软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(8):3793−3820 [doi: 10.13328/j.cnki.jos.006542]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

电动自行车轨迹简化与自适应地图匹配算法*

王东京, 刘继涛, 俞东进

(杭州电子科技大学 计算机学院, 浙江 杭州 310018)

通信作者: 俞东进, E-mail: yudj@hdu.edu.cn

摘　要: 近年来, 随着全球定位系统 (global positioning system, GPS) 的大范围应用, 越来越多的电动自行车装配了GPS传感器, 由此产生的海量轨迹数据是深入了解用户出行规律、为城市规划者提供科学决策支持等诸多应用的重要基础. 但是, 电动自行车上普遍使用的价格低廉的GPS传感器无法提供高精度的定位, 同时, 电动自行车轨迹地图匹配过程因以下原因更具有挑战性: (1) 存在大量停留点; (2) 高采样频率导致相邻轨迹点的距离较短; (3) 电动自行车可行驶的路段更多, 存在大量无效轨迹. 针对上述问题, 提出一种可自适应路网精度的电动自行车轨迹地图匹配方法KFTS-AMM. 该方法融合基于分段卡尔曼滤波算法的轨迹简化算法 (KFTS), 和分段隐马尔可夫模型的地图匹配算法 (AMM). 首先, 利用卡尔曼滤波算法可用于最优状态估计的特性, KFTS能够在轨迹简化过程中对轨迹点进行自动修正, 使轨迹曲线变得平滑并减少了异常点对于地图匹配准确率的影响. 同时, 使用基于分段隐马尔可夫模型的地图匹配算法AMM, 避免部分无效轨迹对整条轨迹匹配的影响. 此外, 在轨迹数据的处理过程加入了停留点的识别与合并, 进一步提升匹配准确率. 在郑州市真实电动自行车轨迹数据的实验结果表明, KFTS-AMM在准确率上相对于已有的对比算法有较大的提升, 并可通过使用简化后的轨迹数据显著提升匹配速度.

关键词: 地图匹配; 轨迹简化; 卡尔曼滤波; 轨迹数据分析; 隐马尔可夫模型; 停留点

中图法分类号: TP18

中文引用格式: 王东京, 刘继涛, 俞东进. 电动自行车轨迹简化与自适应地图匹配算法. 软件学报, 2023, 34(8): 3793–3820. http://www.jos.org.cn/1000-9825/6542.htm

英文引用格式: Wang DJ, Liu JT, Yu DJ. Trajectory Simplification and Adaptive Map Matching Algorithm for Electric Bicycle. Ruan Jian Xue Bao/Journal of Software, 2023, 34(8): 3793–3820 (in Chinese). http://www.jos.org.cn/1000-9825/6542.htm

Trajectory Simplification and Adaptive Map Matching Algorithm for Electric Bicycle

WANG Dong-Jing, LIU Ji-Tao, YU Dong-Jin

(School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China)

Abstract: With the wide application of global positioning system (GPS), more and more electric bicycles are equipped with GPS sensors. Massive trajectory data recorded by those sensors are of great value in many fields, such as users' travel patterns analysis, decision support for urban planners, and so on. However, the low-cost GPS sensors widely used on electric bicycles cannot provide high-precision positioning. Besides, the map matching for the electric bicycles' track data is more complex and challenging due to: (1) many stay points on electric bicycles' trajectories; (2) higher sampling frequency and shorter distance between adjacent track points on electric bicycle's track data; (3) some roads only open for electric bicycles, and the accuracy of matching is sensitive to the quality of the road network. To solve those issues mentioned above, an adaptive and accurate road network map matching algorithm is proposed named KFTS-AMM, which consists of two main components: the segmented Kalman filtering based trajectory simplification (KFTS) algorithm and segmented hidden Markov model based adaptive map matching (AMM) algorithm. Since Kalman filtering algorithm can be used for optimal state estimation, the trajectory simplification algorithm KFTS can make the trajectory curve smoother and reduce the impact of abnormal points

* 基金项目: 工信部工业互联网创新发展工程 (TC200802C, TC200802G); 浙江省自然科学基金 (LQ20F020015)
收稿时间: 2021-01-07; 修改时间: 2021-08-04, 2021-10-08; 采用时间: 2021-11-17; jos在线出版时间: 2023-01-13
CNKI网络首发时间: 2023-01-19

on the accuracy of map matching by fixing the trajectory points automatically in the process of trajectory simplification. Besides, the matching algorithm AMM is used to reduce the impact of invalid trajectory segments on the map matching accuracy. Moreover, stay points identification and merging step are added into the processing of track data, and the accuracy is further improved. Extensive experiments conducted on the real-world track dataset of electric bicycles in Zhengzhou city show that the proposed approach KFTS-AMM outperforms baselines in terms of accuracy and can speed up the matching process by using the simplified track data significantly.

Key words: map matching; trajectory simplification; Kalman filtering; track data analysis; hidden Markov model (HMM); stay points

近年来, 我国已经成为全球电动自行车生产和销售第一大国, 同时, 电动自行车也已经成为人们出行的主要交通工具之一. 据统计, 2018 年我国电动自行车的保有量已经达到了 2.5 亿[1], 随着 GPS 定位技术的不断成熟, 越来越多的电动自行车被安装了 GPS 设备, 由此获取到的海量轨迹数据为分析城市交通状况和用户出行规律提供了很好的基础[2,3]. 地图匹配算法的任务是将 GPS 记录的轨迹匹配到交通工具实际经过的道路上, 这是对轨迹数据进行深度分析和有效利用的必要步骤. 但是, 电动自行车上普遍使用的成本低廉的 GPS 传感器无法提供高精度的定位, 因此往往无法从原始的轨迹数据中直接得知实际经过的准确路线.

目前, 学者对于地图匹配的研究已经有 20 多年的历史, 专注于解决地图匹配问题的文献也有上百篇之多[4]. 然而, 现有的各类算法大多是研究机动车轨迹数据的匹配问题. 对于电动自行车轨迹的地图匹配算法的相关研究还比较少. 相比于机动车的轨迹数据, 电动自行车轨迹数据具有以下 3 个特点.

特点 1: 电动自行车轨迹上存在大量停留点.

特点 2: 电动自行车轨迹采样频率高, 行驶速度较慢, 导致采样点距离较小, 轨迹点密度相应增大.

特点 3: 电动自行车行驶路段更多, 由于路网精度的限制, 部分轨迹可能无法匹配到对应的路网上.

具体而言, 特点 1 会导致地图匹配准确率的下降. 停留点指轨迹上一组连续且相距较近的轨迹点, 其产生原因分为两类: (1) 由于停止或运动速度极慢; (2) 由于车辆反复行驶过某区域[3]. 这两类原因均会导致文献 [4] 中提到的不必要迂回的出现, 从而降低匹配的准确率. 例如, 在图 1 中, 当车辆在轨迹点 p_2 与 p_3 处行驶速度极慢, 为两个停留点, 由于 GPS 的测量误差[5], p_2 与 p_3 在道路 A 上的投影点前后顺序与道路 A 的方向相反, 导致地图匹配算法计算出 $p_2 \rightarrow p_3$ 的路线为道路 A→道路 B→道路 C→道路 D→道路 A, 与事实情况不符.

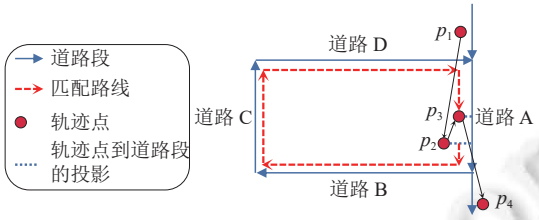
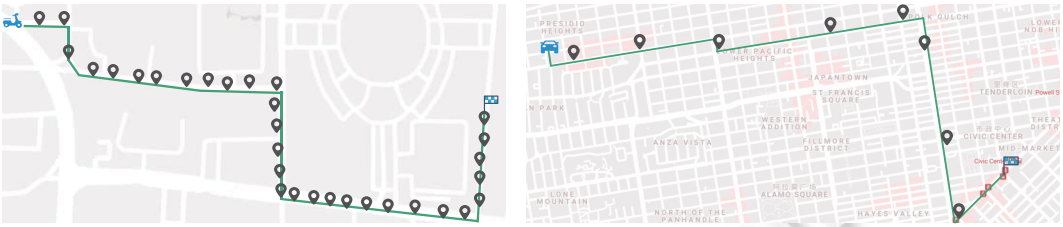


图 1 因停留点导致的轨迹上不必要的迂回

文献 [6] 对 GeoLife[7,8]轨迹数据集中不同出行方式的轨迹进行了分析, 认为自行车与步行出行方式的轨迹数据有更高的方向变化概率与停止概率. 在真实场景中, 机动车只有当等待红绿灯或处于交通拥堵路段时才会导致轨迹上出现停留点, 但是电动自行车用户拥有更大的行驶自由, 例如可以随意在路边停车、在某较小区域内反复行驶, 从而产生更多的停留点. 因此对电动自行车轨迹上的停留点进行预先识别与处理十分必要.

特点 2 会显著地降低地图匹配算法的效率. 电动自行车车速较慢, 因此轨迹点更加密集, 高峰时段下轨迹点密度会进一步提高. 文献 [9] 将采样时间间隔在 1 s 至 10 s 之间的采样方式定义为高频采样, 而时间间隔高于 2 min 的采样方式定义为低频采样. 然而, 这种定义方式只考虑了时间间隔, 忽略了采样点在空间上的距离. 当交通工具速度不同时, 即使采样频率相同, 也可能导致采样点空间距离相差很大. 图 2(a) 显示了一段郑州电动自行车轨迹, 相邻轨迹点平均间距约为 60 m、采样时间间隔约为 12 s, 图 2(b) 显示了一段 CRAWDAD 数据集中旧金山出租车的轨迹[10], 相邻轨迹点平均间距约为 600 m、采样时间间隔约为 30 s. 尽管电动自行车的采样频率约为机动车的 3 倍, 但是轨迹点密度却达到了机动车的 10 倍. 在许多开源轨迹数据集中, 机动车轨迹上的轨迹点密度相对较低.

例如, 开源数据集 T-Drive[11,12]采集了北京市的一万余辆出租车一周内的轨迹数据, 平均采样时间间隔 177 s, 轨迹点间隔 623 m, 文献 [13] 中使用了 2010–2011 年间采集的北京市 6 万余辆出租车轨迹数据, 平均采样时间间隔 70 s, 轨迹点间隔 560 m.



(a) 电动自行车轨迹, 轨迹点间平均距离约为 60 m (b) 出租车轨迹, 轨迹点间的平均距离约为 600 m

图 2 电动自行车与汽车轨迹点间距示例, 地图来源于谷歌地图 (https://www.google.com/maps)

鉴于大量的轨迹数据会带来极大的存储和计算开销, 许多研究人员关注于如何压缩轨迹以提升地图匹配速度[14]. 基于 DP 算法[15]的轨迹简化方法应用最为广泛, 其基于下述假设: 轨迹点的位置记录是交通工具在该时刻位置的真实记录. 由于 GPS 测量存在误差, 当交通工具位于高层建筑旁、桥梁旁时, 误差会急剧增大[5]. 基于 DP 的轨迹简化算法会计算出与原始轨迹形状差异最小的简化轨迹, 因此倾向于保留 GPS 误差较大的轨迹点[4], 进而可能导致匹配准确率下降. 例如, 图 3 中高层建筑旁的轨迹点 GPS 误差很大, 若使用简化后的轨迹进行地图匹配, 将会得到错误匹配结果. 由于卡尔曼滤波算法能够用于最优状态估计, 因此使用卡尔曼滤波算法与轨迹简化算法相结合, 能够在简化过程中不断修正轨迹点. 同时, 现有基于 DP 算法[15]的轨迹简化算法[16–20]依赖于人为设置的距离阈值, 只有简化过程结束后才能计算出轨迹的简化比例.

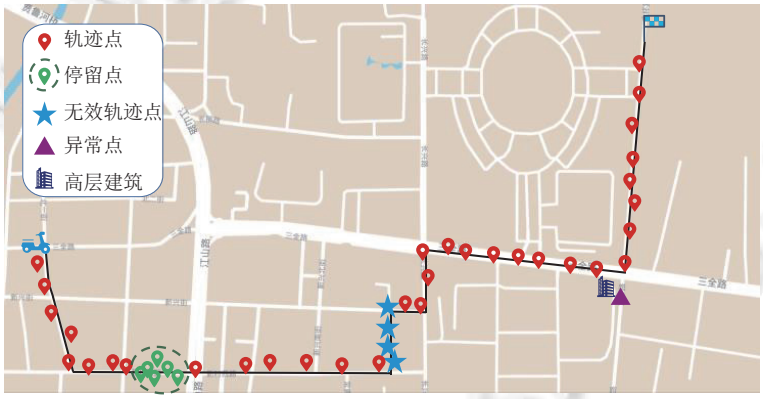


图 3 电动自行车轨迹数据特点

对于电动自行车轨迹数据的特点 3, 我们使用无效轨迹片段来表示轨迹上不存在于给定路网的部分. 当路网精度降低时, 电动自行车的无效轨迹片段会相应增加. 在实际的路网中, 电动自行车能够在一些机动车不能驶入的路段行驶. 例如, 在图 3 中, 星形轨迹点不存在于图中的路网上, 这些轨迹点组成了一个无效轨迹片段. 一些全局匹配算法使用折线间距离度量方法如弗雷歇距离来计算路网上的各条可能路线与待匹配轨迹的相似度[21]. 当轨迹上存在无效轨迹片段时, 这些算法仍会将相似度最大的路线作为匹配结果. 基于状态转移模型的地图匹配算法[14,22–25]则会因无效轨迹片段的存在导致匹配过程中断[4], 无法对轨迹的有效部分计算出匹配路线. 上述两类算法都会因无效轨迹片段的存在而使匹配准确率降低.

为解决现有地图匹配算法因电动自行车轨迹数据的上述特点导致的匹配准确率下降、匹配效率不高的问题, 本文提出了一种可自适应路网精度的电动自行车轨迹地图匹配方法 KFTS-AMM. 该方法融合了基于卡尔曼滤波

算法的轨迹简化算法 (KFTS), 以及分段的隐马尔可夫模型 (hidden Markov model, HMM) 的地图匹配算法 (AMM). 本文的贡献主要集中在以下 3 个方面.

(1) 提出了基于时空聚类的停留点处理算法, 能够准确地识别与合并停留点, 减少轨迹上的迂回, 从而提高地图匹配算法的准确率.

(2) 提出了基于卡尔曼滤波算法的轨迹简化算法 KFTS, 改进了 DPhull 算法, 使其不依赖人为设置的距离阈值, 仅使用轨迹简化比例作为简化过程参数. 由于轨迹简化算法倾向于保留误差较大的轨迹点[4], 因此在轨迹简化过程中加入滤波操作对轨迹点加以修正, 能够提升后续地图匹配过程的准确率.

(3) 使用分段的 HMM 的自适应地图匹配算法 AMM, 能够在匹配的过程中不断检测出无效的轨迹片段, 同时将有效的轨迹片段匹配到给定精度的路网图上, 在不同精度的路网上均能取得良好的准确率.

本文第 1 节回顾了地图匹配算法、轨迹简化算法与轨迹停留点识别的相关研究工作. 第 2 节介绍相关的定义以及问题描述. 第 3 节详细介绍了本文提出的算法. 第 4 节展示了在真实数据集上的对比实验结果. 第 5 节总结了现有工作以及未来工作的重点.

1 相关工作

在本节中, 我们回顾了地图匹配、轨迹简化以及轨迹停留点识别的相关研究工作.

1.1 地图匹配

文献 [26] 提出根据轨迹匹配过程中的采样点范围将地图匹配算法分为局部/增量算法[27–29]与全局算法[21,30]. 局部/增量算法使用轨迹的局部特征进行匹配, 计算速度快, 适用于高频率采样的轨迹, 但当路网密集度较高时, 准确率普遍较低. 全局算法使用整条轨迹进行匹配, 为轨迹寻找一条相似度最高的路线, 计算复杂度较高.

Quddus 等人根据地图匹配算法所使用的技术将地图匹配算法分为以下 4 类[26]: 基于几何关系的匹配算法[31]、基于拓扑关系的匹配算法[32]、基于概率统计的匹配算法[33]和其他匹配算法 (例如: 使用了诸如扩展卡尔曼滤波器[34]、模糊逻辑[35]、证据理论[36]和贝叶斯推理[37]等技术). 但是, 随着近年来又有许多新的方法被提出, 这一分类标准已经不再适用. Chao 等人在文献 [4] 中提出根据核心匹配模型将地图匹配算法分为 4 个类别: 相似度模型[21,27]、状态转移模型[9,14,23,24,38]、候选进化模型[39]和评分模型[40]. 同时, Chao 等人还列举了 3 种给地图匹配带来挑战的数据质量问题, 包括不必要的迂回、异常点和路网中道路的密度, 这 3 种问题均会降低地图匹配算法的准确率. 在基于状态转移模型的地图匹配算法中, 基于隐马尔可夫模型的地图匹配算法[14,22–25,38,41]是最流行的, 同时也被证实具有较高的准确率[42].

一些研究人员致力于提高地图匹配效率. 地图匹配效率的提升可以通过以下 4 种方式[14]: 使用空间索引技术[43,44]以加快对某个轨迹点的近邻点与近邻边的查找速度、避免路网图中最短路径的重复计算[14,45]、使用分布式与并行计算技术[22,46–48]、压缩轨迹以减少参与计算的轨迹点.

上述地图匹配算法设计的实验所使用的数据集多为机动车轨迹数据集, 由于电动自行车轨迹上存在大量停留点、采样频率高、轨迹点密集且存在大量无效轨迹片段的特点, 已有算法不适宜直接应用在电动自行车轨迹数据上.

1.2 轨迹简化

轨迹简化算法是目前最主流的轨迹数据压缩处理方法[49], 将轨迹数据视为一条连续的折线, 通过删除部分对折线形状影响较小的轨迹点, 实现简化后折线与原始折线的近似拟合, 从而完成轨迹数据的压缩. 根据轨迹简化算法是否适用于实时计算, 可以将现有轨迹简化算法划分为在线轨迹简化算法与离线轨迹简化算法[2].

离线轨迹简化算法没有输入轨迹规模的限制, 能够得到全局最优的简化结果. Bellman 算法[50]是最早用于解决轨迹简化问题的算法, 利用了动态规划技术, 但是计算复杂度较高, 达到了 $O(n^3)$. DP 算法[15]是另一种经典的轨迹简化算法, 在最坏情况下的时间复杂度为 $O(n^2)$. 许多研究人员为了提升 DP 算法的效率提出了改进方法, 如 DPhull 算法[16]引入了凸包技术, 使算法在最坏情况下的时间复杂度下降到了 $O(n\log n)$. 现有基于 DP 算法的轨迹简化算法大多依赖于人为设置的距离阈值, 无法直接确定最终的简化比例. 文献 [51] 提出了不使用距离阈值作为参数的

简化方法, 使用递归的方式, 动态调整距离阈值以控制简化比例, 但是易造成轨迹上不同片段简化比例的不平衡. 文献 [52] 提出的简化算法首先依据速度对轨迹分段, 再结合 DP 算法对轨迹简化, 以保留轨迹时空特征.

在线场景中, 需要实时对轨迹数据进行简化, 因此必须根据计算设备的存储与计算性能限制轨迹的输入规模. Sliding window (SW) 算法[17]和 opening window (OW) 算法[18]都使用了能够动态调整大小的滑动窗口作为轨迹点缓冲区, 利用两个轨迹点作为起止点标记一个滑动窗口, 当窗口内某轨迹点与窗口首尾轨迹点连线的欧式距离超过设定阈值时, 将窗口的起始轨迹点向前移动. 这两种算法不同之处在于, SW 算法将起始轨迹点移动到终止轨迹点的前一个轨迹点处, 而 OW 算法则将其移动到距离首尾连线距离最大的轨迹点处. STTrace 算法[19]与 SQUISH 算法[20]设置了固定大小的缓冲区, 同样根据窗口内各轨迹点距首尾轨迹点连线的距离大小来决定是否将窗口向前滑动.

1.3 停留点识别

停留点是指在某一段连续的时间内, 位置变化小于一定距离的一组轨迹点. 当车辆运动速度过于缓慢 (例如用户推行电动自行车)、暂时停止行驶, 或在较小区域内反复行驶时, 就会产生停留点.

停留点识别方法有 3 类: 基于差异的方法[8,53]、基于历史数据与额外位置信息的方法[8]和基于聚类的方法[54,55]. 其中, 基于差异的方法需要通过设置速度、距离或者时间阈值来识别停留点, 这种方法在拥有足够先验知识的情况下才能够有较高的准确率; 基于历史数据与额外位置信息的方法通过用户历史出行数据挖掘用户的出行模式, 进而结合关键位置信息推测停留点; 基于聚类的方法使用聚类算法寻找轨迹点聚类, 但是这种方法同样依赖输入的参数, 如距离和时间阈值. 当距离阈值与时间阈值设置合理时, 基于聚类的方法在没有历史数据和位置信息的情况下仍能达到较高的准确率. 文献 [8] 设置了时间与距离阈值, 线性扫描轨迹以识别停留点, 然后以求均值的方式合并停留点. 然而这种识别方法只考虑了相邻位置的时空关系, 当轨迹在一段区域内反复迂回, 则无法有效识别出停留点.

DBSCAN 算法[56]是一种基于密度的聚类算法, 能够识别任意形状的聚类, 且不需要预先确定聚类个数. 但是由于时空数据的时间属性和空间属性尺度不同, 因此该算法不能很好地解决时空数据的聚类问题. 为此, Birant 等人提出了 ST-DBSCAN 算法[57], 该算法使用了空间邻域半径和时间邻域半径作为聚类内轨迹点的距离阈值, 能够很好地解决时空数据的聚类问题. 本文 StayPointsProcess 算法即使用 ST-DBSCAN 算法完成停留点的识别.

1.4 卡尔曼滤波

卡尔曼滤波算法[58]是一种利用线性系统状态方程, 迭代地对系统每一时刻状态做出最优估计的算法. 许多研究人员对其进行了创新, 使卡尔曼滤波算法也可应用于非线性系统, 例如使用泰勒级数展开将非线性系统线性化的扩展卡尔曼滤波算法 (extended Kalman filtering, EKF)[59,60]和结合了无迹变换 (unscented transform, UT) 的无迹卡尔曼滤波算法 (unscented Kalman filtering, UKF)[61]. 卡尔曼滤波算法在许多领域都有巨大的应用价值, 例如车辆状态预测[62]、动态目标跟踪[63]、传感器信息融合[64]等. 卡尔曼滤波算法能够融合多种传感器数据, 减小仅使用 GPS 传感器对车辆定位时的误差. 文献 [34] 使用车内里程表、陀螺仪与 GPS 传感器数据, 通过卡尔曼滤波算法构建车体的运动模型, 然后结合地图匹配算法对车辆精确定位. 文献 [65] 提出了将卡尔曼滤波算法与地图匹配算法融合入动态的贝叶斯网络中, 同样使用了多种传感器数据对车辆运动模型建模以达到对车辆的精确定位.

2 基本概念和问题描述

在本节中, 我们给出了本文中出现的一些关键变量和符号的说明, 如表 1 所示. 同时, 我们给出了本文相关概念的定义以及所要解决的问题的描述.

定义 1. 轨迹点. 轨迹点 p 为 GPS 传感器的一条记录, 代表了电动自行车的时空状态, 可以由一个三元组表示, $p:<lon,lat,t>$ 其中, lon 和 lat 分别表示电动自行车所处位置的经度和纬度, t 表示这条记录的时间戳. 由于传感器存在测量误差, 因此轨迹点 p 的位置测量值并不完全等于真实位置.

表 1 关键变量说明

变量名	说明
$p \in Tr$	轨迹Tr上的轨迹点p
$G(V,E)$	路网图, 且为有向图
V, E	路网图中的顶点集合和有向边集合, 分别表示道路端点和路段
$e.start, e.end, \forall e \in E$	路段的起点和终点
R	路线

定义 2. 轨迹. 轨迹$Tr: p_1 \to p_2 \to \ldots \to p_n$是由一组按照时间顺序排列的轨迹点组成的序列. 图 4 中$p_1 \to p_2 \to \ldots \to p_{14}$展示了一段轨迹的示例.

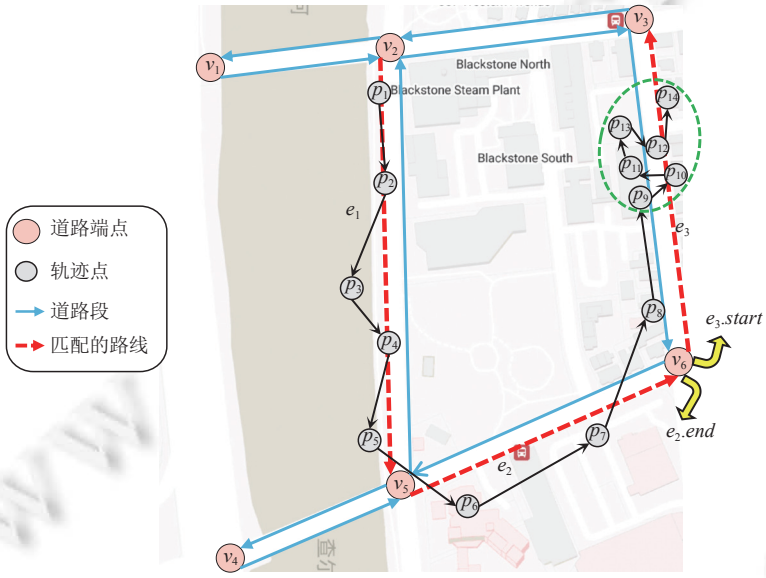


图 4 电动自行车轨迹示例

定义 3. 路网图. 路网图$G(V,E)$是一个有向图, 其中, (1) V是路网图中的顶点集合, 顶点$v \in V$, 表示道路的端点; (2) E是路网图中有向边的集合, 有向边$e \in E$, 在路网中表示道路段, 可表示为一个二元组$e :< e.start, e.end >$, $\forall e.start, e.end \in V$. 在图 4 中, 顶点$v_1, v_2, \ldots, v_6$为路网图上的道路端点, 顶点间的有向边为路网图中的道路段. 顶点与顶点间的有向边共同组成了路网图.

定义 4. 路线. 路线$R: e_1 \to e_2 \to \ldots \to e_n$是由一组路网图中的相连道路段组成的序列, $\forall e_i \in E, e_{i+1}.start = e_i.end, i \in [1, n-1]$. 在图 4 中, $e_1 \to e_2 \to e_3$为图中轨迹$p_1 \to p_2 \to \ldots \to p_{14}$实际经过的路线.

定义 5. 停留点. 轨迹$Tr: p_1 \to p_2 \to \ldots \to p_n$上, 存在一条子轨迹$Tr_{\text{sub}}: p_a \to p_{a+1} \to \ldots \to p_b, 1 \leqslant a \leqslant b \leqslant n$, 且$\forall a \leqslant i \leqslant j \leqslant b, dist(p_i, p_j) \leqslant threshold$, 其中$dist(p_i, p_j)$为轨迹点$p_i$和$p_j$的距离, $threshold$为停留点距离阈值, 则称Tr_{sub}上的轨迹点为停留点. 在图 4 中, 虚线框中的轨迹点$p_9 \to p_{10} \to \ldots \to p_{14}$为一组停留点的示例.

轨迹简化问题描述: 对于给定的原始轨迹$Tr: p_1 \to p_2 \to \ldots \to p_n$, 使用轨迹简化算法将计算出一条新的轨迹$Tr': p'_1 \to p'_2 \to \ldots \to p'_m$, $m < n$, 是原始轨迹Tr的子序列, 即轨迹Tr'上的轨迹点也存在于原始轨迹Tr上, 称Tr'上的轨迹点为特征点. 轨迹简化比例$ratio = m/n, ratio \in (0,1]$是指简化后的轨迹$Tr'$与原始轨迹$Tr$的轨迹点数量之比, $ratio$越大, 轨迹简化过程保留的轨迹点越多.

地图匹配问题描述: 给定一条轨迹$Tr: p_1 \to p_2 \to \ldots \to p_n$和路网图$G(V,E)$, 地图匹配算法将计算出轨迹$Tr$实际经过的路线$R: e_1 \to e_2 \to \ldots \to e_m$. 在图 4 中, 路线$e_1 \to e_2 \to \ldots \to e_m$是轨迹$p_1 \to p_2 \to \ldots \to p_{14}$真实经过的

路线.

3 基于卡尔曼滤波的轨迹简化与自适应地图匹配算法 (KFTS-AMM)

在本节中, 我们详细地描述了本文提出的可自适应路网精度的电动自行车轨迹地图匹配方法 KFTS-AMM. 其整体的架构图如图 5 所示, 主要由 4 个部分组成, 分别是轨迹提取、停留点识别与合并、轨迹简化与地图匹配, (1) 轨迹提取: 删除 GPS 日志中存在属性缺失、冗余和存在明显错误的记录. 对 GPS 日志按照电动自行车设备 ID 与记录时间进行排序, 通过设置时间阈值分割提取出轨迹数据; (2) 停留点处理: 使用时空聚类算法识别停留点, 结合旋转卡壳算法 (rotating calipers algorithm)[66]对停留点进行合并; (3) 轨迹简化: 使用分段的卡尔曼滤波算法对轨迹进行修正, 然后使用基于 DPhull 算法[16]改进的 DPhull-ratio 算法进行轨迹简化; (4) 地图匹配: 使用分段的隐马尔可夫模型 (HMM) 地图匹配算法进行地图匹配.

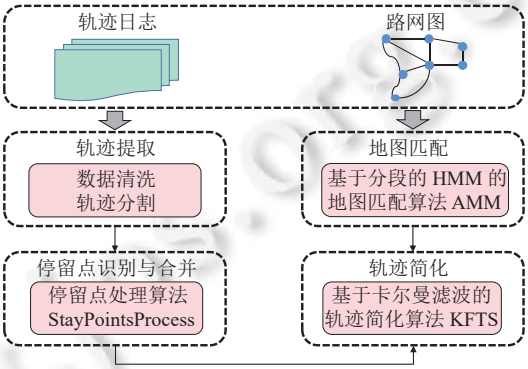


图 5 KFTS-AMM 算法架构图

本文提出的算法涵盖了电动自行车轨迹数据的清洗、轨迹提取、停留点处理、轨迹压缩简化以及地图匹配过程. 在实际应用场景中, 能够兼顾地图匹配的效率和准确率. 同时, 简化后的轨迹可以有效降低数据的存储空间.

3.1 轨迹提取

3.1.1 轨迹数据清洗

本文使用的是郑州市采集的真实电动自行车行驶数据. 数据传输过程中存在部分数据丢失和部分数据传输错误的情况, 因此需要对初始采集数据进行清洗. 原始日志字段包括电动自行车 ID、数据采集时间、数据产生时间、经度与纬度. 其中, 电动自行车设备 ID 和数据产生时间可以唯一确定一条数据. 轨迹数据清洗的过程包括以下 3 个步骤: (1) 删除存在属性缺失的记录; (2) 对电动自行车设备 ID 和时间戳属性均相同的重复记录, 只保留其中一条; (3) 由于已知郑州市的经纬度范围为东经 112.42°至东经 114.14°, 北纬 34.16°至北纬 34.58°, 因此将经度或纬度超出这一范围的异常记录删除.

3.1.2 轨迹数据的分割

交通部门使用安装在电动自行车上的边缘设备采集轨迹数据. 首先按照电动自行车设备 ID 对日志数据进行分割, 然后再按照数据产生时间进行排序, 通过这种方式可获取每辆电动自行车按照时间顺序排列的 GPS 日志序列. 为了从排序后的 GPS 日志中提取电动自行车的轨迹数据, 还需要按照时间间隔对其进行分割. 当电动自行车长时间静止后, GPS 传感器会停止采集数据. 因此, 我们使用了固定的时间阈值来分割不同的轨迹, 我们将时间阈值设置为 3 min. 若排序后的 GPS 日志中相邻记录的数据采集时间超过 3 min 时, 可以认为其属于不同轨迹.

3.2 停留点识别与合并

本节详细分析了电动自行车轨迹的停留点分布特点, 并据此特点提出了电动自行车轨迹上对停留点的处理算

法 StayPointsProcess.

停留点是指在某一段连续的时间内, 位置变化小于一定距离的一组轨迹点. 当车辆运动速度过于缓慢 (例如用户推行电动自行车)、暂时停止行驶或者在某一较小区域内反复行驶时, 就会产生停留点. 在实际场景中, 用户会在骑行电动自行车的途中停车或下车推行并保持一定时间, 此时会随之产生停留点. 图 6 显示了一段真实轨迹与其 GPS 测量记录. 当电动自行车速度接近于零时, 出现了一组停留点, 由于 GPS 测量值在真实位置附近呈混合高斯分布, 因此停留点的 GPS 记录分布在图中椭圆形虚线框范围内. 图 7 展示了 StayPointsProcess 算法的过程, 图中左侧椭圆形虚线框内为一组停留点, 右侧多边形框为停留点的凸包, 凸包直径上的点为停留点合并得到的新轨迹点, 该点位于凸包的直径上. 在图 7 中, 使用均值计算得到的轨迹合并点, 受停留点密度影响, 未分布在这一区域的中心位置, 若以此点作为合并点, 则可能导致合并后的轨迹不平滑.

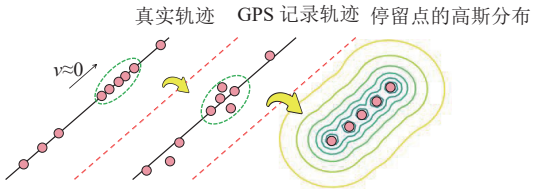


图 6 停留点分布示例

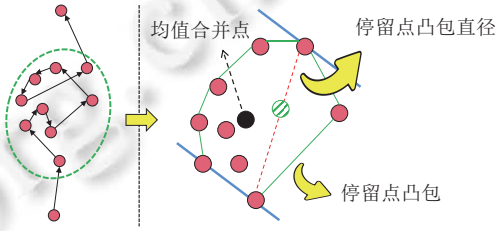


图 7 停留点处理算法 StayPointsProcess 说明

StayPointsProcess 算法首先使用 ST-DBSCAN 算法[57]完成停留点的识别, 然后通过旋转卡壳算法[66]求出停留点的凸包并计算凸包直径, 最后使用电动自行车的平均速度在直径上取出若干点代替停留点, 从而形成新的轨迹. 若轨迹中停留点的个数为 p, 则构造凸包的时间复杂度为 $O(p\log p)$, 设凸包上点个数为 q, $q\leqslant p$ 使用旋转卡壳算法计算此凸包直径的时间复杂度为 $O(q)$, 则停留点的合并过程时间复杂度为 $O(p\log p)$. 若轨迹上的轨迹点个数为 n, $n\geqslant p$, 则 ST-DBSCAN 的平均时间复杂度为 $O(n\log n)$. 综上, StayPointsProcess 算法的整体时间复杂度为 $O(n\log n)$.

算法 1 StayPointsProcess 描述了停留点的处理过程, 其输入包括原始轨迹 Tr、空间邻域半径 eps_1、时间邻域半径 eps_2、停留点最小邻居数 $minPts$ 以及距离计算指标 $metric$, 默认的距离计算指标为曼哈顿距离 (Manhattan distance). 算法输出为新轨迹 Tr'. 算法 1 第 11 行 $Diameter_{\text{stay}}$ 为使用旋转卡壳算法 RC 计算出的停留点凸包直径, 第 12 行 $merge(Diameter_{\text{stay}}, v_{\text{avg}})$ 为停留点的合并过程, 其中 v_{avg} 为轨迹的平均速度.

算法 1 StayPointsProcess 的具体步骤如下.

• 第 1 步 (第 1–4 行): 初始化当前停留点集合、输出轨迹为空. 构造聚类模型并拟合数据, 标记每个轨迹点是否为噪声点. 计算整个轨迹的全局平均速度.

• 第 2 步 (第 6 行): 判断当前点是否为噪声点, 即该点是否不属于任何聚类, 若该点为噪声点, 则执行步骤 3; 若不为噪声点, 则执行步骤 4.

• 第 3 步 (第 7–14 行): 获取当前聚类的标签, 向后遍历轨迹点, 将属于同一聚类的轨迹点加入当前停留点集合. 然后使用旋转卡壳算法计算停留点凸包直径, 在直径上取若干个点作为合并后的轨迹点, 加入输出轨迹. 清空停留点集合.

• 第 4 步 (第 16 行): 将非噪声轨迹点加入输出轨迹.

算法 1. 停留点处理算法 StayPointsProcess.

输入: 原始轨迹 $Tr: p_1\to p_2\to\ldots\to p_n$, 空间邻域半径 eps_1, 时间邻域半径 eps_2, 停留点最小邻居数 $minPts$, 距离计算方法 $metric$;

输出: 停留点识别合并后的轨迹 Tr'.

1. $s_{\text{stay}} \leftarrow \varnothing, Tr' \leftarrow \varnothing$; // 停留点集合 s_{stay} 初始化为空集合
2. $model \leftarrow ST\text{-}DBSCAN(eps_1, eps_2, minPts, metric \leftarrow$ 'Manhattan'); // 构建聚类模型
3. $labels \leftarrow model.fit(Tr)$; // 使用聚类模型拟合数据
4. $v_{\text{avg}} \leftarrow \sum_{i=1}^{n-1} dis(p_i, p_{i+1})/(p_n.t - p_1.t)$; // 计算轨迹的全局平均速度
5. **for** $i = 1$ **to** n **do**
6. **if** $labels[i] \neq -1$ **do** // 判断该点是否属于某个簇类
7. $label_{\text{cur}} \leftarrow labels[i]$;
8. **while** $labels[i] = label_{\text{cur}}$ **do**
9. $s_{\text{stay}}.add(Tr[i])$;
10. $i \leftarrow i + 1$;
11. **end while**
12. $Diameter_{\text{stay}} \leftarrow RC(s_{\text{stay}})$; // 使用旋转卡壳算法计算停留点凸包的直径
13. $Tr'.add(merge(Diameter_{\text{stay}}, v_{\text{avg}}))$; // 在凸包直径上取若干轨迹点, 加入新轨迹
14. $s_{\text{stay}}.clear()$; // 清空当前停留点集合
15. **else**
16. $Tr'.add(Tr[i])$;
17. **end if**
18. **end for**

3.3 轨迹简化

本节详细介绍了本文所提出的基于分段的卡尔曼滤波的轨迹简化算法 KFTS. 该算法首先使用分段的卡尔曼滤波算法修正轨迹曲线, 然后使用基于 DPhull 算法改进的 DPhull-ratio 算法进行轨迹简化, 得到简化后的轨迹.

3.3.1 基于轨迹简化比例的轨迹简化算法 DPhull-ratio

现有的离线与在线轨迹简化算法大多使用距离阈值作为轨迹简化过程的参数[49], 只有当简化过程终止时, 才能计算出轨迹简化比例. 但是, 距离阈值的设置依赖于先验知识和对轨迹数据的初步统计结果, 若设置不够合理, 则无法得到期望的简化比例. 文献 [51] 使用递归的方式, 动态调整距离阈值以控制简化比例, 易造成轨迹上不同片段简化比例的不平衡. 本文在 DPhull 算法的基础上, 提出了基于轨迹简化比例的轨迹简化算法 DPhull-ratio, 能够仅使用轨迹简化比例作为参数, 得到简化结果. 利用优先队列存储轨迹上的子轨迹段, 以达到不同片段简化比例的平衡.

DP 算法的计算过程如图 8 所示, 设待简化的轨迹为 $Tr: p_1 \rightarrow p_2 \rightarrow \ldots \rightarrow p_6$, 距离阈值为 $D_{\text{threshold}}$. 首先, 将轨迹起点 p_1 与终点 p_6 设置为特征点, 并做连线, 由于 p_4 与首尾连线垂直距离最远, 且超过了距离阈值 $D_{\text{threshold}}$, 因此 p_4 被作为轨迹的特征点. 然后, 在 p_4 处分割原轨迹 Tr, 得到子轨迹段 $p_1 \rightarrow p_2 \rightarrow p_3 \rightarrow p_4$ 与 $p_4 \rightarrow p_5 \rightarrow p_6$. 对这两段子轨迹分别递归执行上述过程, 最终确定特征点为 p_1、p_2、p_4 与 p_6, 简化后轨迹为 $Tr': p_1 \rightarrow p_2 \rightarrow p_4 \rightarrow p_6$. 文献 [51] 在 DP 算法的基础上, 实时计算当前简化比例, 若达到输入简化比例, 则简化过程结束. 这种方式易造成轨迹不同片段简化比例的不平衡. DPhull 算法引入了凸包的结构, 能够快速查找距离首尾连线距离最大的轨迹点, 使算法的最坏时间复杂度下降到了 $O(n \log n)$.

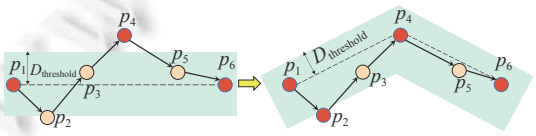


图 8 DP 算法示意图

当需要获得某一设定简化比例的简化结果, 使用 DP 算法 (包括 DPhull) 需要对距离阈值参数多次尝试, 导致计算资源的浪费. 为了解决此问题, 本文提出了 DPhull-ratio 算法, 一次运行即可得到期望的简化比例. DPhull-ratio 算法基于贪心的思想, 维护了一个优先队列, 队列中每个节点记录了一段子轨迹和距离子轨迹首尾连线垂直距离最大的轨迹点, 称其为分割点, 节点的优先级即为分割点距离首尾连线的距离. 这样设置优先级的原因在于, 轨迹段上距离首尾连线垂直距离更大的轨迹点更能代表轨迹段的轮廓. DPhull-ratio 算法每次取出一个队头节点, 在分割点处对子轨迹二分, 并计算分割后两个子轨迹的分割点、优先级用于构建新的队列节点, 最后将新节点加入优先队列.

DPhull 算法在递归过程中, 每次选出距子轨迹段首尾连线距离最大, 且超过距离阈值的轨迹点作为特征点, 这一选择依据与 DPhull-ratio 算法的优先级设置方式相似. 因此, 对于同一条轨迹, 当 DPhull-ratio 算法与 DPhull 算法得到同一简化比例的结果时, 所选取的特征点集合应具有较大相似度, 与原始未简化轨迹的误差也应该较为接近.

在图 9 中, 待简化轨迹为 $Tr: p_1 \to p_2 \to \ldots \to p_{10}$, 设轨迹简化比例为 50%. 优先队列中初始只存在一个节点, 节点代表的轨迹段为 $p_1 \to p_{10}$, 分割点为 p_6, 表示为 $< p_1 \to p_{10}, p_6 >$. 然后, 将其作为队列头节点取出, 令 p_6 为特征点, 并在此处分割. 分割后得到两个新节点, 分别为 $< p_1 \to p_6, p_4 >$ 与 $< p_6 \to p_{10}, p_8 >$, 将其加入队列. 迭代上述过程, 当选取的特征点个数达到总轨迹点个数的 50% 时, 简化过程结束. 算法 2 描述了上述计算过程.

DPhull-ratio 算法增加了维护优先队列的消耗, 设优先队列长度为 n, 则优先队列的插入与删除操作时间复杂度为 $\mathrm{O}(\log n)$. 若轨迹简化比例为 $ratio$, 则用于维护优先队列的时间复杂度为 $\mathrm{O}(ratio \times n \log(ratio \times n))$. 每次取出节点后, 需要使用凸包计算分裂后节点的分割点, 时间复杂度为 $\mathrm{O}(\log n)$, 因为共需取出 $ratio \times n$ 次队头结点, 总时间消耗为 $\mathrm{O}(ratio \times n \log n)$. 综上, DPhull-ratio 算法的时间复杂度为 $\mathrm{O}(ratio \times n \log(ratio \times n) + ratio \times n \log n)$, 即 $\mathrm{O}(ratio \times n \log n)$.

算法 2. 基于压缩率的轨迹简化算法 DPhull-ratio.

输入: 轨迹 $Tr: p_1 \to p_2 \to \ldots \to p_n$, 轨迹简化比例 $ratio$;
输出: 简化后的轨迹 Tr'.

1. $Reserve \leftarrow \{0\}$; // $Reserve$ 数组表示每个轨迹点是否为特征点, 数组长度为 n
2. $Reserve[0] \leftarrow 1$, $Reserve[n-1] \leftarrow 1$; // 初始化首尾轨迹点为特征点
3. $cnt \leftarrow 2$; // cnt 为特征点个数
4. $Init(Heap)$; // $Heap$ 为优先队列, 对其进行初始化
5. **while** $cnt/n < ratio$ **do**
6. $\quad Tr_{\mathrm{sub}}, p_{\mathrm{split}} \leftarrow Heap.top()$; // 选取队头轨迹段, 分割点
7. $\quad Reserve[p_{\mathrm{split}}.index] \leftarrow 1, cnt \leftarrow cnt + 1$; // 分割点设置为特征点, 特征点数加 1
8. $\quad Heap.pop()$; // 将队头元素弹出
9. $\quad Tr_{\mathrm{sub\_1}}, Tr_{\mathrm{sub\_2}} \leftarrow split\left(Tr_{\mathrm{sub}}, p_{\mathrm{split}}\right)$; // 在分割点处二分轨迹段
10. $\quad p_{\mathrm{split\_1}} \leftarrow hull\left(Tr_{\mathrm{sub\_1}}\right)$; // 使用凸包计算子轨迹段 $Tr_{\mathrm{sub\_1}}$ 的分割点
11. $\quad p_{\mathrm{split\_2}} \leftarrow hull\left(Tr_{\mathrm{sub\_2}}\right)$; // 计算子轨迹段 $Tr_{\mathrm{sub\_2}}$ 的分割点
12. $\quad < Tr_{\mathrm{sub\_1}}, p_{\mathrm{split\_1}} > .priority \leftarrow Dis\left(p_{\mathrm{split\_1}}, \overline{Tr_{\mathrm{sub\_1}}}\right)$; // 节点的优先级为分割点与轨迹段首尾连线的垂直距离
13. $\quad < Tr_{\mathrm{sub\_2}}, p_{\mathrm{split\_2}} > .priority \leftarrow Dis\left(p_{\mathrm{split\_2}}, \overline{Tr_{\mathrm{sub\_2}}}\right)$; // 计算第 2 个节点的优先级
14. $\quad Heap.push\left(< Tr_{\mathrm{sub\_1}}, p_{\mathrm{split\_1}} >\right)$; // 将新节点加入优先队列
15. $\quad Heap.push\left(< Tr_{\mathrm{sub\_2}}, p_{\mathrm{split\_2}} >\right)$;
16. **end while**

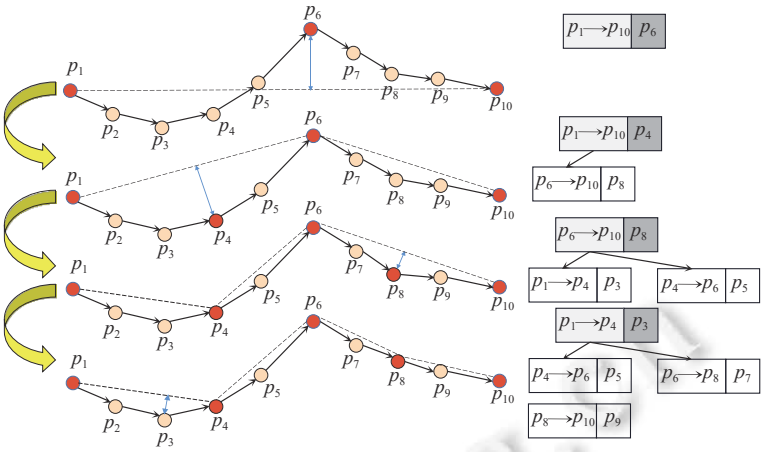


图 9 DPhull-ratio 算法示意图

DPhull-ratio 算法的具体步骤如下.

• 第 1 步 (第 1–4 行): 初始化 *Reserve* 数组, 将首尾节点设置为特征点, 特征点个数设置为 2, 初始化优先队列.

• 第 2 步 (第 5 行): 判断特征点个数与总轨迹点个数比例是否达到轨迹简化比例 *ratio*, 若达到, 则算法结束, 否则执行步骤 3.

• 第 3 步 (第 6–9 行): 取出队头节点, 将分割点设置为特征点, 并在分割点处分割轨迹.

• 第 4 步 (第 10–15 行): 利用凸包分别计算步骤 3 得到的子轨迹段的分割点, 构造两个新节点, 并计算两个新节点的优先级, 然后加入优先队列.

3.3.2 卡尔曼滤波轨迹修正算法

卡尔曼滤波算法可以为每一时刻的状态进行最优估计, 其基于两个方程, 分别是系统状态的预测方程:

$$X_k = AX_{k-1} + q_k \tag{1}$$

和系统状态的观测方程:

$$Z_k = HX_k + r_k \tag{2}$$

在公式 (1) 中, X_k 为系统的状态向量, A 为系统的状态转移矩阵 (表示对系统状态转移的理想估计), q_k 为系统的预测过程噪声, 服从均值为 0, 协方差矩阵为 Q 的正态分布. 在公式 (2) 中, Z_k 为系统的观测向量, H 为系统的观测矩阵, r_k 为观测噪声, 服从均值为 0, 协方差矩阵为 R 的正态分布. 本文所使用的电动自行车轨迹只包含 GPS 数据, 因此卡尔曼滤波轨迹修正算法仅使用轨迹点的经纬度信息用于构建预测与观测方程.

卡尔曼滤波算法迭代地进行预测过程与更新过程. 预测过程使用 2 个公式:

$$\overline{x_k} = Ax_{k-1} \tag{3}$$

$$\overline{P_k} = AP_{k-1}A^{\mathrm{T}} + Q \tag{4}$$

公式 (3) 中, 先验状态向量 $\overline{x_k}$ 为上一时刻的后验状态向量 x_{k-1} 与状态转移矩阵 A 的乘积, 公式 (4) 计算先验卡尔曼误差估计协方差矩阵 $\overline{P_k}$, P_{k-1} 为上一时刻的后验卡尔曼误差估计协方差估计矩阵. 更新过程使用 3 个公式, 分别为:

$$K_k = \overline{P_{k-1}}H^{\mathrm{T}}\left(H\overline{P_{k-1}}H^{\mathrm{T}} + R\right)^{-1} \tag{5}$$

$$x_k = \overline{x_k} + K_k(Z_k - H\overline{x_k}) \tag{6}$$

$$P_k = (I - K_kH)\overline{P_k} \tag{7}$$

公式 (5) 计算得到的卡尔曼增益矩阵 K_k 决定了最优估计更接近预测值还是更接近观测值, 公式 (6) 计算得到的后验状态向量 x_k, 是经过卡尔曼滤波修正后的状态, 公式 (7) 计算后验卡尔曼误差估计协方差矩阵 P_k.

算法 3 KF 描述了对输入轨迹段进行卡尔曼滤波轨迹修正的过程, 算法的输入包括轨迹段Tr_{sub}、坐标预测误差方差var_{predict}、坐标观测噪声方差var_{gps}, 输出为修正过的轨迹Tr'_{sub}. 算法 2 中第 9 行、第 10 行、第 11 行和第 13 行分别对应公式 (3)、公式 (4)、公式 (5), 算法第 13 行对应公式 (6), 算法第 14 行对应公式 (7).

算法 3. 卡尔曼滤波轨迹修正算法 KF.

输入: 轨迹$Tr_{\text{sub}}: p_1 \to p_2 \to \ldots \to p_n$, 预测误差方差$var_{\text{predict}}$, GPS 观测误差方差$var_{\text{gps}}$;
输出: 修正后轨迹Tr'_{sub}.

1. $Tr'_{\text{sub}} \leftarrow \varnothing$;
2. $Init(P,A,H,Q,R)$; // 初始化P、A、H、Q与R
3. **for** $i=1$ **to** n **do**
4. **if** $i=1$ **do**
5. $X \leftarrow \begin{pmatrix} Tr_{\text{sub}}[1].x & Tr_{\text{sub}}[1].y & 0 & 0 \end{pmatrix}^{\text{T}}$; // 初始状态向量
6. **else**
7. $\Delta t \leftarrow Tr_{\text{sub}}[i].t - Tr_{\text{sub}}[i-1].t$;
8. $A[0][2] \leftarrow \Delta t, A[1][3] \leftarrow \Delta t$; // 更新状态转移矩阵
9. $X_{\text{prior}} \leftarrow AX$;
10. $P_{\text{prior}} \leftarrow APA^{\text{T}} + Q$; // 计算先验卡尔曼误差估计协方差矩阵
11. $K \leftarrow P_{\text{prior}}H^{\text{T}}\left(HP_{\text{prior}}H^{\text{T}} + R\right)^{-1}$; // 计算卡尔曼增益
12. $Z \leftarrow \begin{pmatrix} Tr_{\text{sub}}[i].x & Tr_{\text{sub}}[i].y \end{pmatrix}^{\text{T}}$; // 当前时刻轨迹点的 GPS 测量值
13. $X \leftarrow X_{\text{prior}} + K\left(Z - HX_{\text{prior}}\right)$; // 计算后验状态向量
14. $P \leftarrow (I-KH)P_{\text{prior}}$; //计算后验卡尔曼误差估计协方差矩阵
15. $Tr'_{\text{sub}}.add(X[0],X[1])$; // 当前轨迹段中加入修正后的点
16. **end if**
17. **end for**

状态向量X为$(x, y, v_x, v_y)^{\text{T}}$, 其中$x$为电动自行车在该时刻的经度在 GCS_WGS_1984 (world geodetic system 1984, GPS 使用的坐标系统) 投影坐标系下的横坐标, y为电动自行车在该时刻的纬度在 GCS_WGS_1984 投影坐标系下的纵坐标, x和y的单位为 m, v_x与v_y分别为电动自行车在该时刻x方向与y方向上的速度分量, v_x和v_y单位为 m/s. 由于状态向量有 4 个维度, 因此后验估计协方差矩阵P的维度为 4×4, 初始值设置为零矩阵.

KF 算法使用匀速直线运动作为电动自行车的运动模型, 因此状态转移矩阵A设置为 4×4 的单位矩阵. 观测值有 2 个, 分别是电动自行车的经度方向位置x和纬度方向位置y, 因此观测矩阵H设置为 2×4, 初始值设置为$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$. 若$x$与$y$坐标预测误差为$var_{\text{predict}}$, 则系统预测过程噪声的协方差矩阵$Q$为$\begin{pmatrix} var_{\text{predict}} & 0 & 0 & 0 \\ 0 & var_{\text{predict}} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$. 若$x$与$y$坐标测量误差为$var_{\text{gps}}$, 则系统观测过程噪声的协方差矩阵$R$为$\begin{pmatrix} var_{\text{gps}} & 0 \\ 0 & var_{\text{gps}} \end{pmatrix}$.

3.3.3 电动自行车转弯判断

KFTS 算法假设电动自行车匀速直线运动, 但是电动自行车在行驶过程中可能会在路口转弯, 这会影响卡尔曼滤波的状态预测方程的准确度. 因此 KFTS 算法将卡尔曼滤波算法分段进行, 即将转弯前后的轨迹段分别进行

卡尔曼滤波轨迹修正.

存在轨迹段 $p_a \to p_{a+1} \to \ldots \to p_b$, 其平均速度定义为: $\overrightarrow{v_{a\to b}} = \dfrac{dist(p_a, p_b)}{p_b.t - p_a.t}$, 速度的方向由 p_a 指向 p_b, $dist(p_a, p_b)$ 为 p_a 与 p_b 的距离. 对于轨迹 $Tr: p_1 \to p_2 \to \ldots \to p_n$, 设窗口大小为 w (表示窗口内轨迹点的数量). 若某窗口内轨迹段为 $p_1 \to p_2 \to \ldots \to p_w$, 下一窗口内轨迹段为 $p_{w+1} \to p_{w+2} \to \ldots \to p_{2\times w}$, 当两窗口平均速度的夹角大于 45°, 且窗口内轨迹速度方向变化小于 45°时, 可认为电动自行车在两窗口间完成了转弯. 判断是否转弯的条件定义如下: $angle\left(\overrightarrow{v_{1\to w}}, \overrightarrow{v_{w+1\to 2\times w}}\right) > 45°$, 且 $angle\left(\overrightarrow{v_{i\to i+1}}, \overrightarrow{v_{i+1\to i+2}}\right) < 45°$, $i \in [1, w-2] \cup [w+1, 2\times w-2]$, 其中, $angle\left(\vec{v_1}, \vec{v_2}\right)$ 表示速度 $\vec{v_1}$ 与速度 $\vec{v_2}$ 间的夹角. 下文算法 KFTS 描述中的 $Turn(p_i, p_{i+1})$ 表示判断电动自行车是否在两窗口间, 即轨迹点 p_i 与轨迹点 p_{i+1} 间完成了转弯.

3.3.4 基于分段卡尔曼滤波算法的轨迹简化算法

在图 10 中, v 和 v' 分别代表窗口 1 和窗口 2 内轨迹的平均速度. 由于 v 与 v' 的夹角接近 90°, 且窗口 1 与窗口 2 内的轨迹方向变化很小, 因此可认为电动自行车在两个窗口间完成了转弯. 对转弯前后的轨迹分别进行卡尔曼滤波可以减小转弯对卡尔曼滤波的预测方程的影响.

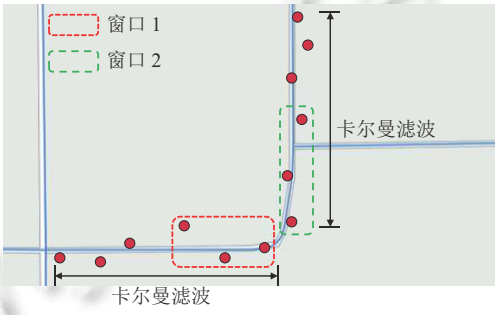


图 10 分段卡尔曼滤波示意图, 虚线框为某窗口内的轨迹段, 窗口内轨迹点数量相同

算法 4 KFTS 描述了基于分段卡尔曼滤波的轨迹简化算法. KFTS 算法的具体步骤如下.

• 第 1 步 (第 1 行): 初始化输出轨迹 Tr_{new} 为空, 当前轨迹片段 Tr_{sub} 为空.

• 第 2 步 (第 3 行): 将当前轨迹点加入轨迹片段 Tr_{sub}.

• 第 3 步 (第 4 行): 判断当前轨迹片段 Tr_{sub} 轨迹点数是否超过窗口大小, 若超过, 则执行步骤 4.

• 第 4 步 (第 5–10 行): 判断轨迹是否完成转弯, 若完成, 则使用算法 2 KF 将当前轨迹片段 Tr_{sub} 进行卡尔曼滤波轨迹修正. 修正后的轨迹片段加入输出轨迹 Tr_{new}, 清空 Tr_{sub}; 若未完成转弯, 则将当前轨迹点加入 Tr_{sub}.

• 第 5 步 (第 13 行): 使用 DPhull-ratio 算法对修正后的轨迹进行简化.

KFTS 算法首先使用卡尔曼滤波修正算法 KF 对轨迹进行修正, KF 算法能够在线性时间内对一段轨迹滤波, 时间复杂度为 O(n). 然后使用 DPhull-ratio 算法对轨迹简化, 由于 DPhull-ratio 的时间复杂度为 O($ratio \times n \log n$), 因此 KFTS 算法整体的时间复杂度为 O($ratio \times n \log n$).

算法 4. 轨迹简化算法 KFTS.

输入: 轨迹 $Tr: p_1 \to p_2 \to \ldots \to p_n$, 窗口大小 w, 预测误差方差 var_{predict}, GPS 观测误差方差 var_{gps}, 轨迹简化比例 $ratio$;

输出: 简化后轨迹 Tr_{new}.

1. $Tr_{\text{new}} \leftarrow \varnothing, Tr_{\text{sub}} \leftarrow \varnothing$;
2. **for** $i = 1$ **to** n **do**
3. $\quad Tr_{\text{sub}}.add(Tr[i])$;

4. **if** $Tr_{\text{sub}}.size() \geqslant w$ **do**
5. **if** $Turn(Tr[i], Tr[i+1])$ do // 判断轨迹是否在此处完成转弯
6. $Tr_{\text{new}}.add\left(KF\left(Tr_{\text{sub}}, var_{\text{predict}}, var_{\text{gps}}\right)\right)$; // 将当前轨迹段进行卡尔曼滤波
7. $Tr_{\text{sub}}.clear()$; // 清空当前轨迹段
8. **else**
9. $Tr_{\text{sub}}.add(Tr[i])$;
10. **end if**
11. **end if**
12. **end for**
13. $Tr_{\text{new}} \leftarrow$ DPhull-ratio$(Tr_{\text{new}}, ratio)$; // 使用 DPhull-ratio 算法进行轨迹简化

3.4 地图匹配

本节介绍了地图匹配部分的内容, 首先介绍了候选点的选择和候选图的构建方法, 其次介绍了如何解决无效轨迹片段导致的候选图中断问题, 最后给出了分段的隐马尔可夫模型 (HMM) 的地图匹配算法 AMM, 其能够自适应路网精度, 解决因路网精度不足导致的匹配准确率下降的问题. 地图匹配算法的应用场景包括离线场景和在线场景[4], 但是大部分电动自行车由于成本问题, 未安装计算设备, 因此本节中提出的地图匹配算法 AMM 只针对离线场景.

3.4.1 候选点选择和候选图构建

我们首先给出候选图的定义, 并且对候选点与候选边进行详细说明.

定义 6. 候选图. 候选图$G_c(C, E_c)$是一个有向图, C 为候选点集合, E_c 为候选点间相连边的集合. 候选图为一个分层的有向图, 有向边只存在于相邻层的节点之间, 且方向由上一层节点指向下一层节点. 图 11 为候选图的示例.

定义 6 中的候选点是对轨迹点实际位置的推测, 候选点所在的路段称为候选边, 通常取轨迹点在候选边上的投影点为候选点, 且轨迹点与候选点距离应小于固定的距离阈值, 该阈值与 GPS 误差范围有关. 候选点选择过程基于 K 近邻算法 (K-nearest neighbor, KNN) 实现, 可以表示为: $CS(p) = \text{KNN}(p)_{k,r}$, 其中$CS(p)$为轨迹点 p 的候选点集合, k 为近邻数, r 为轨迹点和候选点的距离阈值, 对于轨迹$Tr: p_1 \to p_2 \to \ldots \to p_n$, 每一个轨迹点都存在若干候选点, 同一个轨迹点的候选点集合构成了候选图中的候选点层, 因此每一个轨迹点均在候选图中有一个候选点层与之对应. 候选点层$G_c.layer[i]: \left(c_i^1, c_i^2, \ldots, c_i^k\right)$, 表示轨迹点$p_i$的候选点集合. 候选点$c$有 3 个属性, 分别为所在候选边$c.e$、累积概率$c.accum$与前驱点$c.prev$. 有向边$e_{a\to b} \in E_c$, 起点$a$与终点$b$分别为相邻候选层的候选点, $a \in G_c.layer[i]$, $b \in G_c.layer[i+1]$. 在图 11 中, $p_1 \to p_2 \to p_3$为一段轨迹示例, 轨迹点与候选边的距离阈值为 r, 道路段e_1与道路段e_2为轨迹点p_1的两条候选边, 轨迹点p_1在e_1与e_2上的投影点c_1^1与c_1^2为其候选点.

文献 [9] 认为在候选图中, 连接候选点的边$e_{a\to b}$的权值包括空间分析值与时间分析值, 由于本文使用的路网数据没有道路速度的相关信息, 因此本文只使用空间分析值代表$e_{a\to b}$的权值. 空间分析值的定义与文献 [9] 中一致, 表示为: $F_s\left(c_{i-1}^t \to c_i^s\right) = N\left(c_i^s\right) \times p\left(c_{i-1}^t \to c_i^s\right)$, 空间分析值为候选点$c_{i-1}^t$移动到候选点$c_i^s$的概率值, 等于当前候选点的观察概率$N\left(c_i^j\right)$与候选点$c_{i-1}^t$到候选点$c_i^s$的转移概率$p\left(c_{i-1}^t \to c_i^s\right)$的乘积. 观察概率定义为: $N\left(c_i^j\right) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(x_i^j\right)^2/2\sigma^2}$, 其中, 候选点$c_i^j$的观察概率服从方差为$\sigma$的正态分布, x_i^j为轨迹点p_i和候选点c_i^j的距离. 转移概率定义为: $p\left(c_{i-1}^t \to c_i^s\right) = \frac{\min\left(d_{i-1\to i}, w_{(i-1,t)\to(i,s)}\right)}{\max\left(d_{i-1\to i}, w_{(i-1,t)\to(i,s)}\right)}$, 其中, $d_{i-1\to i}$表示轨迹点p_{i-1}与轨迹点p_i的距离, $w_{(i-1,t)\to(i,s)}$表示候选点c_{i-1}^t到候选点c_i^s的最短路径距离. 由于车速限制, 最短路径距离不能过大, 否则连接两候选点的有向边不存在. 文献 [14] 论证了文献 [9] 中的转移概率定义可能会由于密集的轨迹数据与较大的邻域半径而出现概率大于 1 的问题, 因此本文使用文献 [14] 中的转移概率定义方式. 候选点的累积概率为上一层候选点的累积概率与其到该候选点的空间分析值之和的最大值, 计算方法为: $c_i^j.accum = \max\limits_{t\in[1,k]}\left(c_{i-1}^t.accum + F_s\left(c_{i-1}^t \to c_i^j\right)\right)$. 候选点的前驱点的计

算方法为: $c_i^j.prev = \underset{c \in G_c.layer[i-1]}{\arg\max} \left(c.accum + F_s\left(c \to c_i^j\right)\right)$. 在图 12 中, 当前候选点 c_2^1 的前驱点为上一层候选点中累积概率与其到 c_2^1 的空间分析值之和的最大的候选点.

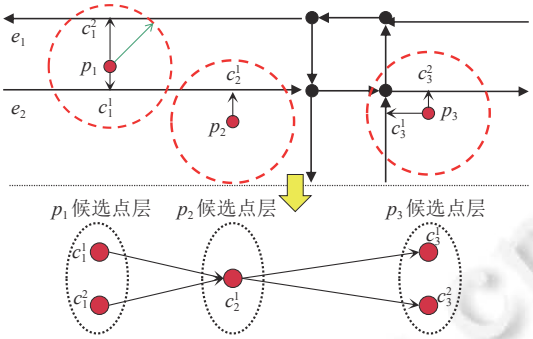


图 11 候选点与候选图示意图, 下方为上方轨迹的候选图

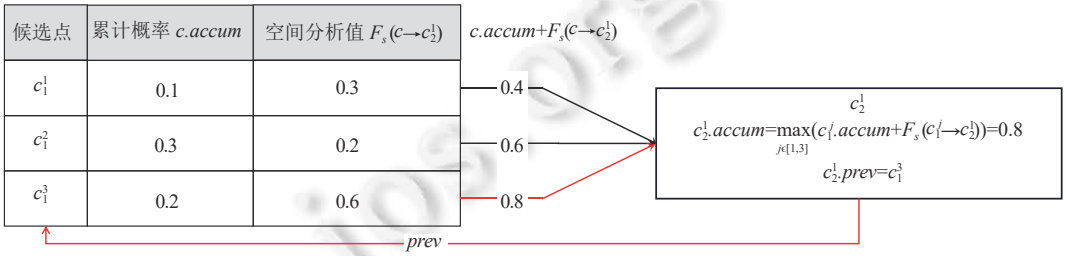


图 12 候选点累积概率与前驱点的计算示例

3.4.2 无效轨迹片段导致的候选图中断

无效轨迹片段是指, 在交通工具行驶的路线中, 某一段不存在于给定路网中的行驶轨迹. 一些使用全局匹配的地图匹配算法, 例如使用弗雷歇距离度量轨迹与路线相似度的算法[21], 当存在无效轨迹片段的轨迹时, 仍会计算出一条路线, 但是这明显与实际情况不符. 使用基于 HMM 的地图匹配算法[14,23–25,37,38]时, 虽然会避免将无效轨迹片段匹配到路网上, 但是会引起候选图中断, 从而导致匹配过程中断, 最终部分有效轨迹片段也无法成功匹配. 在图 13 中, 轨迹片段 $p_5 \to p_8$ 为无效轨迹片段, 该片段上的每个轨迹点的候选点都为空, 当使用基于 HMM 的地图匹配算法时, 在回溯寻找匹配路线时会导致算法中断.

3.4.3 分段的隐马尔可夫模型的地图匹配算法

无效轨迹片段可导致候选图中断, 并且会随着路网精度的降低而不断增多, 降低匹配准确率. 本文提出了分段的 HMM 的匹配算法 AMM, 使用了分段匹配机制, 通过候选图区分有效与无效轨迹片段, 对连续候选图匹配路线.

算法 5 给出了自适应路网精度的地图匹配算法 AMM 的详细描述, 输入包括轨迹 Tr, 路网图 $G(V,E)$, 轨迹点邻域半径 r, 轨迹点近邻数 k. 若输入轨迹 Tr 存在无效轨迹片段, 则输出为轨迹 Tr 实际经过的路线集合 S_{path}; 若输入轨迹 Tr 不存在无效轨迹片段, 则 S_{path} 只包含一条路线. AMM 算法的具体步骤如下.

- 第 1 步 (第 2 行): 使用 KNN 算法计算当前轨迹点 $Tr[i]$ 的候选点集合 $CS(Tr[i])$.
- 第 2 步 (第 3 行): 判断当前轨迹点的候选点集合是否为空, 若不为空, 执行第 3 步; 若为空, 执行第 6 步.
- 第 3 步 (第 4 行): 判断当前候选图是否为空, 若不为空, 执行第 4 步; 若为空, 执行第 5 步.
- 第 4 步 (第 5–14 行): 计算候选图上一层节点与候选图当前层节点间有向边的权值、当前层候选点的累积概率和前驱节点.
- 第 5 步 (第 16–19 行): 当前层候选点为候选图的第一层候选点, 无前驱节点, 计算候选点的累积概率.
- 第 6 步 (第 22–24 行): 为当前候选图匹配路线, 将所得路线加入路线集合 S_{path}, 清空当前候选图. 第 23 行中 *PathInference* 为路线推断算法, 详见算法 6.

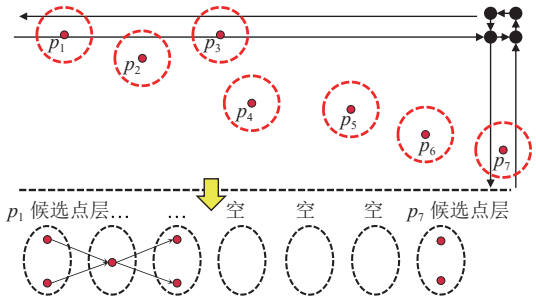


图 13 候选图中断

算法 5. 地图匹配算法 AMM.

输入: 轨迹 $Tr: p_1 \to p_2 \to \ldots \to p_n$, 路网图 $G(V,E)$, 轨迹点邻域半径 r, 轨迹点近邻数 k;
输出: 路线集合 S_{path}.

1. **for** $i = 1$ **to** n **do**
2. $CS(Tr[i]) \leftarrow KNN(Tr[i])_{k,r}$; // KNN 算法寻找轨迹点的候选点
3. **if** $CS(Tr[i]) \neq \varnothing$ **do**
4. **if** $G_c \neq \varnothing$ **do**
5. $C_{\text{last}} \leftarrow G_c.layer[-1]$; // 取出候选图最后一个候选点层
6. **foreach** $c \in CS(Tr[i])$ **do**
7. $G_c.C.add(c)$; // 候选点加入候选图
8. **foreach** $v \in C_{\text{last}}$ **do**
9. $e_{v\to c} \leftarrow F_s(v \to c)$; // 计算空间分析值、有向边权值
10. $G_c.E_c.add(e_{v\to c})$;
11. **end foreach**
12. $c.accum \leftarrow \max\limits_{v\in C_{\text{last}}}(v.accum + F_s(v \to c))$; // 计算累积概率
13. $c.prev \leftarrow \arg\max\limits_{v\in C_{\text{last}}}(v.accum + F_s(v \to c))$; // 寻找候选点的前驱点
14. **end foreach**
15. **else**
16. **foreach** $c \in CS(Tr[i])$ **do**
17. $G_c.C.add(c)$; // 加入最后一个候选点层
18. $c.accum \leftarrow N(c)$; // 第 1 层候选点的累积概率等于观察概率
19. **end foreach**
20. **end if**
21. **else**
22. $R \leftarrow PathInference(G_c)$; // 路线推断
23. $S_{\text{path}}.add(R)$; // 将路线加入输出路线集合
24. $G_c.clear()$; // 清空当前候选图
25. **end if**
26. **end for**

算法 5 第 6 步中使用的 *PathInference* 算法为路线推断算法, *PathInference* 算法的输入为候选图 G_c, 输出为该段轨迹对应的路线 R.

算法 6. 路线推断算法 *PathInference*.

输入: 候选图 G_c;

输出: 路线 R.

1. $c_{\text{cur}} \leftarrow \underset{c \in G_c.layer[-1]}{\arg\max}\ (c.accmu)$; // 选出候选图 G_c 最后一层 $G_c.layer[-1]$ 中累积概率最大的候选点 c_{cur}
2. **for** $i = G_c.layers.size()$ **to** 1 **do** // 从候选图的最后一层回溯到第 1 层
3. $R'.add(c_{\text{cur}}.e)$; // 将候选点所在路段 e 加入逆序路线 R'
4. $c_{\text{cur}} \leftarrow c_{\text{cur}}.prev$; // 当前节点置为当前候选点的前驱
5. **end for**
6. $R \leftarrow R'.reverse()$; // 将逆序路线 R' 逆置并输出路线 R

图 14 为路线推断示例, 图中箭头由候选点指向其前驱节点. 首先从轨迹点 p_4 的候选点中选出累积概率最大的候选点 c_4^3, 然后不断回溯寻找前驱点, 最后可得到路线 $c_1^4.e \rightarrow c_2^2.e \rightarrow c_3^2.e \rightarrow c_4^3.e$, 即为当前候选图对应的实际路线.

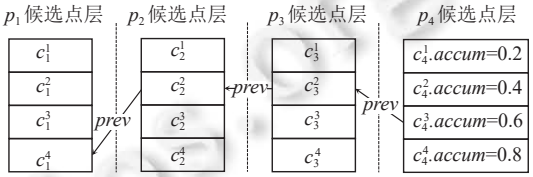


图 14 路线推断示例

3.4.4 地图匹配算法 AMM 复杂度分析

设输入轨迹的轨迹点个数为 n, 每个轨迹点候选点个数为 k. AMM 算法的时间复杂度由以下几个部分组成: (1) 为每个候选点查找候选点以构建候选图, 时间复杂度为 $O(kn)$; (2) 计算候选图中相邻层候选点间的转移概率, 本文使用了文献 [14] 中预先计算路网图顶点最短路径并查表的方式避免重复计算, 候选图层数等于轨迹点个数 n, 每层有 k 个候选点, 因此时间复杂度为 $O(k^2n)$; (3) 路线推断算法 *PathInference* 迭代寻找候选点的前驱点, 时间复杂度为 $O(n)$. 综上, AMM 算法的整体时间复杂度为 $O(nk+k^2n+n)$, 即 $O(k^2n)$.

4 实 验

本节介绍了实验所使用的路网数据和轨迹数据, 给出了实验中地图匹配准确率与地图匹配速度的定义. 同时, 本节设计了 8 组试验, 其中, 实验 1 用于测试基于简化比例的轨迹简化算法 DPhull-ratio 的误差; 实验 2 用于验证轨迹简化算法 KFTS 能否有效修正轨迹; 实验 3 用于测试不同超参数对地图匹配算法 AMM 的匹配准确率的影响; 实验 4 使用多个现有地图匹配算法与本文提出的地图匹配算法 AMM 进行匹配准确率比较; 实验 5 用于测试本文提出的轨迹简化算法 KFTS 对地图匹配准确率的影响; 实验 6 用于验证经过 KFTS 算法简化后的轨迹数据能否提高地图匹配速度; 实验 7 用于测试停留点处理算法 StayPointsProcess 对地图匹配准确率的影响; 实验 8 对 3 个真实电动自行车轨迹案例进行了分析. 实验运行在 Windows 10 操作系统计算机上, 计算机处理器为 Intel Core i5-4590 CPU, 主频为 3.30 GHz, 内存 (RAM) 为 14 GB.

4.1 数据介绍

4.1.1 路网数据

实验使用了郑州市的开放街道地图 (OpenStreetMap, OSM. 来源: https://www.openstreetmap.org) 的开源路网数据集, 其统计信息如表 2 所示, 包含了 OSM 采集到的所有级别的道路, 包含了机动车道路、非机动车道路以及人行小路. OSM 为各类道路设置了不同的标签, 表 3 展示了各类标签与国内道路类型的大致对应情况以及道路数量占比. 路网图拓扑结构如图 15 所示.

表 2 郑州路网数据信息

节点数	边数	经度范围	纬度范围	区域面积
45 198	105 795	东经112.740 8°至东经114.199 1°	北纬34.277 3°至北纬34.351 2°	4 905 km^2

表 3 各类道路信息与数量占比

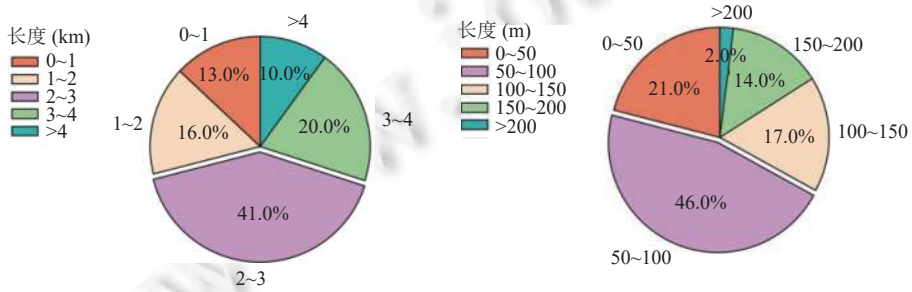
OSM道路标签	国内道路类型	道路数量占比 (%)
Motorway	高速公路	0.4
Trunk	高架、快速路	3.8
Primary	主干道	1.6
Secondary	次干道	8.1
Tertiary	支路	17.5
Unclassified	未分类道路, 多为小型道路	34.0
Service	服务性道路	26.9
Residential	居住区道路	2.1
Footway	人行道	2.6
Cycleway	公园自行车道	0.06

图 15 郑州市路网图

4.1.2 轨迹数据

原始 GPS 日志共有 2 687 214 282 条记录, 删除存在缺失值、冗余或经纬度有明显错误的记录后, 剩余 2 515 378 612 条记录, 经提取得到 10 651 448 条轨迹, 涉及 1 349 641 辆电动自行车, 覆盖的时间范围为 2019 年 9 月 16 日至 2019 年 9 月 30 日, 传感器设置采样时间间隔为 10 s.

提取的轨迹的长度分布如图 16(a) 所示, 约 75% 的轨迹长度分布在 1–4 km, 这说明使用大部分用户倾向于在中距离路程中使用电动自行车. 轨迹上相邻点的平均距离频率分布如图 16(b) 所示, 近半数的轨迹点间距分布在 50–100 m, 由此可以看出约半数用户骑行电动自行车的速度在 18–36 km/h 之间. 本文选取提出轨迹中的 1 000 条轨迹作为轨迹子集, 对其进行人工标记, 完成后续实验.



(a) 轨迹长度频率分布图 (b) 轨迹相邻点距离分布图

图 16 提取轨迹的统计信息

4.2 地图匹配准确率以及速度评价标准

本文实验使用的匹配准确率定义如公式 (8) 所示:

$$\sum_{i}^{|GT|} compare(GT[i], MR[i]) / |GT| \tag{8}$$

其中, GT 表示被标记过的轨迹, $|GT|$ 表示被标记过的轨迹个数, MR 表示轨迹匹配路线, $compare(GT[i], MR[i])$ 用于比较轨迹正确结果 $GT[i]$ 与匹配结果 $MR[i]$, 若完全一致, 则结果为 1. 公式 (8) 衡量了成功匹配的轨迹个数占轨迹总数的比例. 匹配速度定义为匹配总时间除于轨迹总个数: $S_{\text{match}} = T_{\text{match}} / |GT|$.

4.3 实验 1: 基于简化比例的轨迹简化算法 DPhull-ratio 误差测试

实验 1 用于验证本文提出的 DPhull-ratio 算法能否在得到期望简化比例的前提下, 将简化后轨迹与原始轨迹的误差控制在一定范围内. 我们使用了最大垂直欧式距离 $PED_{\max}$ (max perpendicular Euclidean distance) 以衡量轨迹简化的误差. 我们将本文提出的 DPhull-ratio 算法与 DPhull 算法进行了对比, 比较了在轨迹集合简化比例相同时两种算法的 $PED_{\max}$, 同时我们使用了杰卡德相似系数 (Jaccard similarity coefficient) 衡量了两种算法在简化过程中选取的特征点集合的相似度.

设原始轨迹为 $Tr: p_1 \to p_2 \to \ldots \to p_n$, 简化后的轨迹为 $Tr': p_1' \to p_2' \to \ldots \to p_m'$, 数组 $Reserve: \{0,1 | i \in [1,n]\}$ 表示原始轨迹 Tr 上的轨迹点是否存在于简化后轨迹 Tr' 上, $Reserve$ 数组元素取值为 0 或 1, $Reserve[i] = 1$ 表示轨迹点 p_i 在轨迹简化过程中被保留, 否则被舍弃. $PED_{\max}$ 为被简化的轨迹点距离被简化后的轨迹的最大距离, 表示了轨迹简化的误差上界. 最大垂直欧式距离 $PED_{\max}$ 的计算方法如公式 (9) 所示:

$$PED_{\max} = \max_{i \in [1,n]} dist\left(p_i, \overline{p_{i\_prev} \to p_{i\_next}}\right) \times (1 - Reserve[i]) \tag{9}$$

其中, p_{i_prev} 表示轨迹点 p_1 至轨迹点 p_{i-1} 中距离轨迹点 p_i 最近的被保留的轨迹点, p_{i_next} 表示轨迹点 p_{i+1} 至轨迹点 p_n 中距离轨迹点 p_i 最近的被保留的轨迹点, $\overline{p_{i\_prev} \to p_{i\_next}}$ 为这两个轨迹点相连得到的线段, $dist(p_i, \overline{p_{i\_prev} \to p_{i\_next}})$ 表示轨迹点 p_i 距离该线段的垂直距离. 设两种算法计算得到的轨迹保留数组分别为 $Reserve_1$ 和 $Reserve_2$, Jaccard 相似系数的计算方法为: $|Reserve_1 \cap Reserve_2| / |Reserve_1 \cup Reserve_2|$.

我们对 DPhull-ratio 算法与 DPhull 算法在轨迹集合简化比例相同时的 $PED_{\max}$ 指标进行了测试. 具体做法是: 首先, 使用某一距离阈值作为 DPhull 算法的参数, 此时轨迹集合中每条轨迹的简化比例是不同的; 然后, 对于轨迹集合中的每条轨迹, 使用 DPhull-ratio 算法以相同的简化比例对其进行简化, 最终轨迹集合的平均简化比例也相同. 轨迹集合的平均垂直欧式距离 $PED_{\max}$ 随轨迹集合平均简化比例的变化如图 17 所示. 从结果可以看出: (1) 两种算法得到的 $PED_{\max}$ 指标都会随着平均简化比例的上升而下降, 这是因为简化比例越大, 简化过程保留的轨迹点越多, 因此简化后轨迹会更接近原轨迹的形状; (2) 简化比例较大时, DPhull-ratio 算法与 DPhull 算法得到的 $PED_{\max}$ 指标十分接近, 而当简化比例较小, 为 10% 与 20% 时, DPhull-ratio 算法的效果更好. 这说明了 DPhull-ratio 算法通过优先队列来选择特征点的机制是有效的, 同时使用队列节点的分割点与节点包含轨迹段的垂直距离作为节点优先级能够帮助选出更能描绘轨迹轮廓的轨迹点.

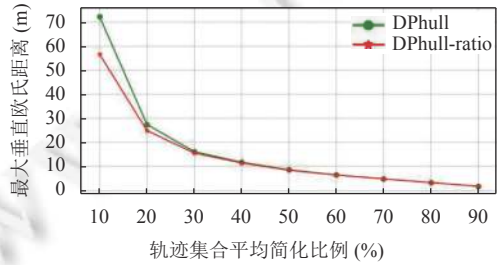


图 17 实验 1: DPhull-ratio 算法和 DPhull 算法的 $PED_{\max}$ 随轨迹集合平均简化比例的变化

DPhull-ratio 算法与 DPhull 算法选取的特征点集合的 Jaccard 相似系数随简化比例的变化如图 18 所示, 从结果可以看出: (1) 两种算法选取的特征点集合的 Jaccard 相似系数较高, 简化比例为 10% 时相似系数也有近 80%; (2) Jaccard 相似系数随简化比例增大而不断提高, 这与图 17 中两种算法得到的 $PED_{\max}$ 的相似度变化趋势一致; (3) 简化比例较低时, Jaccard 相似系数较低, 联系图 17 可以说明, 在简化比例较低时, 根据合理设置的优先级选取特征点的机制更有效, 因此较低简化比例下 Jaccard 相似系数也较低.

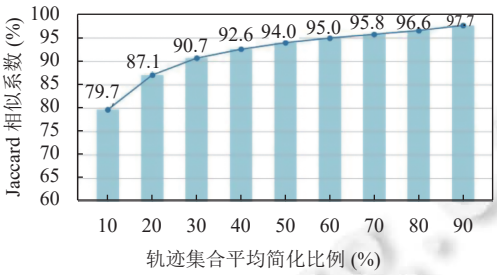


图 18 DPhull-ratio 和 DPhull 所选取的特征点集合的 Jaccard 相似系数随轨迹集合平均简化比例的变化

4.4 实验 2: 轨迹简化算法 KFTS 效果评价

实验 2 用于验证本文提出的轨迹简化算法 KFTS 是否可以有效地修正轨迹点和平滑轨迹, 分别测试了 KFTS 算法与 DPhull-ratio 算法在轨迹简化比例 *ratio* 为 0.2、0.5 与 0.8 的情况下, 简化后轨迹的平均误差 $Error_{\text{mean}}$ 与最大误差 $Error_{\max}$. 轨迹简化比例 *ratio* 表示简化后轨迹点个数与原轨迹点个数的比例. *ratio* 越大, 表示保留的轨迹点越多. 平均误差 $Error_{\text{mean}}$ 定义为: $Error_{\text{mean}}=\sum_{i=1}^{n} dist(prj_i,p_i)/n$, 其中 prj_i 为轨迹点 p_i 在真实路线上的投影点, $dist(prj_i,p_i)$ 表示投影点 prj_i 与轨迹点 p_i 的欧式距离. 最大误差 $Error_{\max}$ 定义为: $Error_{\max}=\max_{i\in[1,n]} dist(prj_i,p_i)$.

实验结果如表 4 所示, 从中可以看出: (1) 在使用不同 *ratio* 的情况下, KFTS 算法均能得到更低的最大误差 $Error_{\max}$ 与平均误差 $Error_{\text{mean}}$; (2) 随着 *ratio* 不断提高, KFTS 算法相比于 DPhull-ratio 算法的评估指标的提升比例不断下降, 这是因为基于 DP 算法[15]的轨迹简化算法倾向于保留 GPS 误差较大的轨迹点. 当 *ratio* 较小时, 大部分 GPS 误差较小的轨迹点因简化而被舍弃, 所以 KFTS 算法相较于 DPhull-ratio 在评估指标上提升较为明显; 当 *ratio* 较大时, GPS 误差较大的轨迹点所占比重减小, 因此 KFTS 算法相比 DPhull-ratio 算法的评估指标的提升比例降低; (3) 当 *ratio* 相同时, 最大误差相较于平均误差的提升比例更高, 这是因为 KFTS 算法可以较好地修正 GPS 误差大的轨迹点. 同时, 评估指标的提升比例均未超过 10%, 原因可能是 KFST 算法中的分段卡尔曼滤波模型不够精细, 仅使用轨迹点的时间与位置属性不能得到更精确的估计值.

表 4 实验 2: KFTS 轨迹简化算法评估结果

轨迹简化比例 *ratio*	评估指标	KFTS (m)	DPhull-ratio (m)	KFTS相比DPhull-ratio的提升比例 (%)
0.2	最大误差 $Error_{\max}$	**43.61**	48.16	9.45
	平均误差 $Error_{\text{mean}}$	**10.47**	11.31	7.43
0.5	最大误差 $Error_{\max}$	**42.54**	46.42	8.36
	平均误差 $Error_{\text{mean}}$	**8.03**	8.59	6.52
0.8	最大误差 $Error_{\max}$	**41.35**	44.08	6.19
	平均误差 $Error_{\text{mean}}$	**6.16**	6.47	4.79

总之, 实验 2 结果证明, 本文提出的轨迹简化算法 KFTS 能够在简化轨迹的同时, 较好地修正轨迹, 有效解决了基于 DP 算法[15]的轨迹简化算法得到的简化后的轨迹误差较大的问题.

4.5 实验 3: 地图匹配算法 AMM 超参数选择

本文提出的地图匹配算法使用的超参数有 2 个, 分别为轨迹点邻域半径 r 和候选点个数 k. 本文设计了实验 3

用于测试不同超参数对于地图匹配算法 AMM 匹配准确率的影响. 实验 3 结果如图 19 所示.

从实验结果中可以观察到, 随着邻域半径 r 的扩大与候选点个数 k 的增加, 匹配准确率不断提高. 当邻域半径 r 选择 50 m、候选点个数 k 选择 4 时, 匹配准确率基本达到最高. 继续扩大邻域半径 r 与增加候选点个数 k 所带来匹配准确率的提高很小, 反而会降低算法的匹配效率. 文献 [5] 指出, 民用 GPS 传感器的测量误差在 3 m 之内, 且在城市路网中, 单方向中可供电动自行车行驶的平行路段最多不会超过 2 条, 因此邻域半径 r 选择 50 m、候选点个数选择 4 时, 不会遗漏候选点. 为了达到最佳实验效果, 后续进行的实验所使用的邻域半径 r 选择 50 m、候选点个数选择 4.

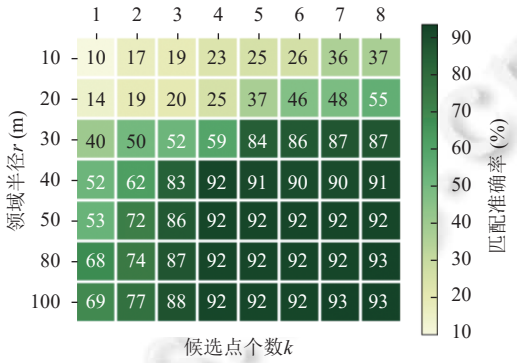


图 19 实验 3: 超参数对于匹配准确率的影响

总之, 实验 3 结果证明, AMM 算法的两个超参数轨迹点邻域半径 r 与候选点个数 k, 分别在选择 4 与 50 m 时, 可以兼顾准确率与效率, 达到最优效果.

4.6 实验 4: 地图匹配算法匹配准确率测试

OSM 对不同级别的道路设置了相应的标签. 在表 5 中, 我们给出了几种常见道路的标签, 道路的标签代表了道路的级别. 表 5 中的道路的级别从 motorway 到 residential 依次降低. 一般情况下道路的精细度较低时, 包含的道路级别也相对较高, 例如只保留城市路网中的主干道路以组成路网的骨架结构. 本文通过选择不同道路标签来模拟不同精度的路网, 其中: (1) 高精度路网包含 motorway、trunk、primary、secondary、tertiary、unclassified、service 与 residential 等 8 种道路标签的道路; (2) 中精度路网包含 motorway、trunk、primary、secondary、tertiary、unclassified 与 service 等 7 种道路标签的道路, 路网的密集程度相对于高精度路网来说有所降低; (3) 低精度路网包含 motorway、trunk、primary、secondary、tertiary 与 unclassified 这 6 种道路标签的道路, 路网的密集程度最低.

表 5 OSM 中道路标签说明

不同精度路网	OSM中的道路标签							
	motorway	trunk	primary	secondary	tertiary	unclassified	service	residential
高精度路网	√	√	√	√	√	√	√	√
中精度路网	√	√	√	√	√	√	√	×
低精度路网	√	√	√	√	√	√	×	×

图 20 展示了郑州市东经 113.775 6°至东经 113.825 4°, 北纬 34.774 1°至北纬 34.813 4°范围内的不同精度的路网图, 道路密度由高精度路网图到低精度路网图不断降低.

实验 4 选取 5 种已有地图匹配算法作为对比算法, 分别是 FMM 算法[14]、ST-Matching 算法[9]、Segmented-MM[29]、IVMM 算法[24]以及 AIVMM 算法[25]. 其中 FMM 算法、ST-Matching 算法、IVMM 算法与 AIVMM 算法均为基于隐马尔可夫模型的地图匹配算法, Segmented-MM 算法属于局部/增量算法. IVMM 算法与 AIVMM 算法在 HMM 模型的基础上加入了互动投票机制, 利用了候选点间的相关性. AIVMM 算法在 IVMM 的基础上加入了两个特定情况的约束, 以减少匹配路线的迂回. FMM 算法为了解决匹配轨迹存在反向移动的问题, 设计了惩罚机制, 使用惩

罚因子 (pf) 来表示惩罚的程度, pf 参数取值范围为 0–100, 取值越高表示对反向移动的修正更强. 本文选取了 pf 分别等于 0、50 与 100 这 3 种情况下的 FMM 算法进行对比, 分别表示为 FMM-1、FMM-2 与 FMM-3. 输入轨迹数据均经过了停留点处理.

(a) 高精度路网图示例　(b) 中精度路网图示例　(c) 低精度路网图示例

图 20　不同精度路网图示例

实验 4 结果如图 21 所示, 从中可以看出: (1) 随着路网精度的下降, 本文提出的 AMM 算法与其他方法的匹配准确率都在下降, 但是其他方法的匹配准确率下降更为明显, 说明了本文提出的 AMM 算法对不同精度路网的适应度更强. 当路网精度下降时, 电动自行车轨迹数据中无法匹配到给定路网的无效轨迹增多, 造成基于 HMM 模型的算法匹配过程中断. Segmented-MM 算法同样无法对无效轨迹进行匹配. 因此对比方法的匹配准确率下降较快; (2) AMM 算法匹配准确率也会随路网精度降低而下降, 原因在于当路网精度不断降低、无效轨迹不断增多时, 候选图中连续层数过少, 导致隐马尔可夫模型不够精确, 因此准确率降低; (3) 由于输入轨迹已进行了停留点处理, 因此 Segmented-MM 针对高频率采样下, 路口处轨迹匹配进行的优化对准确率的提升不明显; (4) FMM-1、FMM-2 与 FMM-3 在相同精度的路网下, 匹配准确率相差不大, 原因在于本文所使用的电动自行车轨迹数据采样频率较高, 相邻轨迹点距离较近, 而文献 [14] 提到的反向移动问题多出现在相邻轨迹点距离较远的情况下.

总之, 实验 4 结果表明, 本文提出的地图匹配算法 AMM 相较于其他对比方法在不同精度路网上的匹配准确率更高, 同时随着路网精度的下降, 匹配准确率的下降幅度更小, 表现出更强的适应度, 有效解决了因路网精度不足而产生的无效轨迹片段使地图匹配算法准确率降低的问题.

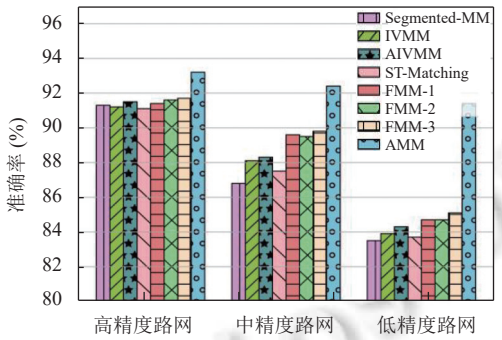


图 21　实验 4: AMM 与对比方法在不同精度路网上的匹配准确率比较

4.7　实验 5: 轨迹简化算法 KFTS 对地图匹配算法准确率的影响

本文提出的轨迹简化算法 KFTS 算法包含了卡尔曼滤波轨迹修正与轨迹简化两个步骤. 实验 5 用于验证 KFTS 算法中的两个关键步骤对地图匹配算法准确率的影响. 首先, 为了验证卡尔曼滤波轨迹修正步骤对于地图匹配准确率的影响, 我们在实验中将 KFTS-AMM 算法与其变体 DPhull-ratio-AMM 进行了对比, KFTS-AMM 算法在轨迹简化前加入了卡尔曼滤波轨迹修正步骤, 而 DPhull-ratio-AMM 则没有轨迹修正步骤. 其次, 为了验证轨迹简化步骤对于地图匹配准确率的影响, 我们评估了 KFTS-AMM 与 DPhull-ratio-AMM 在不同轨迹简化比例下的地图匹配准确率, 其中轨迹简化比例 $ratio$ 设置为 0.2、0.5、0.8 与 1. 轨迹简化比例 $ratio$ 表示简化后轨迹点个数与原轨迹点个数的比例, $ratio$ 越大, 则保留的轨迹点越多. 当 $ratio$ 为 1 时, KFTS-AMM 与 DPhull-ratio-AMM 均不对轨

迹进行简化. 路网数据使用高精度路网图.

实验 5 结果如后文图 22 所示, 从中可以看出: (1) 在轨迹简化比例 *ratio* 相同时, KFTS-AMM 比 DPhull-ratio-AMM 的匹配准确率较高, 说明经过卡尔曼滤波修正后的轨迹可以提高地图匹配的准确率; (2) 随着 *ratio* 不断下降, DPhull-ratio-AMM 的匹配准确率不断下降, 这是因为随着 *ratio* 的下降, GPS 误差大的轨迹点所占比例不断增大, 因此对匹配准确率的影响越来越明显; (3) 当 *ratio* 由 1 下降到 0.8、0.5 时, KFTS-AMM 的匹配准确率变化很小, 可以忽略不计, 说明对采样频率较高的轨迹进行适度简化, 不会降低地图匹配准确率; (4) 当 *ratio* 下降到 0.2 时, KFTS-AMM 的匹配准确率下降较大, 原因是当 *ratio* 下降到 0.2 时, GPS 误差较大的轨迹点所占比例较大, 极大地影响了匹配准确率.

综上, 实验 5 结果证明本文提出的轨迹简化算法 KFTS 中的卡尔曼滤波轨迹修正步骤能够在一定程度上提高地图匹配的准确率, 同时在对轨迹进行适度的简化, 能够在不降低准确率的前提下提升地图匹配效率.

4.8 实验 6: KFTS 算法对地图匹配效率的影响

实验 6 用于测试经轨迹简化算法 KFTS 处理后的轨迹能否给地图匹配算法带来效率上的提升. 实验 6 使用 KFTS 算法在不同轨迹简化比例 *ratio* 下得到轨迹数据作为地图匹配算法 AMM 的输入, 分别测试匹配速度. *ratio* 分别选取 0.2、0.4、0.6、0.8 与 1, *ratio* 越大, 则保留的轨迹点越多. 实验 6 结果如图 23 所示, 平均每条轨迹的匹配时间随着 *ratio* 不断下降而缩短, 说明了经过轨迹简化算法 KFTS 处理后的轨迹可以加快地图匹配的速度.

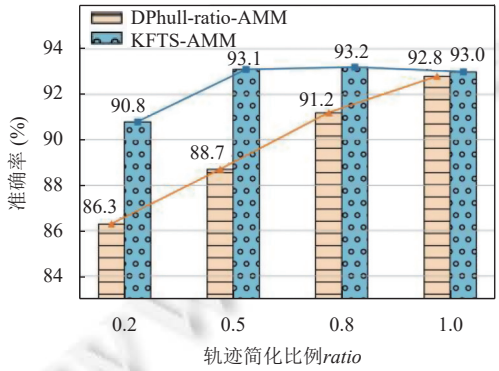


图 22 实验 5: 轨迹简化算法 KFTS 中的关键步骤对地图匹配算法准确率的影响

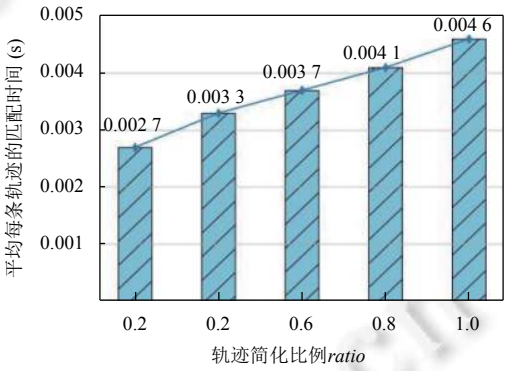


图 23 实验 6: KFTS 轨迹简化算法对地图匹配算法效率的影响

4.9 实验 7: 停留点处理算法对匹配准确率的影响

实验 7 用于评估本文提出的停留点处理算法 StayPointsProcess 对地图匹配算法准确率的影响. 实验 7 分别测试了在不同精度路网上, AMM 算法与未使用 StayPointsProcess 算法的 AMM 算法的地图匹配准确率. 实验 7 结果如后文图 24 所示, 在高、中、低 3 种精度的路网中, 相比于未使用停留点处理的 AMM 算法, 包含了停留处理过程的 AMM 算法均能取得更高的地图匹配准确率. 实验 7 结果表明, 本文提出的停留点处理算法 StayPoints-Process 可以有效识别与合并停留点, 从而解决了电动自行车轨迹数据中存在大量停留点所导致的地图匹配准确率下降的问题[4], 能够有效提升电动自称车轨迹地图匹配算法的性能.

4.10 实验 8: 案例分析

4.10.1 案例 1: 存在停留点轨迹的匹配

图 25 中白色网格为路网图, 红色点为电动自行车轨迹. 图 25(a) 为一段包含停留点的电动自行车轨迹, 椭圆虚线框内为一组停留点. 当未对停留点处理时, 使用 AMM 算法的匹配结果如图 25(c) 所示. 由于 AMM 算法设置了候选点间最短路径的距离上限, 虽然避免了不必要的迂回[4], 但是匹配过程则因停留点的候选点间不存在最短路径而中断. 使用 StayPointsProcess 算法对停留点识别与合并后, AMM 算法匹配结果如图 25(d) 所示, 与正确结果一致.

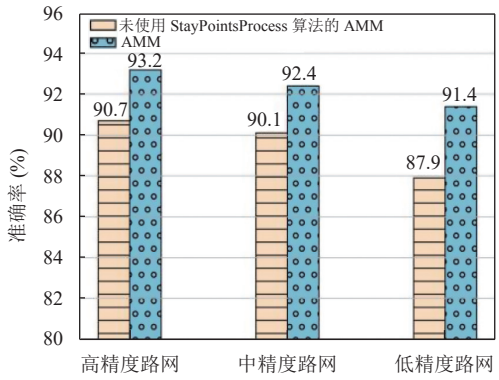


图 24 实验 7: 停留点处理算法 StayPointsProcess 对地图匹配准确率的影响

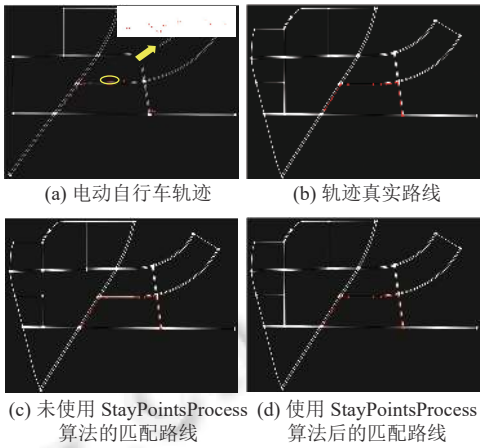


图 25 案例 1: 存在停留点的轨迹

4.10.2 案例 2: 存在 GPS 误差较大轨迹点的轨迹的匹配

图 26 中白色网格为路网图, 红色点为电动自行车轨迹. 图 26(a) 中圆形虚线圆框内的轨迹点 GPS 误差较大. 图 26(c) 与图 26(d) 中的虚折线分别为使用 DPhull-ratio 算法和 KFTS 算法简化后的轨迹, 红色折线分别为匹配后的路线. 图 26(c) 中, 由于误差较大的轨迹点得到了保留, 导致匹配路线错误. KFTS-AMM 对 GPS 误差较大的轨迹点进行了修正, 因此得到了正确匹配结果.

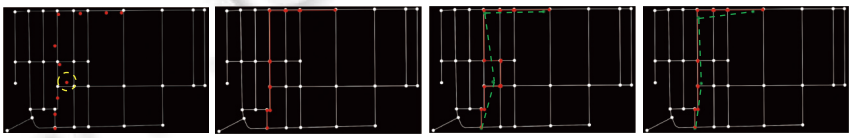


图 26 案例 2: 存在 GPS 误差较大轨迹点的轨迹

4.10.3 案例 3: 存在无效轨迹片段的轨迹的匹配

图 27 中白色网格为路网图, 红色点为电动自行车轨迹. 图 27(a) 中椭圆虚线框内为无效轨迹片段. 图 27(c) 中为 FMM 算法[14]的匹配结果, 由于无效轨迹片段存在, 导致匹配过程中断, 因此只有部分轨迹得到了正确的匹配结果. KFTS-AMM 算法的匹配结果如图 27(d) 所示, 与正确结果一致.

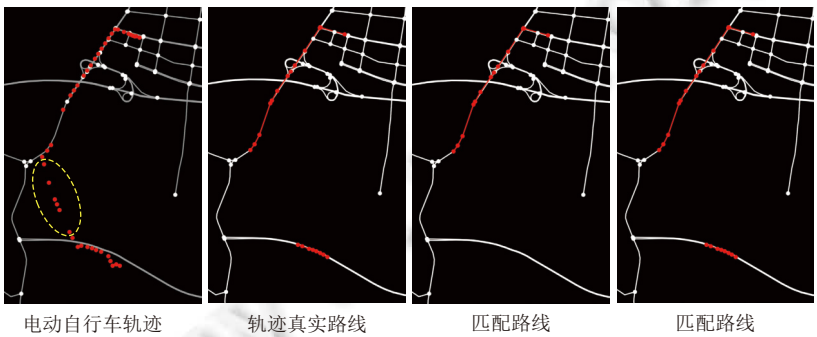


图 27 案例 3: 存在无效轨迹片段轨迹

5 总结与展望

本文提出了一种可自适应路网精度的电动自行车轨迹地图匹配方法 KFTS-AMM. 该方法融合了基于卡尔曼

滤波算法的轨迹简化算法 (KFTS), 以及分段的隐马尔可夫模型的地图匹配算法 (AMM). 本文提出的算法解决了现有地图匹配算法因电动自行车轨迹数据存在停留点、采样频率高、轨迹密度大且受限于路网精度较低而存在无效轨迹片段等特点导致的匹配准确率下降和匹配效率不高的问题. 基于真实的郑州市电动自行车轨迹数据的实验结果证明, 本文提出的算法与现有方法相比, 在不同精度路网图上的适应度更强, 匹配准确率更高, 且简化后的轨迹数据能够显著提升匹配速度. 未来的研究工作包括以下几个部分: (1) 解决地图匹配在隧道、高架桥与高楼附近等多路径效应明显的区域内地图匹配准确率较低的问题; (2) 尝试利用更多传感器数据进行数据融合, 使卡尔曼滤波模型更精确, 进一步提高轨迹修正效果; (3) 本文所提方法不适用于海量轨迹数据处理场景, 在未来的工作中尝试结合大数据与并行化技术, 提升算法的计算效率; (4) 地图匹配过程中获取的无效轨迹片段能够用于更新路网数据, 完整的轨迹对于深入挖掘用户出行规律十分重要, 我们将会在未来的工作中对无效轨迹片段深入地分析与挖掘.

References:

[1] chyxx.com. Analysis on the development trend of China's electric bicycle production, electric bicycle ownership and output of provinces and cities in 2018. 2019 (in Chinese). http://www.chyxx.com/industry/201911/801292.html

[2] Gao Q, Zhang FL, Wang RJ, Zhou F. Trajectory big data: A review of key technologies in data processing. Ruan Jian Xue Bao/Journal of Software, 2017, 28(4): 959–992 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5143.htm [doi: 10.13328/j.cnki.jos.005143]

[3] Zheng Y. Trajectory data mining: An overview. ACM Trans. on Intelligent Systems and Technology, 2015, 6(3): 1–41. [doi: 10.1145/2743025]

[4] Chao PF, Xu YH, Hua W, Zhou XF. A survey on map-matching algorithms. In: Borovica-Gajic R, Qi JZ, Wang WQ, eds. Proc. of the 2020 Australasian Database Conf. Cham: Springer, 2020, 12008: 121–133. [doi: 10.1007/978-3-030-39469-1_10]

[5] Liu JY. Accuracy, errors and bias of GNSS navigation/positioning—Errors in GNSS navigation/positioning (1). Digital Communication World, 2018(12): 1–2 (in Chinese with English abstract). [doi: 10.3969/J.ISSN.1672-7274.2018.12.001]

[6] Zheng Y, Li QN, Chen YK, Xie X, Ma WY. Understanding mobility based on GPS data. In: Proc. of the 10th Int'l Conf. on Ubiquitous Computing. New York: Association for Computing Machinery, 2008. 312–321. [doi: 10.1145/1409635.1409677]

[7] Zheng Y, Xie X, Ma WY. GeoLife: A collaborative social networking service among user, location and trajectory. IEEE Data Engineering Bulletin, 2010, 33(2): 32–39.

[8] Zheng Y, Zhang LZ, Xie X, Ma WY. Mining interesting locations and travel sequences from GPS trajectories. In: Proc. of the 18th Int'l Conf. on World Wide Web. New York: Association for Computing Machinery, 2009. 791–800. [doi: 10.1145/1526709.1526816]

[9] Lou Y, Zhang CY, Zheng Y, Xie X, Wang W, Huang Y. Map-matching for low-sampling-rate GPS trajectories. In: Proc. of the 17th ACM SIGSPATIAL Int'l Conf. on Advances in Geographic Information Systems. New York: Association for Computing Machinery, 2009. 352–361. [doi: 10.1145/1653771.1653820]

[10] Piorkowski M, Sarafijanovic-Djukic N, Grossglauser M. CRAWDAD dataset epfl/mobility (v. 2009-02-24), traceset: cab. 2009. https://crawdad.org/epfl/mobility/20090224/cab [doi: 10.15783/C7J010]

[11] Yuan J, Zheng Y, Xie X, Sun GZ. Driving with knowledge from the physical world. In: Proc. of the 17th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: Association for Computing Machinery, 2011. 316–324. [doi: 10.1145/2020408.2020462]

[12] Yuan J, Zheng Y, Zhang CY, Xie WL, Xie X, Sun GZ, Huang Y. T-drive: Driving directions based on taxi trajectories. In: Proc. of the 18th SIGSPATIAL Int'l Conf. on Advances in Geographic Information Systems. New York: Association for Computing Machinery, 2010. 99–108. [doi: 10.1145/1869790.1869807]

[13] Yuan J, Zheng Y, Xie X. Discovering regions of different functions in a city using human mobility and POIs. In: Proc. of the 18th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: Association for Computing Machinery, 2012. 186–194. [doi: 10.1145/2339530.2339561]

[14] Yang C, Gidófalvi G. Fast map matching, an algorithm integrating hidden Markov model with precomputation. Int'l Journal of Geographical Information Science, 2018, 32(3): 547–570. [doi: 10.1080/13658816.2017.1400548]

[15] Douglas DH, Peucker TK. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The Int'l Journal for Geographic Information and Geovisualization, 1973, 10(2): 112–122. [doi: 10.3138/FM57-6770-U75U-7727]

[16] Hershberger JE, Snoeyink J. Speeding up the douglas-peucker line-simplification algorithm. In: Proc. of the 5th Int'l Symp. on Spatial

Data Handling. Vancouver: University of British Columbia, Department of Computer Science, 1992. 134–143.

[17] Keogh E, Chu S, Hart D, Pazzani M. An online algorithm for segmenting time series. In: Proc. of the 2001 IEEE Int'l Conf. on Data Mining. San Jose: IEEE, 2001. 289–296. [doi: 10.1109/icdm.2001.989531]

[18] Meratnia N, De By RA. Spatiotemporal compression techniques for moving point objects. In: Proc. of the 9th Int'l Conf. on Extending Database Technology. Heraklion: Springer, 2004. 765–782. [doi: 10.1007/978-3-540-24741-8_44]

[19] Potamias M, Patroumpas K, Sellis T. Sampling trajectory streams with spatiotemporal criteria. In: Proc. of the 18th Int'l Conf. on Scientific and Statistical Database Management. Vienna: IEEE, 2006. 275–284. [doi: 10.1109/SSDBM.2006.45]

[20] Muckell J, Hwang JH, Patil V, Lawson CT, Ping F, Ravi SS. SQUISH: An online approach for GPS trajectory compression. In: Proc. of the 2nd Int'l Conf. on Computing for Geospatial Research & Applications. New York: Association for Computing Machinery, 2011. 1–8. [doi: 10.1145/1999320.1999333]

[21] Wei H, Wang Y, Forman G, Zhu Y. Map matching by fréchet distance and global weight optimization. Technical Report, Department of Computer Science and Engineering, Citeseer, 2013. 19.

[22] Lu J, Wang P. MAP matching with hidden Markov model on MapReduce. Computer Applications and Software, 2018, 35(2): 7–15, 73 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-386x.2018.02.002]

[23] Newson P, Krumm J. Hidden Markov map matching through noise and sparseness. In: Proc. of the 17th ACM SIGSPATIAL Int'l Conf. on Advances in Geographic Information Systems. New York: ACM, 2009. 336–343. [doi: 10.1145/1653771.1653818]

[24] Yuan J, Zheng Y, Zhang CY, Xie X, Sun GZ. An interactive-voting based map matching algorithm. In: Proc. of the 11th Int'l Conf. on Mobile Data Management. Kansas City: IEEE, 2010. 43–52. [doi: 10.1109/MDM.2010.14]

[25] Zhang YY, He YL. An advanced interactive-voting based map matching algorithm for low-sampling-rate GPS data. In: Proc. of the 15th IEEE Int'l Conf. on Networking, Sensing and Control. Zhuhai: IEEE, 2018. 1–7. [doi: 10.1109/ICNSC.2018.8361315]

[26] Quddus MA, Ochieng WY, Noland RB. Current map-matching algorithms for transport applications: State-of-the art and future research directions. Transportation Research Part C: Emerging Technologies, 2007, 15(5): 312–328. [doi: 10.1016/j.trc.2007.05.002]

[27] Chawathe SS. Segment-based map matching. In: Proc. of the 2007 IEEE Intelligent Vehicles Symp. Istanbul: IEEE, 2007. 1190–1197. [doi: 10.1109/IVS.2007.4290280]

[28] Wenk C, Salas R, Pfoser D. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In: Proc. of the 18th Int'l Conf. on Scientific and Statistical Database Management. Vienna: IEEE, 2006. 379–388. [doi: 10.1109/SSDBM.2006.11]

[29] Liu MS, Zhang L, Ge JL, Long Y, Che WT. Map matching for urban high-sampling-frequency GPS trajectories. ISPRS Int'l Journal of Geo-Information, 2020, 9(1): 31. [doi: 10.3390/ijgi9010031]

[30] Zheng K, Zheng Y, Xie X, Zhou XF. Reducing uncertainty of low-sampling-rate trajectories. In: Proc. of the 28th IEEE Int'l Conf. on Data Engineering. Arlington: IEEE, 2012. 1144–1155. [doi: 10.1109/ICDE.2012.42]

[31] Taylor G, Blewitt G, Steup D, Corbett S, Car A. Road reduction filtering for GPS-GIS navigation. Trans. in GIS, 2001, 5(3): 193–207. [doi: 10.1111/1467-9671.00077]

[32] Yu M. Improved positioning of land vehicle in its using digital map and other accessory information [Ph.D. Thesis]. Hong Kong: Hong Kong Polytechnic University, 2006.

[33] Gerlach K, Rahmig C. Multi-hypothesis based map-matching algorithm for precise train positioning. In: Proc. of the 12th Int'l Conf. on Information Fusion. Seattle: IEEE, 2009. 1363–1369.

[34] Obradovic D, Lenz H, Schupfner M. Fusion of map and sensor data in a modern car navigation system. Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology, 2006, 45(1): 111–122. [doi: 10.1007/s11265-006-9775-4]

[35] Shang JB, Zheng Y, Tong WZ, Chang E, Yu Y. Inferring gas consumption and pollution emission of vehicles throughout a city. In: Proc. of the 20th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: Association for Computing Machinery, 2014. 1027–1036. [doi: 10.1145/2623330.2623653]

[36] Wang K, Li P, Jin Y, Liu Y. Dual-mode map matching algorithm based on three evidences DS theory. Computer Engineering, 2018, 44(5): 316–321 (in Chinese with English abstract). [doi: 10.19678/j.issn.1000-3428.0046583]

[37] Han JY, Chen KJ, Liu XP. Map-matching based on ensemble learning of naïve Bayesian models. Computer Engineering and Design, 2014, 35(3): 875–879 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-7024.2014.03.026]

[38] Yan SL, Yu J, Zhou HP. IIVMM: An improved interactive voting-based map matching algorithm for low-sampling-rate GPS trajectories. Computer Science, 2019, 46(9): 325–332 (in Chinese with English abstract). [doi: 10.11896/j.issn.1002-137X.2019.09.050]

[39] Taguchi S, Koide S, Yoshimura T. Online map matching with route prediction. IEEE Trans. on Intelligent Transportation Systems, 2018, 20(1): 338–347. [doi: 10.1109/TITS.2018.2812147]

[40] Sharath MN, Velaga NR, Quddus MA. A dynamic two-dimensional (D2D) weight-based map-matching algorithm. Transportation

Research Part C: Emerging Technologies, 2019, 98: 409–432. [doi: 10.1016/j.trc.2018.12.009]

[41] Szwed P, Pekala K. An incremental map-matching algorithm based on hidden Markov model. In: Proc. of the 13th Int'l Conf. on Artificial Intelligence and Soft Computing. Zakopane: Springer, 2014. 579–590. [doi: 10.1007/978-3-319-07176-3_51]

[42] Gao WC, Li GL, Ta N. Survey of map matching algorithms. Ruan Jian Xue Bao/Journal of Software, 2018, 29(2): 225–250 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5424.htm [doi: 10.13328/j.cnki.jos.005424]

[43] Chen BY, Yuan H, Li QQ, Lam WHK, Shaw SL, Yan K. Map-matching algorithm for large-scale low-frequency floating car data. Int'l Journal of Geographical Information Science, 2014, 28(1): 22–38. [doi: 10.1080/13658816.2013.816427]

[44] Zhao DB, Liu XM, Guo L. Real time map matching algorithm of floating car in support of spatial grid index. Journal of Computer-aided Design & Computer Graphics, 2014, 26(9): 1550–1556 (in Chinese with English abstract).

[45] Zeng Z, Zhang T, Zou HX, Wu ZH. Acceleration of map matching for floating car data by exploiting travelling velocity. In: Proc. of the 18th IEEE Int'l Conf. on Intelligent Transportation Systems. Las Palmas: IEEE, 2015. 2895–2899. [doi: 10.1109/ITSC.2015.465]

[46] Zeidan A, Lagerspetz E, Zhao K, Nurmi P, Tarkoma S, Vo HT. Geomatch: Efficient large-scale map matching on Apache spark. ACM/IMS Trans. on Data Science, 2020, 1(3): 21. [doi: 10.1145/3402904]

[47] Almeida AMR, Lima MIV, Macedo JAF, Machado JC. DMM: A distributed map-matching algorithm using the mapreduce paradigm. In: Proc. of the 19th IEEE Int'l Conf. on Intelligent Transportation Systems. Rio de Janeiro: IEEE, 2016. 1706–1711. [doi: 10.1109/ITSC.2016.7795788]

[48] Peixoto DA, Nguyen HQV, Zheng BL, Zhou XF. A framework for parallel map-matching at scale using spark. Distributed and Parallel Databases, 2019, 37(4): 697–720. [doi: 10.1007/s10619-018-7254-0]

[49] Zhang DX, Ding MT, Yang DY, Liu Y, Fan J, Shen HT. Trajectory simplification: An experimental study and quality analysis. Proc. of the VLDB Endowment, 2018, 11(9): 934–946. [doi: 10.14778/3213880.3213885]

[50] Bellman R. On the approximation of curves by line segments using dynamic programming. Communications of the ACM, 1961, 4(6): 284. [doi: 10.1145/366573.366611]

[51] Long H, Zhang SK, Sun PH. Trajectory compression algorithm with adaptive parameter. Application Research of Computers, 2018, 35(3): 685–688, 716 (in Chinese with English abstract). [doi: 10.3969/j.issn.1001-3695.2018.03.010]

[52] Zhang T, Yang ZY. Segmentation-based trajectory simplification algorithm. Application Research of Computers, 2019, 36(7): 2044–2048 (in Chinese with English abstract). [doi: 10.19734/j.issn.1001-3695.2018.02.0090]

[53] Yan ZX, Parent C, Spaccapietra S, Chakraborty D. A hybrid model and computing platform for spatio-semantic trajectories. In: Proc. of the 7th Extended Semantic Web Conf. Heraklion: Springer, 2010. 60–75. [doi: 10.1007/978-3-642-13486-9_5]

[54] Zimmermann M, Kirste T, Spiliopoulou M. Finding stops in error-prone trajectories of moving objects with time-based clustering. In: Proc. of the 2009 Int'l Conf. on Intelligent Interactive Assistance and Mobile Multimedia Computing. Rostock: Springer, 2009. 275–286. [doi: 10.1007/978-3-642-10263-9_24]

[55] Palma AT, Bogorny V, Kuijpers B, Alvares LO. A clustering-based approach for discovering interesting places in trajectories. In: Proc. of the 2008 ACM Symp. on Applied Computing. New York: Association for Computing Machinery, 2008. 863–868. [doi: 10.1145/1363686.1363886]

[56] Ester M, Kriegel HP, Sander J, Xu XW. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining. Portland: AAAI Press, 1996. 226–231.

[57] Birant D, Kut A. ST-DBSCAN: An algorithm for clustering spatial-temporal data. Data & Knowledge Engineering, 2007, 60(1): 208–221. [doi: 10.1016/j.datak.2006.01.013]

[58] Kalman RE. A new approach to linear filtering and prediction problems. Journal of Basic Engineering, 1960, 82(1): 35–45. [doi: 10.1115/1.3662552]

[59] Sunahara Y. An approximate method of state estimation for nonlinear dynamical systems. Journal of Basic Engineering, 1970, 92(2): 385–393. [doi: 10.1115/1.3425006]

[60] Bucy RS, Senne KD. Digital synthesis of non-linear filters. Automatica, 1971, 7(3): 287–298. [doi: 10.1016/0005-1098(71)90121-X]

[61] Julier SJ, Uhlmann JK. Unscented filtering and nonlinear estimation. Proc. of the IEEE, 2004, 92(3): 401–422. [doi: 10.1109/JPROC.2003.823141]

[62] Feng AQ, Qian LP, Ouyang JY, Wu Y. Vehicular networking enabled vehicle state prediction with two-level quantized adaptive Kalman filtering. Computer Science, 2020, 47(5): 230–235 (in Chinese with English abstract). [doi: 10.11896/jsjkx.190300155]

[63] Wang Y, Deng QX, Liu GH, Yin B. Dynamic target tracking and predicting algorithm based on combination of motion equation and Kalman filter. Computer Science, 2015, 42(12): 76–81 (in Chinese with English abstract). [doi: 10.11896/j.issn.1002-137X.2015.12.017]

[64] Wang ZH, Liang DT, Liang D, Zhang JC, Liu HJ. A SLAM method based on inertial/magnetic sensors and monocular vision fusion. Robot, 2018, 40(6): 933–941 (in Chinese with English abstract). [doi: 10.13973/j.cnki.robot.170683]

[65] Smaili C, El Najjar MEB, Charpillet F. A hybrid Bayesian framework for map matching: Formulation using switching Kalman filter. Journal of Intelligent & Robotic Systems, 2014, 74(3): 725–743. [doi: 10.1007/s10846-013-9844-4]

[66] Preparata FP, Ian Shamos M. Computational Geometry: An Introduction. New York: Springer, 1985. [doi: 10.1007/978-1-4612-1098-6]

附中文参考文献:

[1] 产业信息网. 2018年中国电动自行车产量、电动自行车保有量、各省市产量发展趋势分析. 2019. http://www.chyxx.com/industry/201911/801292.html

[2] 高强, 张风荔, 王瑞锦, 周帆. 轨迹大数据: 数据处理关键技术研究综述. 软件学报, 2017, 28(4): 959–992. http://www.jos.org.cn/1000-9825/5143.htm [doi: 10.13328/j.cnki.jos.005143]

[5] 刘基余. GNSS卫星导航定位的精度、误差与偏差——GNSS导航定位误差之一. 数字通信世界, 2018(12): 1–2. [doi: 10.3969/J.ISSN.1672-7274.2018.12.001]

[22] 陆健, 王鹏. 隐马尔可夫模型路网匹配的MapReduce实现. 计算机应用与软件, 2018, 35(2): 7–15, 73. [doi: 10.3969/j.issn.1000-386x.2018.02.002]

[36] 王科, 李鹏, 金瑜, 刘宇. 基于三证据DS理论的双模式地图匹配算法. 计算机工程, 2018, 44(5): 316–321. [doi: 10.19678/j.issn.1000-3428.0046583]

[37] 韩京宇, 陈可佳, 刘茜萍. 基于集成朴素贝叶斯模型的在线地图匹配方法. 计算机工程与设计, 2014, 35(3): 875–879. [doi: 10.3969/j.issn.1000-7024.2014.03.026]

[38] 严盛隆, 于娟, 周后盘. IIVMM: 针对低频GPS轨迹的改进交互式投票匹配算法. 计算机科学, 2019, 46(9): 325–332. [doi: 10.11896/j.issn.1002-137X.2019.09.050]

[42] 高文超, 李国良, 塔娜. 路网匹配算法综述. 软件学报, 2018, 29(2): 225–250. http://www.jos.org.cn/1000-9825/5424.htm [doi: 10.13328/j.cnki.jos.005424]

[44] 赵东保, 刘雪梅, 郭黎. 网格索引支持下的大规模浮动车实时地图匹配方法. 计算机辅助设计与图形学学报, 2014, 26(9): 1550–1556.

[51] 龙浩, 张书奎, 孙鹏辉. 自适应参数的轨迹压缩算法. 计算机应用研究, 2018, 35(3): 685–688, 716. [doi: 10.3969/j.issn.1001-3695.2018.03.010]

[52] 张甜, 杨智应. 基于分段的移动对象轨迹简化算法. 计算机应用研究, 2019, 36(7): 2044–2048. [doi: 10.19734/j.issn.1001-3695.2018.02.0090]

[62] 冯安琪, 钱丽萍, 欧阳金源, 吴远. 车联网络通过两级量化自适应卡尔曼滤波实现车辆状态预测. 计算机科学, 2020, 47(5): 230–235. [doi: 10.11896/jsjkx.190300155]

[63] 王妍, 邓庆绪, 刘赓浩, 银彪. 结合运动方程与卡尔曼滤波的动态目标追踪预测算法. 计算机科学, 2015, 42(12): 76–81. [doi: 10.11896/j.issn.1002-137X.2015.12.017]

[64] 王泽华, 梁冬泰, 梁丹, 章家成, 刘华杰. 基于惯性/磁力传感器与单目视觉融合的SLAM方法. 机器人, 2018, 40(6): 933–941. [doi: 10.13973/j.cnki.robot.170683]

王东京(1991－), 男, 博士, 讲师 , CCF 专业会员, 主要研究领域为推荐系统, 数据挖掘, 业务过程管理.

俞东进(1969－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为软件工程理论和方法, 业务过程管理, 行业大数据.

刘继涛(1996－), 男, 硕士, 主要研究领域为时空数据挖掘.