公交轨迹数据可视化技术笔记

潘志清 2025年2月5日

1. 数据准备

包含所有轨迹记录（约6万条记录）的数据文件大小接近 1GB，在前端加载所有数据并进行可视化渲染是不现实的。因此，需要对数据进行处理，以便在前端进行可视化渲染。

考虑到公交车往往在固定的路线上行驶，因此可以首先根据线路的起止点组合对轨迹数据进行去重，以得到核心的轨迹几何数据（trajectory.json）。而后利用几何数据的索引来表示每一条轨迹数据，以减少数据量。例如我们有`index.csv`文件，其内容如下：

id	s_time	e_time	tra_index
0	2024-09-18 00:00:00	2024-09-18 00:00:00	0
1	2024-09-18 00:00:00	2024-09-18 00:00:00	1

而`trajectory.json`文件中的内容如下：

```
{
    "0": {
        "path": [
            [116.397477, 39.908692],
            [116.397477, 39.908692],
            ...]},}
```

我们在可视化时就可以灵活地根据 `index.csv` 中的 `tra_index` 来获取对应的轨迹数据，以减少总的数据量。使用这种方法，我们可以将原始数据的大小减少到原来的 1/100 左右，即8MB左右。

2. 数据可视化

使用如下函数创建`deck.gl`轨迹图层，该图层仅仅创建一帧图像，即`currentTime`时刻的图像。在实际应用中，我们可以通过不断更新`currentTime`来实现动态轨迹的可视化。可以发现，并不需要将所有的轨迹数据加载到该图层中，而是根据`currentTime`来动态加载当前时刻的活跃轨迹数据。（见 3）

```
function createTripsLayer(data, currentTime) {
    return new TripsLayer({
        data,
        currentTime,
        //...
    });
}
```

3. 可视化性能优化：基于 Webworker 数据切片渲染（时间窗口）

Webworker 技术可以在浏览器的主线程之外运行 JavaScript 代码，因此我们可以将当前帧的轨迹数据的加载和渲染放在 Webworker 中进行，以减轻主线程的负担。我实现了一个时间窗口机制，即 Webworker 会根据当前时间戳检索出当前帧的活跃轨迹数据，并将其传递给主线程进行渲染。

4. 存在的问题

某一时刻（例如下午三点左右）可能存在太多的活跃轨迹数据，这样仍然会给前端的渲染带来较大的压力（即存在卡顿现象）。若不追求过分真实的可视化效果，可以考虑对活跃轨迹数量进行限制，例如最多只同时显示500条活跃轨迹。