apollo_base_map_2_lanelet2_osm

法一

1.遍历所有的point convert_xy_to_latlon.py

```
import xml.etree.ElementTree as ET
import re
import os
from xml.dom.minidom import parseString

def read_file(file_path):
    """Reads the entire content of a file."""
    with open(file_path, 'r') as file:
        return file.read()

def convert_xy_to_latlon(x, y):
    """
    Convert x, y coordinates to lat, lon.
    This function is a placeholder. Replace it with actual conversion logic.
    """
    lat = float(x) / 10000000
    lon = float(y) / 10000000
    return lat, lon

def process_points_to_osm(input_text, output_file_path):
    osm_root = ET.Element("osm")
    node_id = -1  # Starting node ID; this script decrements ID for each point

    points_pattern = re.compile(r"point\s*\{\s*x:\s*([-\d.]+)\s*y:\s*([-\d.]+)\s*\}")

    for match in points_pattern.finditer(input_text):
        x, y = match.groups()
        lat, lon = convert_xy_to_latlon(x, y)

        node_attribs = {
            "id": str(node_id),
            "visible": "true",
            "lat": f"{lat:.8f}",
            "lon": f"{lon:.8f}"
        }
```

```
            ET.SubElement(osm_root, "node", node_attribs)
            node_id -= 1  # Ensure each node ID is unique

    # Convert the ElementTree to an XML string
    rough_string = ET.tostring(osm_root, 'utf-8')
    reparsed = parseString(rough_string)

    # Pretty print to the specified output file with newlines after each node
    with open(output_file_path, "w") as output_file:
        output_file.write(reparsed.toprettyxml(indent="  "))

# Specify the paths to your input and output files
current_directory = os.path.dirname(os.path.abspath(__file__))
input_file_path = os.path.join(current_directory, "base_map.txt")
output_file_path = os.path.join(current_directory, "output.osm")

# Process the input file to generate the OSM output
input_text = read_file(input_file_path)
process_points_to_osm(input_text, output_file_path)

print(f"Generated OSM file saved to: {output_file_path}")

```

2.所有的point_xy转经纬度(暂未完成)

```
#!/usr/bin/python3

__author__ = 'ISmileLi'

from osgeo import gdal, ogr, osr
from pyproj import Transformer

'''
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,
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
'''

def lonLat_to_gauss(lon, lat, from_epsg=4326, to_epsg=21460):
    '''
    经纬度转高斯
    :param lon:
    :param lat:
    :param from_epsg:
    :param to_EPSG:
    :return:
```

```
    '''

    from_spa = osr.SpatialReference()
    '''
    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'
    可以了解官方的这个issue说明：https://github.com/OSGeo/gdal/issues/1546
    '''
    if int(gdal.__version__[0]) >= 3:
        from_spa.SetAxisMappingStrategy(osr.OAMS_TRADITIONAL_GIS_ORDER)
    from_spa.ImportFromEPSG(from_epsg)
    to_spa = osr.SpatialReference()
    to_spa.ImportFromEPSG(to_epsg)
    coord_trans = osr.CoordinateTransformation(from_spa, to_spa)

    t = coord_trans.TransformPoint(lon, lat)
    return t[0], t[1]

def gauss_to_lonLat(x, y, from_epsg=21460, to_epsg=4326):
    '''
    高斯转经纬度
    :param x:
    :param y:
    :param from_epsg:
    :param to_EPSG:
    :return:
    '''

    from_spa = osr.SpatialReference()
    #if int(gdal.__version__[0]) >= 3:
        #from_spa.SetAxisMappingStrategy(osr.OAMS_TRADITIONAL_GIS_ORDER)
    from_spa.ImportFromEPSG(from_epsg)
    to_spa = osr.SpatialReference()
    to_spa.ImportFromEPSG(to_epsg)
    coord_trans = osr.CoordinateTransformation(from_spa, to_spa)

    t = coord_trans.TransformPoint(x, y)
    return t[0], t[1]


def lonLat_to_gauss_proj(lon, lat, from_epsg="EPSG:4326",
to_epsg="EPSG:21460"):
    '''
    使用proj库经纬度转高斯
    :param lon:
    :param lat:
    :param from_epsg:
    :param to_epsg:
```

```
    :return:
    '''
    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
    x, y = transfromer.transform(lon, lat)
    print('lonLat_to_gauss_proj x, y:',x, y)
    return x, y

def gauss_to_lonLat_proj(x, y, from_epsg="EPSG:21460", to_epsg="EPSG:4326"):
    '''
    使用proj库高斯转经纬度
    :param x:
    :param y:
    :param from_epsg:
    :param to_epsg:
    :return:
    '''
    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
    lon, lat = transfromer.transform(x, y)
    print('lonLat_to_gauss_proj lon, lat:', lon, lat)
    return lon, lat

if __name__ == '__main__':
    lon = 116.2446370442708300
    lat = 40.0670713975694400
    x, y = lonLat_to_gauss(lon, lat)
    print('x, y: ', x, y)
    lat_t, lon_t = gauss_to_lonLat(x, y)
    print('lon_t, lat_t: ', lon_t, lat_t)

    '''
    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，
    详细了解可以参考官网文档:
https://pyproj4.github.io/pyproj/stable/api/transformer.html
    '''
    lon_t = 116.2446370442708300
    lat_t = 40.0670713975694400
    x_t, y_t = lonLat_to_gauss_proj(lon_t, lat_t)
    gauss_to_lonLat_proj(x_t, y_t)
```

3.way_id的读取和定义

```
import xml.etree.ElementTree as ET
import re
```

```
from xml.dom.minidom import parseString

def read_file(file_path):
    """Reads the entire content of a file."""
    with open(file_path, 'r') as file:
        return file.read()

def convert_xy_to_latlon(x, y):
    """
    Convert x, y coordinates to lat, lon.
    Placeholder function - replace with actual conversion logic.
    """
    lat = float(x) / 10000000
    lon = float(y) / 10000000
    return lat, lon

def process_roads_to_osm(input_text, output_file_path):
    osm_root = ET.Element("osm")
    nodes = []  # To store node elements temporarily
    ways = []  # To store way elements temporarily

    node_id_start = 1  # Initialize node ID start value
    way_id_start = -123324  # Initialize way ID start value

    road_sections = re.findall(r'road\s*{.*?}(?=\s*road\s*{|$)', input_text, re.DOTALL)

    for section in road_sections:
        way_element = ET.Element("way", id=str(way_id_start), visible="true")
        points = re.findall(r'point\s*{\s*x:\s*([-\d.]+)\s*y:\s*([-\d.]+)\s*}', section)

        for x, y in points:
            lat, lon = convert_xy_to_latlon(x, y)
            node_attribs = {"id": str(node_id_start), "visible": "true", "lat": f"{lat:.8f}", "lon": f"{lon:.8f}"}
            node_element = ET.Element("node", node_attribs)
            nodes.append(node_element)  # Store the node for later addition to the root

            ET.SubElement(way_element, "nd", ref=str(node_id_start))
            node_id_start += 1

        ET.SubElement(way_element, "tag", k="type", v="virtual")
        ways.append(way_element)  # Store the way for later addition to the root

```

```
        way_id_start -= 1

    # Add nodes and ways to the root element in order
    for node in nodes:
        osm_root.append(node)
    for way in ways:
        osm_root.append(way)

    # Convert to string using minidom for pretty printing
    rough_string = ET.tostring(osm_root, 'utf-8')
    reparsed = parseString(rough_string)
    pretty_xml_as_string = reparsed.toprettyxml(indent="  ")

    with open(output_file_path, "w") as output_file:
        output_file.write(pretty_xml_as_string)

input_file_path = "base_map.txt"  # Adjust to your input file's path
output_file_path = "outputroad2way.osm"

input_text = read_file(input_file_path)
process_roads_to_osm(input_text, output_file_path)

print(f"Generated OSM file saved to: {output_file_path}")

```

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

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

#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

#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?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-5-106542743.142^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?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！！！