清 华 大 学

综 合 论 文 训 练

题目：基于浮动车轨迹数据的路况分析

系　　别：计算机科学与技术系

专　　业：计算机科学与技术

姓　　名：吴铮

指导教师：向勇副教授

2017 年 6 月 11 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____ 导师签名：_____ 日　期：_____

中文摘要

随着出租车车载全球定位系统（GPS）装置的普及，浮动车（FCD）轨迹信息越来越容易获得。由于浮动车的轨迹信息有着非常好的城市覆盖率，而且数据量庞大，浮动车数据成为了城市交通路况计算中重要的数据来源。而高密度高准确度的 GPS 信息非常稀有，而且存储和传输成本高，这也使得研究低采样率的浮动车轨迹信息变得很有意义。

在本文中，我们讨论的是低采样率下的浮动车轨迹用于路况计算的情况，主要考虑了相邻两个数据点之间相隔一个路口的情况。本文中的模型采用线性回归作为主要方法，做了一些细节上的优化，在模拟环境下验证了方法的可行性。

本文的创新点主要有：

- 低采样率下分析经过路口的浮动车轨迹
- 没有使用传统的平滑算法

关键词：浮动车数据；低采样率数据；路口；路况估计

ABSTRACT

With the popularity of taxi car global positioning system (GPS) devices, floating car data (FCD) is much easier to be obtained. As the trajectory of the floating car has a very good urban coverage and a large amount of data, floating car data has become an important source of data for urban traffic estimation. However, highly precise GPS traces are rarely available and brings a great pressure to both storage and transmission system hence it is meaningful to study the trajectory information of the floating car data with low sampling rate.

In this paper, We discuss the use of floating car data at low sampling rates for urban traffic situation evaluation taking the situation that adjacent two data points separated by an intersection into consideration. The model in this paper use linear regression as the main method and pay attention to the optimization of some details. We use simulation data to verify the feasibility of this method.

There are some points making our system different:

- Junction analysis of floating car data at low sampling rate
- Abandon the traditional smoothing algorithm

Keywords: Floating Car Data; Low Sampling Data; Road Junction;Traffic Situation Evaluation

目　录

主要符号对照表

GPS	全球定位系统（Global Position System）
FCD	浮动车数据（Floating Car Data）
ITS	智能交通系统（Intelligent Transportation System)
AR	准确率 (Accuracy Rate）

第 1 章　引言

1.1　背景

随着经济发展，汽车的普及率越来越高，已经成为了是人们最常选择的出行手段之一。而汽车持有率的增加导致了很多交通问题，这也就使得实时路况计算变得十分重要。实时路况计算依赖可靠的实时道路轨迹数据，对于高速公路等简单结构的区域，一般采用固定探测装置来进行路况采集[1,2]。而对于相对复杂的城市区域，装有 GPS(Global Position System) 系统的浮动车是最近几年兴起的道路路况计算很好的数据来源。与传统的固定探测装置相比，浮动车不仅部署起来更简单，而且在城市区域拥有非常好的覆盖率，部署的成本也更低。但是高采样率的浮动车轨迹数据很难获得，而且储存和传输成本过高，所以我们主要研究低采样率浮动车轨迹数据下的路况计算[3,4]。

对于复杂的城市区域道路网络来说，各个路段的交叉口是对交通状况产生影响最重要的因素之一，判定路口行驶状况的参数有等待队列长度，车辆停止率等等。在本文中，我们忽略了车辆经过路口时的行驶细节，将信号灯或者其他类似于立交桥的道路切换系统中花费的时间统一计作一个虚拟的路口转向延迟。这样做的好处是可以忽略路口的实际结构，使得模型的可扩展性变得更强。

在我们遇到的实际浮动车轨迹数据中，低采样率的数据（两个相邻点之间相隔超过 30 秒）占到全部数据中的大多数[5]，而且 95% 以上的数据相邻两个点之间至少间隔了一个路口，这也就使得研究使用跨路口的低采样率浮动车轨迹数据判断两段路分别路况的算法变得很有意义。

1.2　研究目标

由于城市路段路口之间的间隔较短，使得车辆花费在路口的等待时间占到了很大比例，使得使用传统的路况估计方法得到的结果会和实际结果有着比较大的偏差，所以我们需要设计一个模型和算法估计城市区域路口的转向延迟，使得数据的利用率变高，结果也更接近实际值，提高整个城市路况估计系统的准确度。

1.3 文章结构

根据本文的内容，文章一共分成 5 个章节：

- 第一章是引言，主要介绍了论文的背景和目标。
- 第二章是综述，具体介绍了论文想要处理的问题和相关的工作。
- 第三章是算法设计，介绍了模型的建立和算法的选择。
- 第四章是实验，介绍了实验的设计和结果分析。
- 第五章是结论，总结了算法的优势和不足，以及今后可能的优化方向。

第 2 章　综述

2.1　问题综述

在计算路况时，由于数据的稀疏性，会出现两个 GPS 坐标之间相隔一个路口甚至更多的情况，但是我们要分别统计每一段路的路况，所以要把两段道路的路况进行分离。建立模型时把每段轨迹的时间分成两段路的用时加上虚拟的路口转向延迟三个部分，对于每个路段，如果存在正反两个方向则两个方向的路况互不干扰单独计算。

拿十字路口举例，如图 2.1所示，一个路口的参数包含 4 个路段每个路段两个方向共 8 个方向的路况参数，路口 12 个方向的转向延迟。当前系统的处理方式是直接把每段轨迹的时间按照经过道路的长度加权分配在每段路上，再由通过同一路段的所有轨迹加权得到该路段的用时，推算出平均速度来表示路况。

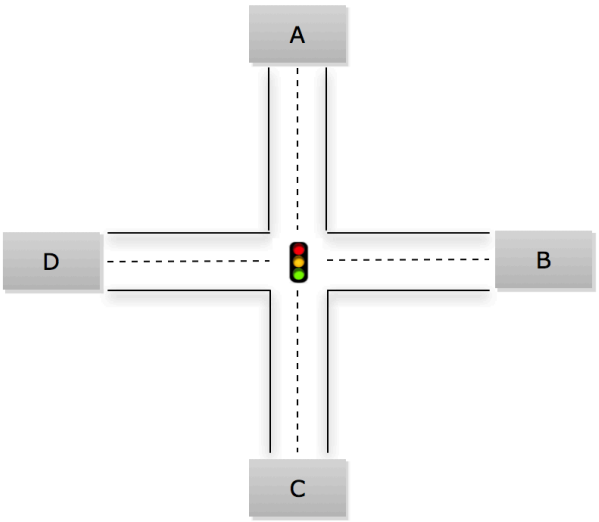

图 2.1　路口示意图

拿 AB 段举例，将路口设为 O 点，这时如果出现 AO 段行驶畅通而 OB 段堵

塞的情况，时间会平摊到 AO 和 OB 两段，这样计算出的结果不能反映真实的路况。又考虑到一定时间内路况不出现大的变化，本文的算法选取一定时间内通过同一路口的数据，使用这些数据直接计算出每一段的精确路况，从而能够更准确地反映每段路分别的路况信息。

2.2 相关工作

2.2.1 平滑算法

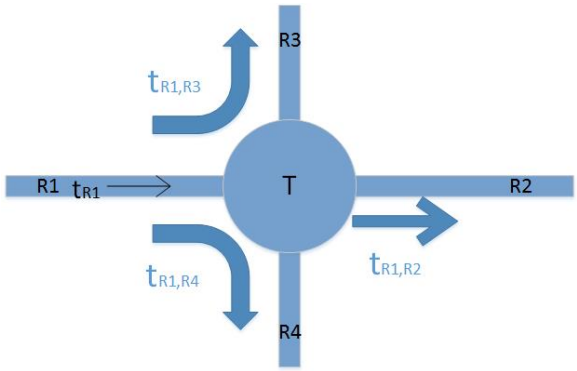

图 2.2　平滑算法

该算法中跨路口的轨迹，如图 2.2所示，拿 $R1 \rightarrow R2$ 举例，包括前一段路 R1 的行驶时间 t_{R1}，转向延迟 $t_{R1,R2}$ 和第二段路的行驶时间 t_{R2}(2-1)。然后计算当前道路的通过时间以及到下一条道路的转向时间的和在总时间中的比重 (2-2)。之后计算实际花费的时间 (2-3)。最后按比例分配行驶时间与转向时间 (2-4)。这种做法优势是实现起来非常简单，通用性强，但是对于某些路况分布不均匀情况可能效果不是很理想。

$$t = t_{R1} + t_{R1,R2} + t_{R2} \tag{2-1}$$

$$percentage = (cover\_rate \times road\_length / road\_speed + turn\_time) / total\_time; \tag{2-2}$$

$$real\_time = interval \times percentage \tag{2-3}$$

$$turn\_delay = real\_time \times turn\_time/(turn\_time + trip\_time) \tag{2-4}$$

2.2.2 路段分解算法

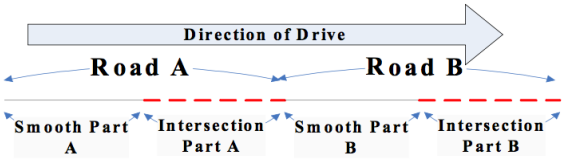

图 2.3 路段分解算法

算法引用自 Yue 发表在 ICCTP 2009 的一篇论文中[5]。该算法将道路分成平滑部分 (Smooth Part) 和路口部分 (Intersection Part)。IP 段长度由路由确定，是由道路等级和历史等待队列长度计算得出的。还将两条相邻的路段定义为道路 A 和道路 B，表示为 RA 和 R_B。两个连续的 GPS 信号分别被定义为 $G_j x_j y_j T_j$ 和 $G_{j+1} x_{j+1} y_{j+1} T_{j+1}$，其中（x，y）分别是匹配的 GPS 坐标，而 T 是相应的时间戳。计算 R_A 段行驶速度的公式如下 (2-5)。

$$TS_A = \frac{L_{TotalA}}{\frac{L_{intersection}}{TS^A_{intersection}} + \frac{L_{TotalA} - L_{intersection}}{TS^A_{SP}}} \tag{2-5}$$

其中，TS_A 代表道路 A 的平均速度，分母左右分别为路口路段的时间和平滑路段的时间。TS^A_{SP} 伪平滑路段的速度，使用所有在平滑路段 A 上前后两殿的平均速度按所占道路比例加权，找不到这样的点则使用该段的瞬时速度代替。$TS^A_{intersection}$ 则是路口路段的速度，使用路口路段车辆平均瞬时速度来计算。这种算法十分依赖瞬时速度，然而我们得到的浮动车 GPS 数据的瞬时速度经常出现非常大的偏差，所以这种算法也不适合我们的系统。

2.2.3 延迟估计算法

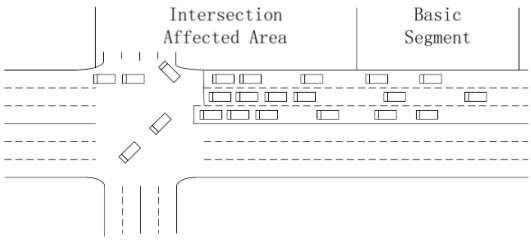

图 2.4 延迟估计算法

算法引用自 Zhao-cheng2014 年发表在 CICTP 的一篇论文中[6]。算法的原理是确定路口受影响区域外的两个最近的 GPS 点，来确定路口行驶时间。在计算路口延迟之前，需要车辆在路口的行驶时间 T_{fcd} 和车辆在加速行驶速度 T_{exp} 的行驶时间。文章中直接采用道路设计速度作为车辆的快速行驶速度。

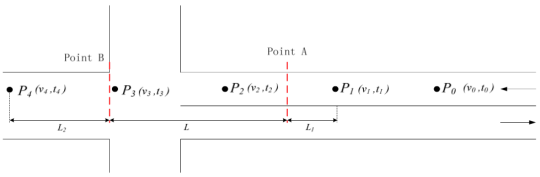

图 2.5 计算原理

算法的计算原理见图 2.5，其中点 A 和点 B 之间的区域是路口影响区域。假设一个跨越十字路口的出租车轨迹包括几个 GPS 点 $P_1P_2...$。选择路口影响区域外的两个最近点（即图 2.5中的 P_1，P_4）。这两点的速度需要大于零。点 P_1 用于估计进入路口影响区域的车辆的时间戳。P_4 点则用于估计离开路口影响区域的车辆的时间戳。可以得出如下的公式:

$$t_A = t_1 + \frac{L_1}{v_1} \tag{2-6}$$

$$t_B = t_4 + \frac{L_2}{v_4} \tag{2-7}$$

$$T_{fcd} = t_B - t_A = \left(t_4 + \frac{L_2}{v_4}\right) - \left(t_1 + \frac{L_1}{v_1}\right) \tag{2-8}$$

参数 v_1，v_4，t_1，t_4 是已知的，可以直接通过 GPS 数据获得。点 P_1 和 P_4 的位置可以通过上面的映射对应算法得出。之后就可以计算 L_1，L_2。因此，延迟估计的公式如下:

$$D = T_{fcd} - T_{exp} = t_B - t_A - \frac{L}{v_{exp}} = \left(t_4 + \frac{L_2}{v_4}\right) - \left(t_1 + \frac{L_1}{v_1}\right) - \frac{L}{v_{exp}} \tag{2-9}$$

该算法的优势是对于路口的判断很精确，但是需求的数据密度还是相对比较多，需要数据间隔在 30s 左右，但是我们的数据不能达到这种要求，间隔基本在 60s 左右，所以这种算法还是无法适用于我们当前的系统。

第 3 章　算法设计

3.1　算法描述

3.1.1　设计思路

前面提到现有的算法如果跨路口的两段路出现一段堵车而另一段通行良好的情况，时间会平摊到两段，不够准确。由于算法要求在线，但历史数据可以从服务器端取得，考虑 5 分钟内路况通常不会出现大的变化，当新的轨迹数据进入系统后，我们可以利用过去五分钟内的历史数据来计算出该轨迹经过路段的准确路况，也保证了算法的在线性。

3.1.2　算法优势

相对于平滑算法，我们对于轨迹数据的利用率更高，能够在相邻道路间速度差异较大时得到比较理想的结果，实现起来也较为简单，算法的复杂度不高，可能很快地计算出结果，能够适应在线系统的要求。模型同样具有可扩展性，只需要简单修改就可以应对多叉路口或者立交桥等复杂路口的情况，也可能扩展到跨越多个路口的情况。

3.2 具体实现

3.2.1 模型建立

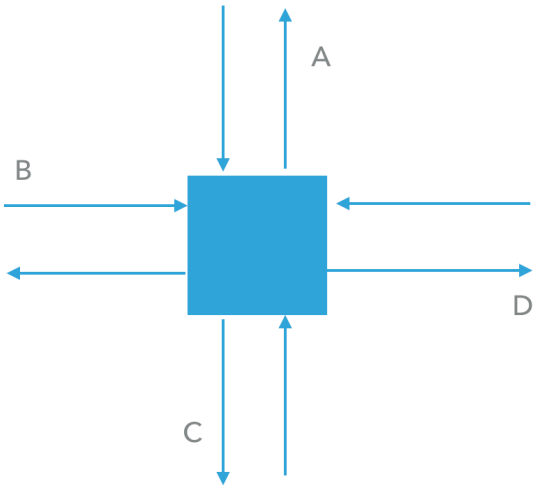

图 3.1 模型示意图

如图3.1所示，我们将路口部分抽象成一个黑盒，一个十字路口就被抽象成A,B,C,D4 段路径正反 8 个方向和黑盒中连接这 8 个方向的 12 组转向延迟。假设进入十字路口的路段为 A1,B1,C1,D1, 离开十字路口的则为 A2,B2,C2,D2。路口的轨迹数据信息就被抽象成了 12 组方程。

$$k1_{A1,B2} \times T_{A1} + k2_{A1,B2} \times T_{B2} + turn_{A1,B2} = t_1 \tag{3-1}$$

$$k1_{A1,C2} \times T_{A1} + k2_{A1,C2} \times T_{C2} + turn_{A1,C2} = t_2$$

.....

$$k1_{D1,C2} \times T_{D1} + k2_{D1,C2} \times T_{C2} + turn_{D1,C2} = t_{12}$$

其中 k1 与 k2 是指轨迹覆盖到当前路段的百分比,turn 是指该方向的转向延迟，包含花费在交通信号灯或者路口切换装置上的时间，把这些时间加在一起抽象成一个转向延迟的概念。

3.2.2 单方向算法

得到 5 分钟内的轨迹数据后按照 12 个方向进行划分，选取出轨迹数量最多的方向先入手。假设该方向 5 分钟内有 n 条轨迹，则我们可以得到 n 组方程，如下所示。再对方程组做线性回归求出拟合直线，之后利用直线的斜率和截距信息计算出花费在第一条路的时间 T_1, 花费在第二条路的时间 T_2 以及转向延迟 turn。

$$\begin{bmatrix} k_{11} & k_{21} & 1 \\ k_{12} & k_{22} & 1 \\ \vdots & \vdots & \vdots \\ k_{1n} & k_{2n} & 1 \end{bmatrix} \times \begin{bmatrix} T_1 \\ T_2 \\ turn \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{bmatrix} \tag{3-2}$$

3.2.3 线性回归

如果数据量多于 3 条，那么方程是无解的，所以我们需要引入线性回归的算法来近似地得出每段路的路况结果。把总时间 t_i 看成随机变量 y，把每条道路的通行时间看成自变量 x_i, 就有 (3-3)。

$$y_i = \theta^T x_i + \epsilon_i \tag{3-3}$$

这时使用最小二乘法求出一条使得所有点到拟合直线垂直距离平方之和 (3-4) 最小的拟合直线。

$$\frac{1}{2}\sum_{i=1}^{n}(h_\theta(y_i - x_i))^2 \tag{3-4}$$

使用最小二乘法对轨迹数据做线性回归也就相当于做了一个最大似然估计，而类似的还有一种方法是使用贝叶斯方法来对轨迹数据做线性回归。Bayes 方法

的关键是计算后验概率，已知先验分布 $p(\theta) = N(0, \tau^2 I)$ 之后，我们使用 bayes 公式得到 (3-5)

$$p(\theta|S) = \frac{p(S|\theta)p(\theta)}{p(S)} \tag{3-5}$$

之后以 θ 为底做积分可以得到 y 在当前的数据样本空间下的分布 (3-6), 再对 y 积分得到线性回归的预测期望值 (3-7)

$$p(y|x, S) = \int_{\theta} p(y|x, \theta)p(\theta|S)d\theta \tag{3-6}$$

$$E[y|x, S] = \int_{y} p(y|x, S)ydy \tag{3-7}$$

Bayes 方法和前面普通的最小二乘法做线性回归最大的区别就是加入了 θ 的先验概率，也就是要找到使得 (3-8) 最小的拟合直线作为结果。

$$\frac{1}{2}\sum_{i=1}^{n}(h_\theta(y_i - x_i))^2 + \frac{\theta^2}{2\tau^2} \tag{3-8}$$

bayes 方法对比最小二乘法最大的优势就是加入一个先验概率来在一定程度上规避掉过拟合的情况，在图3.2中我们选取了一些模拟数据加入了随机的噪声，对比最小二乘法和 Bayes 方法的效果。

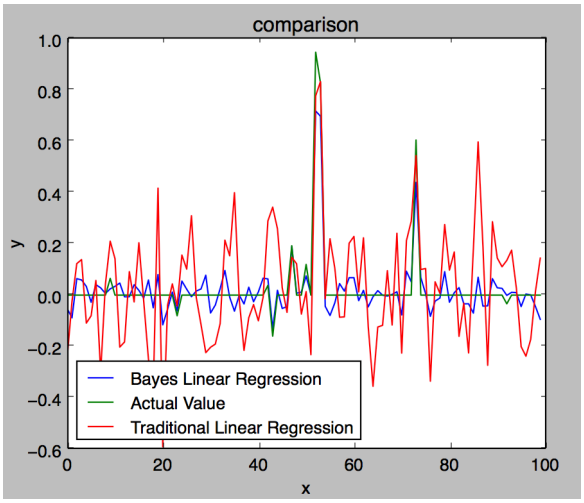

图 3.2　Bayes 对比最小二乘法

图3.2中绿色为原始数据，蓝色为 Bayes 方法回归结果，红色为最小二乘法的回归结果。我们可以看出总体上相比于最小二乘法，Bayes 方法回归出的结果更接近于实际值，但是在实际数据中随着数据量的增加，最小二乘法的精确度会比 Bayes 方法更高，所以我们决定视数据量多少混合使用这两种算法进行计算，具体的结果会在第 4 章实验部分展示。

3.2.4　多方向混合的算法

考虑通过同一路段的三个方向的轨迹，其中有一条为右转的轨迹，而右转方向通常不需要等待信号灯，结果较为准确，需要的轨迹数量更少，如果可以找到右转的轨迹则先使用右转的轨迹计算得出该路段的路况，之后再结合另外两个方向的轨迹信息求解另外两个方向的路况信息。

对于路口整体的轨迹数据，我们使用相对方向直行和左转一般同时放行，得出相对方向直行和左转的红绿灯周期相同，所以转向延迟相近，计算出到一个方向的路况信息之后可以将转向延迟应用于其对应方向的路况计算，减少数据的需求量，综合这些信息我们设计了一个系统来对路口整体路况做计算，流程图如图3.3。

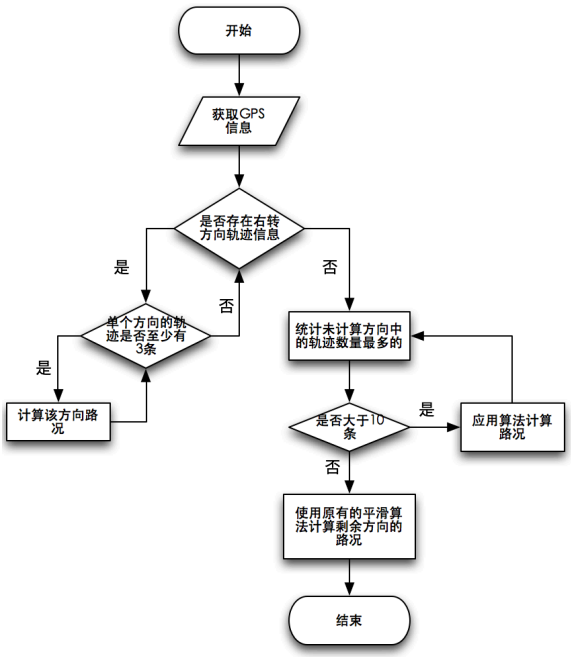

图 3.3　流程图

3.3　细节改进

3.3.1　细化路口延迟

为了细化车辆在路口的转向延迟，我们设计了一个方法来计算车辆在路口的等待队列长度，使用静止点序列 A_n 的平均值的 2 倍和和该序列的最大值来估计等待队列长度。

$$waiting\_queue = \max(A_i | i = 1, 2...n, 2 \times average(A_n)) \tag{3-9}$$

之后我们细化了路口的转向延迟 T(3-10)

$$T = t_2 - t_1 - |p_1 - j_1|/v_1 \tag{3-10}$$

其中 T 为路口延迟，t_2 和 t_1 分别是转弯过程中起始和终止点的时间戳，而 $|p_1 - j_1|$ 表示路口等待队列的长度，v_1 是前一段路平滑路段的平均速度。

3.3.2 排除异常点

由于行驶轨迹数据带有一定的随机性，可能出现驾驶员异常操作或道路匹配错误等原因产生的异常数据，所以我们需要将数据中的异常点排除。对于自变量 y 和拟合向量 $\hat{y} = Py$, 我们可以得到残差向量 $e = y - \hat{y} = (1 - P)y$，计算残差平方和 $RSS = e^T e$ 进而我们将残差标准化 (3-12) 其中 $\hat{\sigma}$ 为残差的参照值 (3-11)，假设数据满足正态分布为 $N(0, 1)$ 的随机样本，那么 r_i 落在 $[-2, 2]$ 的概率 $P(\mu - 2\sigma < U < (\mu + 2\sigma) = 95\%$, 所以如果 $|r_i| > 2$ 那么这个点就被视为异常点，将其删除。如图3.4，该图为一组左转数据的残差图，对于红色的两个点，我们在计算时将其删除，避免对整体结果产生偏差。

$$\hat{\sigma} = \sqrt{RSS/(n - k - 1)} \tag{3-11}$$

$$r_i = \frac{e_i}{\hat{\sigma}\sqrt{1 - p_{ii}}}, i = 1, 2, ..., n \tag{3-12}$$

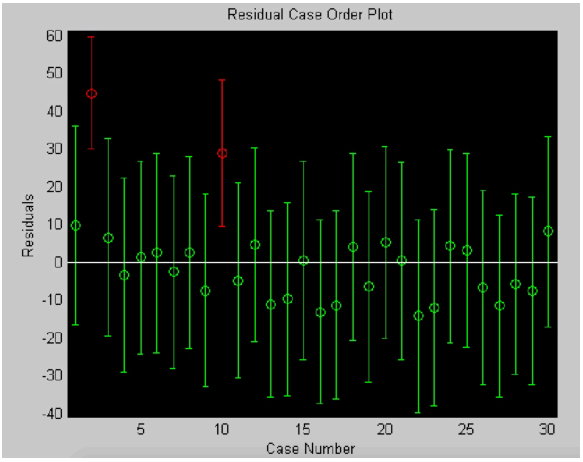

图 3.4　残差图

3.3.3 回归方程式变形

由于有的时候 GPS 轨迹信息的采样间隔不变，比如有些数据固定隔 60s 取一个点，直接3-13对采用线性回归的方法由于 T_{total} 不变不能得到满意的效果，所以要对式子进行变形3-14。这样做可以得到两段路的行驶时间，但是只能将路口延迟近似为一个既得的固定值，会对计算的精度造成影响，所以在 GPS 轨迹信息间隔不固定时还是采用原有的3-13。

$$k_1 \times T_1 + k_2 \times T_2 + T_{turn} = T_{total} \tag{3-13}$$

$$T_1 + \frac{k_2}{k_1} \times T_2 = \frac{T_{total} - T_{turn}}{k_1} \tag{3-14}$$

第 4 章　实验

4.1　实验设计

本文中使用 sumo 作为仿真实验工具，生成模拟道路和信号灯，并且在地图上随机生成车辆获取车辆的轨迹信息来进行仿真试验。参数的设置为每段路的长度为 1500m, 车辆限速初始为 90km\h, 实验过程中会视情况调整，一个交通信号灯的周期为 140s, 其中每个方向直行放行 40s 之后左转放行 30s，右转方向为全时绿灯的状态。

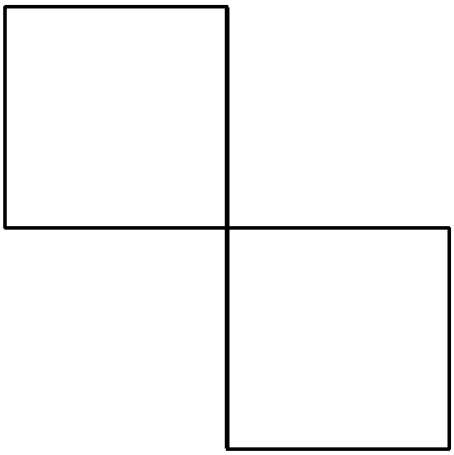

图 4.1　地图

为了让轨迹数据能够集中通过十字路口，主要地图设计成如图4.1，为一个"8" 字形地图，使得随机的车辆能够在地图中随意行驶并且集中地通过中心的路口。

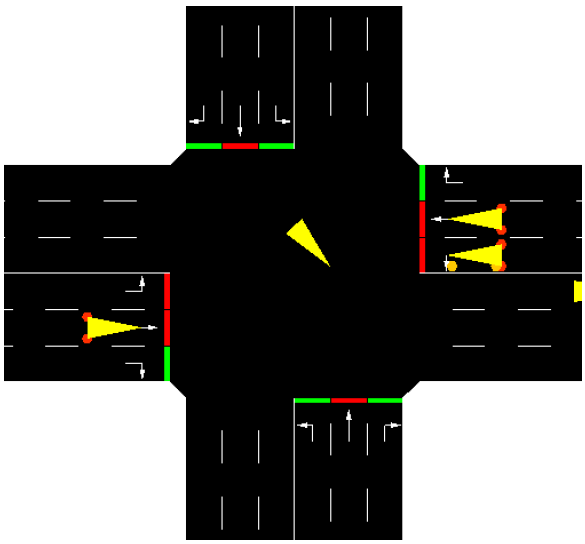

图 4.2　路口放大

模拟过程可以通过 sumo-gui 直观地看到，可以观察每辆车的具体行驶轨迹，路口模拟效果如图4.2, 可以清晰地看到每辆车和红绿灯的状态。

4.2　实验结果分析

4.2.1　对比 Bayes 和最小二乘法

从图4.3中我们可以看到，Bayes 方法在数据量较少的时候更加稳定，准确度也比最小二乘法更高，最小二乘法等价于最大似然估计，在数据量小的时候会出现过拟合的现象，但是随着数据量逐渐增加准确度会超过 Bayes 方法，所以我们会在数据量较少 (少于 14 组时) 使用 Bayes 方法拟合，在数据量较多的时候使用最小二乘法进行更精确的估计，将两种算法结合到一起。

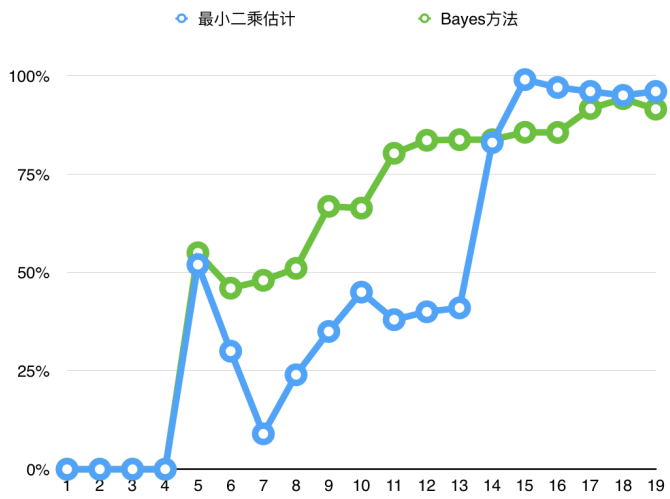

图 4.3　两种方法对比

4.2.2　单方向

利用 sumo 中的轨迹 GPS 数据，选取单方向统计 5 分钟内的轨迹作为输入，之后将估计的两段路的路况与计算出的实际路况进行比对计算出准确率，直行方向和右转方向准确度与轨迹条数的关系见图4.4和图4.5。

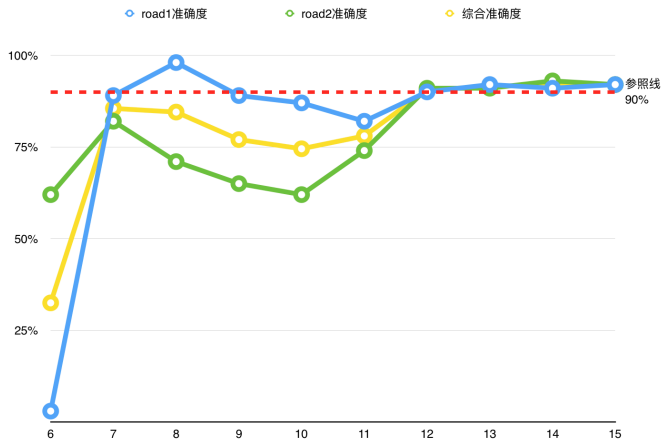

图 4.4　直行方向准确度

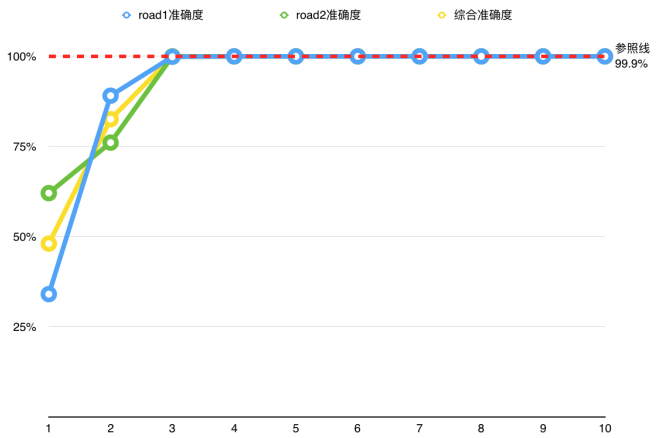

图 4.5　右转方向准确度

从图中我们可以看到直行方向大概需要 12 条轨迹数据来达到总体 90% 的准确率，7 条轨迹到 11 条轨迹之间的结果存在波动，分析数据得出出现波动的原因是其中中间几组数据的信号灯等待时间较长，导致算法回归时直线出现了便宜，数据量再增加直到 12 条之后，个别数据的偏移不会对总体数据造成很大的影响，左转方向与直行方向的情况基本相同。而右转方向因为无需等待交通信号灯，所以转向延迟比较稳定，采用我们的算法，只需要 3 条轨迹信息就可以到达 99.9% 的正确率。

4.2.3　与原有的平滑算法的对比

当单方向两段路路况存在较大差异时，即一段路堵塞而另一段正常行驶，给出方向有 15 组 GPS 轨迹信息，分别使用平滑和线性回归两种算法计算两段路的路况，得出结果如图4.6

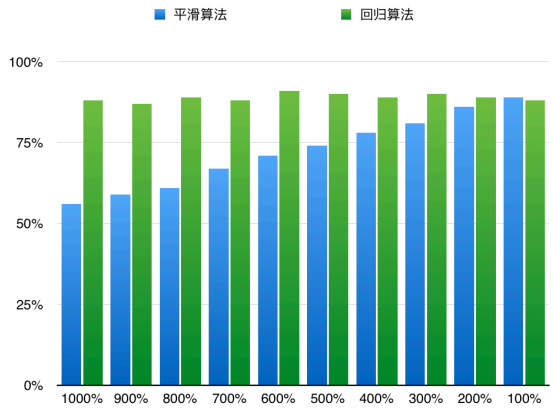

图 4.6　路况差异较大时两种算法的对比

图4.6中横坐标为一条轨迹经过两段路段平均速度的比例，纵坐标为路况计算的准确率。实验中我们通过限制一段路的最高速度使得两个路段的畅通程度产生差异。例如横坐标 1000% 的点，我们取一段路最高限速为 25km\h，另一段路最高限速为 2.5km\h，给出 15 组车辆轨迹数据，分别用两种方法算出两段路的路况与实际值做对比。我们可以看出，当两个方向的平均速度相差 1000% 时，回归算法对比平滑算法优势明显，随着两段路速度差异的变小，算法的准确率差距也逐渐缩小，最后两段路况基本相同时，两种算法的准确率都稳定到 90% 左右。

4.2.4　三方向混合

对于经过同一路段的三个方向轨迹信息进行混合，先拿出右转方向的轨迹信息计算出该路段的路况，再将其应用到直行和左转方向的路况计算中，计算下一路段的路况，得出结果如图4.7。

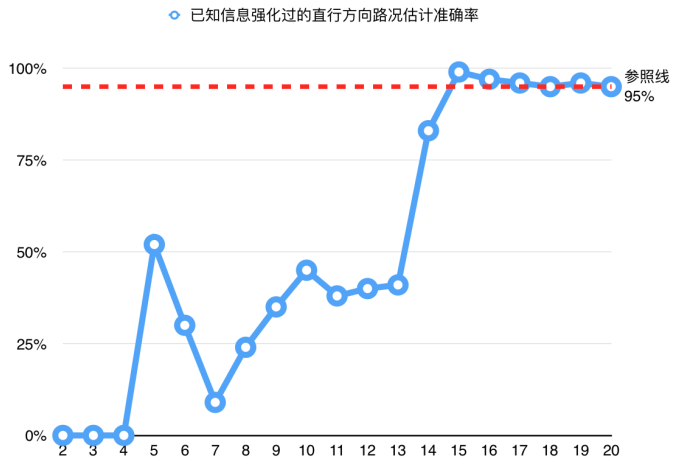

图 4.7　三方向混合准确度

从图中我们可以发现，直行方向此时需要 15 条轨迹数据来达到稳定的准确率，比之前直接对该方向轨迹进行回归的 12 条还多出三条，这种情况出现的原因是确定前一段路况之后，线性回归又三维降低为二维，但是常数项也就是路口延迟的波动对于我回归的影响会变得更严重，但是达到稳定状态后，我们可以得到比前面更优的结果，准确率为 95% 以上，所以我们决定综合使用这两种方法，在轨迹数为 14 条及以下时直接计算该方向路况，在 15 条及以上时使用右转方向的既得信息对回归降低维度，得到更精准的结果。

4.2.5　全方向混合

考虑到相对方向的路口信号灯等待和通行周期相同，如果我们得到一个方向的转向延迟，我们就可以把转向延迟应用到其对应方向的计算中，使准确率得到提高。结果如图4.8。

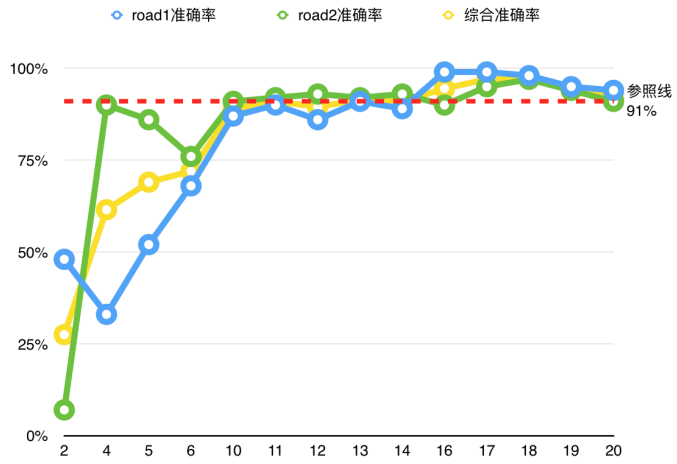

图 4.8　全方向混合准确度

从图中我们可以看出得到对应方向的转向延迟信息后，该方向只需要 10 条轨迹就可以计算出准确率为 91% 左右的路况信息，对应方向提供的转向延迟信息可以帮助筛选该方向的数据，规避掉路口信号灯等待时间过长的轨迹信息带来的影响，所以可以使得路况计算所需的轨迹数量减少。

4.3　小结

通过仿真数据的验证，说明了我们的模型对于轨迹数量较多的稀疏数据能够有很高的路况估计准确率。在我们的模型中，对于一个十字路口，最理想的情况是有四个右转方向各三组以上的轨迹数据，这样就能精确地得到 8 个方向的路况，共需 12 组轨迹，得到 99% 准确率的路况信息。最不理想的情况是没有右转方向的轨迹，这种情况下，我们结合路口对应方向的路况信息，每组对应方向至少需要 12+10=22 组轨迹信息来得到准确率高于 90% 的路况信息，这样加起来就需要至少 22*2=44 组轨迹信息来得到 8 个方向的路况。

第 5 章　结论

相对于传统的平滑算法，我们的算法改善了相邻两段路况相差较大的情况下的路况计算。使用仿真数据检验了算法的可行性，在轨迹数据较为稀疏，数据量较大的情况下能有很好的结果。当数据量不足的时候对于右转方向也可以有很好的判断。

算法的不足是直行和左转方向的路况判断依赖轨迹的条数比较多，出现这种状况的原因是交通信号灯的随机性比较大，又占到了轨迹总时间的很大比例，造成了数据波动，需要一定的数据量才能拟合回准确值。

未来改进的方向有以下几个方面：

- 对算法再做优化，降低路口交通信号灯的影响，使得需求的轨迹数量减少。
- 对算法做扩展，扩充到多个路口的情况，这时需要考虑几个交通信号灯综合作用的情况。
- 将道路车辆数量，浮动车占到总数量的比例，道路等级，历史路况等信息综合起来考虑，并加入到算法中。

插图索引

表格索引

公式索引

参考文献

[1] Yuan Y, Van Lint H, Van Wageningen-Kessels F, et al. Network-wide traffic state estimation using loop detector and floating car data. Journal of Intelligent Transportation Systems, 2014, 18(1):41–50

[2] Wang Y, Papageorgiou M. Real-time freeway traffic state estimation based on extended kalman filter: a general approach. Transportation Research Part B: Methodological, 2005, 39(2):141–167

[3] Rahmani M, Koutsopoulos H N, Ranganathan A. Requirements and potential of gps-based floating car data for traffic management: Stockholm case study. Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on. IEEE, 2010. 730–735

[4] Brockfeld E, Wagner P, Lorkowski S, et al. Benefits and limits of recent floating car data technology–an evaluation study. CDROM-WCTR2007, 2007. C2–830

[5] Yue Y, Zou H, Li Q. Urban road travel speed estimation based on low sampling floating car data. ICCTP 2009: Critical Issues In Transportation Systems Planning, Development, and Management. 2009: 1–7

[6] He Z c, Ye W j. Delay estimation model based on low-sampling-rate floating car data. CICTP 2014: Safe, Smart, and Sustainable Multimodal Transportation Systems. 2014: 387–395

致　谢

衷心感谢导师向勇副教授对本人的精心指导。他们的言传身教将使我终生受益。

感谢清华大学计算机协同工作实验室全体老师和成员的热情帮助和支持。

感谢 ThuThesis，它的存在让我的论文撰写轻松自在了许多，让我的论文格式规整漂亮了许多。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

附录 A　外文资料书面翻译

应用于实时交通传感应用基于隐马尔科夫模型的在线地图匹配

摘要: 在许多智能交通系统（ITS）应用程序中，探测车辆的数据来源，一个关键的步骤是准确地将 GPS 轨迹实时地映射到道路网络。这个过程被称为地图匹配，通常需要考虑数据的噪声和稀疏性，因为（1）高度精确的 GPS 轨迹信息非常稀有，以及（2）密集轨迹对于实况传输和存储是昂贵的。

我们提出了一种基于对噪声和稀疏性鲁棒的隐马尔可夫模型（HMM）的在线地图匹配算法。我们集中在现有的基于 HMM 的算法的两个改进:（1）使用最佳局部化策略，可变滑动窗口（VSW）方法，保证在不确定的未来输入下的在线解决方案质量，和（2）使用机器学习的空间，时间和拓扑信息。我们使用在城市和农村地区的公交路线上收集的现场测试数据来评估我们的算法的准确性。此外，我们还研究了在处理实时输入流时精度和输出延迟之间的关系。

在我们对现场测试数据的测试中，VSW 在精度和输出延迟方面都优于传统的定位方法。我们的研究结果表明，它可以应用于对延迟要求较为苛刻的应用程序，如交通流量传感。

A.1　引言

由大量车辆收集的实时传感器数据，例如城市地区的出租车和公共汽车，为交通感测，交通事故检测，旅行时间预测 [3]，车辆管理 [4] 和路线建议 [5]，[6]。这些系统的可用性取决于通过可用的地图匹配算法提取的数据的可靠性，地图匹配算法做的就是将 GPS 轨迹投影到数字地图上的相应路段。

诸如时间戳，位置和速度的信息通常由探测车辆记录。然而，处理这些大量数据所需的高额到难以负担的存储和带宽导致了现在收集稀疏样本的做法，间隔范围从几十秒到几分钟 [7]。此外，已知安装在这些探头上的传感器容易出现各种错误 [7]，例如不精确的位置和速度测量，重复传输和时间戳不匹配。在诸如交通感测的实时应用中，还要求在未来数据点未给出的时候就执行地上的地图匹配，即算法必须在线。可靠的在线地图匹配算法因此必须考虑这些问题并且保证精度和及时性高的输出。

地图匹配算法被定性为全局或增量/在线。全局算法在生成解之前对整个输入轨迹进行批处理。增量/在线算法采用将输入轨迹划分为更小段并按顺序处理的局部化策略，有时导致次优解。在地图匹配中应用的技术包括几何分析 [12]，信念函数理论 [13]，扩展卡尔曼滤波器 [14] 和隐马尔可夫模型 [11] [15] [16]。这些方法的优势和局限性已在 [9] 中综述。特别地，已经采用基于 HMM 的算法及其变体 [10]，[17] 用于同时评估实际映射的多个假设的能力，以便找到最终的最大似然解。这些方法已经被证明可以抵抗高度噪声测量，例如来自 GSM 塔的位置指纹 [16]，并且它们的精度随着轨迹的时间稀疏度的增加而退化 [10]，[15]，[17]。

我们提出的在线 HMM（OHMM）地图匹配算法受到最近基于 HMM 的方法 [10]，[15] - [17] 的启发。我们解决了在以前的相关工作中没有集中的两个具体问题：（i）需要一个在线算法来管理精度和输出延迟之间的权衡，以及（ii）多个评分函数的融合以估计转换可能性。

大多数现有的增量/在线算法使用简单的定位策略，例如固定滑动窗口和固定深度递归预测。滑动窗口方法简单地将轨迹划分为固定大小的输入序列并且独立地处理它们。较大的窗口尺寸导致更好的精度 [10]，[17] 但更长的输出延迟，反之亦然。递归预测方法将每个点的决定延迟固定数量的步长，以评估未来路径替代方案 [18]。这两种方法虽然易于实现，但可能导致次优解决方案和长输出延迟。当地图匹配算法在实时输入流上操作并且需要在短时间窗口内产生输出时，这显然是不可行的。受实时应用需求的驱使，我们提出了一个最优的局部化策略，使用可变滑动窗口（VSW）将输入分成更小的子问题，并证明找到全局最优解。我们的方法在概念上类似于在线维特比算法 [19]，[20]。此外，我们还开发了 VSW 的次优变体，其提供对延迟性能的最坏情况保证。在第五节中，我们将使用定长滑动窗口（FSW）方法作为基准来比较这两种方法的性能。

其次，我们推导了一种概率评分机制，其在 HMM 中的表现和转移概率的建模中结合了各种传感器数据和拓扑信息：（i）GPS 坐标（ii）车辆速度（iii）速度限制（iv）推断的车辆前进方向，以及（vi）拓扑约束。我们使用支持向量机（SVM）来学习转移概率函数，而不是选择简单的先验模型 [10]，[15]，[17]，然后估计其参数。这种方法的主要优点是它提供了一个数据驱动的框架，用于集成多个过渡评分函数。

我们使用在新加坡包括农村和城市地区的 4 条公交线路上收集的现场测试

数据来评估我们的方法的性能。性能是根据精度和输出延迟标准来测量的。我们的研究结果表明，当操作在实时输入流时，所提出的算法实现了优化或接近最优解，实际上具有低输出延迟。

本文结构如下。第二节根据 HMM 来制定地图匹配问题。方法将在第三节中描述。我们的实验设置在第四部分解释。结果在第五节中介绍。最后，第六部分总结了我们的贡献，并讨论了未来工作的空间。

A.2 地图匹配问题

A.2.1 问题定义

定义 1: 轨迹，$T = (t_n|n = 1, ..., N)$，是由车辆收集的 N 个数据点组成的序列。每个轨迹点由其经度 $(t_n.lon)$，纬度 $(t_n.lat)$，速度 $(t_n.v)$ 和时间戳 $(t_n.t)$ 指定。

定义 2: 段落，$r = (p_m|m = 1, .., M)$，是一条 M 点折线表示的道路段曲线。它由连接顶点的一系列线段 $p_1, , p_m$ 组成。按顺序，其中每个顶点由其经度和纬度指定。段落也由其道路宽度（r.w），速度限制（r.v）和双向行驶（r.d）的允许性定义 (r.d={true,flase})。

定义 3: 数字地图，$G = \{r_k|k = 1, .., K\}$，是一组代表 K 个段落的集合。

给定轨迹 T，地图匹配的目标是找到 T 每个轨迹点到 G 中段落的对应关系。

A.2.2 HMM 方法

在基于 HMM 的地图匹配算法中，候选路径被顺序地生成并基于它们的似然性来评估。当遇到新的轨迹点时，解决方案的过去的假设被扩展以考虑新的观测值。在最后阶段的所有候选中，具有最高联合概率的幸存路径然后被选择为最终解。

对于每个轨迹点，我们首先从该数据最有可能被采集的地点识别一组候选道路段落。这些候选中的每一个表示为马尔可夫链中的隐藏状态，并且具有表现概率，其是在候选段是真实匹配的情况下观察 GPS 点的可能性。直观地，如果点被发现更接近它，我们将给段更高的概率。然后，我们计算链中每对相邻隐藏状态的转移概率，使得后者的概率仅取决于前者，因此服从马尔可夫假设。我们的目标是找到具有最高联合表现和传输概率的马尔科夫链上的最大似然路径。

该过程如图 1 所示。

正式地，我们将表现概率表示为 p(t | h), 其是给定隐藏状态（段）的观察到轨迹点 t 的概率。从隐藏状态到隐藏状态的转移概率是 f(s,h)。给定一个 N 个点轨迹，隐藏状态的似然序列，$T = (t_n|n = 1,, N)$，隐藏状态的似然序列，$S^* = (S_n \in A_n|n = 1, ..., N)$ 满足以下递推关系。

$$V_{n,h} = p(t_n h) \max_{s \in A_{n-1}} \{f(s, h)V_{n-1,s}\}.$$

这里 $V_{0,h} = p(t_0|h)$ 和 A_n 表示第 n 阶段的隐藏状态集合。然后，我们可以从最后一个元素中找到 S^*,$s_N = \arg\max_{h \in A_n}\{V_{N,h}\}$，向后工作找到最大联合概率序列 S_{N-1},...S_1。我们在第 3-E 节将提供一个在线算法找到 S^*。

A.3 OHMM 地图匹配算法

A.3.1 算法的基本流程

- 对于每个轨迹点，找到围绕其半径为 50m 的所有候选分段。施加该阈值的原因是双重的:（1）丢弃具有非常低的表现概率（低于 10^{-4} 的范围）的所有候选，以及（2）避免由于过多候选而导致执行速度的惩罚。
- 针对每个候选分段（隐藏状态）计算表现概率，而将转移概率分配给在隐藏状态上发生的每个边缘。
- VSW 算法在更新的 Markov 链上执行回溯，并给出部分求解（如果可用）。否则，输出将延迟一个阶段。
- 对下一个轨迹点重复上述过程。当到达最后一点时，算法终止。

A.3.2 表现概率

对于在轨迹点附近找到的每个候选分段，我们用 1D 高斯函数对其观测概率进行建模，如下所示。

$$p(observation) = \frac{1}{2w}\int_{-w}^{w} \frac{1}{\sqrt{2\pi\sigma_g^2}} e^{-\frac{(l-d)^2}{2\pi\sigma_g^2}} dl.$$

这里 w 是指道路 r 的半宽 (w=0.5*r.w)，d 是指 t 和 r 间点到曲线之间的大圆距离，σ_g 是 GPS 误差的估计标准偏差。虽然 GPS 误差已知具有非高斯分布，我们采用这种模型是因为它易于实现和在之前的工作被证明有效 [10]，[15]，[17]。我们的方法是不同的，因为我们也考虑道路宽度。这将允许我们更好地在路段之间区分，特别是在路口。

此外，基于假设驾驶员不大可能大大超过速度限制，我们引入超速的惩罚机制。目的是帮助区分从相同交叉点分支出的可能具有不同速度限制的紧密间隔的平行道路。我们注意到在这些情况下，单独的位置测量不足以区分段落，因为记录的轨迹点可能落在它们之间。我们定义惩罚函数 S 如下，

$$S(v_t, v_r) = \frac{v_r}{max(0, v_t - v_r) + v_r}.$$

这里 vt 指传感器记录的时间，而 vr 指该路段的限制速度。如果遵守了速度限制，则 $max(0, v_t - v_r) = 0$。（即惩罚机制不启动）。结合（2）和（3），我们定义表现概率，p(t | r) 如下

$$p(t|r) = S(v_t, v_r)p(oberservation).$$

A.3.3 转移概率

令 i，j 分别表示归属于两个连续轨迹点 t_n 和 t_{n+1} 的一对候选段，其中 $i \in A_n$、$j \in A_{n-1}$。我们将内插路径 $P_{i\to j}$ 定义为当从 i 行进到 j 时车辆最可能采取的段的序列。假设驾驶员选择最短路程的路（路径距离短的被选择可能性高），我们可以使用 A* 路径寻找算法找到插值路径 [21]。对于 K 个片段的给定序列，$P_{i\to j}$ 中的 $(r_1, , r_k)$，我们设计如下的两个评分函数，

方法 1: 距离差异函数 T 测量传感器推测的行进距离和插值路径长度之间的差异，

$$T(d_{i\to j}, D_{i\to j}) = \frac{|d_{i\to j} - D_{t\to j}|}{D_{i\to j}},$$

这里 $d_{i\to j} = \overline{v}_{i\to j}\Delta t$ 是车辆在时间间隔，以平均速度 $\overline{v}_{i\to j}$ 行驶的距离，而 $D_{i\to j}$ 是 $P_{i\to j}$ 的路径长度。上面的方法 1 通过比较其长度与推测（DR）估计评估了假设路径 $P_{i\to j}$ 的可行性。如果 $P_{i\to j}$ 是真实路径，差异会被假定为接近于零。

方法 2：选用动量变化函数 M 测量车辆对于在 $P_{i\to j}$ 中采取的每个路段所引起的平均动量变化，

$$M(v_0, v_1, l_1, ..., v_k, l_k) = \frac{\sum_i^K l_i||v_i - v_{i-1}||}{\overline{v}_{i\to j}\sum_i^K l_i},$$

其中 $(v_1, .., v_k)$ 是 $P_{i\to j}$ 中每个段的车辆的速度矢量，并且 $(l_1...l_k)$ 是相应的段长度。我们假设矢量幅度随时间从 $|v_1|$ 线性变化到 $|v_n|$ ，而它们的方向与段曲线平行。注意，附加参数 v0 是从先前转换的终端速度继承的初始速度矢量。通过类似的逻辑，我们将使用 v_k，当前终端速度作为下一转换中的初始速度，等等。图 2 示出了该概念。上述措施 2 可以被描述为“平滑因子”，其对由许多突然转弯组成的不可行转移做惩罚。

上面介绍的评分函数 T 和 M 为转变 $i \to j$ 提供两种不同的“适合度度量”。这表明，可以通过将这两个度量融合在一起来导出转移概率。我们使用标记为正确或不正确转换的实例训练支持向量机（SVM）分类器，其中特征向量由测量 1 和测量 2 给出的分量分数组成。利用该分类方法，$P_{i\to j}$ 是输入得分组合属于“正确转换”类别的概率。我们将在第 4-B 节更详细地描述训练过程。

A.3.4 在线维特比算法

我们的目标是使用增量法找到全局地图匹配解决方案。这意味着该算法需要在不知道未来输入的情况下沿着马尔可夫链进行不可逆的在线决策，同时确保部分解在组合时产生全局最优解。为了实现这一点，我们应用在线动态规划来解决 (1) 中的递推关系。关键的见解是，当当前幸存路径在马尔科夫链中的某点 (收敛点) 收敛时，所有未来的幸存路径将包含直到收敛点的相同子路径。相关证据可在 [19] 和 [20] 中找到。

我们制定我们的 OHMM 算法的伪码如下。算法 1(OHMM 地图匹配) 递增地处理轨迹点，并且在每个阶段，它输出算法 2 返回的部分解，如果可用的话。否则，它给出一个空输出，并引发一个延迟阶段。算法 2（在线维特比算法）检查

在解链中是否存在任何收敛点，并返回到该点的最大似然子序列 (如果有的话)。

方便地描述算法 1 和 2 在滑动窗口方面的工作原理。当在由窗口覆盖的马尔科夫链中的任何地方找到收敛点时，窗口随着新的轨迹点被处理并从后面收缩而向前扩展。注意，滑动窗口的大小可以根据状态空间的结构而变化，因此名称可变滑动窗口。图 3 示出了 VSW 的工作原理。

然而，VSW 的一个缺点是不能保证最坏情况的窗口大小; 因此在极端情况下输出延迟可以是任意大的。我们通过设置窗口大小的上限来修改算法 1，使得当达到阈值时，算法将输出直到当前阶段的最大似然解。我们将标记这个修改的方法有界滑动窗 (BVSW) 方法。但是与 VSW 不同，这种方法可能导致次优解决方案。

A.4 实验装置

A.4.1 现场测试数据

使用支持 GPS 的智能手机，我们收集了在新加坡的 4 条预定的公共汽车路线的地面实况数据，如图 4 所示。由于我们关心地图匹配结果的路径精度（将在第 6-C 节中定义），因此对实际测试路径（地面实况路径）的了解足以使我们验证算法。为了比较不同环境条件下的表现，我们选择了涵盖新加坡农村和城市地区的 4 条路线。农村路线（R1 和 R2）涉及较少的转弯，并且主要包括通过开放区域的直线路线，例如高速公路。城市路线（U1 和 U2）除了更加高度分支外，还包括密集地包含高层建筑的城市街区。R1，R2，U1 和 U2 的长度分别为 36.3km，11.3km，27.3km 和 32.5km。此外，为了模拟变化的采样频率的轨迹，我们以 10 秒到 5 分钟的采样间隔对我们的原始数据（每 1-3 秒记录一次）进行二次采样。

A.4.2 训练和参数估计

参数 σ_g 需要在（2）中估计体现概率，并且转换概率需要 SVM 训练。

我们通过分析我们的地面实际数据的扰动来估计 σ_g。对于每个轨迹点，我们计算该点离其最近路段中心的大圆距离。然后，基于距离的中值绝对偏差（MAD）计算标准偏差，

$$\sigma_g = 1.4826 median_i(|d_i - median_j(d_j)|).$$

这里 d_i 表示单个轨迹点与其匹配段之间的垂直距离。距离的分布允许我们估计地面真实路径周围的轨迹点的一维扰动。注意在（7）中，MAD 由常数因子 1.4826 缩放，因为我们假设 GPS 测量误差是正态分布的。我们采用 MAD 方法因为它对抗能够更好地对抗数据集中的异常值。在 [7]，[15] 中采用了相同的估计标准差的方法。基于整个数据集，我们获得 σ_g= 6.86m。

为了推断转移概率，我们使用 3,000 个标记的实例训练 SVM 分类器，其中每个实例对应于正确的转换（类别标记为'1'）或不正确的转换（类别标记为'2'）。每个实例是 2D 特征向量（5）和（6）计算得到的得分值，并且两个分量被缩放为 [0,1]。缩放函数是 $(1 + x)^{-1}$，其中 x 是特征的任一分量。使用网格搜索参数空间和 5 重交叉验证，我们发现参数的最佳组合为 C = 0.25 和 g = 0.5，其中 C 是软边际参数，g 是径向基函数（RBF）内核参数。训练结果如图 5 所示。

A.4.3 性能评价

我们将使用两个性能度量来评估我们的算法：精度和输出延迟。

精度被定义为地面实际路径中正确匹配的轨迹点的分数。当轨迹点被映射到包含在地面实况路径中的任何道路段时，记录正确的匹配。这种准确性的测量避免了惩罚"边界类"，其中点位于路口的正中间。我们注意到，精确地确定收集每个轨迹点的路段是不切实际的，原因在于两个原因：（i）"边界类"可以归因于交叉点的任一出口上的路段，以及（ii）可能的误校准的数字地图。因此，路径精度测量是更合适的评估标准。

输出延迟是算法对每个轨迹点引起的平均输出延迟。它通过在获得匹配结果之前花费的时间来量化。

测试如下进行：

- 我们对农村和城市测试路线的不同采样间隔的测试轨迹执行地图匹配，范围从 3 秒到 5 分钟。对于每个路由类别，我们聚合两个测试路由获得的结果。
- 我们比较了三种定位策略在精度和输出延迟方面：VSW，BVSW 和 FSW。

对于 BVSW 和 FWS，测试了不同的窗口尺寸。对于每个窗口大小，我们聚集整组测试数据的结果（4 个测试路由，采样间隔在 10 秒到 5 分钟之间）。

A.5 结果

图 6 展示了城乡测试路线的地图匹配精度的比较。结果表明，除了大于 4 分钟的抽样间隔外，农村路线的准确性比城市路线的准确度大约为 5 %。以小于 1 分钟的间隔，两条路线的精度都高于 0.9。在这两种情况下，精度随着采样间隔的增加而恶化。

图 7 和图 8 中，虚线表示使用 VSW 定位策略实现的最优结果。BVSW 方法在 w = 8 及以上时收敛到 0.921 的最佳精度。在所有情况下，FSW 方法给出一致的较低精度，并且没有收敛到最优解，即使在 w = 20。

在图 8 中，VSW 的平均输出延迟为 82s。与 FSW 相比，它实现了显着更低的延迟，而没有折衷解决方案的最优性。使用 BVSW，在 4 和以上的窗口大小的延迟性能中没有显着的优点。这表明马尔可夫链中的大多数决策点发生在达到窗口界限之前。在 FSW 的情况下，延迟与窗口大小成比例地增加，但是在某个阈值点之后精度增益减小到几乎为零。

A.6 结论和未来工作

在本文中，我们描述了一种用于地图匹配的在线算法，并分析其在地面实况数据上的性能。我们设计了 VSW 和 BVSW 方法来寻找在线解决方案。两者在精度和输出延迟方面都优于先前基于 HMM 的算法中使用的传统 FSW 定位策略。我们还开发了一种数据驱动的方法，用于推断在地图匹配过程中融合传感器测量和拓扑信息的转换概率。总而言之，这些方法提供了用于设计基于在线 HMM 的地图匹配算法的一般框架，其适合于使用浮动车辆数据的实时应用。算法的其它变体可以在估计体现和转移概率时结合附加的传感器数据，例如加速度和高度测量。

对于未来的工作，我们可以探索地图匹配算法的设计与动态参数检测和适应不同的环境设置，如在城市或农村地区，其中 GPS 准确性可能会变化。传感器信息，例如用于 GPS 测量的精度（DOP）值的稀释可能被证明对于实现这个

目标是有用的。此外，我们建议更好的方法 [22] 内插轨迹点，而不是假设它们之间的最短路径。对实际行进路径的更好近似可以提高地图匹配精度。