

# apollo\_base\_map\_2\_lanelet2\_osm

## 法一

### 1.遍历所有的point convert\_xy\_to\_latlon.py

```
1 import xml.etree.ElementTree as ET
2 import re
3 import os
4 from xml.dom.minidom import parseString
5
6 def read_file(file_path):
7     """Reads the entire content of a file."""
8     with open(file_path, 'r') as file:
9         return file.read()
10
11 def convert_xy_to_latlon(x, y):
12     """
13     Convert x, y coordinates to lat, lon.
14     This function is a placeholder. Replace it with actual conversion logic.
15     """
16     lat = float(x) / 100000000
17     lon = float(y) / 100000000
18     return lat, lon
19
20 def process_points_to_osm(input_text, output_file_path):
21     osm_root = ET.Element("osm")
22     node_id = -1 # Starting node ID; this script decrements ID for each point
23
24     points_pattern = re.compile(r"point\s*\{\s*x:\s*([-\.d.]+)\s*y:\s*([-\.d.]+)\s*\}")
25
26     for match in points_pattern.finditer(input_text):
27         x, y = match.groups()
28         lat, lon = convert_xy_to_latlon(x, y)
29
30         node_attribs = {
31             "id": str(node_id),
32             "visible": "true",
33             "lat": f"{lat:.8f}",
34             "lon": f"{lon:.8f}"
35         }
```

```

36     ET.SubElement(osm_root, "node", node_attribs)
37     node_id -= 1 # Ensure each node ID is unique
38
39     # Convert the ElementTree to an XML string
40     rough_string = ET.tostring(osm_root, 'utf-8')
41     reparsed = parseString(rough_string)
42
43     # Pretty print to the specified output file with newlines after each node
44     with open(output_file_path, "w") as output_file:
45         output_file.write(reparsed.toprettyxml(indent="  "))
46
47     # Specify the paths to your input and output files
48     current_directory = os.path.dirname(os.path.abspath(__file__))
49     input_file_path = os.path.join(current_directory, "base_map.txt")
50     output_file_path = os.path.join(current_directory, "output.osm")
51
52     # Process the input file to generate the OSM output
53     input_text = read_file(input_file_path)
54     process_points_to_osm(input_text, output_file_path)
55
56     print(f"Generated OSM file saved to: {output_file_path}")
57

```

## 2.所有的point\_xy转经纬度(暂未完成)

```

1  #!/usr/bin/python3
2
3  __author__ = 'ISmileLi'
4
5  from osgeo import gdal, ogr, osr
6  from pyproj import Transformer
7
8  '''
9  osgeo底层坐标转换使用的库还是proj,下面函数中的espg值需要根据自己的需求进行修改,
10  下文测试使用的是wgs84与中国区高斯-克吕格EPSG码为21460区的转换
11  '''
12
13  def lonLat_to_gauss(lon, lat, from_epsg=4326, to_epsg=21460):
14      '''
15      经纬度转高斯
16      :param lon:
17      :param lat:
18      :param from_epsg:
19      :param to_EPSG:
20      :return:

```

```

21     '''
22
23     from_spa = osr.SpatialReference()
24     '''
25     gdal版本大于3.0以后必须设置转换策略才能正确显示结果，否则结果将会输出'inf'
26     可以了解官方的这个issue说明: https://github.com/OSGeo/gdal/issues/1546
27     '''
28     if int(gdal.__version__[0]) >= 3:
29         from_spa.SetAxisMappingStrategy(osr.OAMS_TRADITIONAL_GIS_ORDER)
30     from_spa.ImportFromEPSG(from_epsg)
31     to_spa = osr.SpatialReference()
32     to_spa.ImportFromEPSG(to_epsg)
33     coord_trans = osr.CoordinateTransformation(from_spa, to_spa)
34
35     t = coord_trans.TransformPoint(lon, lat)
36     return t[0], t[1]
37
38 def gauss_to_lonLat(x, y, from_epsg=21460, to_epsg=4326):
39     '''
40     高斯转经纬度
41     :param x:
42     :param y:
43     :param from_epsg:
44     :param to_EPSG:
45     :return:
46     '''
47
48     from_spa = osr.SpatialReference()
49     #if int(gdal.__version__[0]) >= 3:
50         #from_spa.SetAxisMappingStrategy(osr.OAMS_TRADITIONAL_GIS_ORDER)
51     from_spa.ImportFromEPSG(from_epsg)
52     to_spa = osr.SpatialReference()
53     to_spa.ImportFromEPSG(to_epsg)
54     coord_trans = osr.CoordinateTransformation(from_spa, to_spa)
55
56     t = coord_trans.TransformPoint(x, y)
57     return t[0], t[1]
58
59
60 def lonLat_to_gauss_proj(lon, lat, from_epsg="EPSG:4326",
61     to_epsg="EPSG:21460"):
62     '''
63     使用proj库经纬度转高斯
64     :param lon:
65     :param lat:
66     :param from_epsg:
67     :param to_epsg:

```

```

67     :return:
68     '''
69     transfromer = Transformer.from_crs(from_epsg, to_epsg, always_xy=True) #
        WGS-84对应码->EPSG:4326, 中国高斯对应码: EPSG:21460
70     x, y = transfromer.transform(lon, lat)
71     print('lonLat_to_gauss_proj x, y:', x, y)
72     return x, y
73
74 def gauss_to_lonLat_proj(x, y, from_epsg="EPSG:21460", to_epsg="EPSG:4326"):
75     '''
76     使用proj库高斯转经纬度
77     :param x:
78     :param y:
79     :param from_epsg:
80     :param to_epsg:
81     :return:
82     '''
83     transfromer = Transformer.from_crs(from_epsg, to_epsg, always_xy=True) #
        WGS-84对应码->EPSG:4326, 中国高斯对应码: EPSG:21460
84     lon, lat = transfromer.transform(x, y)
85     print('lonLat_to_gauss_proj lon, lat:', lon, lat)
86     return lon, lat
87
88 if __name__ == '__main__':
89     lon = 116.2446370442708300
90     lat = 40.0670713975694400
91     x, y = lonLat_to_gauss(lon, lat)
92     print('x, y: ', x, y)
93     lat_t, lon_t = gauss_to_lonLat(x, y)
94     print('lon_t, lat_t: ', lon_t, lat_t)
95
96     '''
97     这里要注意pyproj的转换会交换x/y返回, 可以对比osgeo使用打印结果看出来,
98     详细了解可以参考官网文档:
99     https://pyproj4.github.io/pyproj/stable/api/transformer.html
100     '''
101     lon_t = 116.2446370442708300
102     lat_t = 40.0670713975694400
103     x_t, y_t = lonLat_to_gauss_proj(lon_t, lat_t)
104     gauss_to_lonLat_proj(x_t, y_t)

```

### 3.way\_id的读取和定义

```

1 import xml.etree.ElementTree as ET
2 import re

```

```

3 from xml.dom.minidom import parseString
4
5 def read_file(file_path):
6     """Reads the entire content of a file."""
7     with open(file_path, 'r') as file:
8         return file.read()
9
10 def convert_xy_to_latlon(x, y):
11     """
12     Convert x, y coordinates to lat, lon.
13     Placeholder function - replace with actual conversion logic.
14     """
15     lat = float(x) / 10000000
16     lon = float(y) / 10000000
17     return lat, lon
18
19 def process_roads_to_osm(input_text, output_file_path):
20     osm_root = ET.Element("osm")
21     nodes = [] # To store node elements temporarily
22     ways = [] # To store way elements temporarily
23
24     node_id_start = 1 # Initialize node ID start value
25     way_id_start = -123324 # Initialize way ID start value
26
27     road_sections = re.findall(r'road\s*{.*?}(?=\s*road\s*{|\$)', input_text,
28                                re.DOTALL)
29
30     for section in road_sections:
31         way_element = ET.Element("way", id=str(way_id_start), visible="true")
32         points = re.findall(r'point\s*{\s*x:\s*([-\.d.]+\s*)\s*y:\s*([-\.d.]+\s*)\s*}', section)
33
34         for x, y in points:
35             lat, lon = convert_xy_to_latlon(x, y)
36             node_attribs = {"id": str(node_id_start), "visible": "true",
37                             "lat": f"{lat:.8f}", "lon": f"{lon:.8f}"}
38             node_element = ET.Element("node", node_attribs)
39             nodes.append(node_element) # Store the node for later addition to
40                                         the root
41
42             ET.SubElement(way_element, "nd", ref=str(node_id_start))
43             node_id_start += 1
44
45             ET.SubElement(way_element, "tag", k="type", v="virtual")
46             ways.append(way_element) # Store the way for later addition to the
47                                         root
48
49     osm_root.append(nodes)
50     osm_root.append(ways)
51
52     with open(output_file_path, 'w') as f:
53         f.write(osm_root.toxml())
54 
```

```

45     way_id_start -= 1
46
47     # Add nodes and ways to the root element in order
48     for node in nodes:
49         osm_root.append(node)
50     for way in ways:
51         osm_root.append(way)
52
53     # Convert to string using minidom for pretty printing
54     rough_string = ET.tostring(osm_root, 'utf-8')
55     reparsed = parseString(rough_string)
56     pretty_xml_as_string = reparsed.toprettyxml(indent="  ")
57
58     with open(output_file_path, "w") as output_file:
59         output_file.write(pretty_xml_as_string)
60
61 input_file_path = "base_map.txt" # Adjust to your input file's path
62 output_file_path = "outputroad2way.osm"
63
64 input_text = read_file(input_file_path)
65 process_roads_to_osm(input_text, output_file_path)
66
67 print(f"Generated OSM file saved to: {output_file_path}")
68

```

4.

4.

Reference:

1.<https://github.com/fzi-forschungszentrum-informatik/Lanelet2>

2.XML格式定义: [https://gitlab.lrz.de/tum-cps/commonroad-scenarios/-/blob/master/documentation/XML\\_commonRoad\\_2020a.pdf](https://gitlab.lrz.de/tum-cps/commonroad-scenarios/-/blob/master/documentation/XML_commonRoad_2020a.pdf)

3.[https://blog.csdn.net/luochenzhicheng/article/details/125078521?utm\\_medium=distribute.pc\\_relevant.none-task-blog-2~default~baidujs\\_utm\\_term~default-4-125078521-blog-119347699.235^v43^control&spm=1001.2101.3001.4242.3&utm\\_relevant\\_index=5](https://blog.csdn.net/luochenzhicheng/article/details/125078521?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_utm_term~default-4-125078521-blog-119347699.235^v43^control&spm=1001.2101.3001.4242.3&utm_relevant_index=5)

#AutowareAuto 之路径规划系列教程（1）-lanelets2高精地图

4.[https://blog.csdn.net/weixin\\_55366265/article/details/122205190](https://blog.csdn.net/weixin_55366265/article/details/122205190)

#面向自动驾驶的高精度地图框架解析和实战

5.[https://blog.csdn.net/orange\\_littlegirl/article/details/106542743?](https://blog.csdn.net/orange_littlegirl/article/details/106542743?ops_request_misc=%24request_id%24biz_id%24utm_term%24E5%A6%82%E4%BD%95%E5%88%A9%E7%94%A8python%E4%BD%BFapollo%E7%9A%84%E5%9C%B0%E5%9B%BE%E8%BD%AC%E4%B8%BAlanelet2&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-5-106542743.142^v99^pc_search_result_base5&spm=1018.2226.3001.4187)

[ops\\_request\\_misc=&request\\_id=&biz\\_id=102&utm\\_term=%E5%A6%82%E4%BD%95%E5%88%A9%E7%94%A8python%E4%BD%BFapollo%E7%9A%84%E5%9C%B0%E5%9B%BE%E8%BD%AC%E4%B8%BAlanelet2&utm\\_medium=distribute.pc\\_search\\_result.none-task-blog-2~all~sobaiduweb~default-5-](https://blog.csdn.net/orange_littlegirl/article/details/106542743?ops_request_misc=%24request_id%24biz_id%24utm_term%24E5%A6%82%E4%BD%95%E5%88%A9%E7%94%A8python%E4%BD%BFapollo%E7%9A%84%E5%9C%B0%E5%9B%BE%E8%BD%AC%E4%B8%BAlanelet2&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-5-106542743.142^v99^pc_search_result_base5&spm=1018.2226.3001.4187)

[106542743.142^v99^pc\\_search\\_result\\_base5&spm=1018.2226.3001.4187](https://blog.csdn.net/orange_littlegirl/article/details/106542743?ops_request_misc=%24request_id%24biz_id%24utm_term%24E5%A6%82%E4%BD%95%E5%88%A9%E7%94%A8python%E4%BD%BFapollo%E7%9A%84%E5%9C%B0%E5%9B%BE%E8%BD%AC%E4%B8%BAlanelet2&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-5-106542743.142^v99^pc_search_result_base5&spm=1018.2226.3001.4187)

#无人驾驶算法学习（十五）：高精度地图数据存储框架Lanelet2

6.[https://blog.csdn.net/hjk\\_1223/article/details/125708035?](https://blog.csdn.net/hjk_1223/article/details/125708035?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-125708035-blog-106542743.235%5Ev43%5Econtrol&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-125708035-blog-106542743.235%5Ev43%5Econtrol&utm_relevant_index=2)

[spm=1001.2101.3001.6650.1&utm\\_medium=distribute.pc\\_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-125708035-blog-](https://blog.csdn.net/hjk_1223/article/details/125708035?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-125708035-blog-106542743.235%5Ev43%5Econtrol&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-125708035-blog-106542743.235%5Ev43%5Econtrol&utm_relevant_index=2)

[106542743.235%5Ev43%5Econtrol&depth\\_1-utm\\_source=distribute.pc\\_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-125708035-blog-106542743.235%5Ev43%5Econtrol&utm\\_relevant\\_index=2](https://blog.csdn.net/hjk_1223/article/details/125708035?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-125708035-blog-106542743.235%5Ev43%5Econtrol&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-125708035-blog-106542743.235%5Ev43%5Econtrol&utm_relevant_index=2)

#【项目】无人清扫车路径规划：基于ATSP的Lanelet2结构化道路覆盖算法

note:目前还存在bug，欢迎探讨，一切解决，共商结果！感谢来扰！！