基于机器学习的信号灯周期估计

摘要

对于某电子地图服务商来说，城市路网中所有交通信号灯红绿灯周期难以直接获取。本文基于无监督学习的方法，根据该公司大量客户的行车轨迹数据，通过 DBSCAN 聚类、K 聚类估计交通信号灯的周期。并用 ADF 检验和 pettitt 检验对模型估计精度进行了优化。

针对问题一，首先，本文根据行车轨迹数据绘制车辆轨迹图，其次，根据速度公式计算并绘制出每辆车的速度-时间图标，以绿灯开始时间作为时间戳计算信号灯周期长度。并使用 DBSCAN 聚类分析对车辆状态按簇归类，消除噪声车辆。得到初步的周期图，最后对于初步周期图使用分箱法得到准确的周期图。

针对问题二，要求讨论样本车辆比例、车流量、定位误差等因素对问题一模型估计的影响，本文同时考虑了转向减速的影响。采用层次分析法分析得到各影响因素对模型的影响权重，而后基于问题一建立的数学模型，通过起始波的运动模型以及加速度公式消除或减弱各个影响因素对信号周期估计的影响，从而得到精度更高的信号灯估计周期。

针对问题三，要求判断信号灯周期是否变化并指出变化时间与条件。首先，基于问题二建立的数学模型得到信号灯时间周期，通过 ADF 检验判断信号灯周期是否发生变化，而后对于非平稳时间序列，通过 pettitt 检验测出信号灯周期的突变点（可能有多个突变点）并求出新旧周期参数。最后，通过分析信号灯周期变化的规律来确定周期变化所需的时间与条件。

针对问题四, 要求识别某路口连续 2 小时内所有方向样本车辆的轨迹数据。通过 K-means 聚类与肘部法则规划出该路口 12 种行车方向。通过分别将东西、南北方向上的直行、左转、右转这些特征值分类并两两组合成 6 种规划，再分析在流动的时间轴上 6 种规划的规律，发现该路口直行与右转为同一车道，会受直行交通信号灯地控制，假设将右转与直行都视作直行，将 6 种规划降维至 4 种，构成交通信号的显示方式，绘图发现交通信号灯的变化有明显的周期性。而后通过问题二所得数学模型，分析得出该路口信号灯的周期。最后基于问题三中 ADF 检验方法检验该路口信号灯周期是否发生变化并用 pettitt 检验识别周期突变点。最终得出该路口信号灯的整体周期与四个阶段的子周期。

关键字：行车轨迹　DBSCAN 聚类　层次分析法　ADF 检验　pettitt 检验　K-means 聚类

目录

一、 问题重述

1.1 问题背景

某电子地图服务商为了提升导航服务质量，计划使用客户行车轨迹估计交叉路口信号周期。然而，由于许多信号灯未接入网络，无法在线存档和实时更新，则不能直接从交通管理部门获取数据。而大范围的人工调查实际的信号周期长度需要大量的人力金钱以及时间成本。因此该公司计划使用大量客户的行车轨迹数据用以估计交通信号灯周期。

1.2 问题重述

1.2.1 问题一

基于所有车辆的行车轨迹数据，建立一个数学模型来估计每个路口信号灯的红绿周期。对于给定的轨迹数据（附件 1），需要求出每个路口相应方向的信号灯周期，并将结果填入表 1。

1.2.2 问题二

考虑到只能获取部分样本车辆的行车轨迹，并且这些数据可能受到定位误差的影响，基于任务一中的模型改进，添加讨论这些因素（样本车辆比例、车流量、定位误差）对周期估计精度的影响。基于附件 2 中的样本车辆轨迹数据，尝试估计这些路口的信号灯周期，并将结果填入表 2。

1.2.3 问题三

针对信号灯周期可能发生变化的情况，设计一个策略来尽快检测这种变化，并估计新的周期。利用附件 3 中的数据，判断各路口的信号灯周期是否发生变化，求出周期切换的时刻及新旧周期参数，并填入表 3。同时，指出识别周期变化所需的时间和条件。

1.2.4 问题四

基于以上任务建立的模型和某路口连续 2 小时内所有方向样本车辆的轨迹数据，识别出该路口信号灯的周期。

二、 问题分析

2.1 问题一分析

本题需要根据每个附件中的所有车辆路线轨迹来分析对应路口的信号灯周期。首先要从轨迹的变化中得到车辆的速度变化，根据速度时间点来推断信号灯变化的时间点，进而得到周期。由于车辆的行驶状态多样，存在较多噪声数据，因此首先使用 DBSCAN 聚类分析对车辆状态按簇归类，消除噪声车辆。接着基于处理后的数据画出初步的周期图，最后对于初步周期图使用分箱法得到准确的周期图

2.2 问题二分析

题目要求讨论样本车辆比例、车流量、定位误差等因素对问题一模型估计的影响，本文认为除题给误差外，车辆的转向减速也会对交通信号灯周期产生影响，将其考虑在内，采用层次分析法分析得到样本车辆比例、车流量、定位误差和车辆的转向减速对模型的影响权重，并通过每次绿灯开始时间这一时间戳的确定来跨越信号周期，从而估计信号周期的长度并消除或减弱各个影响因素对信号周期估计的影响。

2.3 问题三分析

题目要求判断信号灯周期是否变化，得出信号灯周期变化时刻以及新旧周期参数，同时要求指出变化时间与条件。首先，基于问题二建立的数学模型得到信号灯时间周期，通过 ADF 检验判断信号灯周期是否发生变化，若发生变化，其是非平稳时间序列，若不发生变化，判断其为平稳时间序列。而后对于非平稳时间序列，通过 pettitt 检验测出信号灯周期的每一个突变点，并求出新旧周期参数。最后将所得数据整理汇总填表，通过分析信号灯周期变化的规律来确定周期变化所需的时间与条件。

2.4 问题四分析

已知某路口连续 2 小时内所有方向样本车辆的轨迹数据。通过 K-means 聚类与肘部法则规划出该路口 12 种行车方向。通过分别将东西、南北方向上的直行、左转、右转这些特征值分类并两两组合成 6 种规划，再分析在流动的时间轴上 6 种规划的规律，发现该路口直行与右转为同一车道，会受直行交通信号灯地控制，假设将右转与直行都视作直行，将 6 种规划降维至 4 种，构成交通信号的显示方式，绘图发现交通信号灯的变化有明显的周期性。而后通过问题二所得数学模型，分析得出该路口信号灯的周期。最后用问题三中 pettitt 突变检验方法检验该路口信号灯周期是否发生变化。最终识别出该路口信号灯的整体周期与四个阶段的子周期。

三、 模型假设

所有司机都遵守交通规则。所有司机在看到绿灯亮时立刻启动所有司机看到红灯亮时立刻制动。。定位误差是随机的，不会系统性地偏向某一方向

四、 符号说明

五、 问题一模型的建立和求解

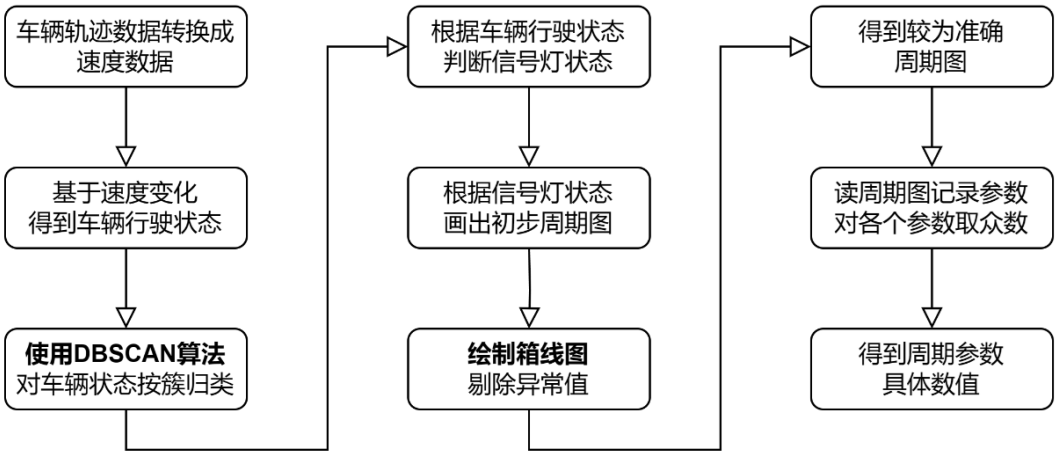

图 **1**　问题一流程图

5.1 模型建立

对于附件表单中给出的数据我们较难了解车辆轨迹的情况。为了便于我们观察与分析，我们首先通过画出车辆轨迹图直观地看出车辆运行的情况。通过车辆轨迹图，可知每个表格中的所有车辆轨迹都相同，也即只有一个信号周期，再对每个车辆的速度进行计算，先根据以下公式，将车辆轨迹进行处理初步得到速度数据

$$v_{i,t}=\frac{\sqrt{\left(x_{i,t}-x_{i,t-1}\right)^2+\left(y_{i,t}-y_{i,t-1}\right)^2}}{T_{track}}$$

接着由公式计算处理，得速度-时间图像（如图）

$$v_{i,t}=\begin{cases}0 & v_{i,t}<\partial\\ v_{i,t} & v_{i,t}\geq\partial\end{cases}$$

对于所有车辆的速度-时间图分析得到：当车辆的坐标不变时，车辆为停止的，也即信号灯在此时刻内为红灯；当车辆的坐标开始变化时，信号灯由红灯变为绿灯。易得，

符号	说明	单位
T_{signal}	信号灯的周期	s
T_{red}	信号灯的红灯周期	s
T_{green}	信号灯的绿灯周期	s
T_{track}	车辆行车轨迹数据的采样间隔	s
t_i	编号为 i 的车辆的行车轨迹数据的时间点	s
$x_{i,t}$	编号为 i 的车辆在时间点时的 X 坐标,	-
$y_{i,t}$	编号为 i 的车辆在时间点时的 Y 坐标	-
y_i	编号为 i 的车辆在时间点时的 Y 坐标	-
$v_{i,t}$	编号为 i 的个车辆在时间点 t 时的速度)	$m/s/$
∂	极小的速度阈值	m/s
$r_{i,t}$	编号为 i 的车辆在时间点 t 是否停止,	-
U_i	编号为 i 的车辆在整体行车速度的数学统计,	-
$t_{0,i}$	编号为 i 的车辆坐标首次出现时间点	t
Δl	最后时间	t
$g_{t',i}$	编号为 i 的车辆开始离开信号交叉口时的时间	t
u_v	车队消散速度	m/s
h	饱和车程	-
u	车辆正常行驶速度	m/s
t'''_i	最后一点停车时速度不为 0 的时间	s
$v'''_{i,t}$	最后一点停车时速度不为 0 的速度	m/s
$x'''_{i,t}$	最后一点停车时速度不为 0 的横坐标	-
$y'''_{i,t}$	最后一点停车时速度不为 0 的纵坐标	-
t''_i	静止后首次启动时的时间	s
$v''_{i,t}$	静止后首次启动时的速度	m/s
$x''_{i,t}$	静止后首次启动时的横坐标	-
$y''_{i,t}$	静止后首次启动时的纵坐标	-

符号	说明	单位
R	地球半径	r
α	显著性水平	-
n	时间序列长度	-
T_n	第 n 个信号灯的时间周期	-
k	车流量密度	-
T_{track}	车辆行车轨迹数据的采样间隔	s
t_i	编号为 i 的车辆的行车轨迹数据的时间点	s
$x_{i,t}$	编号为 i 的车辆在时间点时的 X 坐标,	m
$y_{i,t}$	编号为 i 的车辆在时间点时的 Y 坐标	m
y_i	编号为 i 的车辆在时间点时的 Y 坐标	-
$v_{i\square t}$	编号为 i 的个车辆在时间点 t 时的速度)	-
$r_{i,t}$	编号为 i 的车辆在时间点 t 是否停止,	-
U_i	编号为 i 的车辆在整体行车速度的数学统计,	-

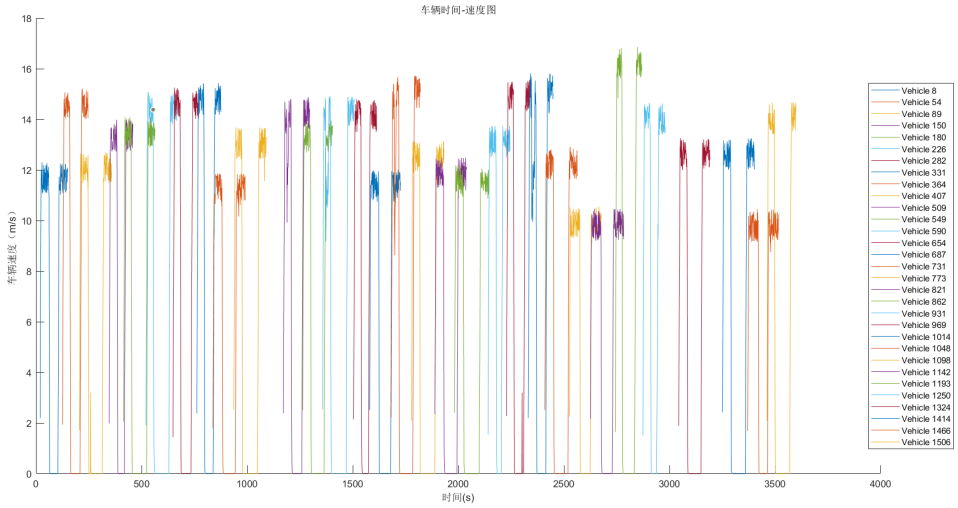

图 **2**　**A1** 中所有车辆的速度-时间图

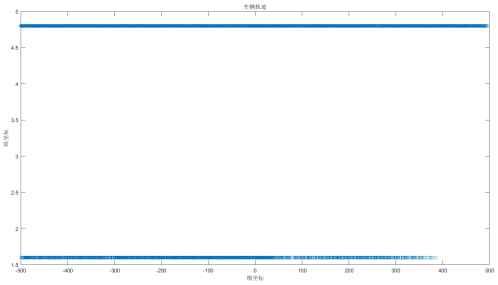

图 **3**　**A1** 表格车辆轨迹

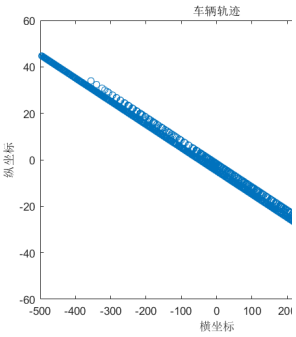

图 **4**　**A2** 表格车辆轨迹

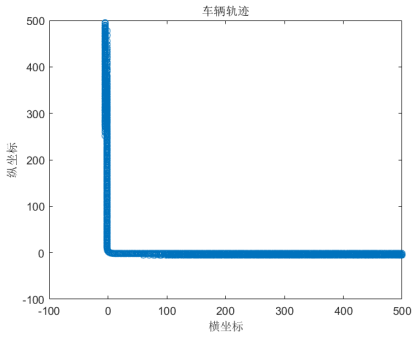

图 **5**　**A3** 表格车辆轨迹

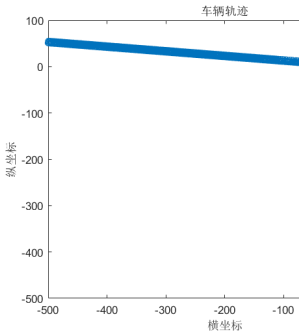

图 **6**　**A4** 表格车辆轨迹

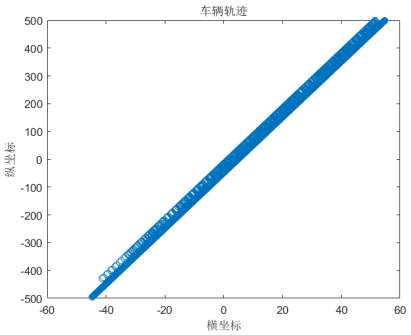

图 **7**　**A5** 表格车辆轨迹

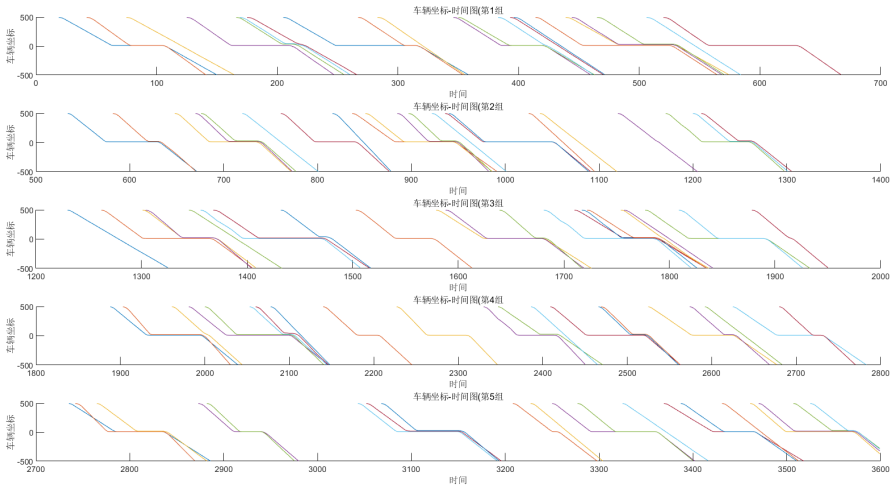

图 **8**　**A1** 中所有车辆的横坐标-时间图（如图 7）

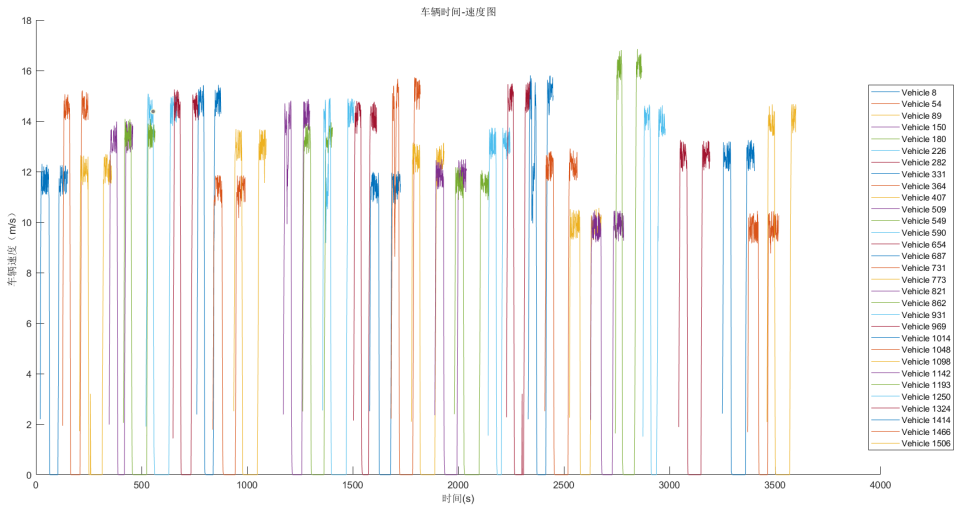

图 **9**　**A1** 中所有车辆的速度-时间图

相邻的两个启动的时刻为一整个周期，而汽车停止的时刻到启动的时刻为红灯时长。由于存在噪声数据，使用 DBSCAN 算法对其进行处理。

5.2 算法描述

5.2.1 DBSCAN 聚类分析

DBSCAN 是一种基于密度的聚类算法, 它具有探测各种形状的聚类簇的能力, 并且能够有效地区分噪声点。该算法的主要思想是, 将样本空间看作是一个多维的空间, 利用密度来描述空间中的数据分布特征。算法的过程如下: 随机选取一个未访问过的点, 并将其标记为已访问。判断该点邻域范围内点数是否大于等于 MinPts, 如果是, 则将这些点都归为一个簇, 并标记为已访问。如果邻域范围内的点数小于 MinPts, 则该点被认为是噪声点, 并将其标记为已处理。继续对剩余未访问的点进行处理, 重复上述过程, 直到所有的点都被处理。最终, 算法输出所有被归为簇的点集以及所有的噪声点。
DBSCAN 算法流程如下：

$$\rho(\boldsymbol{x}) = |N_\epsilon(\boldsymbol{x}_k)|$$

输入: 样本集

$$D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$$

邻域参数

$$(\epsilon, M)$$

输出: 聚类簇

$$C = \{C_1, C_2, \cdots, C_k\}$$

(1) 初始化. 核心点集合

$$\Omega = \varnothing$$

聚类笑数 k:=0 , 未访问的样本集

$$\Gamma := D$$

簇划分 $C = \varnothing$. (2) 对于 $i = 1, 2, \cdots, n$, 确定样本点 x_i 的 ϵ-邻域 $N_\epsilon(x_i)$. 若 $|N_\epsilon(x_i)| \geqslant M$, 则将 x_i 加入核心集 $\Omega := \Omega \cup \{x_i\}$. (3) 生成所有的簇 $C_i(i = 1, 2, \cdots, k)$. while $(\Omega \neq \varnothing)$ do 记录当前未访问的样本集：$\Gamma_{\text{old}} := \Gamma$; 随机选取一个核心点 $x_o \in \Omega$, 初始化当前簇核心点队列 $Q = \{x_o\}$; 更新未访问样本集合 $\Gamma := \Gamma \backslash \{x_o\}$; while $(Q \neq \varnothing)$ do 取出队列 Q 中的首个样本 x_q; **if** $|N_\epsilon(\boldsymbol{x}_q)| \geqslant M$ then 计算 $\Delta := N_\epsilon(\boldsymbol{x}_q) \cap \Gamma$; 将 Δ 中的样本加入到队列 Q; 更新：$\Gamma := \Gamma \backslash \Delta$; end if
end while
置 $k := k + 1$, 生成簇 $C_k = \Gamma_{\text{old}} \backslash \Gamma$; 更新核心点集合 $\Omega := \Omega \backslash C_k$;
end while

据上文的 DBSCAN 算法，我们对车辆轨迹数据进行聚类分析，接着对于聚类分析后的数据进一步处理。因为红灯时速度为 0，绿灯时在运动，为方便我们直观上观察规律，利用如下公式, 对速度进行判断，，绘出周期图（如图）。

$$s_{i,t} = \begin{cases} 1 & v_{i,t} \neq 0 \\ 0 & v_{i,t} = 0 \end{cases}$$

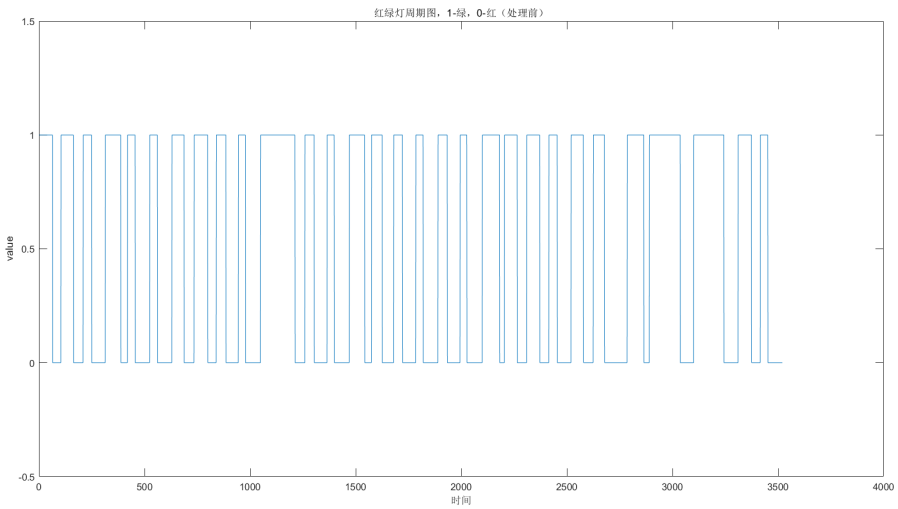

图 **10**　初步计算的周期

5.2.2 分箱法处理异常值

可以观察到，周期图中有些周期有异常，因此对于这些周期使用箱线图剔除异常值首先周期的上、下四分位数 Q_2、Q_1 和四分位距 IQR : Q_1: 下四分位数，数据由小到大排列后第 25 位置的数据；Q_2: 上四分位数，数据由小到大排列后第 75 位置的数据；IQR : 四分位距，$IQR = Q_2 - Q_1$。

在箱线图中，将满足 $x_{i,j,t} > Q_2 + 1.5IQR$ 或 $x_{i,j,t} < Q_1 - 1.5IQR$ 的周期作为异常值并剔除，在后续模型求解中不予考虑。

根据处理过后的数据，求得最终的周期图

最后通过读图以及下图公式

$$T_{signal} = T_{red} + T_{green}$$

计算得到周期的各项数据。

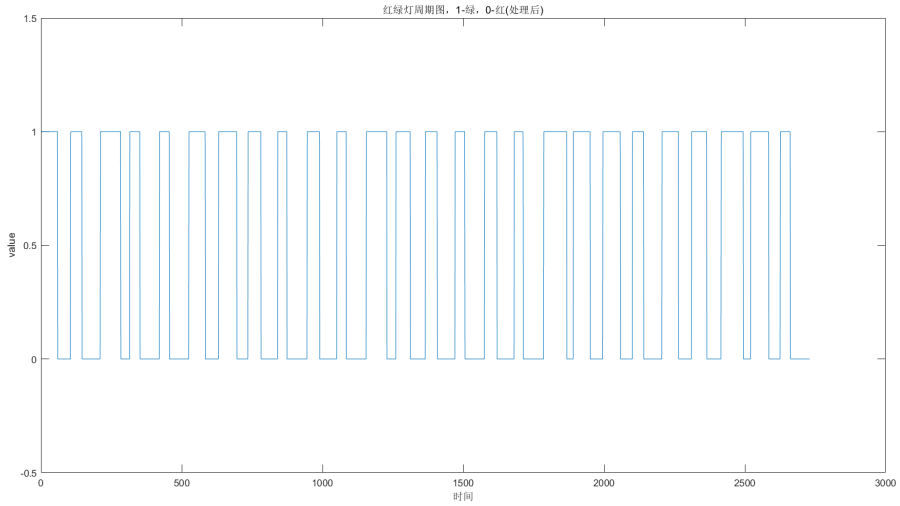

图 **11**　最终的的周期图

路口	A1	A2	A3	A4	A5
红灯时长（秒）	69	53	67	60	47
绿灯时长（秒	36	35	38	28	41
周期（秒）	105	88	105	88	88

5.3 求解结果

六、 问题二模型

在问题一的基础上，题目要求考虑各种因素如：定位误差、车流量、车辆比例等因素对模型估计精度的影响。因此，我们在分析影响因素的基础上对问题一已建立的模型进行优化，从而提高模型估计精度。同时，我们额外分析了转向减速这一因素对模型估计精度的影响。

6.1 递阶层次结构模型

根据题意，分析得递阶层次结构图如下

6.2 准则层分析

6.2.1 样本车辆比例

样本车辆比例的减少会导致数据的代表性下降，这可能影响模型最后结果的可信度，即精度，故本文调整模型，使其在不同的样本车辆比例下，模型预测的稳定性和可靠性都保持良好。

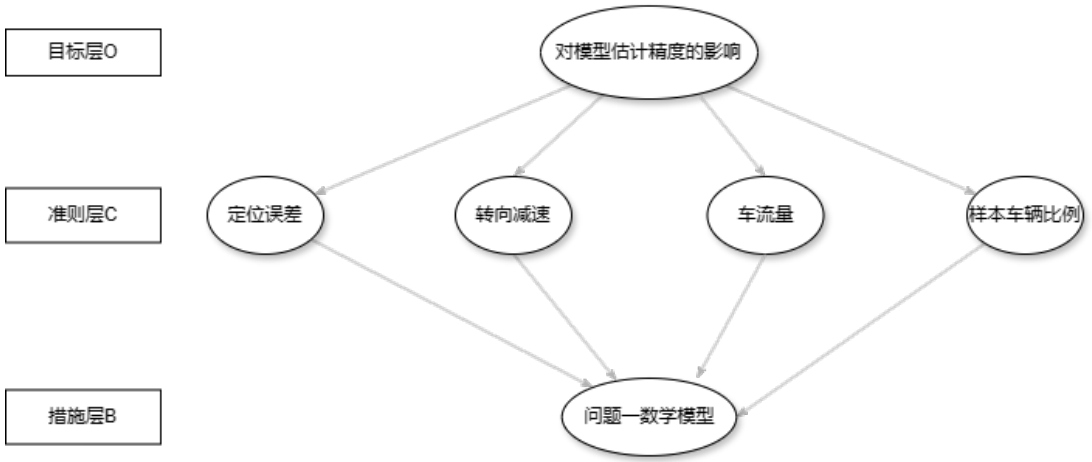

图 **12**　递阶层次结构图

车流量 由于行车数据稀疏，难以保证每个信号周期内的轨迹数据完整，从而会降低模型精度

定位误差 由于定位误差的存在，使采样点可能无法捕获车辆的实际启动的时间戳，从而影响数据的可靠性，导致模型估计精度降低。

转向减速 由于我们对车辆静止的判断是基于速度是否小于一个既定的阈值，而车辆在转向时会减速，可能会达到所设定阈值一下造成误判。影响模型估计精度。
建立判断矩阵 判断矩阵是表示本层所有因素针对上一层某一个因素的相对重要性的比较。判断矩阵的元素 aij 用 1-9 标度方法给出。

标度	含义
1	表示两个因素相比，具有同样重要性
3	表示两个因素相比，一个因素比另一个因素稍微重要
5	表示两个因素相比，一个因素比另一个因素明显重要
7	表示两个因素相比，一个因素比另一个因素强烈重要
9	表示两个因素相比，一个因素比另一个因素极端重要
2、4、6、8	上述两相邻判断的中值
倒数	aij=1/aji

1-9 标度方法

我们在这里做 $\frac{n(n-1)}{2}$ 次两两判断，得出判断矩阵 A
对于构建的判断矩阵 A，其中元素若满足 $a_{ij}a_{jk} = a_{ik}$,$\forall i, j, k = 1, 2, \cdots, n$ 则其为正互反矩阵。

	车流量	车辆比例	定位误差	转向减速
车流量	1	1	2	5
车辆比例	1	1	2	4
定位误差	1/2	1/2	1	2
转向减速	1/5	1/4	1/2	1

判断矩阵 A

6.2.2 一致性检验

对于正互反矩阵 A，当且仅当其最大特征根 $\lambda_{\max} = n$，有 n 阶正互反矩阵 A 为一致矩阵，且当正互反矩阵 A 非一致时，必有 $\lambda_{\max} > n$。因而我们可以由 $\lambda_{\max}$ 是否等于 n 来检验判断矩阵 A 是否为一致矩阵。由于特征根连续地依赖于，故 $\lambda_{\max}$ 比 n 大得越多，A 的非一致性程度也就越严重，$\lambda_{\max}$ 对应的标准化特征向量也就越不能真实地反映出 $X = \{x_1, \cdots, x_n\}$ 在对因素 Z 的影响中所占的比重。因此，对决策者提供的判断矩阵有必要作一次一致性检验，以决定是否能接受它。
对判断矩阵的一致性检验的步骤如下：
（i）计算一致性指标 CI
$CI = \frac{\lambda_{\max} - n}{n-1}$
（ii）查找相应的平均随机一致性指标 RI。对 $n = 1, \cdots, 9$，Saaty 给出了 RI 的值，如下表所示：

n	1	2	3	4	5	6	7	8	9
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45

RI 的值是这样得到的，用随机方法构造 500 个样本矩阵：随机地从 1 9 及其倒数中抽取数字构造正互反矩阵，求得最大特征根的平均值 $\lambda'_{\max}$，并定义
$RI = \frac{\lambda'_{\max} - n}{n-1}$
计算一致性比例 $CR\,CR = \frac{CI}{RI}$ 当 $CR < 0.10$ 时，认为判断矩阵的一致性是可以接受的，否则应对判断矩阵作适当修正。

6.2.3 得出层次排序

准则	车流量	样本车辆比例	定位误差	转向减速
准则层权重	0.4091	0.3182	0.1818	0.0909
影响度排序	1	2	3	4

层次排序

由表可知车流量对模型估计精度的影响最大，其次是样本车辆比例、定位误差、转向减速。

6.3 建立模型减小各影响因素对模型估计精度影响

以 B1 为代表：主要建模过程与问题一类似, 数据处理后画出车辆 x 坐标-时间图

使用 DBSCAN 聚类去除车辆噪声得到每个信号周期内的车辆坐标-时间图由公式 () 得到速度-时间图

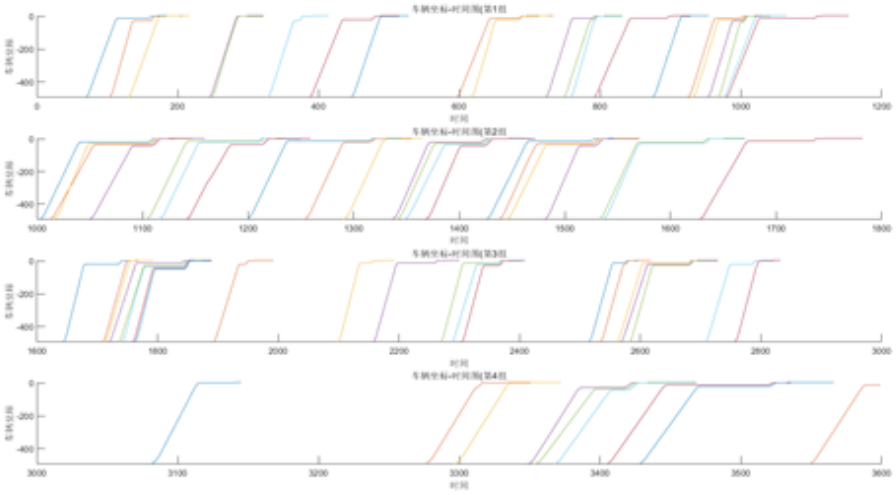

图 **13**　车辆 **x** 坐标**-**时间图

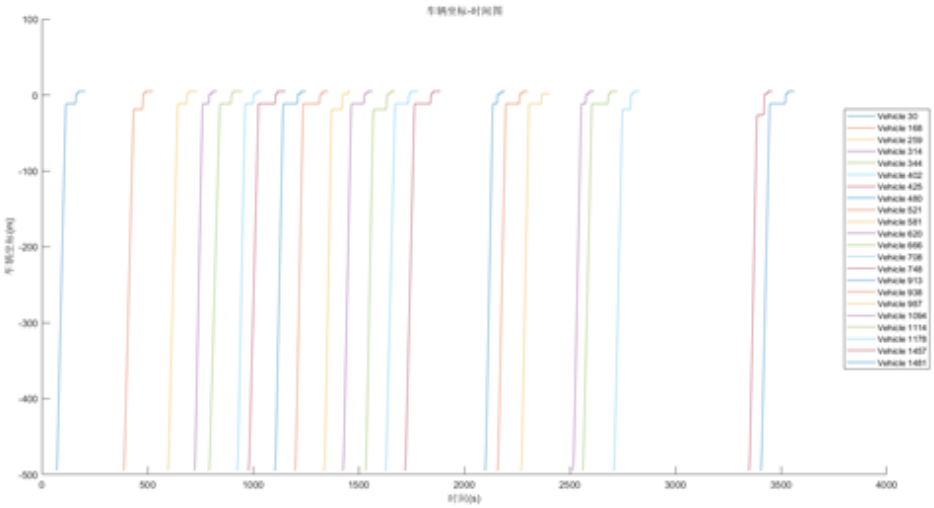

图 **14**　处理后车辆坐标**-**时间图

6.4 降低各影响因素对模型估计精度的影响

问题一中我们以绿灯开始的时间为时间戳定义信号周期长度，由于车流量与样本车辆比例的影响，路口排队车辆和第一辆车之间存在延迟时间，样本数据中排队第一辆车也可能与实际有偏差。使得等待红灯的车辆启动时间不一定为绿灯开始时间。我们考虑这部分时间差，将排队车辆开始离开路口的时间设为，则绿灯启动时间戳如下：

$$t_{ij}=t'_{ij}-\Delta l/u_v$$

Δl: 车辆停止位置到停车线的距离

U_v 车队消散速度，在交通波理论中也称为起始波，可以表示为：

$$U_v=\frac{u}{hku-1}$$

该方程为起始波的运动学模型。其中，h：饱和车流量；u：车辆的正常行驶速度；k：阻塞交通密度流量。由上述方程可知起始波的大小与行进距离和行驶速度有关，在不饱和循环中，车辆的行进速度和行驶速度可以近似地视为常数，因此本文将车队的起始波速度近似地视为常数。其一般是稳定的，查阅文献可知合理的范围在 15km/h—20km/h。经过上述处理后得到更为精确的信号灯周期图

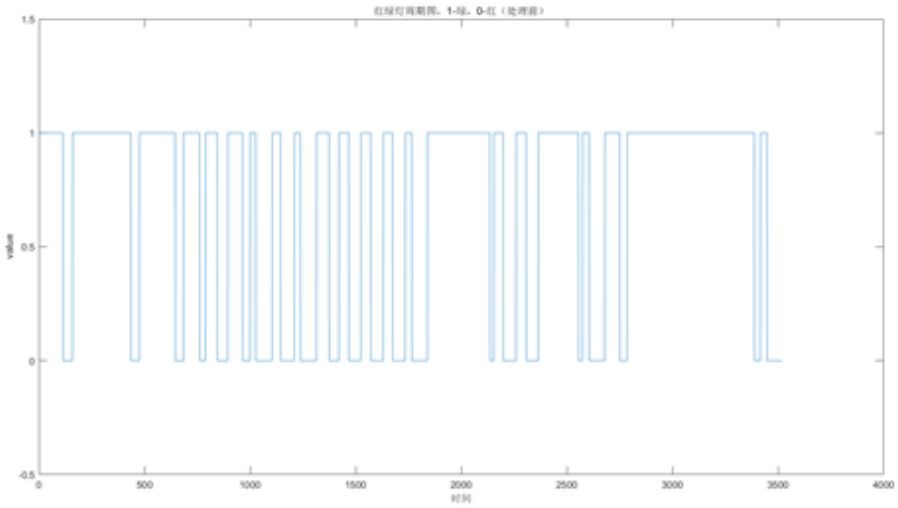

图 **15**　处理前

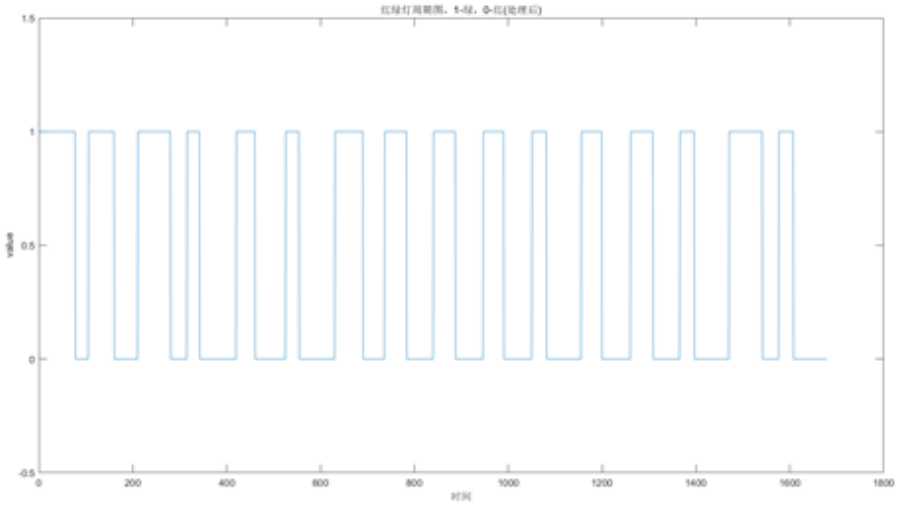

图 **16**　处理后

两张图表对比可以明显看出，在问题一的模型上对影响因素的干扰进行相应处理后的信号周期更加均匀，证明所建立的数学模型估计精度更高。因此，我们继续采用此种方式得出剩余四个路口相应方向的信号灯周期

七、 问题三模型

7.1 求解 C1-C6 各自的信号灯周期

此处信号灯周期的求解基于问题二建立的数学模型得出，便不再赘述。得出如下 6 个路口分别的信号灯周期图由图示可估计 C1-C6 中，C1、C2、C6 信号灯周期发生变化，C3、C4 信号灯周期无变化。其中 C4 信号灯周期十分

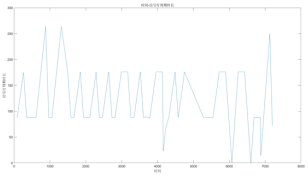

图 **17** **C1**

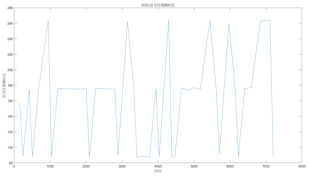

图 **18** **C2**

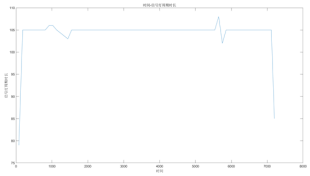

图 **19** **C3**

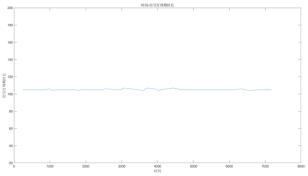

图 **20** **C4**

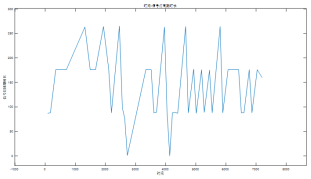

图 **21** **C5**

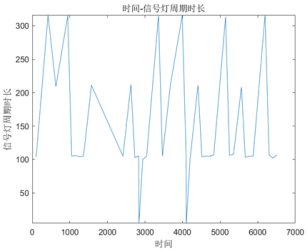

图 **22** **C6**

平稳

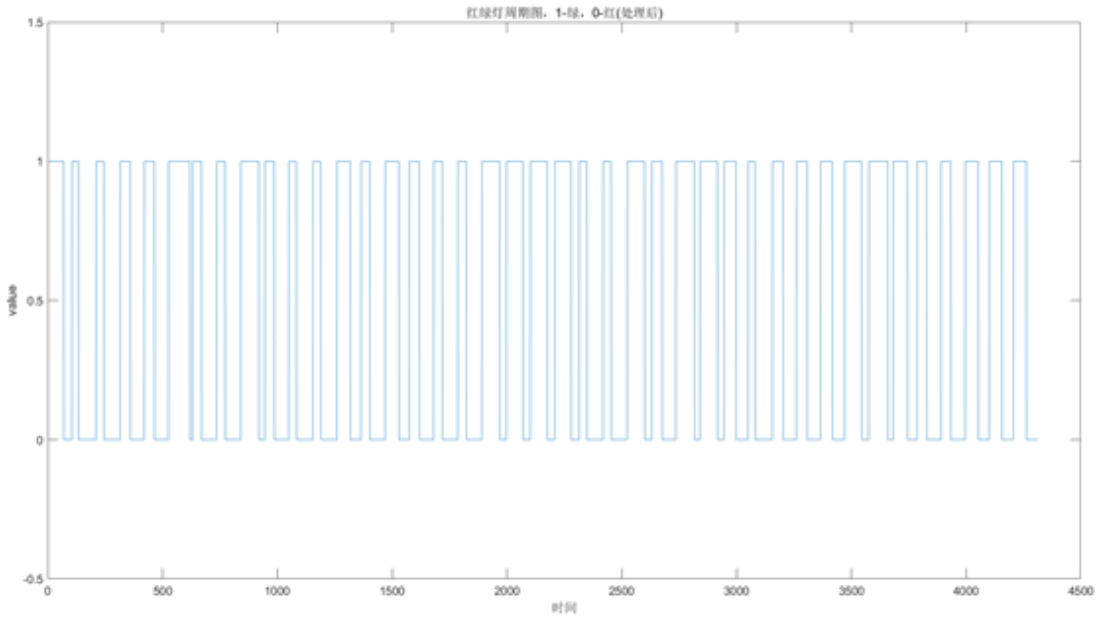

图 **23** **C4** 周期图

而后，我们通过 ADF 检验验证 C1-C6 信号灯周期变化与否的判断与预期的吻合。

7.2 ADF 检验信号灯周期是否发生变化

信号灯周期在连续 2 小时内按顺序排列可以看作是一个时间周期 Ct，通过对时间序列进行平稳性检验，平稳时间序列在其平均值附近呈现出不断波动，而非平稳时间序列在不同时期往往具有不同的均值，使用问题一中方程中的

方法，提取相邻循环下的绿灯开始时间，然后按 2 小时的时间顺序排列周期。本文使用了增强的 Dickey-Fuller（ADF）检验，这是时间序列平稳性最常用的检验方法。ADF 的测试模型如下：

$$\Delta C_t = \rho C_{t-1} + \delta x^{`}{}_t + \beta_b \sum_{b=1}^{a} \Delta C_{t-b} + \varepsilon_t$$

其中，t：时间变量

X_t：可选的外生变量，包括漂移或漂移和时间趋势

ρ 和 δ 是要估计的参数；ε_t：残差

若 $|\rho|<1$，则 C_t 是一个稳态序列，通过检查 ρ 严格小于 0；

假设 $H_0 : \rho=0$ ，则 C_t 包含单位根，且是非平稳时间序列。

测试的 ADF 统计量：

$$t_\rho = \hat{\rho} / S(\hat{\rho})$$

如果样本计算的 ADF 统计量值小于临界值，则证明其对应的时间序列是平稳时间序列，否则为非平稳时间序列。

经过检验得到如下结果 与 7.1 猜想相符

路口	C1	C2	C3	C4	C5	C6
ADF 检验结果	符合 H0	符合 H0	拒绝 H0	拒绝 H0	符合 H0	符合 H0
周期是否变化	是	是	否	否	是	是

7.3 pettitt 突变检测法估计周期变化点

Pettitt 突变检测法是一种非参数检验方法，前提是时间序列存在趋势变化。该方法通过检查时间序列的平均值变化的时间来确定非平稳时间序列的周期变化点。首先，确定整个时间序列的第一个突变点，然后将原始时间序列分为两个时间序列，并检测出新的突变点; 可能有多个突变点。最后，应根据具体原因分析确定突变点。

对于给定的时间序列 $C_t = 1, 2, 3, ..., z$，用 U_t 对其变化进行统计

$$U_{t,z} = U_{t-1,z} + V_{t,z}$$

$$V_{t,z} = \sum_{q=1}^{z} \text{sgn}(C_t - C_q)$$

对于函数 sgn(x) 有

统计值表示序列中最可能的突变点

$$Q_t = \max_{1 \le t \le z} |U_t|$$

用 P 检验统计量，从而确定突变点是否显著

$$P = 2\exp\{-6Q_t^2/(z^3 + z^2)\}$$

对所得数据结果整理得 C1-C6 这 6 个不相关路口相应方向的信号灯周期是否变化与周期切换时刻，以及新旧周期参数如下表所示：

7.4 结果分析

对信号灯周期突变时刻进行分析可知，各个路口信号灯周期切换时，若切换后信号灯总周期增大，切换后的绿灯时长在总时长中占比会提高，若切换后信号灯总周期减小，切换后的红灯时长在总时长中占比会提高。故本文猜想周期变化的条件之一为车流量，当车流量增加时，信号灯周期会增加，当车流量减少时，信号灯周期会降低，且信号灯周期切换时间点都在信号灯完整显示一周期后，在新周期开始前完成切换。

```
发现信号灯周期发生突变!
突变发生在第 247 s。
新周期参数为:
 T_signal(2) = 89.000000
红灯时长为: T_red(2) = 35.000000
绿灯时长为: T_green(2) = 54.000000
旧周期参数为:
 T_signal(1) = 158.000000
红灯时长为: T_red(1) = 27.000000
绿灯时长为: T_green(1) = 131.000000
发现信号灯周期发生突变!
突变发生在第 686 s。
新周期参数为:
 T_signal(5) = 177.000000
红灯时长为: T_red(5) = 18.000000
绿灯时长为: T_green(5) = 159.000000
旧周期参数为:
 T_signal(2) = 88.000000
红灯时长为: T_red(4) = 36.000000
绿灯时长为: T_green(4) = 52.000000
发现信号灯周期发生突变!
突变发生在第 1829 s。
```

图 **24**　得到的数据

八、 问题四模型

首先对所有的数据进行分析，使用 MatLab 画出所有车辆的轨迹图当 P 小于给定的显著性水平（经过文献查询后，本文取 0.05），则认为存在统计学上显着的突变点。

通过 pettitt 检验对 C1-C6 信号灯周期进行检验得到以下结果（图中仅展示部分数据）：

可以看出这是一个十字路口，涉及方向较多。

由我国的道路交通法可知，车辆有如图 12 种方式通过十字路口：朝南直行，朝北直行，朝西直行，朝东直行，朝南右转，朝南左转，朝北右转，朝北左转，朝西右转，朝西左转，朝东右转，朝东左转。

考虑对数据进行聚类，相同方向的车辆轨迹聚为一类，所以应基于 K-means 聚类算法的十字路口交通方向识别并考虑将其聚为 12 类。剔除在十字路口不停止的车辆后提取对于计算十字路口周期有帮助的车辆，即：东西直行 + 右转,

路口	C1	C2	C3	C4	C5	C6
周期是否变化	是	是	否	否	是	是
周期 1 红灯时长（秒）	7	88			351	60
周期 1 绿灯时长（秒）	81	33			50	44
周期 1 总时长（秒）	88	55			126	104
周期切换时刻	263	247			176	944
周期 2 红灯时长（秒）	40	35			4223	50
周期 2 绿灯时长（秒）	135	54			26	265
周期 2 总时长（秒）	175	89			62	315
周期切换时刻	351	1214			88	4514
周期 3 红灯时长（秒）	32	30			6071	64
周期 3 绿灯时长（秒）	56	146			23	40
周期 3 总时长（秒）	88	176			153	104
周期切换时刻	1319	3413			176	5669
周期 4 红灯时长（秒）	22	27				64
周期 4 绿灯时长（秒）	66	60				39
周期 4 总时长（秒）	264	87				103
周期切换时刻	1583	4647				6302
周期 5 红灯时长（秒）	32	32				23
周期 5 绿灯时长（秒）	56	145				84
周期 5 总时长（秒）	88	177				107
周期切换时刻		5705				
周期 6 红灯时长（秒）		47				
周期 6 绿灯时长（秒）		41				
周期 6 总时长（秒）		88				

南北直行 + 右转, 东西左转, 南北左转四种，然后根据这四类车辆的启动时间算出十字路口总周期。

8.1 数据处理

计算车辆速度及提取速度变化信息的步骤与前三问一致，但与前三问的处理方法不同的是，此文因为涉及到十字路口的转向问题，所以应算出每个车辆航向角，从而判断车辆的转向。车辆航向角的计算是基于连续的 x 轴, y 轴坐标，计算车辆的航向角，即车辆的行驶方向，计算方式如下：

$$\mathrm{t}(x) = [\min(x), \max(x), \min(y), \max(y), \max(\theta), \min(\theta)]$$

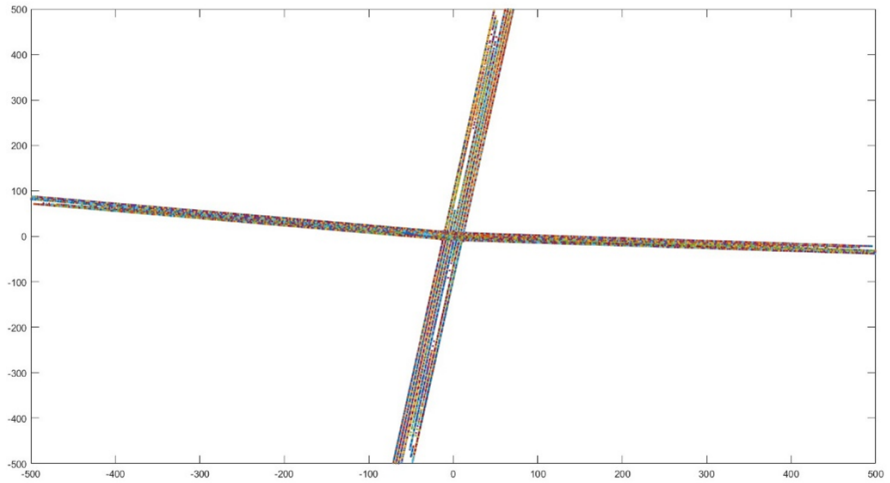

图 **25**　所有车辆轨迹图

在得到各车辆的时间、轨迹、速度及航向角后，我们选择最能代表车辆特征的轨迹的 x 最大值、x 最小值、y 最大值、y 最小值、航向角最大值最小值作为每个车辆的特征向量进行聚类。特征向量表示如下：

$$\mathrm{t}(x) = [\min(x), \max(x), \min(y), \max(y), \max(\theta), \min(\theta)]$$

8.2 模型的建立

为了实现车辆的对题目进行分类，我们利用 K-Means 聚类算法进行分类。K-Means 聚类算法是一种无监督算法，它通过数据之间的内在关系把样本划分为若干类别，使得同类别样本之间的相似度高，而不同类别之间的样本相似度低（即增大类内聚，减少类间距）。K-Means 聚类算法的步骤是先随机选择 □ 个对象作为初始聚类中心，然后计算每个对象与每个种子聚类中心之间的距离，并将每个对象分配到最近的聚类中心。集群中心和分配给他们的对象代表一个集群。每次迭代，基于集群中已有的对象，重新计算集群的中心。这个过程不断重复，直到中心点和样本划分的类别同时收敛。

8.2.1 k-means 聚类分析

为了实现车辆的对题目进行分类，我们利用 K-Means 聚类算法进行分类。K-Means 聚类算法是一种无监督算法，它通过数据之间的内在关系把样本划分为若干类别，使得同类别样本之间的相似度高，而不同类别之间的样本相似度低（即增大类内聚，减少类间距）。K-Means 聚类算法的步骤是先随机选择 □ 个对象作为初始聚类中心，然后计算每个对象与每个种子聚类中心之间的距离，并将每个对象分配到最近的聚类中心。集群中心和分配给他们的对象代表一个集群。每次迭代，基于集群中已有的对象，重新计算集群的中心。这个过程不断重复，直到中心点和样本划分的类别同时收敛。

建立 K-means 聚类算法步骤如下：

step1

建立每个车辆的特征向量 $t(x)$；

Step2

由肘部法则和需求 K 值

肘部法的核心指标是 SSE(Sum Of The Squared Errors，误差平方和)，SSE 是所有样本的聚类误差，代表了聚类效果的好坏。肘部法的核心思想是随着聚类数 K 的增大，样本划分会更加细致，每个簇聚合程度会逐渐提高，那么 SSE 会逐渐变小。当 K 小于真实类数的时候，K 的增加会大幅度的增加每个簇的聚合程度，下降得会很快；当 K 到达真实类数的时候，则会逐渐趋于平缓，那么 SSE 与 K 关系图将会是一个肘部的形状，而这个肘部便是 K 值的真实类数。

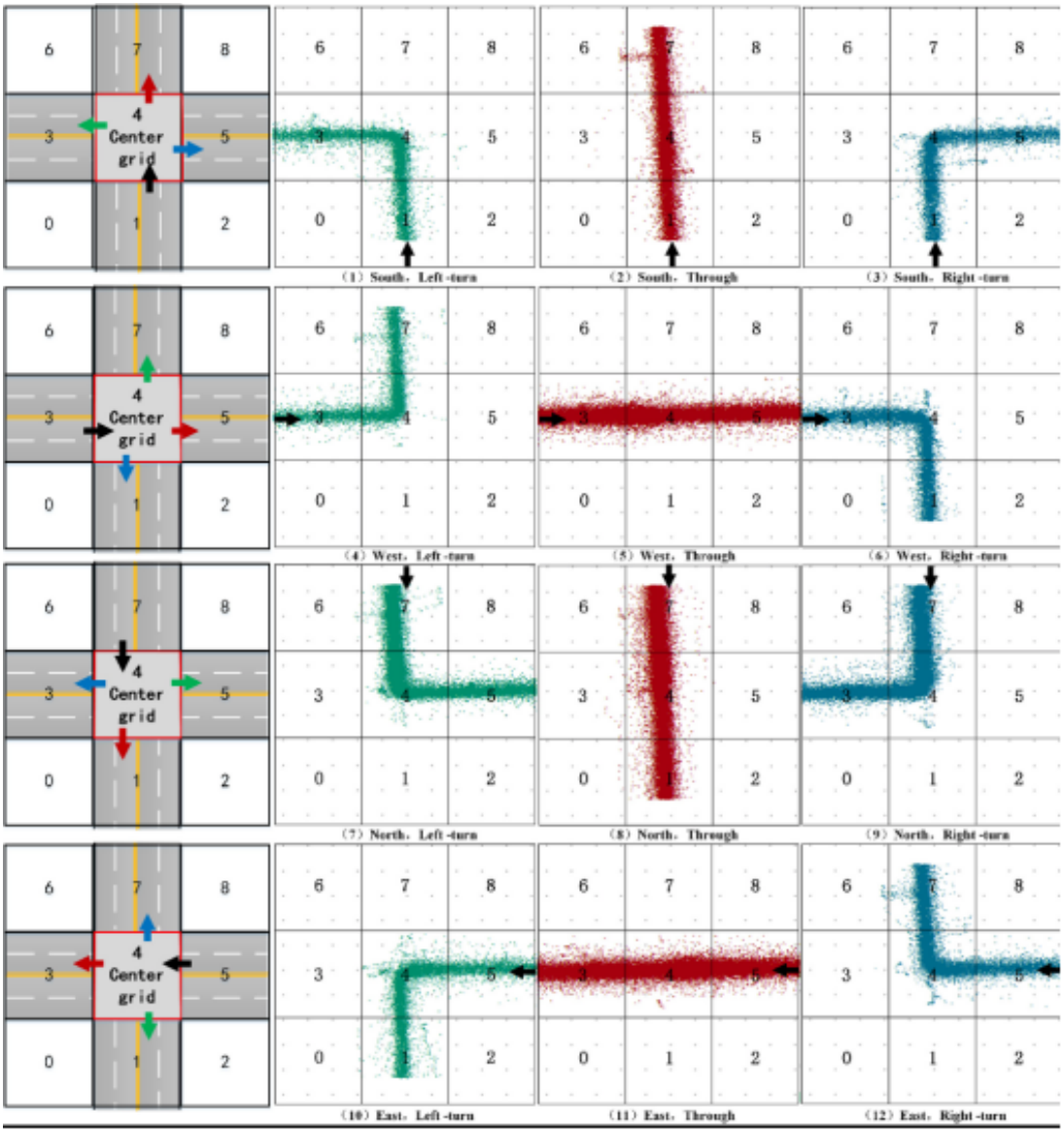

图 **26**　十字路口行车轨迹

如图 20 所示，此时选择 K 为 12 时，SSE 极小，最接近真实类数，又因为十字路口由 12 个转向，所以我们将 K 设置为 12，将所有车辆聚为 12 类；

step3

建立 K-means 聚类算法

具体算法流程如下：1) 初始化：确定聚类数 k ，并随机挑选 k 个样本点作为初始聚类中心。2) 计算距离：对于每个数据点，计算其到每个聚类中心的欧氏距离 3) 更新划分：按照“欧氏距离最小原则”，将数据点合并到最近的类。4) 更新聚类中心：重新计算每个维度中每个类别的数据点的平均值，并将获得的平均值点用作新的聚类中心。5) 判断是否收敛：如果聚类中心的变化没有超过预设阈值或者没有导致聚类标准函数 E 的变化超过设定阈值，则收敛；否则转到 2) 具体的应用流程如图所示

8.2.2 K-means 聚类效果

如图所示，车辆在十字路口的方向可分为以下 12 种：朝南直行，朝北直行，朝西直行，朝东直行，朝南右转，朝南左转，朝北右转，朝北左转，朝西右转，朝西左转，朝东右转，朝东左转。

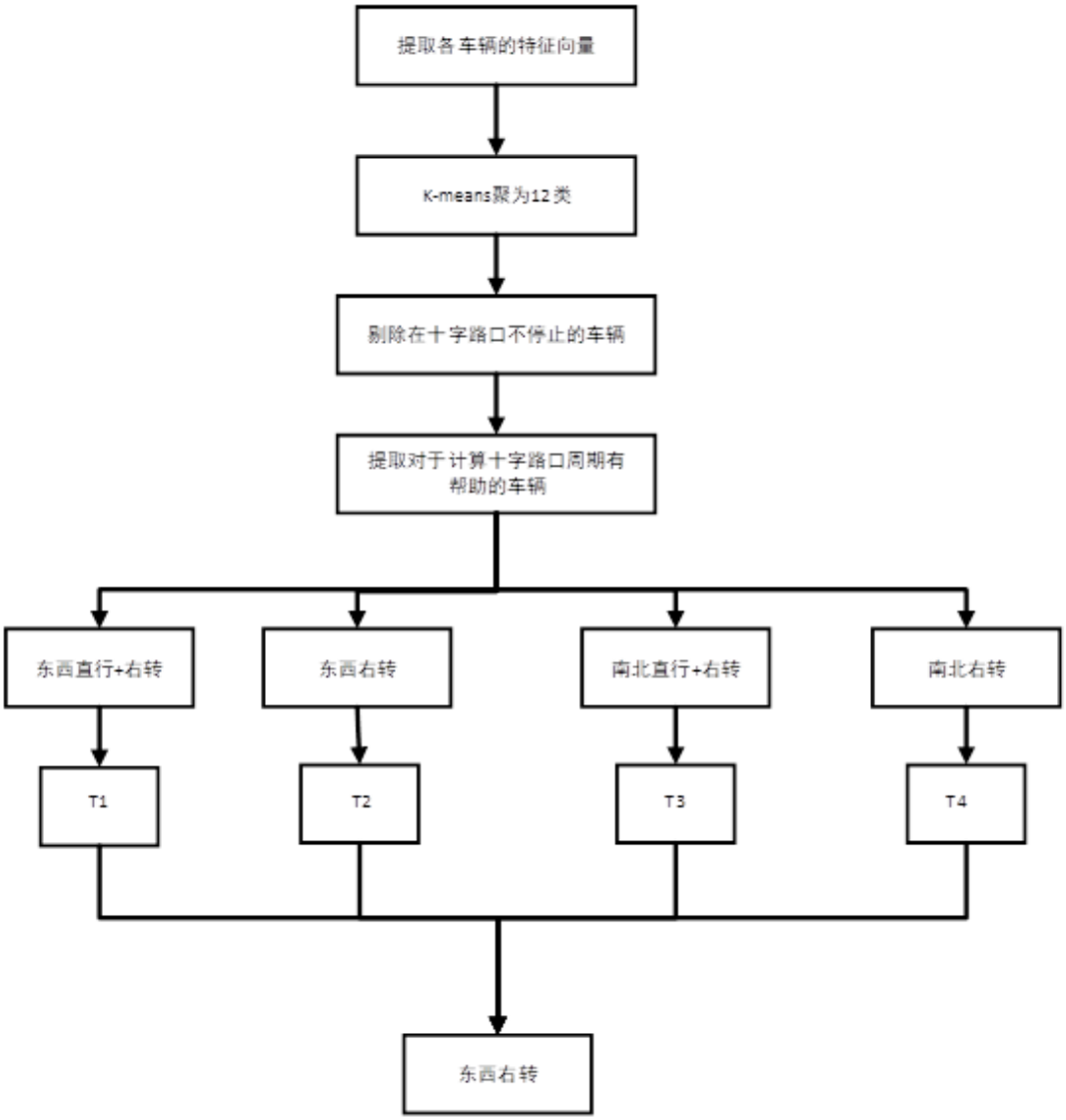

图 **27**　聚类流程图

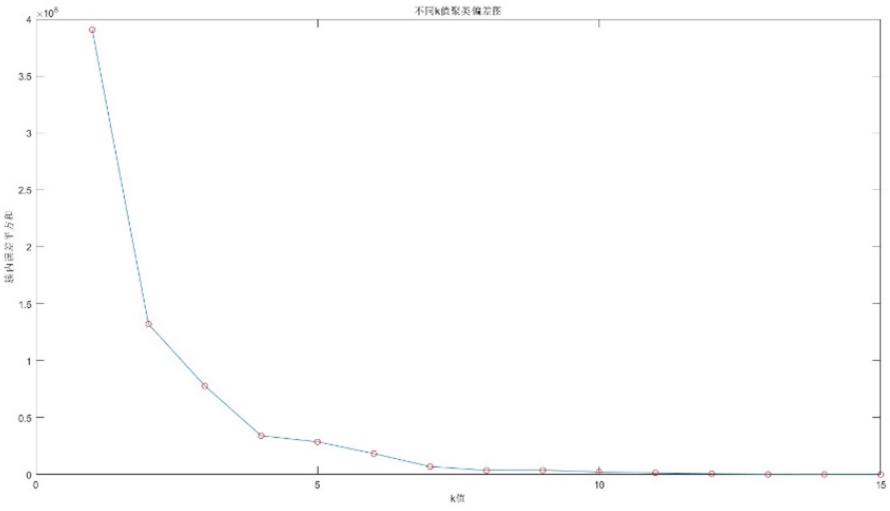

图 **28**　不同 **k** 值聚类偏差图

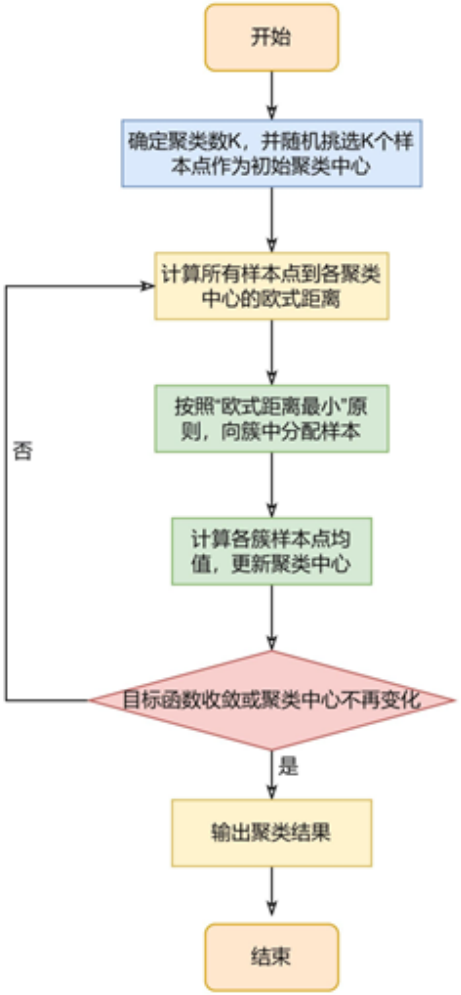

图 **29**　**K-means** 流程图

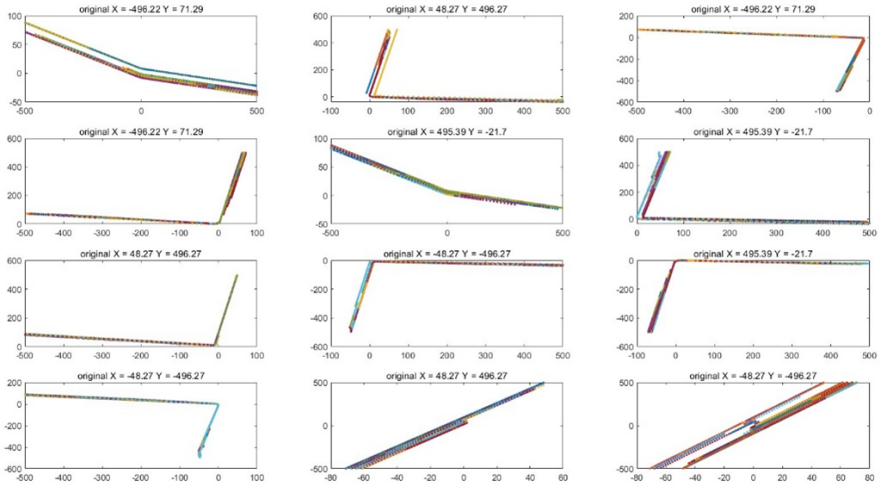

图 **30**　**K-means** 聚类效果

8.3 信号灯周期计算

8.3.1 十字路口信号灯周期模型建立

由我国交通规则可知，直行通常可以右转，为简化模型，我们将根据东西直行 + 右转，南北直行 + 右转，东西左转，南北左转这四类转向进行计算周期，这四类转向的顺序如图所示 如下图所示，十字路口信号灯周期计算方式

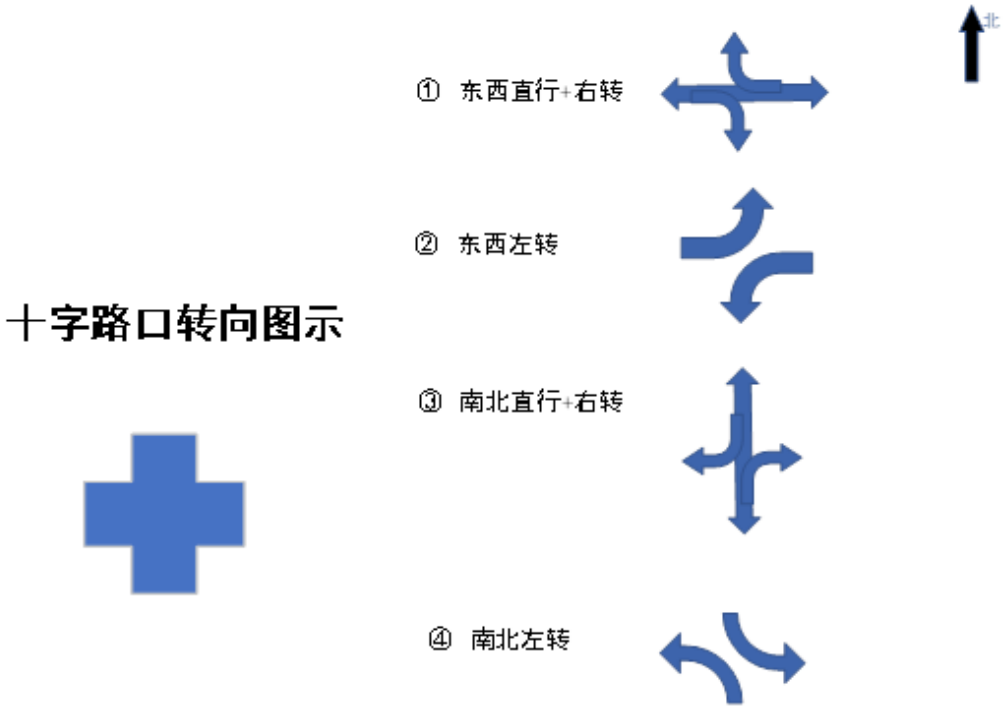

图 **31**　十字路口转向图示

如下

$$T_{Green-west-east}\mathrm{T}_{Green-south-north}\mathrm{T}_{Green-w-e-left}\mathrm{T}_{Green-s-n-left}$$

分别是各东西直行 + 右转的指示绿灯时长，南北直行 + 右转的指示绿灯时长，东西左转的指示绿灯时长，南北左转的指示绿灯时长。

$$T_{all} = T_{Green-west-east}+\mathrm{T}_{Green-south-north}+\mathrm{T}_{Green-w-e-left}+\mathrm{T}_{Green-s-n-left}$$

T_{all} 为四个转向灯时长之和，即为十字路口的整个周期时长。

8.4 十字路口信号灯周期计算方法

为计算 $T_{Green-west-east}\mathrm{T}_{Green-south-north}\mathrm{T}_{Green-w-e-left}\mathrm{T}_{Green-s-n-left}$, 我们使用 matlab 建立这四个转向的车辆启动时间矩阵 M

$$M = [\mathrm{M}_{west-east},\mathrm{M}_{south-north},\mathrm{M}_{w-e-left},\mathrm{M}_{s-n-left}]$$

东西直行 + 右转转向的车的启动时间建立为矩阵:

$$\mathrm{M}_{west-east}$$

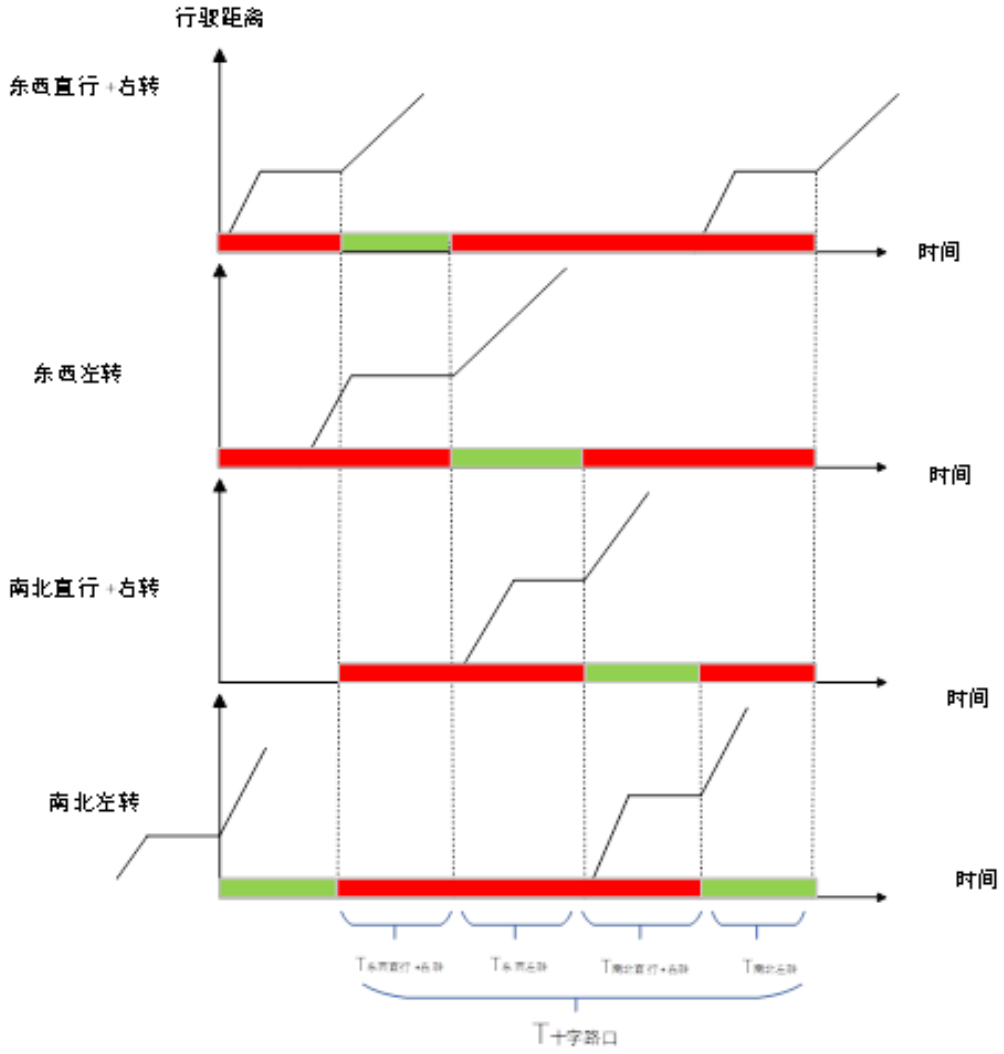

图 **32**　十字路口信号灯周期计算示意图

南北直行 + 右转转向的车的启动时间建立为矩阵

$$\mathrm{M}_{south-north}$$

东西左转转向的车的启动时间建立为矩阵

$$\mathrm{M}_{w-e-left}$$

南北左转转向的车的启动时间建立为矩阵

$$\mathrm{M}_{s-n-left}$$

建立了四个转向的车辆启动时间矩阵 M 后，用 matlab 将每一列的时间序列用不同的颜色形状的图案画出，如图所示 图片放大后可见，该时间矩阵呈现十分规律的周期性，但是同一时间点有许多相近的点聚在一起，即红灯结束，绿灯亮起后有很多车近乎同时启动出发。

为了计算方便以及模型的准确性，我们将聚集到一起的点进行聚类，并只保留每一类中的第一个数据，剔除其他噪声数据。因为红灯结束，绿灯亮起后出发的第一辆车的启动出发时间最接近绿灯亮起的时间，所以保留的每类第一个数据可以近似认为是绿灯亮起的时间点。

数据处理后，只保留每个绿灯亮起后出发的第一辆车的启动出发时间，可得到矩阵：

$M_{Green} = [\mathrm{M}_{Green-west-east},\mathrm{M}_{Green-south-north},\mathrm{M}_{Green-w-e-left},\mathrm{M}_{Green-s-n-left}]$ 由于绿灯亮起后出发的第一

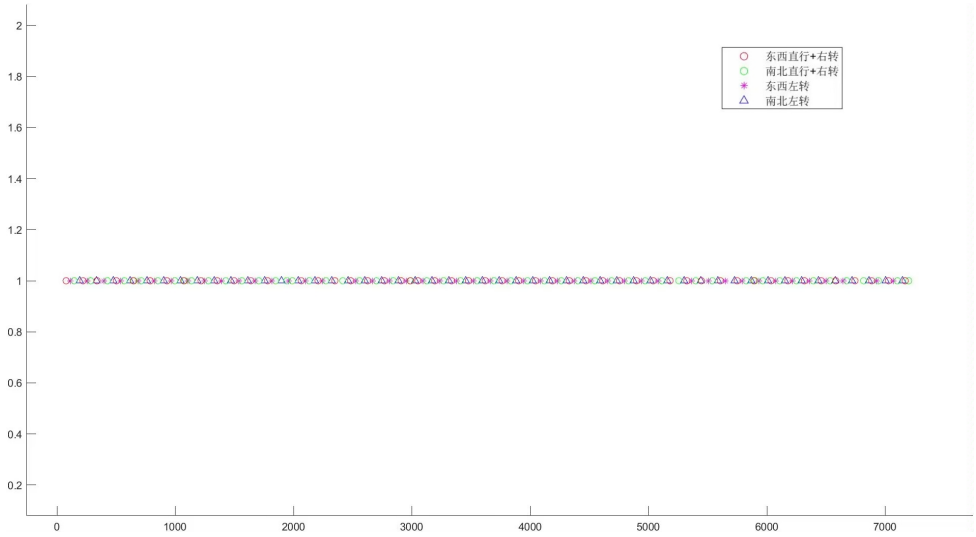

图 **33**　图示与整体图像

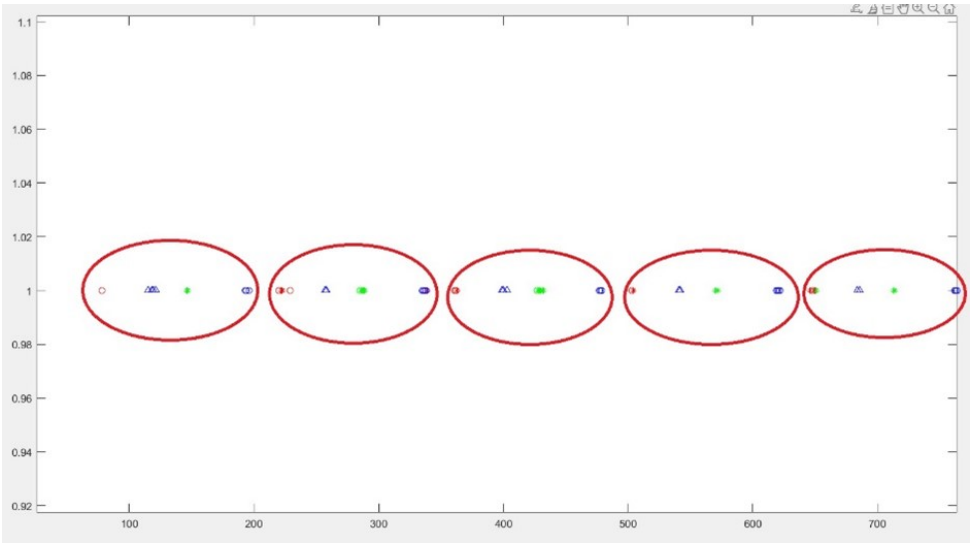

图 **34**　四个转向的车辆启动时间序列图

辆车的启动出发时间最接近绿灯亮起的时间，所以该矩阵的每一列数据可以近似认为是绿灯亮起的时间点。
用 matlab 将 M_{Green} 每一列的时间序列用不同的颜色形状的图案画出，如图所示： 图片放大后可见，该时间矩阵呈现十分规律的周期性，将图上的时间序列点进行前向差分就能得出东西直行 + 右转的指示绿灯，南北直行 + 右转的指示绿灯，东西左转的指示绿灯，南北左转的指示绿灯。

8.4.1 计算整个周期时长

step1
M_{Green} 进行前向差分

$$diff(M_{Green}) = M_{Green}(:, 2:end) - M_{Green}(:, 1:end-1)$$

step2
$diff(M_{Green})$ 每列求平均得

$$T_{Green} = [T_{Green-west-east}, \mathrm{T}_{Green-south-north}, \mathrm{T}_{Green-w-e-left}, \mathrm{T}_{Green-s-n-left}] = mean(diffM_{Green}, 'all')$$

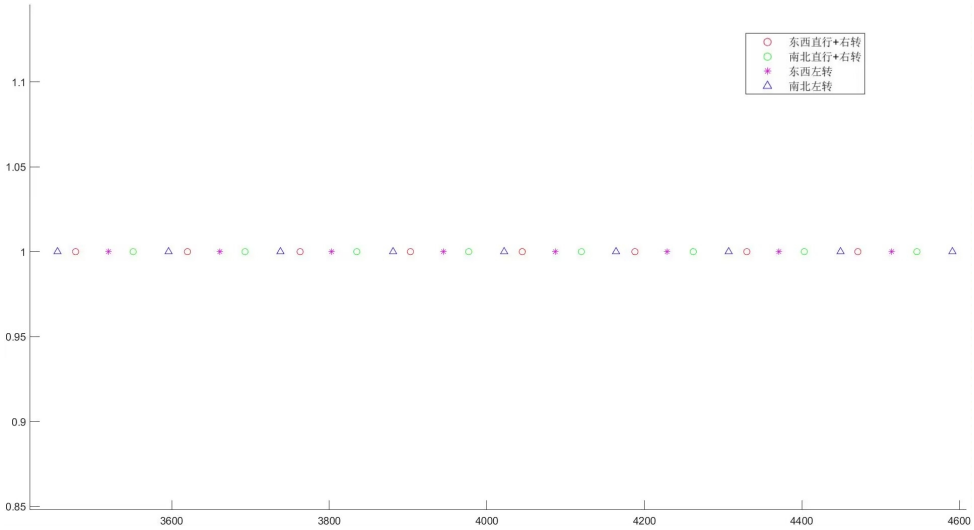

图 **35**　图示与整体图像

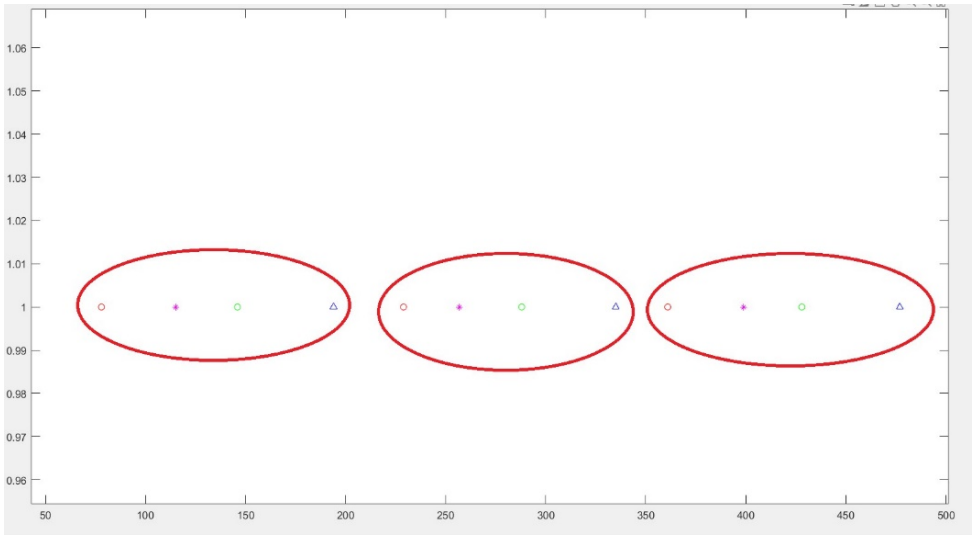

图 **36**　四个转向的绿灯亮起时间序列图

$$T_{Green} = [T_{Green-west-east}, \mathrm{T}_{Green-south-north}, \mathrm{T}_{Green-w-e-left}, \mathrm{T}_{Green-s-n-left}]$$

T_{Green} 是一个 1*4 的矩阵，它的各列分别是各东西直行 + 右转的指示绿灯时长，南北直行 + 右转的指示绿灯时长，东西左转的指示绿灯时长，南北左转的指示绿灯时长。

step3

T_{Green} 进行后向差分

$$diffback(M_{Green}) = M_{Green}(2:m,:) - M_{Green}(1:m-1,:)$$

step4

$diffback(M_{Green})$ 求众数 $T_{all} = \frac{1}{m}\sum_{i=1}^{m-1} diffback(M_{Green})$ T_{all} 即为十字路口的整个周期时长：

$$T_{all} = T_{Green-west-east} + \mathrm{T}_{Green-south-north} + \mathrm{T}_{Green-w-e-left} + \mathrm{T}_{Green-s-n-left}$$

路口 D	绿灯时长（秒）	红灯时长（秒）	总周期（秒）
东西直行 + 右转	27	115	142
东西左转	37	105	142
南北直行 + 右转	28	114	142
南北左转	50	92	142
总周期（秒）	142		

十字路口信号灯周期计算结果

8.4.2 计算结果

九、 模型的评价

本文运用的数学模型有 DBSCAN 聚类、层次分析法、ADF 检验、pettitt 检验、K 聚类等基础数学模型，现对其优缺点进行分析

9.1 模型的优点

- DBSCAN 聚类能够发现能够发现任意形状的簇，对噪声点具有较好的鲁棒性。且不需要事先指定簇的个数。能够处理数据集中簇的密度不均匀的情况。
- 层次分析法能够将复杂的决策问题分解为层次结构，便于分析和决策。且能够考虑多个因素之间的相对重要性，提高决策的科学性。
- DF 检验用于检验时间序列数据的平稳性，适用于许多经济和金融领域的数据分析。且能够检验时间序列数据中是否存在单位根，判断序列是否具有平稳性。
- Pettitt 检验用于检测时间序列数据中的突变点或结构突变，适用于环境、气候等领域的数据分析。且它是非参数检验方法，不需要对数据分布做出假设。
- K-means 聚类简单且高效，适用于大规模数据集的聚类。并且易于实现和解释，对于凸形簇效果较

9.2 模型的缺点

- DBSCAN 聚类对于高维数据和具有不同密度的簇效果可能不佳。且对于参数的选择比较敏感，如邻域半径和最小密度阈值的设定。
- 层次分析法在构建层次结构和确定权重时可能存在主观性和不确定性。对于层次结构的设计和权重的确定需要较多的专业知识和经验。
- ADF 检验只能检验序列的一阶差分平稳性，对于高阶差分的平稳性检验不适用。且在样本量较小的情况下可能存在一定的不稳定性。
- Pettitt 检验对于突变点的定位可能不够精确。且在样本量较小的情况下可能存在一定的不稳定性。
- K-means 聚类需要事先指定簇的个数，对初始簇中心的选择敏感。且对异常值和噪声点敏感，对非凸形簇效果不佳。

参考文献

[1] 周志华, 王珏. 机器学习及其应用 2009[M]. [出版地不详]: 机器学习及其应用 2009, 2009.

[2] 周英, 卓金武, 卞月青. 大数据挖掘[M]. [出版地不详]: 大数据挖掘, 2016.

[3] 卓金武. MATLAB 在数学建模中的应用[M]. [出版地不详]: MATLAB 在数学建模中的应用, 2014.

[4] TIAN X, CHEN D, YAN X, et al. Estimation method of intersection signal cycle based on empirical data[J]. Journal of Transportation Engineering Part A Systems, 2021, 147(3):04021001.

[5] 陈鹏, 闫聪, 鲁光泉, 等. 一种基于浮动车轨迹数据的交叉口信号周期估计方法[Z]. [出版地不详: 出版者不详], 2019.

[6] 司守奎, 孙玺菁. 数学建模算法与应用[M]. [出版地不详]: 数学建模算法与应用, 2011.

附录 A　代码文件列表

表 1　**Add caption**

文件名	文件描述
pro1.m	问题 1 求解
pro2.m	问题 2 求解
pro3.m	问题 3 求解
pro4.m	问题 4 求解
plots-pro4.m	问题四十字路口绘制

附录 B　代码

pro1.m

```
clc
clear
close all

%%

% 读取数据
data = readtable('A1.csv');
time = data.time;
vehicle_ids = unique(data.vehicle_id);
x = data.x;
y = data.y;
figure(6)
plot(x,y,'o')
xlabel('横坐标');
ylabel('纵坐标');
title(['车辆轨迹']);

line = mode(x);  %等红灯的线
waiting_vehicel = [];
flag = 0;
green_start_last = 0;
T_red_start_last = 0;
% 初始化周期参数
T_signal = []; % 信号灯周期  red + green
T_red = []; % 红灯时长
T_green = []; % 绿灯时长
T_circle = [];% 组合
```

```
29
30 % 设定速度阈值，用于判断车辆是否停下
31 speed_threshold = 0.5; % 例如，0.5 m/s
32
33 % 对每辆车进行分析
34 for i = 1:length(vehicle_ids)
35     idx = data.vehicle_id == vehicle_ids(i);  %找出特定编号的车
36     vehicle_x = x(idx);
37     vehicle_y = y(idx);
38     vehicle_time = time(idx);
39     % 计算速度
40     vehicle_dx = diff(vehicle_x);
41     vehicle_dy = diff(vehicle_y);
42     vehicle_dt = diff(vehicle_time);
43     vehicle_speed = sqrt(vehicle_dx.^2 + vehicle_dy.^2) ./ vehicle_dt;
44
45     for j = 1:length(vehicle_time)
46         % 创建逻辑数组，标记同时满足两个条件的行
47         logical_idx = (data.time == vehicle_time(j)) & (data.vehicle_id ==
    vehicle_ids(i));
48
49         % 使用 find 函数找到这些逻辑为真的行的索引
50         row_indices = find(logical_idx);
51         if j == 1
52             data.v(row_indices) = vehicle_speed(1);
53         else
54         data.v(row_indices) = vehicle_speed(j-1);
55         end
56     end
57
58     % 找到车辆停止等待红灯的点
59     stop_indices = find(vehicle_speed < speed_threshold);
60
61     % 如果没有找到停止的点，则跳过此车辆
62     if isempty(stop_indices)
63         continue;
64     end
65
66     % 如果车距离信号灯太远，则跳过
67     if vehicle_x(stop_indices(1)) <(line-10) && vehicle_x(stop_indices(1)) >(line
    +10)
68         continue;
69     end
70     % 假设第一辆停下的车开始的时间为红灯开始时间
71
72     T_red_start = vehicle_time(stop_indices(1));
73
```

```
    % 假设车辆再次开始移动时为绿灯开始时间
    green_start = vehicle_time(stop_indices(end) + 1);


    %淘汰比第一辆停下的车晚来的车
    if  i >1 && T_red_start > T_red_start_last && T_red_start < green_start_last
        continue;
    end
    %停车时间
    T_red_current = green_start - T_red_start;
    %总周期T - 停车时间
    T_green_current = green_start - green_start_last - T_red_current;
    % 筛选每个周期第一辆停下的车
    % T_red = [T_red, T_red_current];
    % T_green = [T_green, T_green_current];
    % 如果停止时间太短的点，则跳过此车辆
    if  length(stop_indices)<10
        green_start = green_start_last;
        continue;
    end

    if  i >1 && T_red_start < T_red_start_last
        T_signal = [T_signal(1:end-1) , green_start - green_start_last];
       T_red = [T_red(1:end-1), T_red_current];
       T_green = [T_green(1:end-1), T_green_current];
       waiting_vehicel = [waiting_vehicel(1:end-1),vehicle_ids(i)];
    else
    % 更新红灯时长（第一个停下到再次移动的时间差）
    T_signal = [T_signal , green_start - green_start_last];
    T_red = [T_red, T_red_current];
    % 更新绿灯时长（如果有多辆车停下，取最后一个停下到第一个再次移动的时间差）
    T_green = [T_green, T_green_current];
    waiting_vehicel = [waiting_vehicel,vehicle_ids(i)];
    end

    % 更新信号灯周期的开始时间为下一个红灯的开始时间
    green_start_last = green_start;
    T_red_start_last = T_red_start;
end

%每个周期第一辆停下的车所停时间
T_waiting_vehicel = horzcat(waiting_vehicel',T_red');

% 估计信号灯周期
t =zeros(length(T_red),1);
for i = 1:length(T_red )
    t(i) = i;
```

```
end
T_circle = [t,(T_signal)',T_red',T_green',waiting_vehicel'];
T_circle = array2table(T_circle);

%画出每个周期第一辆停下的车的轨迹图
figure(1)
    hold on;
    n =size(T_waiting_vehicel);
    for j = 1:n(1)
        vehicle_data = data(data.vehicle_id == T_waiting_vehicel(j,1), :);
        plot(vehicle_data.time, vehicle_data.x, '-', 'DisplayName', ['Vehicle '
    num2str(T_waiting_vehicel(j,1))]);
    end

    xlabel('时间(s)');
    ylabel('车辆坐标(m)');
    title(['车辆坐标-时间图']);
    legend('show');

    hold off;
% 初始化 C 为一个空矩阵
figure(5)
    hold on;
    n =size(T_waiting_vehicel);
    for j = 1:n(1)
        vehicle_data = data(data.vehicle_id == T_waiting_vehicel(j,1), :);
        plot(vehicle_data.time, vehicle_data.v, '-', 'DisplayName', ['Vehicle '
    num2str(T_waiting_vehicel(j,1))]);
    end

    xlabel('时间(s)');
    ylabel('车辆速度（m/s）');
    title(['车辆时间-速度图']);
      ylim([-5,20]);
    legend('show');

    hold off;
C = [];

% 遍历 A 和 B 的每个元素
for i = 1:length(T_red)
    % 将 A(i) 个 1 添加到 C 中
    C = [C; ones(T_green(i), 1)];
    % 将 B(i) 个 0 添加到 C 中（如果 B(i) 不为 0）
    C = [C; zeros(T_red(i), 1)];

end
```

```
166
167 %处理前
168 figure(2)
169 plot(C)
170
171 ylim([-0.5 1.5]);
172 xlabel('时间');
173 ylabel('value')
174 title(['红绿灯周期图，1-绿，0-红（处理前）']);
175 hold off
176 %%
177
178 % 数据处理->删除离群值
179 T_circle = rmoutliers(T_circle,"DataVariables","T_circle2");
180 T_circle = table2array(T_circle);
181 T_signal = T_circle(:,2);
182 T_red = (T_circle(:,3));
183 T_green = (T_circle(:,4));
184
185 figure(8)
186     hold on;
187     n =size(T_circle);
188     for j = 1:n(1)
189         vehicle_data = data(data.vehicle_id == T_circle(j,5), :);
190         plot(vehicle_data.time, vehicle_data.v, '-', 'DisplayName', ['Vehicle '
    num2str(T_waiting_vehicel(j,1))]);
191     end
192
193     xlabel('时间(s)');
194     ylabel('车辆速度（m/s）');
195     title(['车辆时间-速度图(processed)']);
196       ylim([-5,20]);
197     legend('show');
198
199     hold off;
200 % 显示结果
201 disp(['估计的信号灯周期（秒）: ', num2str(mean(T_signal(:,1)))]);
202 disp(['估计的红灯时长（秒）: ', num2str(mode(T_red))]);
203 disp(['估计的绿灯时长（秒）: ', num2str(mean(T_signal(:,1))-mode(T_red))]);
204
205 C = [];
206
207 % 遍历 A 和 B 的每个元素
208 for i = 1:length(T_red)
209     % 将 A(i) 个 1 添加到 C 中
210     C = [C; ones(T_green(i), 1)];
211     % 将 B(i) 个 0 添加到 C 中（如果 B(i) 不为 0）
```

```
    C = [C; zeros(T_red(i), 1)];

end
figure(3)
plot(C)

ylim([-0.5 1.5]);
xlabel('时间');
ylabel('value')
title(['红绿灯周期图，1-绿，0-红(处理后)']);
hold off


%%
% 画图
figure(4)
data1 = table2array(data);
num_vehicle_ids = length(vehicle_ids);
max_legend_entries = 21; % 每个图例中的最大条目数

num_subplots = ceil(num_vehicle_ids / max_legend_entries); % 计算需要的子图数量

for subplot_index = 1:num_subplots
    start_index = (subplot_index - 1) * max_legend_entries + 1;
    end_index = min(subplot_index * max_legend_entries, num_vehicle_ids);

    subplot(num_subplots, 1, subplot_index);
    hold on;

    for i = start_index:end_index
        vehicle_data = data(data.vehicle_id == vehicle_ids(i), :);
        plot(vehicle_data.time, vehicle_data.x, '-', 'DisplayName', ['Vehicle '
    num2str(vehicle_ids(i))]);
    end

    xlabel('时间');
    ylabel('车辆坐标');
    title(['车辆坐标-时间图(第' num2str(subplot_index) '组']);
    %legend('show');

    hold off;
end
```

pro2.m

```
clc
clear
close all
```

```
%%

% 读取数据
data = readtable('A1.csv');
time = data.time;
vehicle_ids = unique(data.vehicle_id);
x = data.x;
y = data.y;
figure(6)
plot(x,y,'o')
xlabel('横坐标');
ylabel('纵坐标');
title(['车辆轨迹']);
line = mode(x);  %等红灯的线
waiting_vehicel = [];
flag = 0;
green_start_last = 0;
T_red_start_last = 0;
% 初始化周期参数
T_signal = []; % 信号灯周期  red + green
T_red = []; % 红灯时长
T_green = []; % 绿灯时长
T_circle = [];% 组合

% 设定速度阈值，用于判断车辆是否停下
speed_threshold = 0.5; % 例如，0.5 m/s

% 对每辆车进行分析
for i = 1:length(vehicle_ids)
    idx = data.vehicle_id == vehicle_ids(i);  %找出特定编号的车
    vehicle_x = x(idx);
    vehicle_y = y(idx);
    vehicle_time = time(idx);
    % 计算速度
    vehicle_dx = diff(vehicle_x);
    vehicle_dy = diff(vehicle_y);
    vehicle_dt = diff(vehicle_time);
    vehicle_speed = sqrt(vehicle_dx.^2 + vehicle_dy.^2) ./ vehicle_dt;

    % 找到车辆停止等待红灯的点
    stop_indices = find(vehicle_speed < speed_threshold);
    %停车时间
    % stop_time = vehicle_time(stop_indices)
    %flag =flag +1

    % 如果没有找到停止的点，则跳过此车辆
```

```
51
52      if isempty(stop_indices)
53          continue;
54      end
55
56      % 如果车距离信号灯太远，则跳过
57      if vehicle_x(stop_indices(1)) <(line-10) && vehicle_x(stop_indices(1)) >(line
        +10)
58          continue;
59      end
60      % 假设第一辆停下的车开始的时间为红灯开始时间
61
62      T_red_start = vehicle_time(stop_indices(1));
63
64      % 假设车辆再次开始移动时为绿灯开始时间
65      green_start = vehicle_time(stop_indices(end) + 1);
66
67
68      %淘汰比第一辆停下的车晚来的车
69      if  i >1 && T_red_start > T_red_start_last && T_red_start < green_start_last
70          continue;
71      end
72
73      T_red_current = green_start - T_red_start;
74      T_green_current = green_start - green_start_last - T_red_current;
75      % 筛选每个周期第一辆停下的车
76      % T_red = [T_red, T_red_current];
77      % T_green = [T_green, T_green_current];
78      % 如果停止时间太短的点，则跳过此车辆
79      if  length(stop_indices)<10
80          green_start = green_start_last;
81          continue;
82      end
83
84      for j = 1:length(vehicle_time)
85          % 创建逻辑数组，标记同时满足两个条件的行
86          logical_idx = (data.time == vehicle_time(j)) & (data.vehicle_id ==
        vehicle_ids(i));
87
88          % 使用 find 函数找到这些逻辑为真的行的索引
89          row_indices = find(logical_idx);
90          if j == 1
91              data.v(row_indices) = vehicle_speed(1);
92          else
93          data.v(row_indices) = vehicle_speed(j-1);
94          end
95      end
```

```
96
97      if  i >1 && T_red_start < T_red_start_last
98          T_signal = [T_signal(1:end-1) , green_start - green_start_last];
99         T_red = [T_red(1:end-1), T_red_current];
100        T_green = [T_green(1:end-1), T_green_current];
101        waiting_vehicel = [waiting_vehicel(1:end-1),vehicle_ids(i)];
102     else
103     % 更新红灯时长（第一个停下到再次移动的时间差）
104     T_signal = [T_signal , green_start - green_start_last];
105
106     T_red = [T_red, T_red_current];
107     % 更新绿灯时长（如果有多辆车停下，取最后一个停下到第一个再次移动的时间差）
108     T_green = [T_green, T_green_current];
109
110
111     waiting_vehicel = [waiting_vehicel,vehicle_ids(i)];
112     end
113
114     % 更新信号灯周期的开始时间为下一个红灯的开始时间
115     green_start_last = green_start;
116     T_red_start_last = T_red_start;
117
118 end
119
120 %每个周期第一辆停下的车所停时间
121 T_waiting_vehicel = horzcat(waiting_vehicel',T_red');
122
123 % 估计信号灯周期
124
125 t =zeros(length(T_red),1);
126 for i = 1:length(T_red )
127     t(i) = i;
128 end
129 T_circle = [t,(T_signal)',T_red',T_green'];
130 T_circle = array2table(T_circle);
131
132 %画出每个周期第一辆停下的车的轨迹图
133 figure(1)
134     hold on;
135     n =size(T_waiting_vehicel);
136     for j = 1:n(1)
137         vehicle_data = data(data.vehicle_id == T_waiting_vehicel(j,1), :);
138         plot(vehicle_data.time, vehicle_data.x, '-', 'DisplayName', ['Vehicle '
    num2str(T_waiting_vehicel(j,1))]);
139     end
140
141     xlabel('时间(s)');
```

```
    ylabel('车辆坐标(m)');
    title(['车辆坐标-时间图']);
    legend('show');

    hold off;
%速度-时间图
figure(5)
    hold on;
    n =size(T_waiting_vehicel);
    for j = 1:n(1)
        vehicle_data = data(data.vehicle_id == T_waiting_vehicel(j,1), :);
        plot(vehicle_data.time, vehicle_data.v, '-', 'DisplayName', ['Vehicle '
    num2str(T_waiting_vehicel(j,1))]);
    end

    xlabel('时间(s)');
    ylabel('车辆速度（m/s）');
    ylim([-5,20]);
    title(['车辆时间-速度图']);
    legend('show');

    hold off;

    % 初始化 C 为一个空矩阵
C = [];

% 遍历 A 和 B 的每个元素
for i = 1:length(T_red)
    % 将 A(i) 个 1 添加到 C 中
    C = [C; ones(T_green(i), 1)];
    % 将 B(i) 个 0 添加到 C 中（如果 B(i) 不为 0）
    C = [C; zeros(T_red(i), 1)];

end

%处理前
figure(2)
plot(C)

ylim([-0.5 1.5]);
xlabel('时间');
ylabel('value')
title(['红绿灯周期图，1-绿，0-红（处理前）']);
hold off
%%

% 数据处理->删除离群值
```

```
T_circle = rmoutliers(T_circle,"quartiles", ...
        "DataVariables",["T_circle2","T_circle3","T_circle4"]);
T_circle = table2array(T_circle);
T_signal = T_circle(:,2);
T_red = (T_circle(:,3));
T_green = (T_circle(:,4));

% 显示结果
disp(['估计的信号灯周期（秒）: ', num2str(mean(T_signal(:,1)))]);
disp(['估计的红灯时长（秒）: ', num2str(mode(T_red))]);
disp(['估计的绿灯时长（秒）: ', num2str(mean(T_signal(:,1))-mode(T_red))]);

C = [];

% 遍历 A 和 B 的每个元素
for i = 1:length(T_red)
    % 将 A(i) 个 1 添加到 C 中
    C = [C; ones(T_green(i), 1)];
    % 将 B(i) 个 0 添加到 C 中（如果 B(i) 不为 0）
    C = [C; zeros(T_red(i), 1)];

end
figure(3)
plot(C)

ylim([-0.5 1.5]);
xlabel('时间');
ylabel('value')
title(['红绿灯周期图，1-绿，0-红(处理后)']);
hold off


%%
% 画图
figure(4)
data1 = table2array(data);
num_vehicle_ids = length(vehicle_ids);
max_legend_entries = 21; % 每个图例中的最大条目数

num_subplots = ceil(num_vehicle_ids / max_legend_entries); % 计算需要的子图数量

for subplot_index = 1:num_subplots
    start_index = (subplot_index - 1) * max_legend_entries + 1;
    end_index = min(subplot_index * max_legend_entries, num_vehicle_ids);

    subplot(num_subplots, 1, subplot_index);
    hold on;
```

```
    for i = start_index:end_index
        vehicle_data = data(data.vehicle_id == vehicle_ids(i), :);
        plot(vehicle_data.time, vehicle_data.y, '-', 'DisplayName', ['Vehicle '
    num2str(vehicle_ids(i))]);
    end

    xlabel('时间');
    ylabel('车辆坐标');
    title(['车辆坐标-时间图(第' num2str(subplot_index) '组']);
    %legend('show');

    hold off;
end
```

pro3.m

```
clc
clear
close all

%%

% 读取数据
 data = readtable('C2.csv');
% data = readtable('C2.csv');
 %data = readtable('C3.csv');
% data = readtable('C4.csv');
% data = readtable('C5.csv');
%data = readtable('C2.csv');
%%



time = data.time;
vehicle_ids = unique(data.vehicle_id);
x = data.x;
y = data.y;
figure(7)
plot(x,y,'o')
xlabel('横坐标');
ylabel('纵坐标');
title(['车辆轨迹']);

linex = mode(x);  %等红灯的线
liney = mode(y);  %等红灯的线
waiting_vehicel = [];
flag = 0;
```

```
green_start_last = 0;
T_red_start_last = 0;
% 初始化周期参数
T_signal = []; % 信号灯周期  red + green
T_red = []; % 红灯时长
T_green = []; % 绿灯时长
T_circle = [];% 组合
T_green_record = [];
% 设定速度阈值，用于判断车辆是否停下
speed_threshold = 0.1; % 例如，0.5 m/s

% 对每辆车进行分析
for i = 1:length(vehicle_ids)
    idx = data.vehicle_id == vehicle_ids(i);  %找出特定编号的车
    vehicle_x = x(idx);
    vehicle_y = y(idx);
    vehicle_time = time(idx);
    % 计算速度
    vehicle_dx = diff(vehicle_x);
    vehicle_dy = diff(vehicle_y);
    vehicle_dt = diff(vehicle_time);
    vehicle_speed = sqrt(vehicle_dx.^2 + vehicle_dy.^2) ./ vehicle_dt;

    for j = 1:length(vehicle_time)
        % 创建逻辑数组，标记同时满足两个条件的行
        logical_idx = (data.time == vehicle_time(j)) & (data.vehicle_id ==
    vehicle_ids(i));

        % 使用 find 函数找到这些逻辑为真的行的索引
        row_indices = find(logical_idx);
        if j == 1
            data.v(row_indices) = vehicle_speed(1);
        else
        data.v(row_indices) = vehicle_speed(j-1);
        end
    end

    % 找到车辆停止等待红灯的点
    stop_indices = find(vehicle_speed < speed_threshold);

    % 如果没有找到停止的点，则跳过此车辆
    if isempty(stop_indices)
        continue;
    end

    % 如果车距离信号灯太远，则跳过
    if abs(vehicle_x(stop_indices(1)) -linex)>20 && abs(vehicle_y(stop_indices(1))
```

```
-liney)>20
        continue;
    end
    % 假设第一辆停下的车开始的时间为红灯开始时间

    T_red_start = vehicle_time(stop_indices(1));

    % 假设车辆再次开始移动时为绿灯开始时间
    green_start = vehicle_time(stop_indices(end) + 1);


    %淘汰比第一辆停下的车晚来的车
    if  i >1 && T_red_start > T_red_start_last && T_red_start < green_start_last
        continue;
    end
    %停车时间
    T_red_current = green_start - T_red_start;
    %总周期T - 停车时间
    T_green_current = green_start - green_start_last - T_red_current;
    % 筛选每个周期第一辆停下的车
    % T_red = [T_red, T_red_current];
    % T_green = [T_green, T_green_current];
    % 如果停止时间太短的点，则跳过此车辆
    if  length(stop_indices)<2
        green_start = green_start_last;
        continue;
    end

    if  i >1 && T_red_start < T_red_start_last
        T_signal = [T_signal(1:end-1) , green_start - green_start_last];   %总周期T
 = T_green - 上一次的T_green
       T_red = [T_red(1:end-1), T_red_current];
       T_green = [T_green(1:end-1), T_green_current];
       T_green_record = [T_green_record(1:end-1),green_start];
       waiting_vehicel = [waiting_vehicel(1:end-1),vehicle_ids(i)];
    else
    % 更新红灯时长（第一个停下到再次移动的时间差）
    T_signal = [T_signal , green_start - green_start_last];
    T_red = [T_red, T_red_current];
    % 更新绿灯时长（如果有多辆车停下，取最后一个停下到第一个再次移动的时间差）
    T_green = [T_green, T_green_current];
     T_green_record = [T_green_record,green_start];
    waiting_vehicel = [waiting_vehicel,vehicle_ids(i)];
    end

    % 更新信号灯周期的开始时间为下一个红灯的开始时间
    green_start_last = green_start;
```

```
    T_red_start_last = T_red_start;
end

%每个周期第一辆停下的车所停时间
T_waiting_vehicel = horzcat(waiting_vehicel',T_red');

% 估计信号灯周期
t =zeros(length(T_red),1);
for i = 1:length(T_red )
    t(i) = i;
end
T_circle = [t,(T_signal)',T_red',T_green',T_green_record'];
T_circle = array2table(T_circle);

%画出每个周期第一辆停下的车的轨迹图
figure(1)
    hold on;
    n =size(T_waiting_vehicel);
    for j = 1:n(1)
        vehicle_data = data(data.vehicle_id == T_waiting_vehicel(j,1), :);
        plot(vehicle_data.time, vehicle_data.x, '-', 'DisplayName', ['Vehicle '
    num2str(T_waiting_vehicel(j,1))]);
    end

    xlabel('时间(s)');
    ylabel('车辆坐标(m)');
    title(['车辆坐标-时间图']);
    legend('show');

    hold off;
% 初始化 C 为一个空矩阵
figure(6)
    hold on;
    n =size(T_waiting_vehicel);
    for j = 1:n(1)
        vehicle_data = data(data.vehicle_id == T_waiting_vehicel(j,1), :);
        plot(vehicle_data.time, vehicle_data.v, '-', 'DisplayName', ['Vehicle '
    num2str(T_waiting_vehicel(j,1))]);
    end

    xlabel('时间(s)');
    ylabel('车辆速度（m/s）');
    title(['车辆时间-速度图']);
      ylim([-5,20]);
    legend('show');

    hold off;
```

```
C = [];

% 遍历 A 和 B 的每个元素
for i = 1:length(T_red)
    % 将 A(i) 个 1 添加到 C 中
    C = [C; ones(T_green(i), 1)];
    % 将 B(i) 个 0 添加到 C 中（如果 B(i) 不为 0）
    C = [C; zeros(T_red(i), 1)];

end

%处理前
figure(2)
plot(C)

ylim([-0.5 1.5]);
xlabel('时间');
ylabel('value')
title(['红绿灯周期图，1-绿，0-红（处理前）']);
hold off
%%

% 数据处理->删除离群值
T_circle = rmoutliers(T_circle,"quartiles", ...
        "DataVariables",["T_circle2","T_circle3","T_circle4"]);
T_circle = table2array(T_circle);
T_signal = T_circle(:,2);
T_red = (T_circle(:,3));
T_green = (T_circle(:,4));

% 显示结果
disp(['估计的信号灯周期（秒）: ', num2str(mean(T_signal(:,1)))]);
disp(['估计的红灯时长（秒）: ', num2str(max(T_red))]);
disp(['估计的绿灯时长（秒）: ', num2str(mean(T_signal(:,1))-mode(T_red))]);

C = [];

% 遍历 A 和 B 的每个元素
for i = 1:length(T_red)
    % 将 A(i) 个 1 添加到 C 中
    C = [C; ones(T_green(i), 1)];
    % 将 B(i) 个 0 添加到 C 中（如果 B(i) 不为 0）
    C = [C; zeros(T_red(i), 1)];

end
figure(3)
plot(C)
```

```
215
216 ylim([-0.5 1.5]);
217 xlabel('时间');
218 ylabel('value')
219 title(['红绿灯周期图，1-绿，0-红(处理后)']);
220 hold off
221
222
223 %% Pettitt突变点检测
224 figure(8)
225 plot(T_circle(:,5),T_circle(:,2));
226 xlabel('时间');
227 ylabel('信号灯周期时长')
228 ylim([20,200]);
229 title(['时间-信号灯周期时长']);
230 T_sig = T_circle(:, 2);  % 提取红灯时间序列
231
232 % 初始化存储结果的变量
233 change_points = [];
234
235 for i = 2:length(T_sig)-2
236     if abs(T_sig(i) - T_sig(i-1)) > 50 && abs(T_sig(i) - T_sig(i+1)) < 10 && abs(
    T_sig(i) - T_sig(i+2)) < 10
237         change_points = [change_points, [T_circle(i,5); T_circle(i,3);T_circle(i,4)
    ;T_circle(i,2)]];
238     end
239 end
240 k = size(change_points);
241 % 输出检测结果
242 if ~isempty(change_points)
243     fprintf('检测到以下突变点：\n');
244
245     for i = 1:k(2)
246         fprintf('突变点时间：%d红灯时间：%d\n绿灯时间：%d\n 周期时间：%d\n',
    change_points(1,i), change_points(2,i), change_points(3,i),change_points(4,i));
247     end
248 else
249     fprintf('未检测到符合条件的突变点。\n');
250 end
251 %%
252 %{
253 % 获取信号灯周期 T_signal
254 T_signal = T_circle(:, 2);
255
256 T_change = [];  % 存储检测到的周期变化信息
257
258 n = length(T_signal);  % 计算信号灯周期的个数
```

```
critical_value = 0.05;  % 设置显著性水平对应的临界值

% 初始化变量
tau = zeros(n, 1);  % 存储Pettitt检测的统计量
K = zeros(n, 1);    % 存储Pettitt检测的累积量
T_flag = 0;  % 记录累积的突变点位置

% 计算Pettitt检测的统计量和累积量
for t = 2:n
    if t > length(T_signal)
        break;
    end

    K(t) = max(T_signal(1:t)) - min(T_signal(1:t));
    tau(t) = sum(sign(T_signal(t) - T_signal(1:t-1)));

    % 计算Pettitt检测的统计量的绝对值
    abs_tau = abs(tau);

    % 计算Pettitt检测的检测量
    U_t = abs_tau - max(abs_tau) / 2;

    % 找到突变点的位置
    [~, idx] = max(U_t(1:t));
    T_flag = T_flag + idx;

    % 输出检测结果
    if U_t(idx) > critical_value
        T_change = [T_change, [T_circle(T_flag, 5); T_red(T_flag); T_green(T_flag);
     T_signal(idx)]];
        fprintf('发现信号灯周期发生突变！\n');
        fprintf('突变发生在第 %d s。\n', T_circle(T_flag, 5));
        fprintf('新周期参数为:\n T_signal(%d) = %f\n', T_flag, T_signal(idx));
        fprintf('红灯时长为: T_red(%d) = %f\n', T_flag, T_red(T_flag));
        fprintf('绿灯时长为: T_green(%d) = %f\n', T_flag, T_green(T_flag));
        fprintf('旧周期参数为:\n T_signal(%d) = %f\n', idx-1, T_signal(idx-1));
        fprintf('红灯时长为: T_red(%d) = %f\n', T_flag-1, T_red(T_flag-1));
        fprintf('绿灯时长为: T_green(%d) = %f\n', T_flag-1, T_green(T_flag-1));

        % 更新信号灯周期序列，继续检测后续的周期变化
        T_signal = T_signal(idx+1:end);
        n = length(T_signal);
        tau = zeros(n, 1);
        K = zeros(n, 1);

    end
end
```

```
%}
%%
% 画图
figure(5)
data1 = table2array(data);
num_vehicle_ids = length(vehicle_ids);
max_legend_entries = 21; % 每个图例中的最大条目数

num_subplots = ceil(num_vehicle_ids / max_legend_entries); % 计算需要的子图数量

for subplot_index = 1:num_subplots
    start_index = (subplot_index - 1) * max_legend_entries + 1;
    end_index = min(subplot_index * max_legend_entries, num_vehicle_ids);

    subplot(num_subplots, 1, subplot_index);
    hold on;

    for i = start_index:end_index
        vehicle_data = data(data.vehicle_id == vehicle_ids(i), :);
        plot(vehicle_data.time, vehicle_data.x, '-', 'DisplayName', ['Vehicle '
    num2str(vehicle_ids(i))]);
    end

    xlabel('时间');
    ylabel('车辆坐标');
    title(['车辆坐标-时间图(第' num2str(subplot_index) '组']);
    %legend('show');

    hold off;
end
```

pro4.m

```
clear;
clc
data = readtable("D.csv");

% 删除异常值
data(data.vehicle_id == 5428,:) = [];
 data(data.vehicle_id == 5431,:) = [];
  data(data.vehicle_id == 5468,:) = [];
%% 参数
speed_threshold = 1; % 设定速度变化阈值（单位：m/s）
heading_change_threshold = 30; % 设定航向角变化阈值（单位：度）
buffer_radius = 10; % 缓冲区半径，可以根据实际情况调整
stop_speed_threshold = 0.1; % 停止速度阈值（单位：m/s），用于确定车辆是否停止
threshold = 10; % 周期时长变化阈值

```

```
%% 聚类数据准备
% 特征提取
% 提取唯一的vehicle_id
unique_vehicle_ids = unique(data.vehicle_id);

% 初始化cell数组来存储每个vehicle_id的候选交叉口位置和停止/启动事件
intersection_candidates_cell = cell(numel(unique_vehicle_ids), 1);
stop_start_events_cell = cell(numel(unique_vehicle_ids), 1);

% 初始化停止和启动事件数组
all_stop_events = [];
all_start_events = [];
%停下来的车及其停止启动时间
all_start_car = [];
t = zeros(size(unique_vehicle_ids,1),6);

% 循环遍历每个vehicle_id
for i = 1:numel(unique_vehicle_ids)
    current_vehicle_id = unique_vehicle_ids(i);

    % 提取当前vehicle_id的所有数据
    vehicle_indices = find(data.vehicle_id == current_vehicle_id);
    vehicle_data = data(vehicle_indices, :);

    % 提取x和y坐标以及时间
    time_vector = vehicle_data.time;
    x_vector = vehicle_data.x;
    y_vector = vehicle_data.y;

    % 计算速度
    dx = diff(x_vector);
    dy = diff(y_vector);
    dt = diff(time_vector);
    speeds = sqrt(dx.^2 + dy.^2) ./ dt;
    speeds = [NaN; speeds]; % 在速度数组前添加一个NaN以匹配原始数据长度

    % for j = 1:length(time_vector)
    %     % 创建逻辑数组，标记同时满足两个条件的行
    %     logical_idx = (data.time == time_vector(j)) & (data.vehicle_id ==
    unique_vehicle_ids(i));
    %
    %     % 使用 find 函数找到这些逻辑为真的行的索引
    %     row_indices = find(logical_idx);
    %
    %     data.v(row_indices) = speeds(j);
    %
    % end
```

```
62
63  % 识别速度变化点
64  %加速
65  speed_change_indices = find(abs(diff(speeds)) > speed_threshold);
66
67  % 计算航向角
68  headings = atan2(diff(y_vector), diff(x_vector));
69  headings_deg = rad2deg(headings);
70
71  % 识别转向事件
72  heading_change_indices = find(abs(diff(headings_deg)) >
    heading_change_threshold);
73
74  % 交叉验证：寻找同时满足速度变化和转向事件的点
75  intersection_indices = intersect(speed_change_indices, heading_change_indices);
76
77  % 提取候选交叉口位置
78  if ~isempty(intersection_indices)
79      intersection_candidates = [time_vector(intersection_indices + 1), x_vector(
    intersection_indices + 1), y_vector(intersection_indices + 1)];
80  else
81      intersection_candidates = [];
82  end
83
84  % 存储候选交叉口位置到cell数组
85  %intersection_candidates_cell{i} = intersection_candidates;
86
87  % 特征向量
88  t(i,:) = [min(x_vector),max(x_vector),min(y_vector),max(y_vector),max(
    headings_deg),min(headings_deg)];
89  % 识别停止和启动事件
90  stop_idx = find(speeds < stop_speed_threshold);
91  start_idx = find(diff([false; speeds >= stop_speed_threshold]));
92  if isempty(stop_idx)
93      continue;
94  end
95  stop_idx = stop_idx(stop_idx < length(time_vector) - 1); % 过滤出有效的stop_idx
96  start_idx = start_idx(start_idx < length(time_vector) - 1); % 过滤出有效的
    start_idx
97
98  %
99  % red_start = time_vector(stop_idx(1));
100 %
101 % % 假设车辆再次开始移动时为绿灯开始时间
102 % green_start = time_vector(stop_idx(end) + 1);
103
104 % 提取停止和启动事件的时间、x坐标和y坐标
```

```
    stop_events = [time_vector(stop_idx + 1), x_vector(stop_idx + 1), y_vector(stop_idx + 1)];
    start_events = [time_vector(start_idx + 1), x_vector(start_idx + 1), y_vector(start_idx + 1)];

    % 确定停止和启动事件是否在候选交叉口缓冲区内
    if ~isempty(intersection_candidates)
        intersection_x = intersection_candidates(:, 2);
        intersection_y = intersection_candidates(:, 3);

        in_intersection_stop_idx = zeros(size(stop_events, 1), 1);
        for j = 1:size(stop_events, 1)
            if sqrt((stop_events(j, 2) - intersection_x).^2 + (stop_events(j, 3) - intersection_y).^2) <= buffer_radius^2
                in_intersection_stop_idx(j) = 1;
            end
        end

        % 在缓冲区半径内检查启动事件
        in_intersection_start_idx = zeros(size(start_events, 1), 1);
        for j = 1:size(start_events, 1)
            if sqrt((start_events(j, 2) - intersection_x).^2 + (start_events(j, 3) - intersection_y).^2) <= buffer_radius^2
                in_intersection_start_idx(j) = 1;
            end
        end
        % 将整数索引数组转换为逻辑数组
        logical_stop_idx = in_intersection_stop_idx ~= 0;
        logical_start_idx = in_intersection_start_idx ~= 0;
        % 提取交叉口附近的停止和启动事件
        in_intersection_stop_events = stop_events(logical_stop_idx, :);
        in_intersection_start_events = start_events(logical_start_idx, :);

        % 将当前车辆的停止和启动事件添加到总数组中
        %all_stop_events = [all_stop_events; in_intersection_stop_events(:, 1)]; % 只存储时间
        %all_start_events = [all_start_events; in_intersection_start_events(:, 1)]; % 只存储时间
        if ~isempty(in_intersection_stop_events)
            all_start_car = [all_start_car;[unique_vehicle_ids(i), in_intersection_stop_events(1,1),in_intersection_stop_events(end,1)+1]];
        end
        % 存储当前车辆的停止和启动事件到cell数组
        %stop_start_events_cell{i} = {in_intersection_stop_events, in_intersection_start_events};
    end
end
```

```
%% k聚类


% k_means
means = mean(t); %求均值
stds = std(t);   %求标准差
data1 = (t-means)./stds; %标准化

%判断k值
K = 15;A = zeros(K,2);
for k=1:K
    [~,~,sumd,~] = kmeans(data1,k);
    sse=sum(sumd.^2);
    A(k,1) = k;
    A(k,2) = sse;
end
figure(1);
plot(A(:,1),A(:,2));
hold on
plot(A(:,1),A(:,2),'ro');
title('不同k值聚类偏差图');
xlabel('k值');ylabel('簇内误差平方和');

%统计项目数
k = 12;
[idx,C,sumd,D] = kmeans(data1,k);

%%
%聚类 12图

% 创建 4x3 的子图网格
nRows = 4;
nCols = 3;
figure; % 创建一个新的图形窗口

% 初始化子图索引
subplotIndex = 1;

for j = 1:12 % 假设您只需要处理前12个索引
    Idx = find(idx == j); % qf_Idx 需要在此前定义，并且其长度至少为12
    unique_vehicle_id = unique_vehicle_ids(Idx);

    for i = 1:numel(unique_vehicle_id)
        current_vehicle_id = unique_vehicle_id(i);

        % 提取当前 vehicle_id 的所有数据
        vehicle_indices = find(data.vehicle_id == current_vehicle_id);
```

```
        vehicle_data = data(vehicle_indices, :);

        % 提取 x 和 y 坐标以及时间（时间在这里可能不需要用于绘图）
        x_vector = vehicle_data.x;
        y_vector = vehicle_data.y;

        % 创建或选择子图
        subplot(nRows, nCols, subplotIndex);
        plot(x_vector, y_vector, '.');

        title(sprintf('original X = %s Y = %s', num2str(x_vector(1)),num2str(
    y_vector(1)))); % 添加标题
        % xlim([xmin xmax]); % xmin 和 xmax 需要定义为您想要显示的范围
        % ylim([ymin ymax]); % ymin 和 ymax 同样需要定义
         hold on
        % 更新子图索引


        % 如果subplotIndex超过了子图的总数，则退出循环
        if subplotIndex > nRows * nCols
            break;
        end

        % 强制更新图形窗口


        % 暂停0.1秒
        %pause(0.1);
    end
    hold off
      subplotIndex =subplotIndex+1;
    % 如果subplotIndex超过了子图的总数，则退出外层循环
    if subplotIndex > nRows * nCols
        break;
    end
end

%%
%提取南北直行，东西直行，和转向的轨迹

% 创建 4x3 的子图网格
nRows = 4;
nCols = 3;
figure; % 创建一个新的图形窗口

% 初始化子图索引
subplotIndex = 1;
```

```
237
238 for j = 1:12 % 假设您只需要处理前12个索引
239     Idx = find(idx == j); % qf_Idx 需要在此前定义，并且其长度至少为12
240     unique_vehicle_id = unique_vehicle_ids(Idx);
241
242     for i = 1:numel(unique_vehicle_id)
243         flag = find(all_start_car(:,1) == unique_vehicle_id(i));
244         if isempty(flag)
245             continue;
246         end
247
248         current_vehicle_id = unique_vehicle_id(i);
249
250         % 提取当前 vehicle_id 的所有数据
251         vehicle_indices = find(data.vehicle_id == current_vehicle_id );
252         vehicle_data = data(vehicle_indices, :);
253
254         % 提取 x 和 y 坐标以及时间（时间在这里可能不需要用于绘图）
255         x_vector = vehicle_data.x;
256         y_vector = vehicle_data.y;
257
258         % 创建或选择子图
259         subplot(nRows, nCols, subplotIndex);
260         plot(x_vector, y_vector, '.');
261
262         title(sprintf('original X = %s Y = %s', num2str(x_vector(1)),num2str(
    y_vector(1)))); % 添加标题
263         % xlim([xmin xmax]); % xmin 和 xmax 需要定义为您想要显示的范围
264         % ylim([ymin ymax]); % ymin 和 ymax 同样需要定义
265          hold on
266         % 更新子图索引
267
268
269         % 如果subplotIndex超过了子图的总数，则退出循环
270         if subplotIndex > nRows * nCols
271             break;
272         end
273
274         % 暂停0.1秒
275         %pause(0.1);
276     end
277     hold off
278       subplotIndex =subplotIndex+1;
279     % 如果subplotIndex超过了子图的总数，则退出外层循环
280     if subplotIndex > nRows * nCols
281         break;
282     end
```

```
end
%% 出发时间 k_means初始聚类
figure
for j = 1:12 % 假设您只需要处理前12个索引
    Idx = find(idx == j); % qf_Idx 需要在此前定义，并且其长度至少为12
    unique_vehicle_id = unique_vehicle_ids(Idx);

    for i = 1:numel(unique_vehicle_id)
        flag = find(all_start_car(:,1) == unique_vehicle_id(i));
        if isempty(flag)
            continue;
        end

        current_vehicle_id = unique_vehicle_id(i);

        vehicle_data = all_start_car(flag, :);    %车辆的编号和出发时刻




        % 创建或选择子图
        if j == 1 || j== 5  %东西 直的

            plot(vehicle_data(3),1,'ro');  %red

            hold on
        end


        if j == 3|| j == 6 %we右
            plot(vehicle_data(3),1,'r*');  %red
           hold on

        end

        if j == 7|| j == 8  %sn右
            plot(vehicle_data(3),1,'g*');  %green
        hold on
        end

        if j == 11 || j == 12   %南北  直的
            plot(vehicle_data(3),1,'go');  %green
        hold on
        end

        if j == 4    %we左拐
            plot(vehicle_data(3),1,'b^');  %blue
```

```
        hold on
        end

        if j == 2|| j == 10    %sn左拐
            plot(vehicle_data(3),1,'bo');  %green
         hold on
        end
        hold on

    end


end


hold off




%% 聚类处理后，消除噪声

 % 创建一个新的图形窗口
  south_north = [];
  west_east = [];
  sn_left = [];
  we_left = [];
 buffer_cluster = 15;
    figure
for j = 1:12 % 假设您只需要处理前12个索引
    Idx = find(idx == j); % qf_Idx 需要在此前定义，并且其长度至少为12
    unique_vehicle_id = unique_vehicle_ids(Idx);

    for i = 1:numel(unique_vehicle_id)
        all_start_car1 = sortrows(all_start_car,3);
        flag = find(all_start_car1(:,1) == unique_vehicle_id(i));
        if isempty(flag)
            continue;
        end

        %current_vehicle_id = unique_vehicle_id(i);

        vehicle_data = all_start_car1(flag, :);    %车辆的编号和出发时刻


```

```
377
378         % 创建或选择子图
379         if j == 1 || j== 5 || j == 3|| j == 6  %东西 直的 +右拐
380           if isempty(west_east)||abs(vehicle_data(3) - current_west_east) >
    buffer_cluster  %每类只保留的一个
381               current_west_east = vehicle_data(3);
382
383               west_east = [west_east;current_west_east];
384               plot(current_west_east,1,'ro');
385           end
386
387             hold on
388         end
389
390 %{
391         if j == 3|| j == 6 %we右
392             %plot(vehicle_data(3),1,'r*');  %red
393            hold on
394
395         end
396 %}
397 %{
398         if j == 7|| j == 8  %sn右
399             %plot(vehicle_data(3),1,'g*');  %green
400         hold on
401         end
402 %}
403         if j == 11 || j == 12 ||j == 7|| j == 8  %南北  直的 +右转
404             if isempty(south_north)||abs(vehicle_data(3) - current_south_north) >
    buffer_cluster  %每类只保留的一个
405               current_south_north = vehicle_data(3);
406
407               south_north  = [south_north;current_south_north];
408               plot(current_south_north,1,'go');  %green
409             end
410         hold on
411         end
412
413         if j == 4    %we左拐
414             if isempty(we_left)||abs(vehicle_data(3) - current_we_left) >
    buffer_cluster  %每类只保留的一个
415               current_we_left = vehicle_data(3);
416
417               we_left  = [we_left;current_we_left];
418               plot(current_we_left,1,'m*');  %green
419             end
420
```

```
        hold on
        end

        if j == 2|| j == 10    %sn左拐
            if isempty(sn_left)||abs(vehicle_data(3) - current_sn_left) >
    buffer_cluster  %每类只保留的一个
                current_sn_left = vehicle_data(3);

                sn_left = [sn_left;current_sn_left];
                plot(current_sn_left,1,'b^');  %green

                hold on
            end
        end
        hold on

    end


end
%legend('show')

hold off
south_north = clusterAndKeepMin(south_north);
  west_east = clusterAndKeepMin(west_east);
  sn_left = clusterAndKeepMin(sn_left);
  we_left = clusterAndKeepMin(we_left);
  figure
  hold on
  plot(sn_left,1,'b^', 'DisplayName','南北左转');

  plot(we_left,1,'m*', 'DisplayName', '东西左转');

  plot(south_north,1,'go', 'DisplayName', '南北直行+右转');

  plot(west_east,1,'ro', 'DisplayName', '东西直行+右转');

  hold off
  T_circle = [];
  T_circle = [west_east(1:47),we_left(1:47),south_north(1:47),sn_left(1:47)];
  T = diff(T_circle);
  diff_T_circle = zeros(size(T_circle));
mode(T)
% 计算每一列当前列减前一列的差值
% for n = 1:size(T_circle, 2)
%
%     diff_T_circle(:, n) = T_circle(:, n) - T_circle(:, n-1);
```

```
% end

 % T_circle = concatenateAndFilter(T_circle)
% %%
%   south_north = [];
%   west_east = [];
%   sn_left = [];
%   we_left = [];
%  buffer_cluster = 15;
% figure
%
% for j = 1:12
%     Idx = find(idx == j);
%     unique_vehicle_id = unique_vehicle_ids(Idx);
%
%     for i = 1:numel(unique_vehicle_id)
%         flag = find(all_start_car(:,1) == unique_vehicle_id(i));
%         if isempty(flag)
%             continue;
%         end
%
%         vehicle_data = all_start_car(flag, :);
%
%         if j == 1 || j == 5 || j == 3 || j == 6
%             % 东西直行 + 右转
%             if isempty(west_east) || (vehicle_data(3) - current_west_east) > buffer_cluster
%                 current_west_east = vehicle_data(3);
%                 west_east = [west_east; current_west_east];
%                 plot(current_west_east, 1, 'ro', 'DisplayName', '东西直行+右转');
%             end
%         elseif j == 11 || j == 12 || j == 7 || j == 8
%             % 南北直行 + 右转
%             if isempty(south_north) || (vehicle_data(3) - current_south_north) > buffer_cluster
%                 current_south_north = vehicle_data(3);
%                 south_north = [south_north; current_south_north];
%                 plot(current_south_north, 1, 'go', 'DisplayName', '南北直行+右转');
%             end
%         elseif j == 4
%             % 东西左转
%             if isempty(we_left) || (vehicle_data(3) - current_we_left) > buffer_cluster
%                 current_we_left = vehicle_data(3);
%                 we_left = [we_left; current_we_left];
%                 plot(current_we_left, 1, 'm*', 'DisplayName', '东西左转');
```

```
%             end
%         elseif j == 2 || j == 10
%             % 南北左转
%             if isempty(sn_left) || (vehicle_data(3) - current_sn_left) >
    buffer_cluster
%                 current_sn_left = vehicle_data(3);
%                 sn_left = [sn_left; current_sn_left];
%                 plot(current_sn_left, 1, 'b^', 'DisplayName', '南北左转');
%             end
%         end
%         hold on
%     end
% end
% hold off
%%
figure
plot(1,'ro')
hold on
plot(1,'go')
plot(1,'m*')
plot(1,'b^')
legend('东西直行+右转', '南北直行+右转', '东西左转', '南北左转');
```

plot-pro4.m

```
clear;clc
data = readmatrix("D.csv");

% 提取唯一的vehicle_id
unique_vehicle_ids = unique(data(:, 2));
for i = 1:numel(unique_vehicle_ids)
    current_vehicle_id = unique_vehicle_ids(i);

    % 提取当前vehicle_id的所有数据
    vehicle_indices = find(data(:, 2) == current_vehicle_id);
    vehicle_data = data(vehicle_indices, :);

    % 提取x和y坐标以及时间
    time_vector = vehicle_data(:, 1);
    x_vector = vehicle_data(:, 3);
    y_vector = vehicle_data(:, 4);
    plot(x_vector,y_vector,'.');
    hold on;

end
```