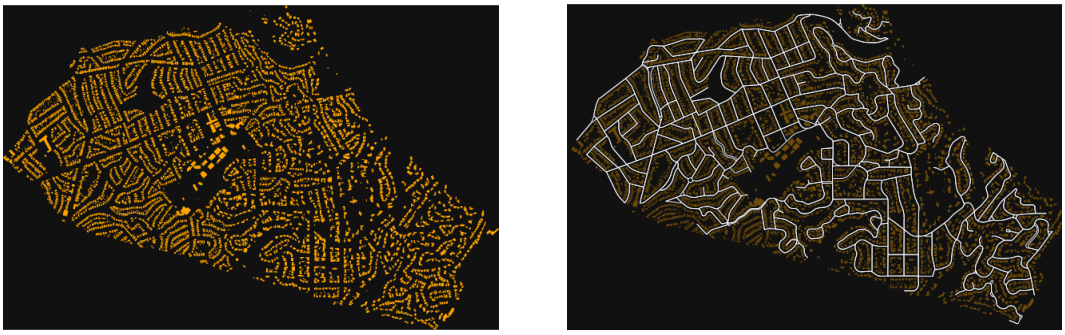
(a) Road name

(b) Road network

Figure 2: City road

3.2 Building information

OSMnx can also obtain information about the building, including its outline, location and so on. After that, we can use the building information for location and query operations. For instance, we can find all the parks in the area, bus stops and so on. Combine the building footprint with the road network and we can get a relatively nice map:



(a) Building foot-print (b) Combine road

Figure 3: City foot-print

3.3 Folium Interactive map

Since OSMnx already includes Folium package functionality and there is a good conversion interface between the two package, using Folium to draw beautiful interactive maps is a good choice. Not only that, Folium also supports the addition of markers, movement, and zooming on interactive maps, which also facilitate user interaction.

Folium maps not only make it easy to interact with users, but also show the details of each location more clearly.

The following image clearly shows the beautiful and interactive performance of using Folium maps:

(a) zoom out (b) zoom in

Figure 4: Folium map

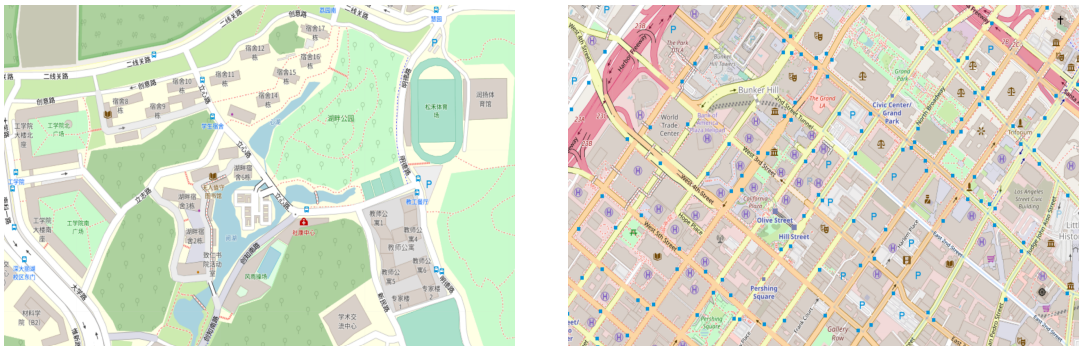
4 Local location

As a navigation system, locating the position of self-devices is an indispensable function. A computer doesn't have the same sense of location as a human. It has to rely on all the data available to determine where local location is.

The IP address is associated with the geolocation database. IP addresses are unique identifiers used by devices on the Internet. The geographic location database contains a large number of IP addresses and their corresponding geographic location information, which can be accessed and queried. When user type a web address into your browser, your computer asks a DNS server on the network to translate it into an IP address. Once the IP address is obtained, the computer can compare the IP address with the geolocation database and determine the geolocation information corresponding to the IP address. These geolocation databases usually determine the country, region, city, and even the latitude and longitude coordinates of an IP address based on the geographic location information of the IP address.

Python's geocoder package has the function to get an IP address and convert it into latitude and longitude, which our navigation system can use for getting the current location.

For instance, since I am now located in Southern University of Science and Technology, the longitude and latitude of the current location according to the IP query is (39.90403, 116.407526). If I change VPN to USA -01 T TikTok and use the same function to find local position is (34.0522,-118.2437) in Califonia LosAegeles.



(a) SUStech (b) Califonia LosAegeles

Figure 5: Local location

5 Location query

For navigation systems, queries are also an essential feature. If a user is going to a strange place, the first step he must take is to look up the location of that place in the navigation system. So we need to design a function that takes a place-name string and returns its position.

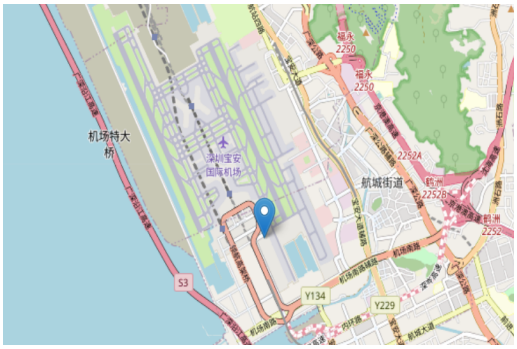
To implement this function, I used Amap's third party API. Requests a third-party api to get the latitude and longitude of the location using the Requests. Amap needs to register and create applications on Amap Open platform in advance to obtain user's exclusive key (My key: bff294f3fd631c85a1d6077987 54d4c4). Through the key and Amap's third party API interface: https://restapi.amap.com/v3/geocode/geo?, user can query the latitude and longitude of the region through Amap's built-in server.

Based on this latitude and longitude information, OSMnx can be used to obtain information about the road network around the region and draw a Folium Map based on this information.
In the figure, the marker is the position of longitude and latitude queried. Although there is some gap with the central point of the region, it is enough for us to locate this region.

6 Route planning

Route planning is extremely important in the function of navigation system. When client need to go to a place, navigation system need to plan a route that works best for the client.

(a) Shenzhen North Railway Station



(b) Bao'an Airport

Figure 6: Local query

6.1 Simple route planning

For simple route planning, the basic idea is to drive the shortest distance or take the least time. Because the navigation system already has positioning and query functions, the system can easily obtain the longitude and latitude of the starting and destination points. The system obtains the node closest to the starting and destination points as the actual start and end points and plan the two routes that drive the shortest distance or take the least time.

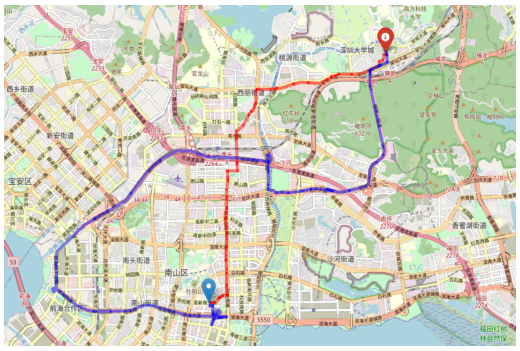
Generally, the time taken is calculated based on the length of each edge divided by the maximum speed limit.

$$time = \frac{length}{max\ \ speed}$$

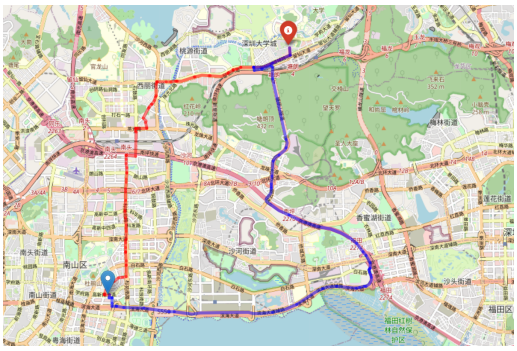
But most of the time, especially on congested urban roads, cars are not traveling at the maximum speed limit on the road, so we can weigh the ratio of distance traveled to time spent to get a better result. Of course distance travelled and time spent should be standardised first.

$$time = \frac{length}{max\ \ speed} + \alpha * length$$

The following figure is the route plan from Southern University of Science and Technology to Guangdonghai Campus of Shenzhen University. The red lines in the figure below represent the shortest routes. The blue line represents the optimal route calculated with the formula weighed.



(a) $\alpha = 0$



(b) $\alpha = 0.5$

Figure 7: Optimal route

As you can see, the blue lines almost all fall on the Nanping Expressway.

6.2 Travel mode

The most basic way for people to travel can be divided into three aspects: car, bicycle and walking. But for some roads, such as those with many steps, cars and bicycles are not passable. The same narrow roads are accessible to cyclists and pedestrians but not to cars.

For the package OSMnx, it can divide the data network into many layers, including "drive", "bike", "walk" and so on. Depending on how the customer chooses to travel, the navigation system can then select the appropriate layer and plan the route on the appropriate layer.

Take Manhattan as example, the navigation system can separately obtain layers that match the user's travel mode for route planning.

(a) drive (b) bike (c) walk

Figure 8: Travel mode

Of course, we can also apply the previous section to various layers.

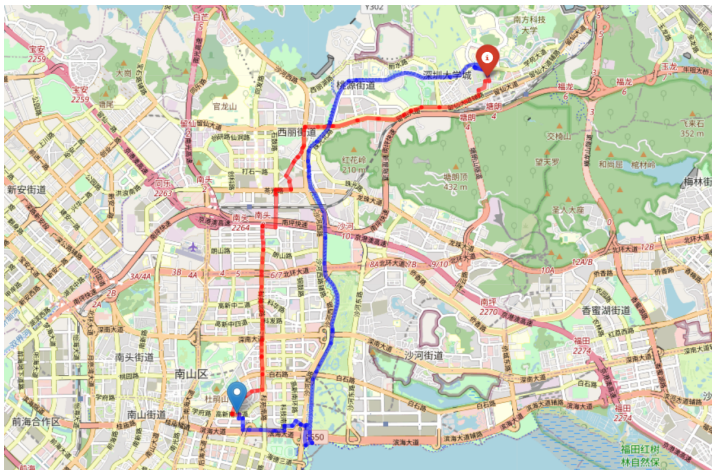


Figure 9: Bike Road

As you can see, if you want to go from Southern University of Science and Technology to Shenzhen University by bike, it is fastest to walk along the blue route (The bike route along the DaSha River).

6.3 Public transport

Of course, people are more likely to choose public transport such as railway and subway. How should the navigation system take public transport routes into account in the route planning process.

Take railway lines in Guangdong Province as an example.

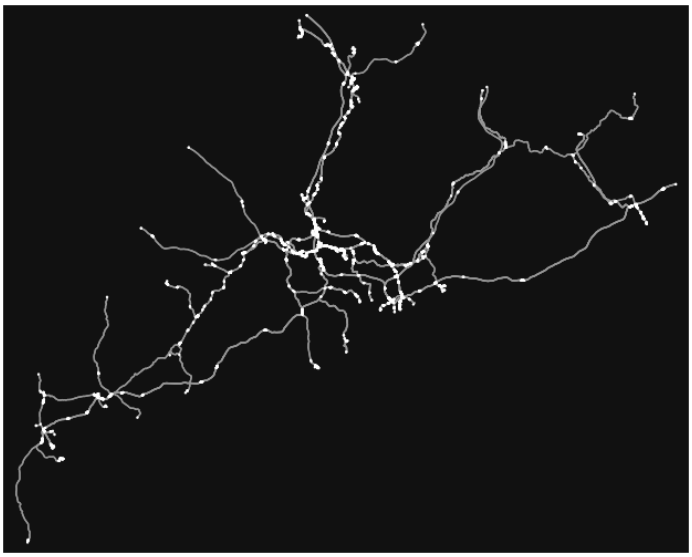
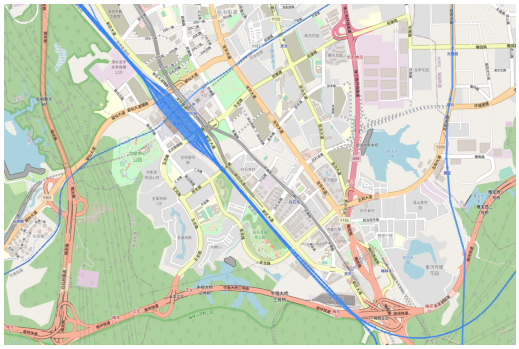
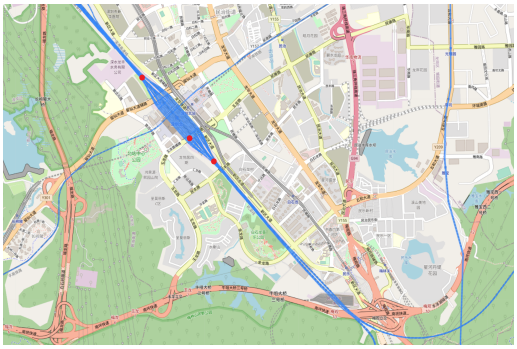


Figure 10: Railway in Guangdong

If the client needs to travel from Southern University of Science and Technology to Chaoshan, taking the train is a good choice due to the long distance. Then the navigation system should find a train station closest to the user. Above all, the system needs to read the unsimplified railway information of Guangdong Province.



(a) ShenZhenBei railway station



(b) DBSCAN algorithm

Figure 11: Optimal route

As you can see, in the unsimplified map, there are a lot of nodes and lines clustered around the train station. This is also an important basis for navigation systems to determine whether a node is near a train station. So we use DBSCAN algorithm to cluster all nodes in the map. The navigation system treats these points as train stations and looks for the shortest path based on their latitude and longitude.

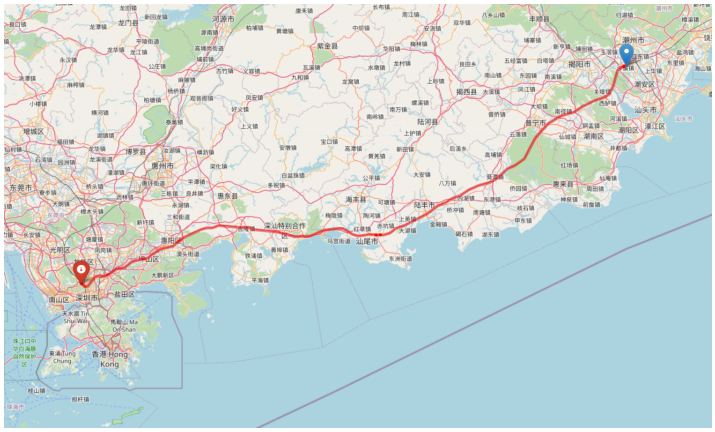


Figure 12: Shenzhen to Chaoshan

Once the path is in place, the navigation system simply guides the customer to the corresponding train station according to the start and end of the path.

6.4 Avoid congestion

Because the road network data does not contain the real-time traffic flow information, it is unrealistic to judge the road congestion situation according to the real-time traffic flow. We can only assume that the probability of congestion in central areas is higher, so navigation systems should try to avoid central roads when planning routes.

Road centrality can be measured by the following two criteria:

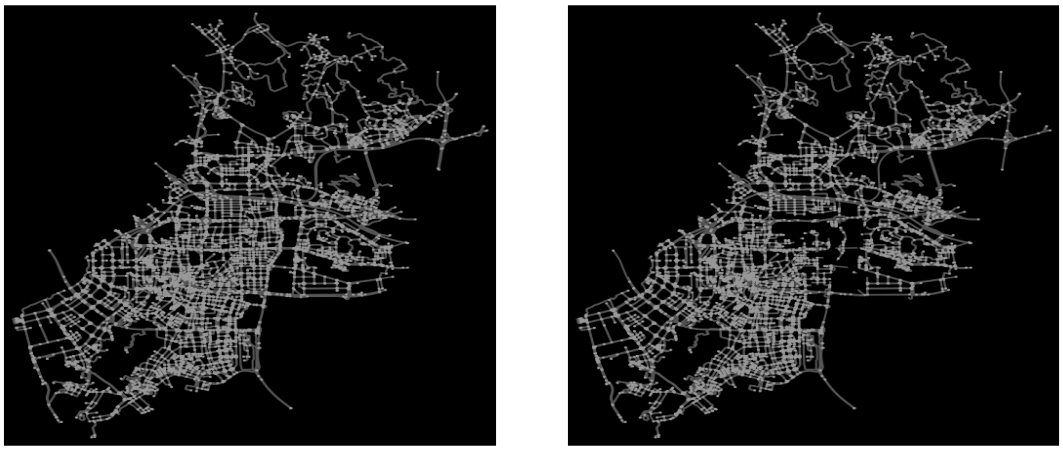
1. Closeness centrality: measures the average distance from one node to all other nodes, that is, the inverse of the average length of the shortest path from one node to the other nodes. The higher this index is, the closer this node is to other nodes in the network and the faster it can transmit information to other nodes. Therefore, it plays an important role in information transmission and communication.

2. Betweenness centrality: measures the importance of a node in a network, namely how many shortest paths pass through it. In a network, the higher the betweenness centrality of a node, the more important it plays as an intermediary between other nodes. Such nodes tend to have greater control in the network, for example, in information transmission, logistics and transport.

Take Closeness centrality as example, the navigation system can calculate the Closeness centrality of each node. In order to avoid congestion, when planning routes, the system will remove nodes with high Closeness centrality and adjacent edges of these nodes. The road network is reconstructed by using the processed nodes and edges. In the example, we discard nodes where the Closeness centrality is greater than 0.03.

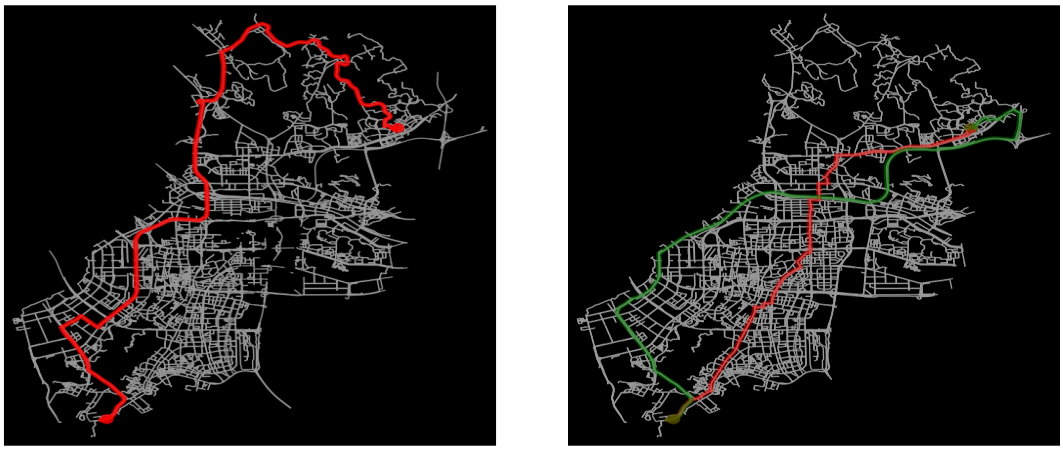
Planning in the reconstructed road network can avoid roads with high congestion probability.

As you can see in Figure 15, this action actually avoid passing the road with high probability of congestion.



(a) Origin (b) Reconstruction

Figure 13: Reconstruct network



(a) Avoid Congestion (b) Normal

Figure 14: Congestion route plan

7 Deviation detection

In real-world situations, users often deviate from navigation routes because they are unfamiliar with local routes. A qualified navigation system needs to detect in time whether the user is deviating from the route and replan the route for user.

Since the navigation system has the function of local positioning, we can set the navigation system to detect once every 5 seconds, read the current longitude and latitude, and re-plan the path. If the path is a subset of the last planned route, it indicates that there is no deviation from the route. But rearranging lines every five seconds consumes computing resources. Because the local positioning is not particularly accurate, this may also lead to errors.

Obtain the longitude and latitude of the local position each time, find the nearest edge from the local position in the optimal path and calculate the specific distance. If the distance exceeds a certain threshold, the user is considered to have deviated from the route and an optimal path is planned for the user.

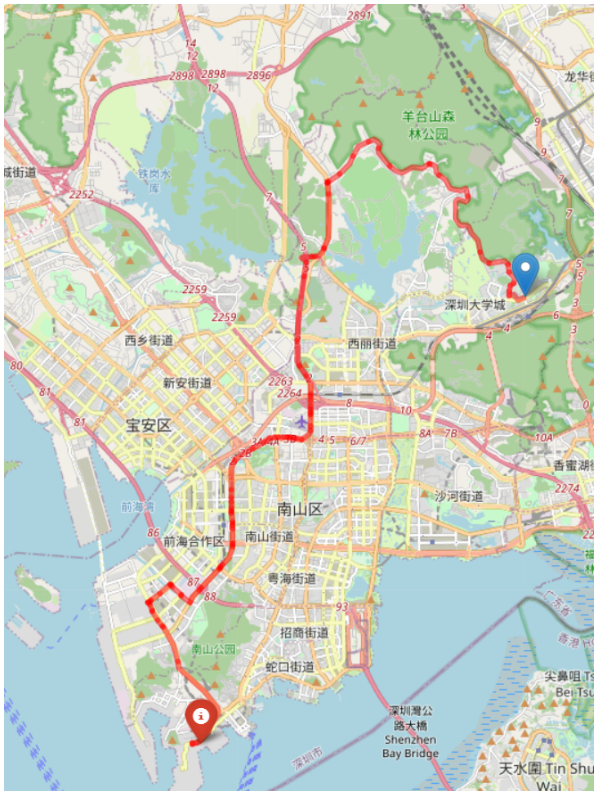


Figure 15: Shenzhen to Chaoshan

8 Weekness of the navigation system

1. The points obtained by clustering are not necessarily stations, but also train maintenance stations and so on.
2. DBSCAN parameters are difficult to determine
3. Inaccurate local positioning cause error when deviation detection