使用行车轨迹估计交通信号灯周期问题

摘要

获取交通信号灯的红绿周期并通过电子地图展示给用户,可以提供极大的便利,但由于许多信号灯无法从交通管理部门获取所有信号灯的数据，需要使用大量用户的行车轨迹估计交通信号灯的周期。

针对问题一，在信号灯周期固定且所有行车轨迹数据已知的条件下，首先通过每辆车的运动状态来定义每辆车对于红灯的反应时间并利用数据计算出结果。然后利用均匀分布的**极大似然估计** (MLE) 来近似红灯时长。利用傅里叶变换（Fourier Transform）得到一些可能性较大的信号灯周期长度。最后再利用**自相关函数（ACF）**求出这些周期中自相关系数最大的周期，即为模型估计的信号灯周期。

针对问题二，文章讨论了样本车辆不全、车流量大小和轨迹数据定位误差对模型估计精度的影响。通过定量分析自相关系数的变化，发现样本车辆不足或车流量较小会导致信号周期判断变得复杂，而较大的车流量有助于提高估计的准确性。此外，模型通过设定状态的速度、加速度判断阈值来规避、减小定位误差的影响。

针对问题三，针对信号灯周期可能变化的情况，文章建立了一个数学模型来快速检测周期变化并计算新周期。该模型通过定义最小时间单元、构建自适应时间窗口、设置周期变化的判断指标，以及明确周期变化时刻和检测条件，实现了对信号灯周期变化的实时监测。

针对问题四，分析某路口连续 2 小时内所有方向样本车辆的轨迹数据，以确定不同行驶方向的信号灯周期。针对路口多方向的问题，文章通过 K-Means **聚类**分析识别车辆行驶方向，将车辆根据通过路口前后的行驶方向进行分组。并合并对向车道的相同转向方向的车辆轨迹数据,基于这些数据对每个信号灯的周期进行分析,从而得到不同信号灯的周期结果。

交通信号灯问题与出行息息相关，通过分析样本车辆的行驶轨迹，研究其特性及规律，建立合理有效的数学模型，对人们的便利生活、出行方便都具有重大的现实意义，可解决部门交通信号灯未接入网络，无法获取数据以及临时交通信号灯的周期无法被及时获取的情况。因此本文所建立的精确度较高的预测模型能够为电子地图服务商提供数据参考，具有现实意义。

关键词：傅里叶变换、自相关函数、自适应时间窗口、K-Means 聚类

目录

1 问题重述

电子地图服务商需要准确获取城市路网中所有交通信号灯的红绿周期，以便为用户提供更好的服务。但是由于存在信号灯未联网、人工检测不实际等原因。该公司计划使用大量客户的行车轨迹数据估计交通信号灯的周期。附件中收集了不同情况下不同路口的车辆轨迹数据。数据包括时间点、车辆编号以及车辆的具体坐标。

为了较准确地估计交通信号灯的周期，本文需要分析研究下列问题：

问题一：在信号灯周期固定不变且所有行车轨迹都已知的情况下，根据数据，建立合理的数学模型，求出相应路口相应方向的信号灯周期。

问题二：讨论样本车辆不全、车流量、轨迹数据存在定位误差等因素对上述模型估计精度的影响。考虑上述因素，并根据数据，求出相应路口相应方向的信号灯周期。

问题三：在信号灯周期可能发生变化的情况下，建立适当的数学模型，尽快检测出其周期变化，指出每次周期变化的时刻，求出变化后的新周期。同时，指明该模型识别出周期变化所需的时间和条件。

问题四：考虑所有因素可能造成的影响，建立合适的数学模型，根据数据识别出相应路口信号灯的周期。

2 问题分析

2.1 问题 1 的分析

问题 1 要求在已知所有车辆的行车轨迹且信号灯周期固定不变的条件下，利用路口的车辆轨迹数据，建立合理有效的数学模型来求解出路口相应方向的红灯、绿灯周期。

针对问题 1 的周期判断问题，经过数据的处理以及查阅文献，首先通过每辆车的运动状态来**定义每辆车对于红灯的反应时间**并利用数据计算出结果。然后根据假设，所有对红灯做出反应的车辆的反应时间在 0 到 T_{red} 中成均匀分布。于是，我们对所有的反应时间，利用**均匀分布的极大似然估计**，得到车辆反应时间的**区间上限** $\hat{T}$。再利用区间上限的估计值去近似红灯时长。

然后，我们的目标转向计算**完整的信号灯周期**。根据分析，我们将存在车辆**由静止等待转向启动加速**的时刻定义为**信号灯周期的开始时刻**。根据所有的开始时刻，我们利用**傅里叶变换**得到所有可能的信号灯周期长度。最后再利用自相关系数（AC）的值，选取最可能的信号灯周期 C，再结合红灯时长，最终求解得到绿灯时长 T_{green}。

2.2 问题 2 的分析

问题 2 要求考虑样本车辆不全、车流量、轨迹数据存在定位误差等因素对上述模型估计精度的影响。

上述因素对模型估计精度的影响，会通过影响我们对于车辆运动状态的判断，从而间接影响我们计算红灯时长以及信号灯周期。因此，除了定性分析阐述影响外，我们还**定量**地利用**自相关系数**的变化来体现它们的影响，并实现可视化直观地展示它们的影响。

2.3 问题 3 的分析

问题 3 的目标是在信号灯周期可能发生变化的条件下，快速识别这种变化及其新周期。

我们采用了**基于问题 1 模型的方法**，通过将 2 小时的车辆轨迹数据分割成多个**最小时间单元**，构建了一个数学模型来评估不同时间单元间的周期变化：这个过程首先需要定义周期变化的判断指标，然后设定最小时间单元长度。并对数据量不足的时间单元进行合并，实现构建**自适应时间窗口**，这一点至关重要。最后，根据**指标差异**判断周期变化的时刻并计算新周期。

2.4 问题 4 的分析

问题 4 的目标是分析某路口连续 2 小时内所有方向样本车辆的轨迹数据，以确定不同行驶方向的信号灯周期。

首先，针对多方向的问题，我们通过 K-Means **聚类分析**识别车辆行驶方向，将车辆根据通过路口前后的行驶方向进行分组。此后，我们合并对向车道的相同转向方向的车辆轨迹数据，基于这些数据对每个信号灯的周期进行分析，从而得到不同组的信号周期结果。

3 模型假设

本文做出如下**假设**：

(1) 每一辆车在信号灯周期内任一时刻到达路口处的概率是均匀分布的。

(2) 所有司机都是遵守交规的，即不会出现类似闯红灯的行为，这样他们的行驶轨迹才对于红绿灯的时长判断有价值。

(3) 司机在观察到前方路口为红灯的瞬间即开始减速。

(4) 每个路口使用该电子地图软件的用户占所有车辆的比例是相同的。

(5) 停止在路口等待红灯的司机在绿灯亮起的瞬间就启动加速。

(6) 根据对数据的观察，我们发现，某些左转的车辆经历了：静止等待、加速后减速再静止等待的过程，假设这是他们**进入待转弯区**的运动原因。

4 符号说明

符号	说明
C	信号灯周期时长（绿灯与红灯时长之和）
$\hat{T}$	反应时间的区间上限的极大似然估计值
T_{red}	当前路口红灯时长
T_{green}	当前路口绿灯时长
t_i	第 i 辆车遇到红灯时红灯剩余的时长
t_i^{dec}	第 i 辆车在观察到红灯后的减速时长
t_i^{stop}	第 i 辆车在红灯路口停止等待红灯的时长
ρ_k	时间序列在滞后 k 时自相关系

5 数据的预处理

5.1 车辆速度的计算

为了更直观地表示每辆车的运动状态，从而反映出信号灯的不同状态对车辆运动状态的影响，我们针对车辆行驶位置与时间数据，利用下面的公式计算出每个时刻记录中的车辆的瞬时速度。

$$v = \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{\Delta t}$$

考虑到数据每秒采样一次，我们将 Δt 代入为 1 秒。对附件中的所有数据，我们都运用上面的思路进行预处理计算出相应车辆的瞬时速度，为后面的行驶状态分析提供便利。

5.2 异常数据的修正

通过对数据可视化后的浏览和检查，我们发现部分原始数据存在异常，对数据预处理的手段是对**异常数据进行修正**。下面是我们对异常数据的定义以及处理思路。

针对题设所给的时刻与车辆位置信息，计算出每时刻相应车辆的速度信息，以表现该时刻对应车辆的运动状态。根据观察运动状态，发现某些车辆的运动状态如下图所示：在静止状态下突然起步后又突然静止。这表示车辆的位置信息在静止的时刻发生了突变，是位置信息异常导致的，故需要修正为静止状态。注意，我们只修正了相应车辆的运动状态即速度，并没有修改其位置信息。

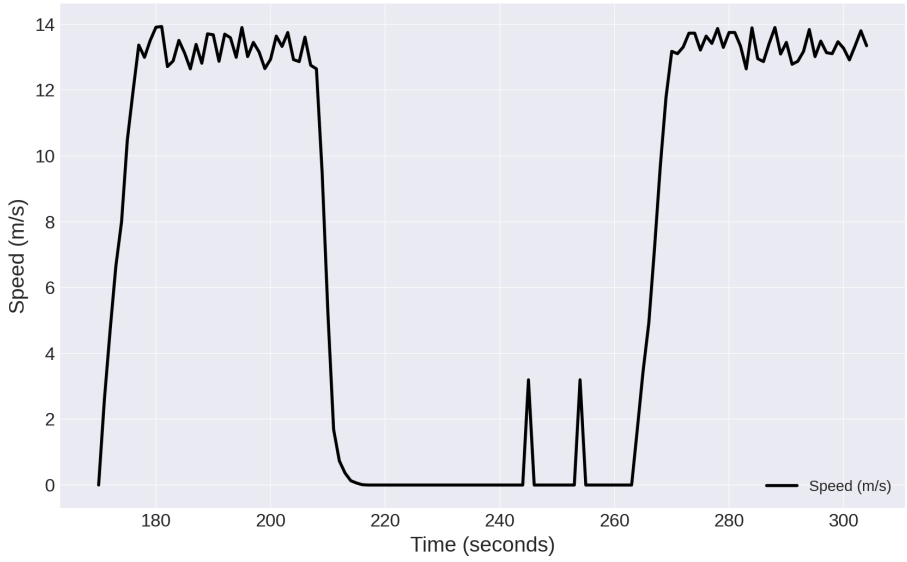

图 1：需要修正的异常情况

上图是以附件 A2 中车号为 74 的行驶状态为例的，不难发现：在 240 秒到 260 秒之间出现了两次有静止开始的跳变，这是异常数据，需要利用上面的修正方法进行修正。

5.3 不完整数据的舍弃

在处理涉及车辆运动状态的数据集时，我们经常需要确保数据的准确性和完整性，以便模型能够准确预测和分析车辆的行为。在这种情况下，对于那些记录中出现的最后一秒的车辆数据，我们决定将其从数据集中删除。这一决策基于一个重要考虑：最后一秒的数据可能不足以反映车辆的最终运动状态，因为它可能是受到截断的，导致我们无法准确判断该车辆的行为模式。

如果保留这些不完整的记录，可能会导致数据分析和模型训练时出现误差，因此对整个研究的影响是负面的。例如，在进行交通流预测或车辆行为分析时，这些不完整的数据可能会引入噪声，从而影响模型的泛化能力和预测精度。

因此，为了保证模型的准确性和数据分析的可靠性，选择删除这些最后一秒的记录是必要的。通过这样的数据清洗，我们可以减少潜在的误导信息，确保数据集中的每一条记录都能够真实可靠地反映车辆的运动状态，从而为后续的数据分析和模型建立提供坚实的基础。

6 模型的建立与求解

6.1 问题 1 的模型建立与求解

问题 1 要求在已知所有车辆的行车轨迹且信号灯周期固定不变的条件下，利用路口的车辆轨迹数据，建立合理有效的数学模型来求解出路口相应方向的红灯、绿灯周期。本文通过以下个步骤进行建模：

- **步骤一：**定义每辆车的对红灯的反应时长为其开始减速到其静止后重新启动的时间。
- **步骤二：**根据假设 (1)，利用**均匀分布的极大似然估计**，求出相应路口的红灯时长。
- **步骤三：**利用车辆启动信息定义信号灯周期起始时刻和偏移时长，并计算出每一时刻启动的车辆数，利用**傅里叶变换**求出可能的若干信号灯周期。
- **步骤四：**利用**自相关系数**，寻找出最可能的信号灯周期。

6.1.1 对于每辆车对红灯的反应时长的定义

首先，利用预处理得到的车辆速度数据对每辆车的运动状态进行可视化分析，经过仔细地观察以及分析，我们发现：车辆在行驶的过程中有四种状态：近似匀速、刹车减

速、启动加速以及静止等待。下图是附件 A1 中 8 号车辆的运动状态随时间变化的图像，图像中可以很清晰地观察到上述的四种状态。

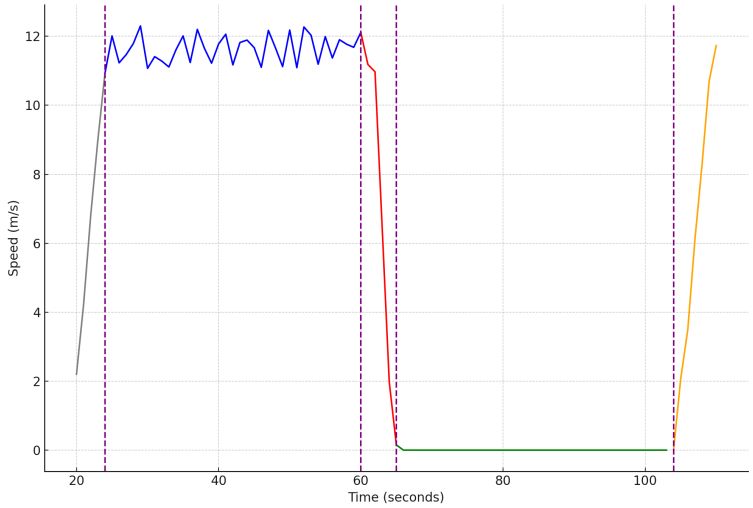

图 2：A2-8 车的速度时间图像

注：蓝色、红色、绿色、橙色分别表示启动近似匀速、刹车减速、静止等待和启动加速状态。

根据四种状态的分析，我们发现，能体现出红灯对运动状态的影响的只有刹车减速阶段、静止等待阶段以及启动加速阶段。为了让车辆流动更准确地反映交通灯的变化，考虑下面的因素：

- **刹车减速：**当司机看到前方的红灯时，会开始减速，这个阶段的开始标志着车辆对红灯的反应开始。
- **静止等待：**车辆完全停下来等待红灯变绿，这个阶段是汽车等待红灯的直接体现。
- **启动加速：**红灯变绿后，车辆重新加速离开交叉口。这个阶段标志着车辆对红灯的反应结束。

在我们的模型中，对于上面三个状态有着严谨的定义：

- **刹车减速：**车辆由匀速时的速度开始减速，直至最后的速度小于 $0.5m/s$
- **静止等待：**车速小于 $0.5m/s$

- **启动加速：**启动的时刻我们定义为：当前的速度小于 $0.5m/s$ 且当前的加速度大于 $0.5m/s^2$

综合上面对于车辆对红灯的反应的阐述，我们定义：每辆车对于红灯的反应时长等于其刹车减速的时长加上其静止等待红灯的时长。即：

$$t_i = t_i^{dec} + t_i^{stop}$$

6.1.2 利用均匀分布的极大似然估计求出路口红灯时长

根据我们的假设，认为所有对红灯做出反应的车辆在红灯期间任何时间到达路口的概率是相同的。

在上述假设下，我们令红灯的持续时长是 $t = T_{red}$，那么所有对红灯做出反应的车辆对红灯做出反应的时长的概率密度函数应该是均匀的。在这种情况下，所有对红灯做出反应的车辆的反应时间也将遵循一个从 0 到 T_{red} 的均匀分布。

那么，因为所有对红灯做出反应的车辆的反应时间也将遵循一个从 0 到 T_{red} 的均匀分布，所以我们可以用所有的反应时间的区间上限的极大似然估计值来近似红灯时长 T_{red}。

结合均匀分布的区间上限极大似然估计（MLE）的公式：

$$\hat{T} = max\{t_1, t_2, ...\}$$

我们用所有反应时间的极大似然估计来近似红灯的时长，即：

$$T_{red} = \hat{T}$$

6.1.3 利用傅里叶变换求解出若干可能的信号灯周期

首先，根据假设（5），我们设定：存在车辆由静止等待到启动加速的时刻都定义为信号灯周期的开始。这一定义使我们能够将车辆的运动状态与信号灯的颜色变换直接关联，从而更精确地分析交通流的响应时间和信号灯周期的效果。在这个设定下，我们可以观察到，每当信号灯变为绿灯，静止的车辆开始加速，标志着新的周期的开始。此外，这种方法也便于我们收集数据和进行时序分析，因为每个周期的起点都是明确且一致的，这样一来，通过测量从一个绿灯开始到下一个绿灯开始的时间间隔，我们可以准确地确定信号灯的周期长度。

针对已知信号等周期开始的时间序列，我们的目标是求解出信号灯的周期 C。于是，我们考虑**傅里叶变换**：它能将时域中的信号转换为频域中的信号，它能帮助我们识别周期性的变化。傅里叶变换是一种将时域、空域数据转化为频域数据的方法，任何波形（时域）都可以看做是不同振幅、不同相位正弦波的叠加（频域）。对于一条具备周期性的时间序列，它本身就很接近正弦波，所以它包含一个显著的正弦波，周期就是该正弦波的周期。

那么，将傅里叶变换的模型应用于本题的数据，我们可以针对每一个路口求出若干可能的信号灯周期。为了更直观地表现结果，我们下面展示对 A4 的处理结果：

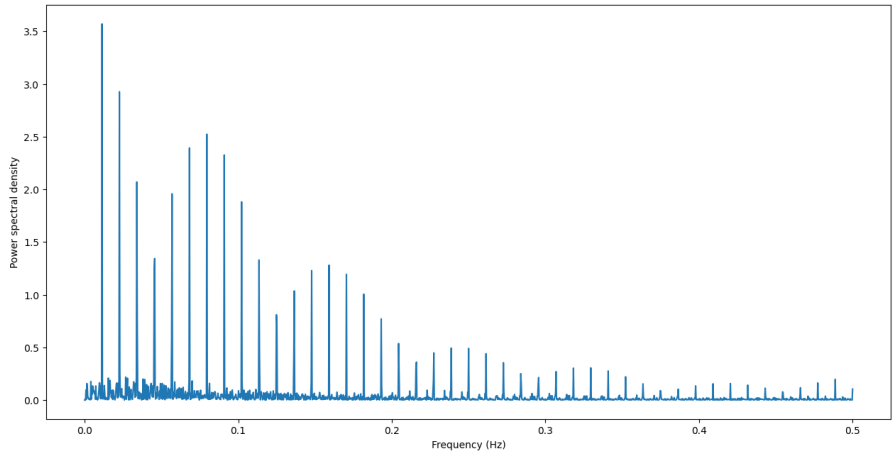

图 3：A4 序列的功率谱密度 (PSD) 图

PSD 显示信号中每个频率包含多少总功率。相应地，周期是通过计算信号的傅里叶变换的幅值平方得到的。这为后文的选择信号灯周期做好了处理。

6.1.4 利用自相关系数寻找出最可能的信号灯周期

为了从上述傅里叶变换得到若干可能的信号周期求解出真正的信号周期，我们考虑**自相关系数**模型。自相关函数（ACF）是一个重要的统计工具，用于分析时间序列数据。在时间序列分析中，自相关函数帮助我们了解数据中各个时间点的值与其过去值的关系程度。这种分析对于识别数据中的周期性模式非常有用。

具体来说，它计算时间序列在不同时间滞后时的相关系数。对于一个时间序列 Y_t，在滞后 k 时自相关系数 ρ_k 的值为：

$$\rho_k = \frac{E[(Y_t - \mu)(Y_{t-k} - \mu)]}{\sigma^2}$$

其中，μ 是时间序列的均值，σ 表示时间序列的方差。

利用 ACF 模型，我们对上面的求出的若干可能的信号灯周期进行处理，让自相关系数最大的的信号灯周期作为真正的信号灯周期。其可视化结果我们将在本文后面展示。

6.1.5 表格填写

根据上述模型，我们利用附件中的数据，计算出每个路口的红灯时长以及信号灯的周期长度。结合题干条件，信号灯只包括红灯和绿灯，我们求出绿灯时长：

$$T_{green} = C - T_{red}$$

于是，我们将结果填入表格中：

路口	A1	A2	A3	A4	A5
红灯时长（秒）	76	59	83	72	67
绿灯时长（秒）	29	29	22	16	21

表 1

6.2 问题 2 的讨论分析与求解

问题二要求讨论样本车辆不全、车流量、轨迹数据存在定位误差等因素对上述模型估计精度的影响。并根据数据，求出相应路口相应方向的信号灯周期。我们对于上述影响因素的讨论思路如下：

- 通过 ACF 量化样本车辆不全给信号灯周期判断带来的影响
- 通过 ACF 量化车流量的大小给信号灯周期判断带来的影响
- 模型设定的状态阈值规避了较小的定位误差带来的扰动

6.2.1 样本车辆不全给信号灯周期判断带来的影响

我们的模型是通过检测样本车辆在启动加速阶段的开始来确定交通信号灯周期的开始。这种方法依赖于样本车辆在信号灯变绿时准确记录其启动时刻。然而，如果样本车辆的数据不完整或不充分，例如由于数据传输中断或传感器故障，这将直接影响我们捕捉到的信号灯周期起点时间的准确性。

起点判断的延迟或误差会导致信号灯周期分析的不准确。这种不准确性可能表现为对信号周期长度的误判，或者对绿灯时间窗口的误读等。为了定量地体现这种影响，我们继续采取问题 1 的思路：先利用傅里叶变换求出可能的若干信号灯周期，然后计算出每一个可能的周期的自相关系数，并将其与利用问题 1 中的的数据计算出来的结果相比较。

于是我们发现：当样本车辆充足时，自相关函数（ACF）通常能够清晰地指出一个显著的周期，这个周期的自相关系数明显高于其他值，从而使我们能够较为精确地判定信号灯的真实周期。然而，在样本车辆比例较低的情况下，问题变得复杂。我们发现在这种情况下，尽管计算出多个可能的信号周期，但这些周期的自相关系数差距不大，这导致我们难以从中明确区分出哪一个是信号灯的真正周期。这种现象可能是由于数据不足造成的信号噪声和统计波动，从而使得周期性信号不够显著，或者多个不同的周期伪装成了可能的信号灯周期。

下面，我们可视化地展示了路口 A1 和 B3 可能的周期的自相关系数，更直观地体现这种对比。

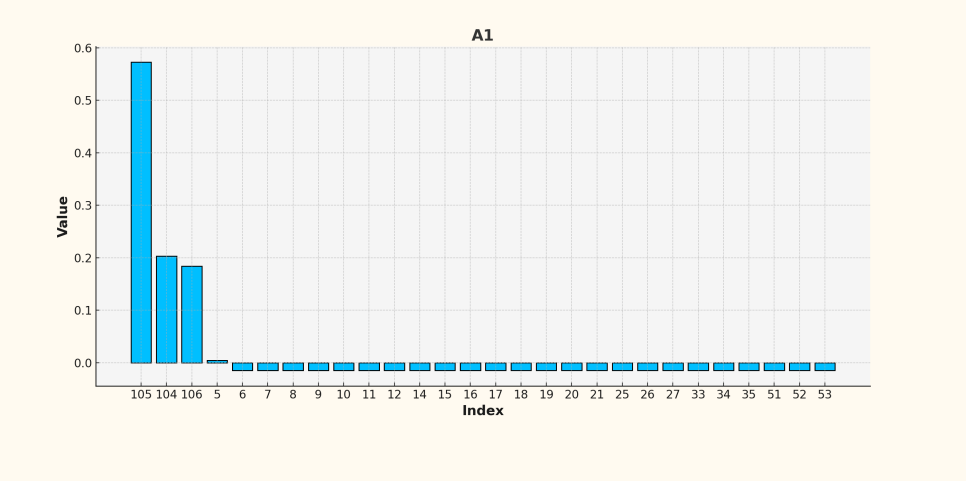

图 4：A1 路口可能的周期时间的自相关系数情况

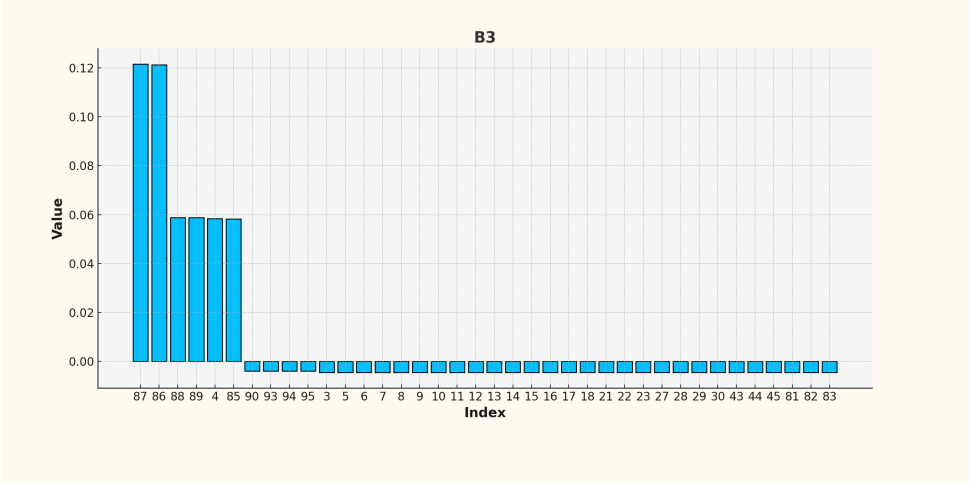

图 5：B3 路口可能的周期时间的自相关系数情况

观察上面两张图，不难发现，A1 路口的周期可以很精确地判断出来：105 秒。而 B3 路口的周期可能为 87 秒或 86 秒，难以准确地判断出来：一方面，这是因为它们的自相关系数太过于接近；另一方面，它们的值也只在 0.12 附近，并没有显著地大。

6.2.2 车流量的大小给信号灯周期判断带来的影响

车流量的大小是影响信号灯周期分析准确性的一个关键因素。在我们的交通模型中，车流量对于信号灯周期的测定具有双重影响：

一方面，当车流量较大时，我们能够收集到更多的数据点标记信号灯周期的开始时刻。这是因为每当信号灯变绿时，较大的车流量意味着更多的车辆开始移动，从而提供了更多用于分析的样本点。随着数据点的增多，我们使用傅里叶变换和自相关函数（ACF）计算出的信号灯周期也会更加精确。

另一方面，车流量的大小也与我们模型的统计假设相匹配。在较大的车流量下，车辆对红灯的反应时长更加接近于均匀分布，这样的数据分布有助于我们准确计算各阶段的持续时间，尤其是红灯时长。而在车流量较小的情况下，由于车辆较少，收集到的数据点减少，可能会出现偶然性较大的情况，这些偶然因素可能导致对红灯时长的估计偏小。

接下来，我们定量地分析路口车流量大小不同对判断相应路口的信号灯周期产生的影响：观察它们可能的周期的自相关系数。根据假设（4），在样本比例相等的情况下，我们分析附件 2 中的路口，观察车流量差异较大的路口的可能的周期的自相关系数分布。发现：车流量较小的路口也面临了它们可能的信号灯周期的自相关系数差距不大的问题。

下面展示 B2 和路口可能的周期的 ACF 值，将其与图 4 进行对比，更直观地体现这种影响。

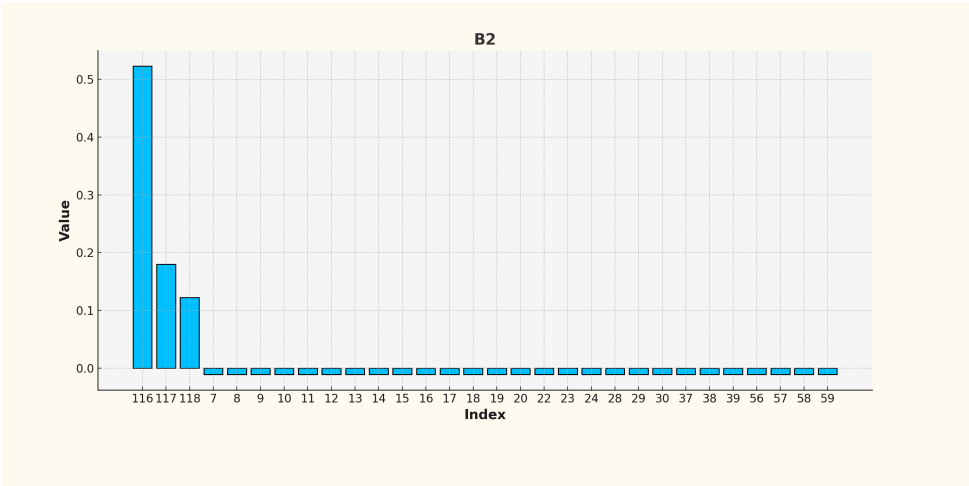

图 6：B2 路口可能的周期时间的 ACF 值情况

对比上面两图，车流量较大的路口 B2 可以较清晰地判断其信号灯周期，而 B3 面临着上述难以判断出真正的信号灯周期的问题。

6.2.3 模型规避了定位误差带来的影响

上文已经阐述清楚，我们计算红灯时间 T_{red} 和绿灯时间 T_{green} 只与车辆的三种状态识别有关。那么结合我们对于三种状态的定义，我们分别来讨论定位误差可能对三种运动状态造成的影响，依此来间接地定性探讨定位误差对预估红灯时长和绿灯时长的影响。通过查阅资料发现，我们在这里只需要考虑较小的定位误差，因为：现代定位技术，如北斗和其他卫星导航系统，通常能够达到较高的精度，误差范围大约在几米之内。在这种情况下，较小的定位误差成为更常见的情形，而较大的误差则相对罕见。

(1) **较小的定位误差不会对刹车减速的判断造成影响：**在我们的研究中，我们将刹车减速的过程定义为车速从初始的均匀速度减至 $0.5m/s$ 以下。这种定义意味着，只要车辆的速度明显下降到接近静止状态，就标志着刹车减速的开始。因此，尽管定位误差可能会影响车辆的精确位置测量，但由于我们关注的主要是速度的显著变化而非具体位置，这种误差不会对我们确定刹车减速起始点产生实质性影响。换句话说，即使存在定位偏差，只要速度的减少趋势清晰可辨，我们就能准确判断刹车的启动时刻。这种方法减少了对精确地理位置数据的依赖，使得速度变化成为评估刹车效果的主要依据，从而提高了研究的实用性和可靠性。

(2) **较小的定位误差不会对静止等待的判断造成影响：**在我们的研究中，处理异常数据的策略专注于修正那些可能由定位误差导致的数据突变。具体来说，我们识别出那些从静止等待状态突然发生改变的数据点，并通过算法修正这些异常，以确保它们不会影响整体数据的准确性。这种修正策略是基于对设备定位技术的理解，包括其潜在的误差范围和可能导致的数据扭曲。因此，尽管定位技术可能存在一定的不精确性，我们的处理方法确保了这些误差不会误导我们对车辆是否处于静止等待状态的判断。

(3) **较小的定位误差不会对启动加速的判断造成影响：**在我们的研究中，起步加速的定义非常明确：当车辆的速度小于 $0.5m/s$ 且加速度大于 $0.5m/s^2$ 时，认为车辆开始起步加速。这一定义允许我们准确地捕捉到车辆从静止状态过渡到运动状态的瞬间。由于这种过渡主要通过速度和加速度的变化来识别，因此，即便存在轻微的定位误差，也不会对起步加速的判断造成影响。

轻微的定位误差，通常不足以改变加速度的计算结果，因为这些计算主要依赖于速度的变化率，而非绝对位置。此外，我们的数据处理算法已经包括了多种机制来补偿可能的测量误差，确保分析结果的准确性。

因此，我们的方法确保了即使在存在轻微定位误差的情况下，也能可靠地识别和记录起步加速的时刻。这种精确的定义和强大的数据处理能力使我们能够进行深入的车辆动态分析，为交通规划和车辆行为研究提供支持。

6.2.4 表格填写

沿用上面的模型以及公式，利用附件中的数据，我们计算得到答案，填入表格当中：

路口	B1	B2	B3	B4	B5
红灯时长（秒）	80	87	74	82	97
绿灯时长（秒）	25	29	13	23	19

表 2

6.3 问题 3 的模型建立与求解

问题 3 要求在考虑信号灯周期有可能发生变化的前提下，尽快检测出这种变化，以及变化后的新周期。基于问题 1 中建立的数学模型，我们对 2 小时的车辆轨迹分割为若

干最小时间单元进行处理，并通过合并单元的方式构建自适应时间窗口，建立判断不同时间窗口内周期是否变化的数学模型。具体通过以下步骤进行建模：

- **步骤一：**确定最小时间单元长度，对分割后的若干最小时间单元进行分别分析，基于问题 1 中建立的数学模型，定量计算最小时间单元间的指标差异。
- **步骤二：**通过合并最小时间单元来构建自适应时间窗口，计算不同时间窗口对应的周期，对某些因数据量过少而无法定量计算指标或计算结果有明显错误的最小时间单元进行局部合并，直至可定量计算出指标。
- **步骤三：**定义周期变化的判断指标，即确定一定的限差。若不同时间窗口间指标差异超过限差，则认为该差异并非合理误差，是由周期变化引起的。
- **步骤四：**根据指标差异及时间窗口中第一辆与最后一辆车的停止再起步时刻，估计周期变化时刻，合并周期相近的时间窗口。

6.3.1 划定最小时间单元

本题数据给出 7200 秒内通过路口的所有车辆数据，但实际车流密度是稀疏的。我们尝试令最小时间单元 $N = 72, 120, 360, 480, 600$ 等值，比较输出结果。

实际上，当 N 过小时，最小时间单元包含的车的数量就很稀疏，会出现前后车辆出现间隔大于一个周期的现象。此时计算时间窗口的周期就会过大，是正常周期的两倍或三倍等。

当 N 过大时，计算周期的时间窗口就会更大，如果红绿灯有周期变化，时间窗口会横跨两个周期。

因此，我们需要在保证每个时间窗口有足够样本车辆的情况下尽可能缩小时间窗口大小。在实际建模过程中，我们选择 $N = 360$ 作为处理本题数据的最小时间单元。

6.3.2 构建自适应时间窗口

在划分过最小时间单元后，我们通过循环，将包含 $vehicle_id$（车辆编号），$redtime$（停车时长）$stoptime$（停车时刻）$starttime$（起步时刻）的数据，按照 $starttime$（起步时刻）与时间窗口的关系进行划分。初始时间窗口就是最小时间单元。

之后，用 Python 的 pandas 库中的 groupby() 函数依据 $starttime$ 分组，并 count() 函数统计每个 $starttime$ 有几条记录，把记录转换成时间序列上的信号函数。

然后，计算时间窗口内红灯时间估计值。这个过程中我们同时用了极大似然估计和矩估计。由于时间窗口内包含的样本车辆可能过少，矩估计通过均值与方差的计算会存在有较大偏差。最后我们选择用极大似然估计求时间窗口内红灯时间估计值。

最后，沿用问题 1 的算法，通过傅里叶变换和自相关函数，计算时间窗口内信号函数的周期。最后，如果周期小于时间窗口内红灯时间的估计值，就合并将该时间窗口与下一最小单元合并为新的时间窗口，重复这个过程直到构建出合适的时间窗口。

6.3.3 定义周期变化的判断指标

构建过过时间窗口后，我们沿用前述模型，计算每个时间窗口内的周期和红灯时间估计值。如果当前时间窗口周期或红灯时间的估计值与前一周期的红灯时间的估计值的间隔大于 10 秒，则判断为发生周期变化。

6.3.4 明确周期变化时刻以及检测条件

在明确发生周期变化时刻后，我们为了尽可能精确时间窗口内的周期变化时间，我们选择周期变化后下一个周期内第一个检测出的车辆起步时间作为最终的周期变化时刻。

理想情况下，若车流量足够大，在周期变化后，我们可以在最小时间单元对应的时间内检测出红绿灯周期变化。但是如果车流量较少，最小时间单元覆盖范围不一定能包含完整周期，就需要数倍最小时间单元内检测出周期变化。所以我们的模型对车流密度有很高的要求，最理想的情况是每当路口变红灯时都有车辆停在此处等红灯，并获取这个数据。

6.3.5 填写表格

根据上述建模思想以及附件中的数据，我们计算得出结果，并填写入表格中。

路口	C1	C2	C3	C4	C5	C6
周期 1 红灯时长（秒）	59	71	83	80	58	78
周期 1 绿灯时长（秒）	29	17	22	25	30	27
周期切换时刻	3600s	无	无	2968s	无	2526s
周期 2 红灯时长（秒）	52			70		62
周期 2 绿灯时长（秒）	36			35		43
周期切换时刻				3705s		3358s
周期 3 红灯时长（秒）				81		80
周期 3 绿灯时长（秒）				24		25

表 3

6.4 问题 4 的模型建立与求解

问题 4 要求基于某路口连续 2 小时内所有方向样本车辆的轨迹数据，求解。基于问题 3 中建立的数学模型，首先识别该路口车辆行驶方向，基于不同行驶方向的样本车辆轨迹数据分析对应的信号灯周期。具体通过以下步骤进行建模：

- **步骤一：**基于经历相同信号灯的车辆具有相近的通过路口前行驶方向与通过路口后的行驶方向，根据两行驶方向对车辆进行 K-Means **聚类分析**，最终得到经历不同信号灯的若干组车辆轨迹数据。
- **步骤二：**基于对向左转、直行、右转的信号灯具有相同周期，**合并对向相同转向方向**的车辆轨迹数据。
- **步骤三：**基于问题 3 中建立的数学模型，分别对单个信号灯进行周期分析，得出最终的若干组周期结果。

6.4.1 K-Means 聚类分析判断车辆行驶方向

K-Means 算法的基本原理是：通过迭代的方式，将数据点划分到最接近的类簇中心点所代表的类簇中，然后根据每个类簇内的所有点重新计算该类簇的中心点（取平均值），再不断重复此过程，直至类簇中心点的变化很小或达到指定的迭代次数。

本例中，以车辆驶进路口的速度矢量与 X 正半轴的夹角作为 X1，以车辆驶离路口的速度矢量与 X 轴正半轴夹角作为 X2。以不同车辆的 X1 作为横轴，X2 作为纵轴，绘制平面图如下（共有 3222 个点，颜色深浅可表示点数多寡）：

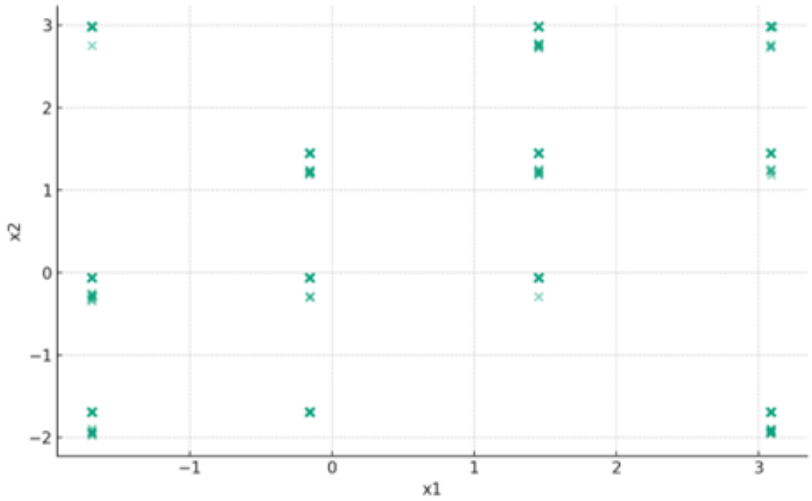

图 6：行驶方向特征图

从平面图中可以获知车辆驶进路口的方向共有四个，车辆驶离路口的方向也共有四个，不同行驶路线的车辆共有十二种，即车辆从四个方向进入路口时都可选择左转、直行、右转。并且，相同行驶路线的车辆组内距离较小，不同行驶路线的车辆组间距离较大，可由 K-Means 聚类算法得到较好的分类结果。目标种类为十二类，则设置初值 K=12。

聚类结果如下：

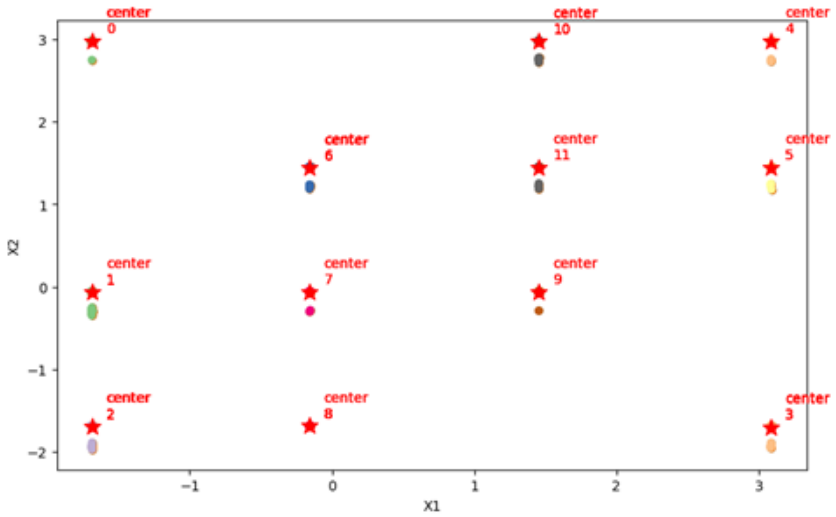

图 7：行驶方向 K-Means 聚类图

6.4.2 合并具有对向相同转向方向的车辆估计数据

我们注意到在现实生活中，正常情况下，对向相同转弯方向的信号灯具有相同周期，如图所示，当前方向左转信号灯与对向左转信号灯同步变化。

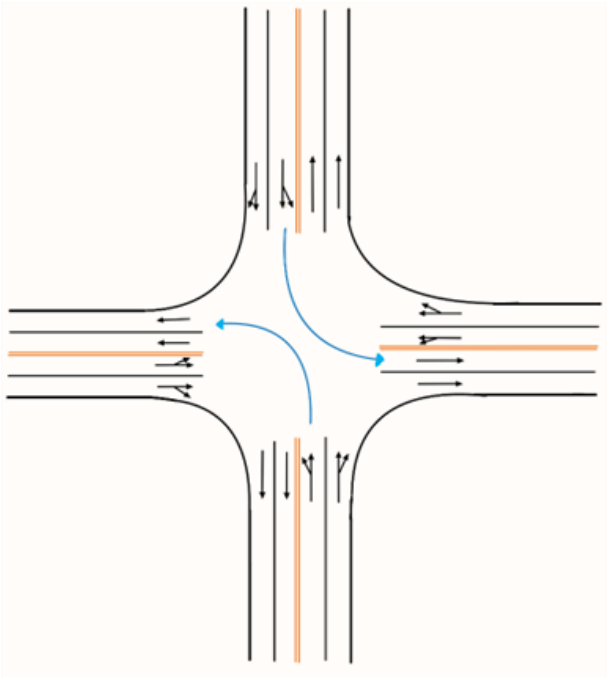

图 8：路口车辆方向

则不同行驶路线的十二种车辆可两两合并，分别对应一个对向的信号灯周期。具体信号灯编号为上下路口右转 1、左转 2、直行 3，左右路口左转 4、直行 5、右转 6，进

一步将车辆的轨迹数据划分为六类。相对于分为十二类而言，分为六类有助于增加每一类的数据量，提高信号灯周期可信度，并且仅基于符合生活实际的假设。

6.4.3 分别对单个信号灯进行周期分析

基于前述模型，利用附件 4 中的数据，计算出该路口每个方向的红灯时长以及信号灯周期长度，最终将结果填入到表格中：

信号灯	1	2	3	4	5	6
周期 1 红灯时长（秒）	94	118	98	90	106	102
周期 1 绿灯时长（秒）	48	24	44	52	36	40
周期切换时刻	无	2040s	无	680s	无	1923s
周期 2 红灯时长（秒）		96		117		94
周期 2 绿灯时长（秒）		46		25		48
周期切换时刻		2461s		5362s		
周期 3 红灯时长（秒）		120		99		
周期 3 绿灯时长（秒）		22		43		
周期切换时刻				6072s		
周期 4 红灯时长（秒）				111		
周期 4 绿灯时长（秒）				31		

表 4

7 模型的改进与推广

7.1 模型的优点与创新

7.1.1 傅里叶变换与自相关函数结合

单独利用傅里叶变换求解周期，在本例数据量较小的情况下，如果仅以最大幅值对应的频率作为周期倒数的唯一值，大概率会得到错误的周期；单独利用自相关函数

（ACF）求解周期，由于周期的不确定性，无法断定信号灯周期的可能值，而是依赖经验判断，且需要计算较多不同周期的自相关系数。模型先利用傅里叶变换求解可能的周期（取前 15 个幅值对应的周期时能兼顾效率与准确度），再计算这些周期的自相关系数，大量减少了计算量。

7.1.2 最小时间单元与自适应时间窗口结合

由于样本车辆的数量不确定性，一个固定的周期变化检测时间窗口很难兼顾准确性与普适性。如果时间窗口设置过大，则在数据密度较大时会造成数据浪费，不能及时检测周期变化；如果时间窗口设置过小，则在数据密度较小时会造成周期识别有误，模型失去可靠性。基于以上状况，我们提出了最小时间单元与自适应时间窗口结合的思路。一方面，最小时间单元设置为较小值可以保证数据密度较大时，能够尽快检测周期变化；另一方面，当数据密度较小时，最小时间单元内无法正常识别周期，则与下一最小时间单元合并为自适应时间窗口，直至可以正常识别周期，兼顾了效率与准确。

7.2 模型的局限性

7.2.1 假设存在理想性

假设所有对红灯做出反应的车辆的反应时间均匀分布是一个理想化的假设，实际情况中，司机的反应时间可能受多种因素影响，如天气、司机疲劳度、车辆类型等，这些因素的多样性和复杂性使得实际分布可能远离均匀分布。

7.2.2 计算的复杂度较大

使用傅里叶变换和自相关系数（ACF）的方法虽然能够较好地估计周期，并且尽可能减少了计算量，但在计算上还是较为复杂和耗时，特别是在实时或近实时的交通管理应用中，这可能成为一个限制。

7.3 模型的改进

为了同时改善模型假设的理想性和降低计算复杂性，我们可以采取一种综合的方法来提高模型的现实适用性和效率。首先，通过引入更多与驾驶行为、天气条件和车辆类型相关的变量，我们可以更准确地模拟司机的反应时间和交通流动。这种方法通过使用混合分布模型（如正态分布与均匀分布的混合）或非参数统计模型来替代单一的均匀分布假设，能更好地捕捉到数据的真实分布。

同时，为了降低计算复杂性，我们可以探索使用简化或近似的数学方法，如采用快速傅里叶变换（FFT）代替传统的复杂傅里叶分析，或者实施基于滑动窗口的周期检测算法。这些方法既能减少对计算资源的需求，也能提高处理速度。

未来，我们可以将样本比例、车流量、定位误差等会对模型精度产生影响的因素建立定量模型，评估影响因素的具体影响，进而确定提升模型精度应该关注的诸方面，使模型更具有普适性与可推广性。

参考文献

[1] 李建春, 陶崇瑾, 陈立新. 基于车路协同的红绿灯配时优化控制策略 [J]. 数字技术与应用,2023,41(12):43-45.DOI:10.19695/j.cnki.cn12-1369.2023.12.13.

[2] 赵谦, 马文越, 郑超, 等. 基于单路口车流量统计的交通灯配时系统研究 [J]. 电子测量技术,2023,46(08):82-91.DOI:10.19651/j.cnki.emt.2211293.

[3] 李奕衡, 刘羽飞, 王登, 等. 红绿灯周期时长的挖掘方法、电子设备及计算机程序产品:CN202111580642.X[P].20211222.

附录

Python

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as  plt
import math

'''
判断异常点
'''
file ='C1'#选择文件
data=pd.read_csv(file+'.csv')# 读文件
data_id=data.copy()
data_id=data_id.groupby("vehicle_id")# 以车辆id分类
print("总车数： ",len(data_id))
totaldata=pd.DataFrame()
totaldata.insert(loc=0,column='time',value=0)
totaldata.insert(loc=1,column='vehicle_id',value=0)
totaldata.insert(loc=2,column='x',value=0)
totaldata.insert(loc=3,column='y',value=0)
totaldata.insert(loc=4,column='velocity',value=0)
totaldata.insert(loc=5,column='acceleration',value=0)
for i,d in enumerate(data_id):
    dat=d[1].to_numpy()# d[0]是id，d[1]是DataFrame格式数据
    # 包含最后一秒的车的数据剔除
    #print(dat[:,0])
    if 7199 in dat[:,0]:
        break
    # time vehicle_id x坐标，y坐标
    j=0
    velocity=np.zeros(len(dat))# 速度
    acceleration=np.zeros(len(dat))# 加速度
    # 这里是插入到末尾，第5列和第6列
```

```
    dat=np.insert(dat,4,velocity,axis=1)#axis=1是插入列
    dat=np.insert(dat,5,acceleration,axis=1)
    for id,pt in enumerate(dat):
        if id+1==len(dat):
            pt[4]=dat[id-1][4]
            pt[5]=dat[id-1][5]
            break
        # 计算速度
        pt[4]=math.sqrt((pt[2]-dat[id+1][2])*(pt[2]-dat[id+1][2])
                        +(pt[3]-dat[id+1][3])*(pt[3]-dat[id+1][3]))
    for id,pt in enumerate(dat):
        if id+1==len(dat):
            pt[4]=dat[id-1][4]
            pt[5]=dat[id-1][5] #最后的速度与加速度和倒数第二个一样
            break
        # 计算加速度
        pt[5]=dat[id+1][4]-pt[4]
    ''' 纠正错误数据(v,a应该为0的改为0)'''
    for id,pt in enumerate(dat):#如果前后两个的速度小于0.5
        if dat[(id+1)%len(dat)][4]<0.5 and dat[id-1][4]<0.5 and id!=len(dat) and id!=0:
            if pt[4]>0.5:# 但是该时刻速度不为0
                print('错误车'+str(d[0]))
                print('错误时间'+str(int(pt[0])))
                pt[4]=0
                pt[5]=0
                dat[id-1][5]=0# 前一秒的加速度也该变为0
                totaldata.loc[len(totaldata)-1]=dat[id-1]
        totaldata.loc[len(totaldata)]=pt
totaldata.to_csv(file+'_va.csv')
'''
寻找因为看到红灯减速或停车的点
'''
def isStop(id,dat):
    #如果索引越界，返回False
```

```
    if id>=len(dat):
        return False
    i=id
    a=0
    #如果前后速度为0，直接返回True
    if dat[id-1][4]<0.5 and dat[id][4]<0.5 and dat[(id+1)%len(dat)][4]<0.5:
        return True
    # 记录减速过程
    while dat[i%len(dat)][5]<=0:# 停车点（v、a为0）也算
        a+=abs(dat[i%len(dat)][5])
        i=i+1
        if i==len(dat):
            break
    #如果速度减到0.5以内，说明减速时刻看到红灯了
    if dat[i%len(dat)][4]<0.5:
        return True
    return False
data=pd.read_csv(file+'_va.csv')# 读文件
data_id=data.loc[:,'time':'acceleration'].copy()
data_id=data_id.groupby("vehicle_id")# 以车辆id分类
times=pd.DataFrame()
times.insert(loc=0,column='vehicle_id',value=0)
times.insert(loc=1,column='redtime',value=0)
times.insert(loc=2,column='stoptime',value=0)
times.insert(loc=3,column='starttime',value=0)
#起步次数(多次起步我们认为一次红灯没通过)
times.insert(loc=4,column='startNum',value=0)
# 遍历每一辆车
for i,d in enumerate(data_id):
    startNum=0# 起步次数
    # time vehicle_id x，y，v,a
    dat=d[1].to_numpy()# d[0]是index，d[1]是DataFrame格式数据
    stoptime=0
    starttime=0
```

```
100
101     for id,pt in enumerate(dat):
102         if isStop(id,dat):
103             #如果目前速度都为0，但是此刻加速度大于0.5，认为开始启动
104             if dat[id][4]<0.5 and dat[id][5]>0.5:
105                 starttime= dat[id][0]
106                 if len(times)!=0:
107                     # 覆盖原有记录
108                     if times.loc[len(times)-1]['vehicle_id']==dat[id][1]:
109                         startNum=startNum+1
110                         stoptime=times.loc[len(times)-1]['stoptime']
111                         times.loc[len(times)-1]=[pt[1],
112                         int(starttime-stoptime),stoptime,starttime,startNum]
113                         continue
114                 times.loc[len(times)]=[pt[1], int(starttime-stoptime),stoptime,starttime,startNum]
115             if isStop(id-1,dat):# 如果上一秒是启动点，继续
116                 continue
117             stoptime=dat[id][0]# 记录停车时间
118 times=times.to_numpy()
119 #接近点合并
120 i=0
121 while i<len(times):
122     j=0
123     while abs(times[i][3]- times[(i+j+1)%len(times)][3])<20:#小于20合并
124         times[i+j+1][3]=0
125         j+=1
126     i+=1
127 tmp=times.copy()
128 ids=[]
129 for i,t in enumerate(tmp):
130     if t[3]==0:
131       ids.append(int(i))
132 times=np.delete(times,ids,axis=0)
133 datatime={'vehicle_id':times[:,0], 'redtime':times[:,1], 'stoptime':times[:,2],
```

```
            'starttime':times[:,3], 'startNum':times[:,4]}

times=pd.DataFrame(datatime)
times.to_csv(file+"_time.csv")
data=pd.read_csv(file+'_time.csv')
data_id=data.loc[:,'vehicle_id':'starttime'].copy()
data_id=data_id.groupby('starttime').count()
time_car=np.zeros(7200)#记录索引对应时刻有无车辆
print('有效统计车数：',len(data_id))
#有记录就是1
for i in data_id.index:
    time_car[int(i)]=int(1)
data={'count':time_car[int(data_id.index[0]):]}
dftime_car=pd.DataFrame(data)
dftime_car.to_csv(file+'_timecar.csv')#保存文件
```

Python

```
from __future__ import division
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as signal
from scipy.fftpack import fft, fftfreq
from statsmodels.tsa.stattools import acf
import math
'''
某一时刻有多少辆车启动
间隔N秒进行合并，如果N秒算出正确周期，保留，否则合并下一个N秒继续计算
'''
''' 计算周期'''
def period_find(record):
    record = record.to_numpy()
    record_by_first = record.copy()[:,0]   # 获取第一列数据
```

```
    # 对数据进行傅里叶变换
    fft_series = fft(record_by_first)
    power = np.abs(fft_series)  # 计算每个频率的功率
    sample_freq = fftfreq(fft_series.size)  # 计算对应的频率
    # 选择正频率部分
    pos_mask = np.where(sample_freq > 0)
    freqs = sample_freq[pos_mask]
    powers = power[pos_mask]

    top_k_seasons = 20 # 选择功率最大的前20个频率
    if len(powers)<20:
        top_k_seasons=len(powers)-1
    # 获取这些频率的索引
    top_k_idxs = np.argpartition(powers, -top_k_seasons)[-top_k_seasons:]
    # 计算对应的周期，提取浮点数附近的整数
    fft_periods1 = (1 / freqs[top_k_idxs]).astype(int)
    fft_periods2 = (1 / freqs[top_k_idxs] + 1).astype(int)
    fft_periods3 = (1 / freqs[top_k_idxs] - 1).astype(int)
    fft_periods = np.append(fft_periods1, fft_periods2)
    fft_periods = np.append(fft_periods, fft_periods3)
    max = 0
    max_index = 0
    # 寻找具有最大自相关值的周期
    for lag in fft_periods:
        acf_score = acf(record_by_first, nlags=lag)[-1]
        # 计算给定lag的自相关值
        if acf_score > max:
            max = acf_score
            max_index = lag
    print(f"max fft acf: {max}")
    print(f"周期为: {max_index}")
    return max_index

if os.path.exists('tmp')==False:
```

```
52      os.mkdir('tmp')
53
54  file ='C1'
55  N=360# N最好可以被7200整除
56  data=pd.read_csv(file+'_time.csv')
57
58  data_id=data.loc[:,'vehicle_id':'startNum'].copy()
59  result=pd.DataFrame()
60  result.insert(loc=0,column='cross-part',value=0)
61  result.insert(loc=1,column='period',value=0)
62  result.insert(loc=2,column='recordNum',value=0)#记录数
63  result.insert(loc=3,column='firsttime',value=0)
64  result.insert(loc=4,column='endtime',value=0)
65  result.insert(loc=5,column='max-redtime',value=0)#极大似然估计
66  result.insert(loc=6,column='gmm-redtime',value=0)#矩估计
67
68  recordNum=0
69  isRight=False
70  j=0
71  redtime=0
72  while True:
73      dftime_car=pd.DataFrame()
74      dftime_car.insert(loc=0,column='count',value=0)#该时间点几辆车出发
75      dftime_car.insert(loc=1,column='part',value=0)#是第几段
76      dftime_car.insert(loc=2,column='recordNum',value=0)#该段几记录数
77      dftime_car.insert(loc=3,column='firsttime',value=0)#该段起始时间点
78      dftime_car.insert(loc=4,column='endtime',value=0)#该段终止时间点
79      k=0
80      if j+k+1>int(7200/N):
81          break
82      isRight=False
83      while isRight==False:
84          if N*(j+1+k)>7200:
85              break
```

```
86          #vehicle_id redtime stoptime starttime
87          partdata1=data_id[data_id['starttime']<N*(j+1+k)].copy()
88          partdata1=partdata1[partdata1['starttime']>=N*j].copy()
89          #index:starttime
90          #vehicle_id redtime stoptime
91          partdata=partdata1.groupby('starttime').count()
92          time_car=np.zeros(7200)
93          print('第'+str(len(result))+'时间段记录数:',len(partdata))
94          recordNum=(len(partdata))#记录数
95          #没有记录，则退出
96          if recordNum==0:
97              j=j+1
98              break
99          #time_car第i个索引的值就是i时刻有几辆车(改成1)
100         for i in partdata.index:
101             time_car[int(i)]=int(partdata[i:i]['vehicle_id'])
102         '''绘制频率图代码'''
103         # for i,yi in enumerate(time_car[int(partdata.index[0]):]):
104         #     if yi !=0:
105         #         plt.scatter(i, 1,s=10,c='b') # 绘制散点图
106         #         plt.vlines(i,1,0)
107         #         plt.axhline(0)#添加横线
108         # plt.show()
109         dftime_car.drop(dftime_car.index, inplace=True,axis=0)#清空数据
110         #从partdata.index[0]，即该时间段第一个有记录的车辆开始
111         for i in time_car[int(partdata.index[0]):N*(j+1+k)]:
112             dftime_car.loc[len(dftime_car)]=[i,str(j),len(partdata),
113                 int(partdata.index[0]),int(partdata.index[-1])]
114         dftime_car['recordNum']=len(partdata)
115         record=dftime_car.copy()
116         record=record.groupby('part')
117         k+=1
118         for record in record:
119             max_index=period_find(record[1])
```

```
120             #周期大于红灯认为是正常的
121             newpartdata1=partdata1[partdata1['startNum']==0]
122             redtime1=newpartdata1['redtime'].max()
123             if max_index>redtime1:
124                 isRight=True
125         j+=k
126         if recordNum!=0:
127             newpartdata1=partdata1[partdata1['startNum']==0]
128             # 极大似然估计，等于一次通过路口的车的最大等红灯数
129             redtime1=newpartdata1['redtime'].max()
130             # 矩估计，均值加根号3倍标准差
131             ave=newpartdata1['redtime'].mean()
132             var=newpartdata1['redtime'].var()
133             redtime2=ave+math.sqrt(3*var)
134             result.loc[len(result)]=[file+'-'+str(len(result)),max_index,recordNum,
135                 partdata.index[0],partdata.index[-1],redtime1,redtime2]
136         print(result)
137 result.to_csv('tmp/'+file+'_period-N'+str(N)+'.csv')
```

Python

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import math
5
6  '''
7  寻找因为看到红灯减速或停车的点
8  记录起步次数
9  '''
10 def isStop(id,dat):
11     #如果索引越界，返回False
12     if id>=len(dat):
13         return False
14     i=id
```

```
    a=0
    #如果前后速度为0，直接返回True
    if dat[id-1][4]<0.5 and dat[id][4]<0.5 and dat[(id+1)%len(dat)][4]<0.5:
        return True
    # 记录减速过程
    while dat[i%len(dat)][5]<=0:# 停车点（v、a为0）也算
        a+=abs(dat[i%len(dat)][5])
        i=i+1
        if i==len(dat):
            break
    #如果速度减到0.5以内，说明减速时刻看到红灯了
    if dat[i%len(dat)][4]<0.5:
        return True

    return False



''' 返回a点到b点的欧氏距离平方'''
def distance(a, b):
    d = (a[1] - b[1]) * (a[1] - b[1]) + (a[2] - b[2]) * (a[2] - b[2])
    return d


file = 'D'
data = pd.read_csv(file + '.csv')  # 读文件
print(data.keys())
data_id = data.copy()
data_id=data_id.groupby("vehicle_id")# 以车辆id分类
print("总车数：",len(data_id))

totaldata=pd.DataFrame()
totaldata.insert(loc=0,column='vehicle_id',value=0)
totaldata.insert(loc=1,column='start_azimuth',value=0)#起始速度方向与x轴夹角
```

```
totaldata.insert(loc=2,column='end_azimuth',value=0)# 终点速度方向与x轴夹角

index = 0
for i, d in enumerate(data_id):
    dat = d[1].to_numpy() # d[0]是id，d[1]是DataFrame格式数据
    # time vehicle_id x坐标，y坐标
    # 包含最后一秒的车的数据剔除
    # print(dat[:,0])
    if 7199 in dat[:, 0]:
        break
    for id, pt in enumerate(dat):
        if id == 0:
            totaldata.loc[index,'vehicle_id'] = pt[1]
            #计算起始方向与x轴正方向夹角的角度，结果为正表示从 X
                轴逆时针旋转的角度，结果为负表示从 X 轴顺时针旋转的角度
            start_azimuth = math.atan2(dat[id + 1][3] - pt[3], dat[id + 1][2] - pt[2])
            totaldata.loc[index,'start_azimuth'] = start_azimuth
        if id + 1 == len(dat):
            end_azimuth = math.atan2(pt[3] - dat[id - 1][3], pt[2] - dat[id - 1][2])
            totaldata.loc[index,'end_azimuth'] = end_azimuth
            index = index + 1
            break

# totaldata.to_csv("D_azimuth.csv")
totaldata=pd.read_csv("D_azimuth.csv")

'''
对于该例子来讲，D中路口为十字路口，
每个路口都可以直行，左转，右转，共有12种转弯方向
故初值设置为12
'''
K = 12 # 选定K个聚类中心
data = totaldata.to_numpy()
label = np.zeros(len(data))  # 标签，标志每个点属于哪个类
```

```
82  center = data[0:K].copy()  # 初始聚类中心，这里设成是前K个点
83  # for i in range(K):
84  #     center[i][0] = 'Z' + str(0) + '-' + str(i)
85
86  # 前K个点打标签
87  for i in range(K):
88      label[i] = i
89
90  ischange = True
91
92  index = 0  # 记录while循环次数
93  while (ischange == True):
94      # 根据到每个聚类中心距离分成K类
95      for i, pt in enumerate(data):
96          minid = 0 # 距离哪个聚类中心最近，初始到0
97          mindis = distance(center[0], pt)  # 初始为到0的距离
98          for j in range(K): # 计算到每个聚类中心的距离
99              if distance(center[j], pt) < mindis:
100                 mindis = distance(center[j], pt)
101                 minid = j
102         label[i] = minid # 记录第i个点属于哪一类
103
104     # 根据新分成的K类建立新的聚类中心
105     newcenter = center.copy()
106     for ct in newcenter:  # 归0
107         ct[1] = 0
108         ct[2] = 0
109     Nums = np.zeros(K) # 记录每个类有多少数
110     for j in range(K):  # 遍历每一类
111         for i, pt in enumerate(data): # 遍历每一个点
112             if label[i] == j:
113                 Nums[j] = Nums[j] + 1
114                 newcenter[j][1] += pt[1]
115                 newcenter[j][2] += pt[2]
```

```
116     print('各类数目', Nums)
117     for j in range(K):
118         newcenter[j][1] /= Nums[j]
119         newcenter[j][2] /= Nums[j]
120     '''每次迭代都画图，可以看出每次迭代聚类中心的变化'''
121     # name = data[:, 0]
122     # X1 = data[:, 1]
123     # X2 = data[:, 2]
124     # plt.scatter(X1, # x轴数据
125     #             X2, # y轴数据
126     #             s=100, # 点大小
127     #             marker='.', )
128     # for i, pt in enumerate(center):
129     #     plt.scatter(pt[1], pt[2], s=150, c='red', marker='.', )
130     #     plt.annotate('center' + '\n' + str(i), c='red', xy=(pt[1], pt[2]), xytext=(pt[1]
        + 0.1, pt[2] + 0.1))
131
132     # # 标注点名
133     # # for i in range(len(name)):
134     # #     plt.annotate(str(name[i]) + '\n' + str(int(label[i])), xy=(X1[i], X2[i]),
        xytext=(X1[i] + 0.1, X2[i] + 0.1))
135     # plt.ylabel('X2')
136     # plt.xlabel('X1')
137     # plt.show()
138     ischange = False
139     for j in range(K): #
        因为精度原因，不能简单用等号判断迭代前后中心位置变没变，这里设定一个极小的容差
140         if (newcenter[j][1] - center[j][1]) * (newcenter[j][1] - center[j][1]) > 0.000001:
141             ischange = True # 超过容差，认为前后中心位置改变
142             break
143         if (newcenter[j][2] - center[j][2]) * (newcenter[j][2] - center[j][2]) > 0.000001:
144             ischange = True
145             break
146
```

```
147         print('新聚类中心：', newcenter)
148         if ischange == True:
149             center = newcenter.copy()
150             index = index + 1
151
152 # 画图，0，1，2，3等等是某一类
153 name = data[:, 0]
154 X1 = data[:, 1]
155 X2 = data[:, 2]
156 plt.scatter(X1, # x轴数据
157                 X2, # y轴数据
158                 s=100, # 点大小
159                 marker='.',
160                 c=label, cmap='Accent')
161 for i, pt in enumerate(center):
162     plt.scatter(pt[1], pt[2], s=150, c='red', marker='*', )
163     plt.annotate('center' + '\n' + str(i), c='red', xy=(pt[1], pt[2]), xytext=(pt[1] + 0.1,
            pt[2] + 0.1))
164
165 # 标注点名
166 for i in range(len(name)):
167     plt.annotate(str(name[i]) + '\n' + str(int(label[i])), xy=(X1[i], X2[i]), xytext=(X1[i]
            + 0.1, X2[i] + 0.1))
168 plt.ylabel('X2')
169 plt.xlabel('X1')
170 plt.show()
171
172 '''
173 对向的红绿灯时间相同，可归于同一类型
174 0-9
175 1-10
176 2-11
177 3-6
178 4-7
```

```
5-8
'''
totaldata.insert(loc=3,column='label',value=0)# 终点速度方向与x轴夹角
for i,dat in enumerate(totaldata.to_numpy()):
    if label[i]==9:
        label[i]=0
    if label[i]==10:
        label[i]=1
    if label[i]==11:
        label[i]=2
    if label[i]==6:
        label[i]=3
    if label[i]==7:
        label[i]=4
    if label[i]==8:
        label[i]=5
    totaldata.loc[i,'label']=label[i]
totaldata.loc[:,['vehicle_id','label']].to_csv('D_label.csv')
```