中国科学: 信息科学 2017 年 第 47 卷 第 11 期: 1523–1537
SCIENTIA SINICA Informationis
机器学习专题 · 论文

一种基于空间编码技术的轨迹特征提取方法

乔少杰[1], 韩楠[2], 李天瑞[3], 熊熙[1], 元昌安[4], 黄江涛[4*], 王晓腾[3]

1. 成都信息工程大学网络空间安全学院, 成都 610225
2. 成都信息工程大学管理学院, 成都 610103
3. 西南交通大学信息科学与技术学院, 成都 611756
4. 广西师范学院计算机与信息工程学院, 南宁 541004
* 通信作者. E-mail: hjt@gxtc.edu.cn

收稿日期: 2017–01–08; 接受日期: 2017–06–26; 网络出版日期: 2017–11–09
国家自然科学基金 (批准号: 61772091, 61100045, 61363037)、教育部人文社会科学研究规划基金 (批准号: 15YJAZH058)、教育部人文社会科学研究青年基金 (批准号: 14YJCZH046)、四川省教育厅 (批准号: 14ZB0458)、广西自然科学基金重点项目 (批准号: 2014GXNSFDA118037) 和成都信息工程大学引进人才科研启动项目 (批准号: KYTZ201715, KYTZ201750) 资助

摘要 GPS 数据存在位置精度偏差而且易受噪声干扰, 大规模数据挖掘前需要进行轨迹特征提取. 本文提出基于 GeoHash 的空间编码技术 GeoHashTree 对时空点进行索引, 提高邻域轨迹点查询效率. 将 GeoHashTree 应用于轨迹聚类, 提出一种改进的基于密度的轨迹聚类算法, 使聚类中最近邻点查询时间复杂度由 $O(n^2)$ 降为 $O(n\log n)$. 以提取角度变化点为基础, 通过聚类对角度变化点进行深层次特征提取, 实现特征点的准确识别. 大量真实 GPS 数据上的实验结果表明: 相比传统算法, 基于 GeoHashTree 空间索引结构的轨迹聚类算法时间开销平均提升 90.89%, 同时保证聚类结果的准确性. 可视化结果表明: 在大规模数据集上, 轨迹特征提取方法能够准确找到角度变化点, 有效挖掘各类特征点. 此外, 算法不依赖路网数据, 可根据路网实时改变时新增的轨迹数据进行动态更新.

关键词 轨迹, 大数据, 编码方法, 聚类分析, 特征提取

1 引言

智能手机、车载 GPS 终端等设备将网络与现实结合, 帮助我们更好地理解并改善生活方式. 其中, 位置信息扮演着重要的角色. 随着大数据分析处理技术快速发展, 位置数据中蕴藏的价值逐渐被人们所认知和利用 [1]. 在众多位置数据中, 轨迹数据具有独特价值. 不同于签到数据, 轨迹数据是连续、完整的具有时效性的行为展示, 包含出行规律、社交信息等特征, 因此理解并利用好轨迹数据具有重要意义 [2]. 轨迹数据可以由配备 GPS 功能的智能设备采集, 数据能够反映个体的出行规律、行为习惯等特点. 掌握了这些信息, 当我们在使用生活服务类 APP 时, 就可以通过判断、分类, 准确地推送符

引用格式: 乔少杰, 韩楠, 李天瑞, 等. 一种基于空间编码技术的轨迹特征提取方法. 中国科学: 信息科学, 2017, 47: 1523–1537, doi: 10.1360/N112017-00008
Qiao S J, Han N, Li T R, et al. A trajectory feature extraction approach based on spatial coding technique (in Chinese). Sci Sin Inform, 2017, 47: 1523–1537, doi: 10.1360/N112017-00008

合用户需求的信息 [3]. 将单一个体的轨迹信息进行综合, 可以得到群体的移动规律并加以利用. 例如, 动物迁徙行为分析、瞬时人群聚集分析等 [4]. 现实生活中由于人群聚集、疏导滞后等因素导致的踩踏事故时有发生, 如能够实时采集人群的轨迹数据, 通过预测技术, 判断出人群聚集趋向, 可以为集会管理者及时进行疏导提供指示. 此外, 人群聚集情况的实时分析在人流疏导、实时预警等方面具有重要价值.

本文研究大规模轨迹数据特征点提取方法的动机主要基于如下几点考虑:

(1) 在轨迹点获取过程中, 由于 GPS 数据采集设备位置精度的原因, 导航过程中会出现轨迹点偏离道路的问题, 需要通过轨迹特征的提取来保证系统匹配修正的精度和效率 [5].

(2) 移动对象数据存在诸多不确定性因素, 如位置的不确定性、对象存在的不确定性、对象运动行为的不确定性等 [1], 导致原始轨迹数据会存在一定的噪声点, 会影响轨迹大数据查询、挖掘和预测的准确性. 因此, 在进行轨迹大数据挖掘前非常重要的工作是对原始轨迹数据进行特征提取.

(3) 轨迹数据可以准确地反映道路拥堵情况, 通过对轨迹特征点的提取, 可以反映出交通流量, 进而影响道路的规划方案, 实现智能交通控制. 此外, 通过对轨迹数据进行聚类分析, 可以发现功能区设置的缺陷, 帮助我们完善社区配套设施建设.

2 相关工作

轨迹数据往往包含误差, 因此需要对轨迹数据进行特征提取, 包括去噪、校准、分段、化简、地图匹配等一系列操作 [6]. 其中, 轨迹数据去噪主要过滤错误采样数据和冗余数据, 去除因采样设配硬件问题产生的漂移点、重复点等, 从而为后续研究排除干扰 [7]. 常用方法包括规则过滤、Bayes 滤波、Kalman 滤波和粒子滤波等, 通过将轨迹数据匹配映射到路网数据中也可以完成轨迹去噪工作. 此外, 在去噪过程中通常也可以完成对轨迹的校准. 轨迹由于采样经度等因素影响具有偏差, 轨迹校准利用前后轨迹点, 对轨迹连续性、完整性进行提升, 得到高质量数据.

轨迹地图匹配是轨迹特征提取中的重要组成部分, 原始轨迹数据仅由单纯的 GPS 点构成, 为了直观感受轨迹数据, 需要将其匹配到地图数据上. 轨迹数据采集中存在噪声、采样频率低、信号丢失等不足, 通过将轨迹数据映射到路网数据中可以一定程度上抵消数据采集中的偏差. 此外, 地图匹配可以结合多元化的地理信息, 针对轨迹数据, 提供临近点查询、道路规划等功能. Yuan 等 [8] 提出了基于交叉投票方法的地图匹配方法, 该方法针对低采样率情况下轨迹地图匹配率较低的特点, 采用集成学习的思想将不同匹配结果相结合, 从而提高了匹配成功率. Shan 等 [9] 提出了地图自动补全与更新系统, 使用地图匹配算法对轨迹数据集中的数据进行匹配, 可以去除由 GPS 设备本身产生的噪声数据.

目前, 针对轨迹几何特征提取算法的研究较多, 如, Douglas-Peucker 及其衍生算法已经被广泛应用, 但其计算量较大, 容易造成形状特征失真, 在复杂路网环境下容易导致误匹配 [5]. 针对路网要素特点, Guo 等 [10] 提取道路要素特征点, 将复杂的路网匹配转化为特征点匹配, 可以解决复杂的多对多匹配问题. 这一方法需要对道路数据进行预处理, 无法满足系统实时性要求. 针对大规模浮动车数据量庞大, 地图匹配时对实时性要求较高的特点, 文献 [11] 提出一种离线计算与在线匹配相结合的浮动车轨迹实时局部地图匹配方法. 上述特征提取方法在提取效率和准确度上都有着不错的效果, 然而这些特征提取算法往往仅针对曲线几何特征进行提取, 没有考虑实际车辆行驶过程中由于驾驶习惯不同或紧急避让等突发事件因素的干扰. 此外, 在道路交叉口处的复杂几何特征计算量相对较大, 效率偏低会影响后续位置匹配和误差修正的进行 [5].

为了提取有意义的轨迹特征点, Feng 等 [12] 提出了一种新型轨迹降维方法抽取有意义的轨迹结构

特征, 提高了异常轨迹检测的准确性. Zabalza 等 [13] 将奇异谱分析技术应用于二维轨迹时空点的提取, 通过显著性检验研究确定有意义的特征点. Sauer 等 [14] 利用经过索引的轨迹数据高效抽取特征点, 解决了大规模时间戳稀疏的轨迹数据特征提取不准确的问题. 近年来, 将提取的轨迹特征点应用于路径规划研究的工作得到学者的广泛关注. Chen 和 Zhang 等 [15,16] 提出一种基于密度的聚类算法用于发现乘客上下客的热点区域, 构建公交车路线图进而剪枝无效的停车站实现最优的路线规划. 此外, Chen 等 [17] 提出了一种两阶段包裹投递路径规划方法, 离线计算包裹投递最短路线, 并在线自适应规划行车路线, 达到实时响应请求的目的.

综上, 现有轨迹特征提取算法面临的主要困难和挑战包括: (1) 需要进行数据预处理, 数据规模大无法满足具体应用实时性的需求; (2) 未考虑路网中道路复杂多变性; (3) 地图匹配算法精度不高等问题. 本文利用 GeoHash 空间索引技术降低邻域轨迹点查询的时间代价, 提高后期轨迹特征提取的效率. 通过计算轨迹角度变化特征点对其进行聚类特征抽取, 保证轨迹挖掘结果的准确性.

3 问题描述

不同类型的轨迹数据具有不同特征, 随着各类具有 GPS 功能的移动终端的普及, 轨迹数据的获取变得简单高效. 但由于数据采集精度的误差及移动对象运动的诸多不确定性等因素导致原始轨迹数据不能直接用于位置精度要求较高的应用中, 需要对轨迹数据进行分析并提取特征点. 本文将城市路网中的移动对象轨迹作为研究对象, 提取轨迹特征. 首先给出轨迹的定义, 描述如下.

定义1 (轨迹) 给定 GPS 点序列 T=$\{p_1, p_2, p_3, \ldots, p_n\}$, p_i=(α_i, β_i, t_i), 其中, α 表示纬度坐标, β 表示经度坐标, t 表示时间戳, 对于任意 $i < j$ 有 $p_i.t < p_j.t$, $p_i \in P$, P 为 GPS 点集, 于是称 T 为轨迹.

为了从轨迹数据中获取信息, 通常涉及到轨迹分类、轨迹比较等操作. 例如, 无法找到两条由相同 GPS 点组成的轨迹. 原始的轨迹数据之间并不能直接进行上述操作, 需要对轨迹数据进行转化, 常见做法是进行地图匹配. 已有地图匹配方法存在两方面不足: (1) 依赖大量的路网数据, 且要求路网数据尽可能精准, 一旦城市路网规划发生变化, 地图数据无法实时做出改变, 导致匹配精度下降; (2) 路网匹配计算本身需耗费大量时间, 增加了数据处理的复杂度. 基于路网的轨迹数据包含一定的路网特征, 可以加以利用. 本文从轨迹数据本身出发, 尽可能利用轨迹数据的自身信息完成轨迹特征提取.

通常利用网格对轨迹数据进行空间分割, 根据轨迹相对于网格的位置进行匹配. 基于网格的划分方法计算效率高, 但存在边界问题 [7], 即两条轨迹本应被识别为相似轨迹, 但是在划分后的空间中并不相连. 此外, 基于路网的轨迹数据中常常存在长直路线, 使用网格表示法容易产生较多冗余. 对轨迹数据直接进行基于位置的聚类是常用的方法. 当移动对象长时间停留在某一位置时, 由于采样频率恒定, 因此在该处会产生大量的数据点, 通过聚类就可以发现移动对象的兴趣点. 此种方法对于稀疏数据具有一定效果, 例如基于位置的签到数据等. 但是此方法常常会产生诸多无意义的聚类点, 例如移动对象在交通路口等待的情况. 另外当移动对象进入室内丢失 GPS 信号时, 该方法也无法正确获取兴趣点. 因此, 停留点提取算法得到普遍应用. 停留点提取算法在以智能手机为 GPS 终端, 相对稀疏的轨迹数据上有较好应用, 但是对于车载 GPS 等设备采集的轨迹数据, 其提取效果并不理想.

结合停留点提取算法, 本文提出一种新的轨迹特征提取方法, 引入角度变化作为特征, 当移动对象在路网中运动时, 其角度变化多数情况下意味着行驶道路的更改, 假设移动对象在同一道路上的运动特征保持不变, 于是关注的变化就是行驶道路的变化. 当移动对象行驶角度发生变化时, 认为其运动特征相应发生改变, 因此角度变化点对应特征变化点, 即轨迹特征点.

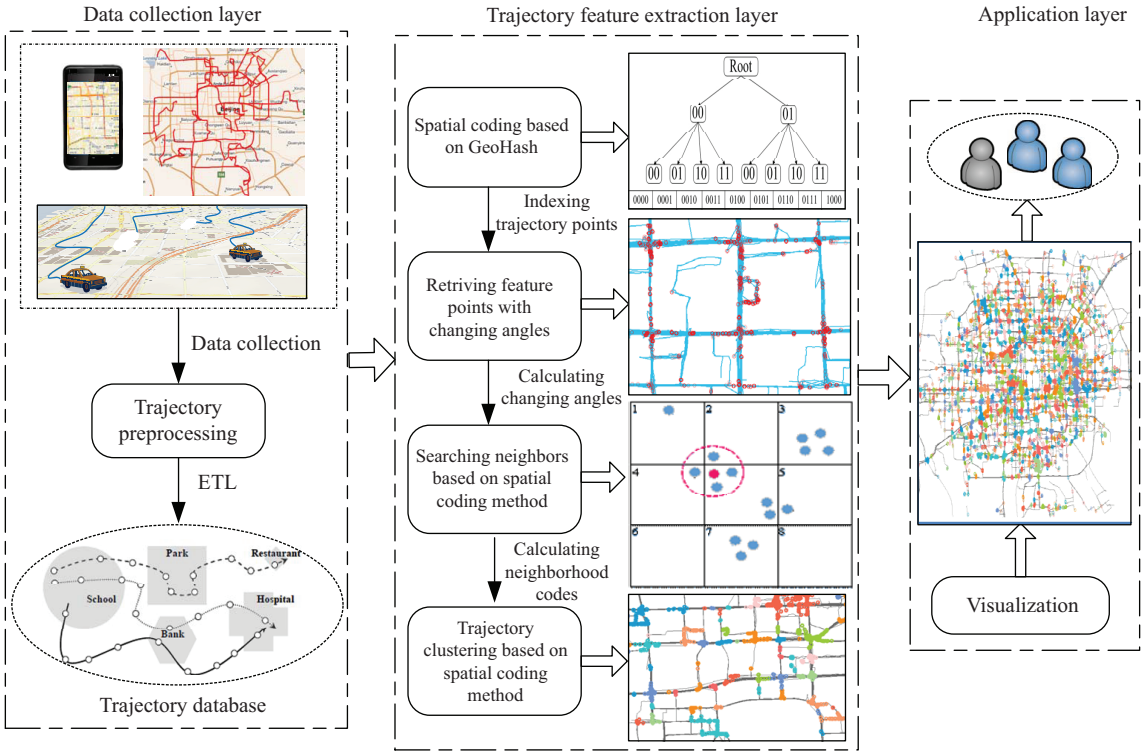

图 1　(网络版彩图) 基于空间编码技术的轨迹特征提取工作原理

Figure 1　(Color online) Working mechanism of trajectory feature extraction based on spatial coding technique

本文提出一种新的基于 GeoHash 空间编码 [18] 的轨迹特征提取方法, 工作原理如图 1 所示. 算法包含 3 个层次: (1) 数据感知层, 将配有 GPS 功能的移动便携设备采集的数据通过轨迹点提取、转换和加载 (ETL 过程) 存储到轨迹数据库中; (2) 轨迹特征提取层, 首先对轨迹点进行 GeoHash 编码, 构建 GeoHashTree; 然后, 计算轨迹角度变化较大的偏转点置于特征点集合中; 再者, 利用 8 邻域查找算法查询临近轨迹点并利用密度聚类思想计算轨迹点之间的距离进而实现轨迹的聚集操作; (3) 应用层, 对提取的轨迹特征点进行可视化输出并应用于基于位置的各类服务中.

经过特征提取后的轨迹数据具有广阔的应用场景, 以智慧交通中交通流预测问题作为概貌进行描述. 智能交通旨在建立起一种在较大范围内, 全方位发挥作用的实时、准确、高效的综合运输和管理系统. 应用轨迹特征提取点, 可以在路口处预测每天哪一时段, 具体在哪一区域是机动车出行的高峰期. 通过预测交通流可以辅助决策使路网上的交通流量处于最佳状态, 改善交通拥挤和阻塞状况, 最大限度地提高交通的通行能力, 提高整个公路运输系统的机动性、安全性和运输效率.

4　基于 GeoHash 的区域编码技术

在特征点提取过程中, 需要对特征点进行聚类, 传统基于密度的聚类算法对于轨迹大数据的支持不足, 因此考虑引入空间编码技术优化聚类过程. GPS 数据主要包含经纬度坐标, 使用传统方法分别在其精度和纬度值上建立索引, 不能满足轨迹聚类中的邻近点快速查询的需求. 为了解决这一问题, 本文引入 GeoHash 空间编码技术对轨迹点进行索引, 将二维经纬度数据转化成一维编码字符串, 从

表 1 纬度空间编码方法举例

Table 1 Example of latitude spatial coding method

Area (°)	0-area (°)	1-area (°)	Code
$-90 \sim 90$	$-90 \sim 0$	$0 \sim 90$	1
$0 \sim 90$	$0 \sim 45$	$45 \sim 90$	0
$0 \sim 45$	$0 \sim 22.5$	$22.5 \sim 45$	1
$22.5 \sim 45$	$22.5 \sim 37.5$	$37.5 \sim 45$	1
$33.75 \sim 45$	$33.75 \sim 39.375$	$39.375 \sim 45$	1
$39.375 \sim 45$	$39.375 \sim 42.1875$	$42.1875 \sim 45$	0
$39.375 \sim 42.1875$	$39.375 \sim 40.7812$	$40.7812 \sim 42.1875$	0
$39.375 \sim 40.7812$	$39.375 \sim 40.0781$	$40.0781 \sim 40.7812$	0

而满足了空间索引需求. GPS 坐标点覆盖整个地球表面, 因此将地球表面视作一完整平面, 南北包括 90°N ∼ 90°S, 东西包括 180°W ∼ 180°E.

编码过程可以视作对空间区域的划分过程, 通过划分逐渐逼近实际值. 以经度为例, 将西经 180° 至东经 180° 划分为 $[-180, 0]$ 和 [0, 180] 两个区间, 若待编码点落在左边区间, 则第一位编码为 0, 否则为 1. 如果坐落在 [0, 180] 区间, 那么进一步将 [0, 180] 区间分为 [0, 90] 和 [90, 180] 两个区间进行第 2 位编码, 以此类推, 递归编码. 编码的总长度根据所需精度进行选择. 纬度编码过程与经度相似, 通过迭代编码后得到长度相同的经度和纬度编码, 然后依次取纬度第 i 位编码和经度第 i 位编码合成空间编码的第 $2i$ 和 $2i+1$ 位 (i 从 0 开始取值). 下面给出一个具体编码示例, 如下所示.

例 1 空间编码过程举例. 以北京的纬度 39.904844 为例进行编码, 表 1 展示了对其进行长度为 8 的逐次逼近的编码过程, 通过不断递归并缩小范围, 最终得到纬度的编码结果为 10111000.

算法 1 GeoHash(lat, lng, n)

for each i=0 to $n/2$ **do**
  $\overline{\text{la}} \leftarrow (b+t)/2$;
  if lat $> \overline{\text{la}}$ **then**
    $\alpha[i] \leftarrow 1$, $b \leftarrow \overline{\text{la}}$;
  else
    $\alpha[i] \leftarrow 0$, $t \leftarrow \overline{\text{la}}$;
  end if
  $\overline{\text{ln}} \leftarrow (l+r)/2$;
  if lng $> \overline{\text{ln}}$ **then**
    $\beta[i] \leftarrow 1$, $l \leftarrow \overline{\text{ln}}$;
  else
    $\beta[i] \leftarrow 0$, $r \leftarrow \overline{\text{ln}}$;
  end if
end for
$c \leftarrow \text{combine}(\alpha, \beta)$;
return c.

算法 1 给出 GeoHash 空间编码方法的具体实现过程, 其中输入参数: lat 和 lng 分别表示 GPS 点纬度和经度坐标, n 表示初始编码长度, c 表示输出的编码, b 和 t 分别表示当前区域纬度最小值和最大值, 初始值分别是 $-90.0°$ 和 $90.0°$, l 和 r 分别表示当前区域经度最小值和最大值, 初始值分别是

$-180.0°$ 和 $180.0°$. 算法第 2 ~ 7 行实现纬度编码, 第 8 ~ 13 行实现经度编码. 第 2 行取当前纬度范围的中点; 第 3 ~ 7 行判断纬度 lat 在当前范围内所处区间, 其中 n 表示编码长度, 若在右区间则编码为 0, 否则编码为 1, 然后缩小当前范围; 第 8 行取当前经度范围的中点; 第 9 ~ 13 行判断经度 lng 在当前范围内所处区间, 若在左区间则编码为 0, 否则编码为 1, 然后缩小当前范围; 第 15 行调用 combine(·) 方法, 将 α 和 β 编码进行合并; 第 16 行输出编码 c. 算法 1 采用二分查找的思想, 其时间复杂度是 $O(\log(n))$, 其中 n 表示初始编码长度.

5 基于 GeoHashTree 的轨迹聚类算法

5.1 轨迹聚类算法

轨迹聚类通常采用 DBScan 算法 [19], 当特征点较多时, 迭代计算特征点之间距离的计算开销非常大, 需要根据数据的特征进行优化. 对于轨迹特征点, 并不需要遍历计算数据集中所有的特征点与当前点的距离, 因为所提取的特征点所在的簇的分布半径通常在某一街区范围内, 仅需关注小范围内的相邻特征点. GeoHash 空间编码算法采用类似二分查找的思想, 逐步缩小搜索范围, 最终定位到查找对象. 本节所提聚类算法的优势在于: 基于 GeoHash 空间编码算法的二分查找特性, 在 DBScan 中引入空间索引查找邻近点, 仅计算相邻区域内特征点的距离, 从而缩小计算范围, 降低计算复杂度.

轨迹聚类时对 DBScan 中 getNeighbors 进行改进, 使用 searchNeighbors 进行替换查找轨迹邻域点, 从而在搜索过程中增加空间编码特性. 为了快速地查找邻近特征点, 本文在 GeoHash 空间编码基础上提出 GeoHashTree, 对数据集中的特征点进行编码和查找. GeoHashTree 是一种类似前缀树的数据结构, 如图 2 所示, Root 节点作为所有节点的根, 每一层子节点均表示对父节点所包含区域的一次划分. 图 2 中 GeoHashTree 完成了区域的两次划分, 编码长度为 4, 每一个子节点包含对应区域内所有轨迹点.

searchNeighbors 算法基本思想是将特征点逐一加入 GeoHashTree 中, 将待查点进行 GeoHash 编码, 查询后输出邻近点集合作为结果, 具体内容如算法 2 所示: 第 1 行利用轨迹数据建立一棵 GeoHashTree; 第 2 行将待查找点 p 进行 GeoHash 编码; 第 3 ~ 5 行将编码后的待查找点作为输入, 在中心区域内查找; 第 6 ~ 10 行根据邻域计算方法, 利用 getNeighbors(·) 函数计算得到中心区域的 8 个相邻的区域, 其中 e 表示邻域半径; 第 11 行输出查找结果集合 D.

算法 2 searchNeighbors(p, e)

```
1: create a GeoHashTree;
2: c ← encode(p);
3: for each i ∈ GeoHashTree(c) do
4:     D.add(i);
5: end for
6: for each n ∈ getNeighbors(c) do
7:     for each i ∈ GeoHashTree(n) do
8:         D.add(i);
9:     end for
10: end for
11: return D.
```

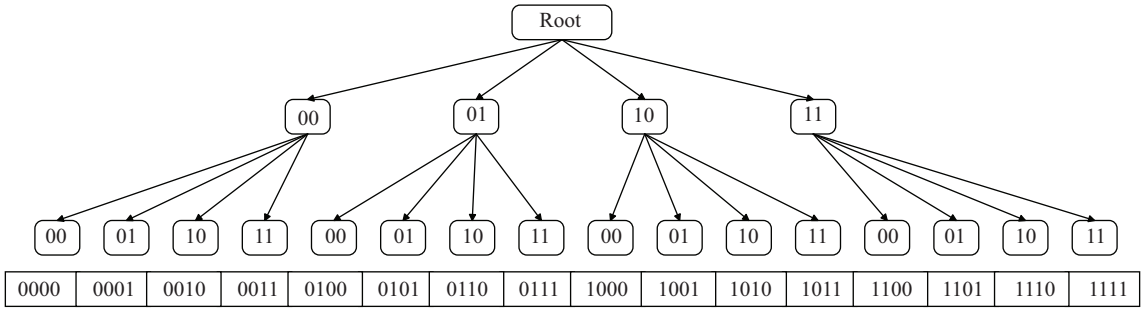

图 2 GeoHashTree 数据结构举例

Figure 2 Example of the data structure of a GeoHashTree

算法 3 基于 DBScan 的改进轨迹聚类算法 —— DBScan*(D, e, θ)

```
1: C ← 0;
2: for each unvisited point p in D do
3:    mark p as Visited;
4:    N ←searchNeighbors(p, e);
5:    if sizeof(N) < θ then
6:       mark p as Noise;
7:    else
8:       C ← next cluster;
9:       ExpandCluster(p, N, C, e, θ);
10:   end if
11: end for
```

基于 DBScan 算法给出改进的轨迹聚类算法 DBScan*, 如算法 3 所示. 基于 DBScan 的轨迹聚类算法分为 DBScan 和 ExpandCluster 两部分. 在 DBScan 中, 遍历轨迹点集合 D 中所有轨迹点, 第 4 ~ 10 行计算点 p 与所有轨迹点的距离, 将距离小于邻域半径 e 的点存入集合 N 中, 如果 N 小于最少点数 θ 则标记 p 为噪声, 否则以 p 为核心建立新簇 (第 8 行) 并调用 ExpandCluster 算法递归访问 N 中轨迹点 (第 9 行). 改进的 ExpandCluster 算法的不同点在于在查找点 p 的邻近点时利用本文提出的 searchNeighbors 算法得到 p 的 8 邻域, 并在 8 邻域中查找, 无需遍历 p 的所有邻近点, 找到符合所有满足小于半径 e 的邻域点.

综上, DBScan* 算法相比 DBScan 算法的优势在于: DBScan 算法在计算过程中需要多次扫描所有轨迹点, 计算两点之间的距离, 当数据量增大时, 算法计算开销巨大. 针对轨迹点数据的特征, DBScan* 算法引入空间编码技术, 对轨迹点进行编码, 建立查找树. 当需要计算距离时, 利用 searchNeighbors 算法查找临近范围内的点, 从而大大减少了计算量, 提高了算法的运行速率.

5.2 算法复杂性分析

传统的 DBScan 算法采用基于密度的思想, 算法理论上计算复杂度为 $O(n^2)$, 其中 n 为轨迹数据中移动对象数量. 本文针对轨迹点的空间特性, 考虑到聚类对象时分布在平面空间中的经纬度数据, 引入空间编码方法, 对轨迹点进行编码, 并建立索引数据结构 GeoHashTree, 实现了对空间聚类算法的改进和优化. 在改进算法中, 当需要通过遍历其他对象来判断另一对象是否为核心对象时, 首先对所有对象建立查找树, 并对该对象进行编码. 通过查找树, 查找该对象邻近范围内的数据点, 采用

GeoHashTree 使得查找范围逐级递减, 从而大大地减少了计算量, 提高了算法的时间效率. 在采用空间索引的情况下, 聚类算法的时间复杂度下降为 $O(n\log n)$, 本文将在 7.2 小节实验中证明改进聚类算法的时间性能优势.

6 轨迹特征提取算法

为了提取轨迹角度变化特征点, 需要完成轨迹点间距离计算和轨迹角度变化计算, 主要定义如下.

定义2 (轨迹点距离) 已知点 A 和 B, ϕ_1, ϕ_2 分别表示 A, B 的纬度, λ_1, λ_2 分别表示 A, B 的经度, γ 为地球半径, 则轨迹点 A, B 两点距离定位为

$$d(A,B)=2\gamma\times\arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2-\phi_1}{2}\right)+\cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right). \tag{1}$$

定义3 (轨迹角度) 已知轨迹点 A, B, C, 上述 3 点构成的角度 $\angle ACB$ 定义为

$$\cos\angle ACB=\frac{d(B,C)^2+d(A,C)^2-d(A,B)^2}{2d(B,C)\times d(A,C)}. \tag{2}$$

理想情况下, 算法通过对相邻的三个轨迹点进行计算便可得到该段轨迹的角度, 但是轨迹通常发生偏转的过程并不会仅包含一个点, 角度偏转的过程可能包含两个或者三个轨迹点, 单纯计算相邻三个轨迹点的角度并不能得到准确的位置. 为了能够准确计算角度偏转点, 需要以当前点为中心选取角度计算所需的采样点, 因此定义角度计算点采样间隔 τ_i, 如下所示.

定义4 (角度采样点间隔) 角度采样点间隔定义为

$$\tau_i=\frac{a}{2k}\sum_{j=i-k}^{2k}d(p_j,p_{j+1}), \tag{3}$$

其中, k 为计算平均速度的窗口大小, a 为平均速度与采样间隔的相关系数. 通过计算移动对象在当前点 p_i 的平均速度, 可以得到采样间隔.

定义5 (角度阈值 δ) 轨迹发生偏转的最小角度定义为角度阈值, 用 δ 表示.

本文通过角度计算提取轨迹中角度变化的点, 如算法 4 所示: 第 1 ~ 9 行从轨迹数据集 T 中提取角度变化点, 其中 3 ~ 9 行遍历轨迹数据集, 按照角度采样点间隔 τ_i 给出的窗口长度, 使用 calAngle 函数计算当前轨迹点与前序后序轨迹点之间的夹角, 若角度值超过阈值 δ, 则认为该点为角度变化点, 将其加入到集合 D 中; 第 10 行对 D 中轨迹点应用 DBScan* 聚类算法 (详见第 5.1 小节) 得到聚类集合; 第 11 ~ 13 行遍历聚类所得到的聚类集合, 取集合的中点作为特征点数据加入轨迹特征点集合 W 中; 第 14 行输出 W.

特征点提取算法以提取轨迹角度变化点为基础, 首先粗略提取角度变化点集合, 该集合中的数据对轨迹角度变化的反应较为直接, 包含了一定数量的噪声. 角度阈值 δ 的设置会对提取效果产生一定影响. 考虑到轨迹数据所具有的统计特性, 当数据量逐渐增大时, 角度变化点的分布出现聚集, 因此使用基于密度的聚类方法, 对角度变化点集合进行聚类, 可以很好地弥补提取精度不高的问题, 同时降低结果对于 δ 的敏感度, 因为随着数据量的增长, 聚类算法所反映出的是轨迹数据集的整体特征.

算法 4 基于角度变化的轨迹特征点提取算法

```
for each t in T do
    P ← t.getPoints(); //P 表示轨迹点集合
    for i ∈ [0, |P|) do
        τ_i ← get_τ_i(p_i);
        if calAngle(p_{i−τ_i}, p_i, p_{i+τ_i}) > δ then
            D.add(p_i);
        end if
    end for
end for
C ← DBScan*(D, e, θ); //e 表示邻域半径, θ 表示最小邻域点数
for each c in C do
    W.add(getCore(c));
end for
return W.
```

表 2 轨迹数据集描述

Table 2 Description of trajectory datasets

Parameter	Value
Time interval	2008/02/02 ∼ 2012/02/08
No. of taxies	10357
No. of trajectories	>25000
No. of trajectory points	>15000000
Length of trajectories	>9000000 km

7 实验及算法性能分析

根据前文所提出的理论, 利用真实环境下采集的 GPS 数据进行实验. 实验分别对基于空间索引的聚类算法和轨迹特征提取算法进行对比分析, 针对算法的准确率、时间效率等方面进行实验.

7.1 实验环境及数据集描述

实验所用数据集来源于微软亚洲研究院郑宇研究员所领导的 T-Driver 项目 [20], 数据采集自北京市真实路网中的出租车 GPS 设备, 包含 10357 辆出租车超过一周的行驶轨迹数据, 具体描述如表 2 所示.

原始轨迹数据存储在数据库中, 按时间排序, 组成轨迹点的属性包括: 采集日期、采集时间、经度、纬度. 本文中所提到的算法均采用 Java 程序设计语言实现, 利用 Eclipse Juno 集成开发环境. 实验硬件平台为 Intel(R) Core(TM)2 Duo P8700 2.53 GHz CPU, 3 GB 内存, 操作系统平台为 Windows 7.

7.2 基于空间编码的轨迹聚类算法性能分析

本节实验对比传统 DBScan 算法和基于空间索引编码的 DBScan* 聚类算法的时间开销, 如图 3 所示. 实验数据集规模由 1000 个对象逐渐增加至 8000, 通过观察可以发现, 随着实验数据集轨迹数量增加, 未采用空间索引的 DBScan 算法的时间开销显著增长, 而采用 GeoHashTree 空间索引的 DBScan* 算法则增长平缓, 始终保持在较低水平. DBScan* 算法相比于 DBScan 算法, 在平均时间开销上具有

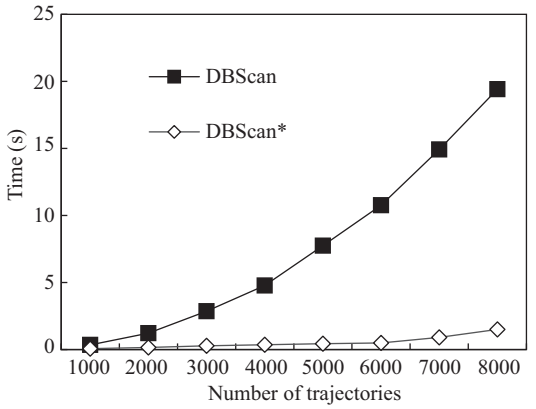

图 3　不同算法时间开销对比

Figure 3　Time cost comparison of distinct algorithms

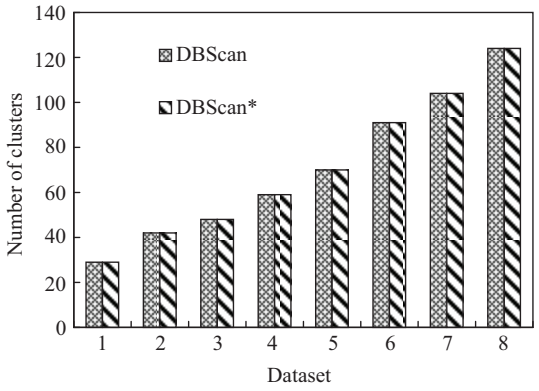

图 4　不同算法聚簇个数对比

Figure 4　Number of clusters comparison of algorithms

表 3　GeoHashTree 构建时间代价

Table 3　Time cost of GeoHashTree generation

No. of trajectory points	2500	5000	10000	20000	100000	250000	500000
Time (ms)	15	31	63	98	266	679	2310

90.89% 的提升, 主要原因在于: 采用了空间编码思想, 应用 GeoHashTree 空间索引结构, 该结构使得聚类过程中邻近点查询算法复杂度由 $O(n^2)$ 降为 $O(n\log n)$, 因此使得聚类算法整体效率得到提升. 图 4 显示了在不同规模数据集上两种聚类算法发现的聚簇个数. 实验结果表明, 采用 GeoHashTree 空间索引结构的聚类算法与传统聚类算法具有相同聚簇个数, 即两种算法的聚类准确性一致.

上述实验结果表明采用 GeoHashTree 空间索引的聚类算法在提升时间性能的同时保证了聚类的准确性. 采用 GeoHashTree 索引结构的聚类算法需要提前建立索引结构, 由此产生的时间开销如表 3 所示.

实验结果表明: 随着数据规模的不断增加, 索引结构建立所需时间开销缓步增加, 当轨迹点的规模达到 500000 级别时, 时间开销才上升至秒级. 尽管采用 GeoHashTree 空间索引结构的聚类算法需要额外的时间开销, 但相比于聚类过程, 索引结构建立所需的时间开销非常小, 可以忽略不计.

采用空间索引的 DBScan* 算法同样易受参数的影响, 观察聚类半径和最小邻域点数对聚类结果的影响, 结果如图 5 和 6 所示. 图 5 显示了聚类半径对聚类结果的影响, 通过观察可以发现: 随着聚类半径的增加, 形成的聚类个数逐渐下降, 这与预期结果相吻合. 针对轨迹数据, 由于其通常受到路网走向及宽度的限制, 因此轨迹角度变化点的出现范围有限. 结合图 5 特征点提取算法的具体实验, 可以发现: 聚类半径选择在 100 米范围内具有较为良好的聚类效果. 与传统 DBScan 算法不同, 改进算法的主要应用场景为轨迹特征点聚类, 因此在参数的选择上可以根据轨迹数据的特征进行分析.

针对最小邻域点数这一参数的选择, 在不同规模数据集上实验结果如图 6 所示, 可以发现: 随着最小邻域点数增加, 生成的聚簇数量逐渐下降, 这一点在不同规模数据集上表现一致, 与预期结果相符.

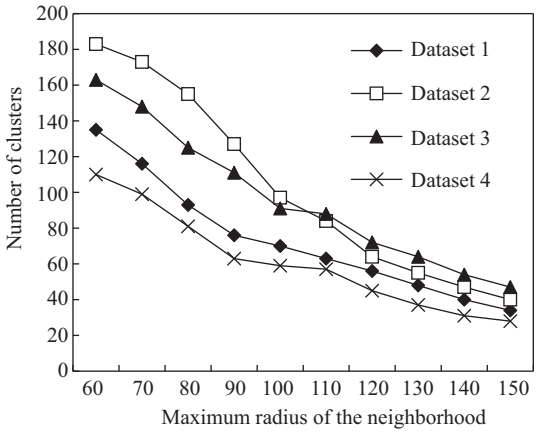

图 5 不同聚类半径下 DBScan* 算法聚类结果

Figure 5 Clustering results of DBScan* algorithm under different cluster radiuses

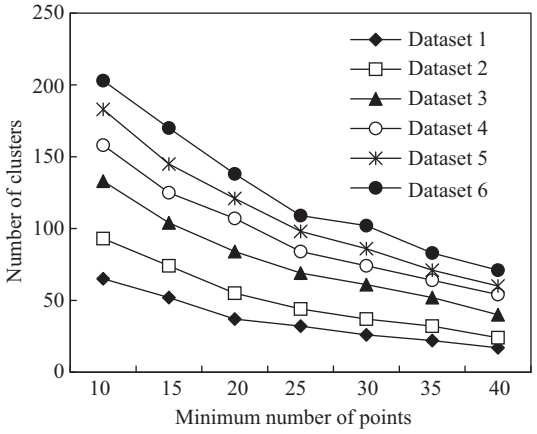

图 6 不同最小邻域点数下 DBScan* 算法聚类结果

Figure 6 Clustering results of DBScan* under different minimum number of points

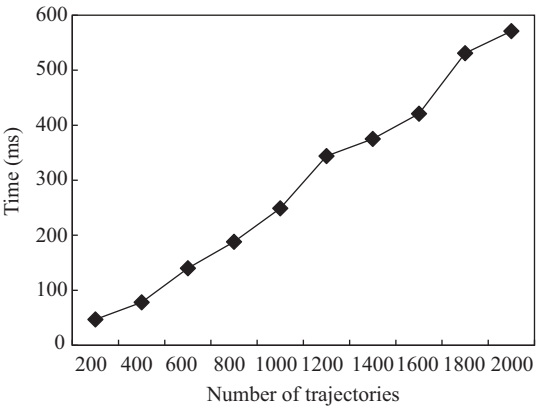

图 7 不同轨迹数量下特征点提取时间

Figure 7 Time cost of feature points extraction under different number of trajectories

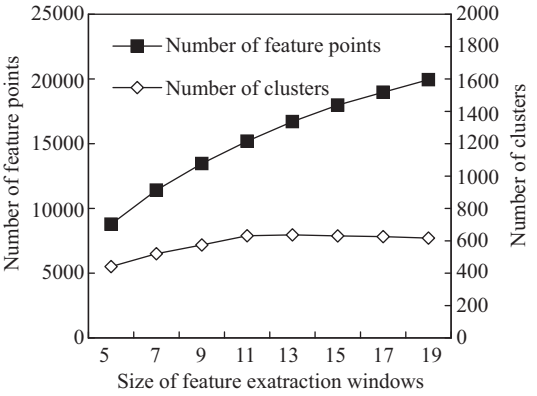

图 8 特征提取结果与提取窗口大小的关系

Figure 8 Relationship between feature extraction results and the size of extraction windows

7.3 特征提取算法时间性能分析

本小节实验考察基于 GeoHashTree 空间编码的特征点提取算法的时间效率, 图 7 展示了轨迹特征提取算法的时间开销与轨迹数量之间的关系. 实验结果表明, 随着轨迹数量不断增长, 特征提取算法保持线性增长, 证明了本文提出的轨迹特征提取算法具有较好的可伸缩性. 原因在于使用 GeoHashTree 对轨迹点进行索引可以快速查询到给定点的邻近节点, 使得聚类算法的效率得到极大的提升, 保证移动对象规模增大的时候, 利用聚类操作提取轨迹特征点的时间开销不会剧烈增加.

在特征提取算法中提取窗口值的选取对结果会产生影响, 本节实验考察特征提取窗口取值对结果的影响, 实验结果图 8 所示. 可以发现随着提取窗口不断增大, 特征点数量不断增长, 即提取算法识别出了更多的角度变化点, 但是经聚类提取后的聚类结果开始不断增加, 然后趋于稳定. 通过实验可以发现, 当提取窗口取值介于 9 ~ 13 时提取算法可以获得最佳提取结果.

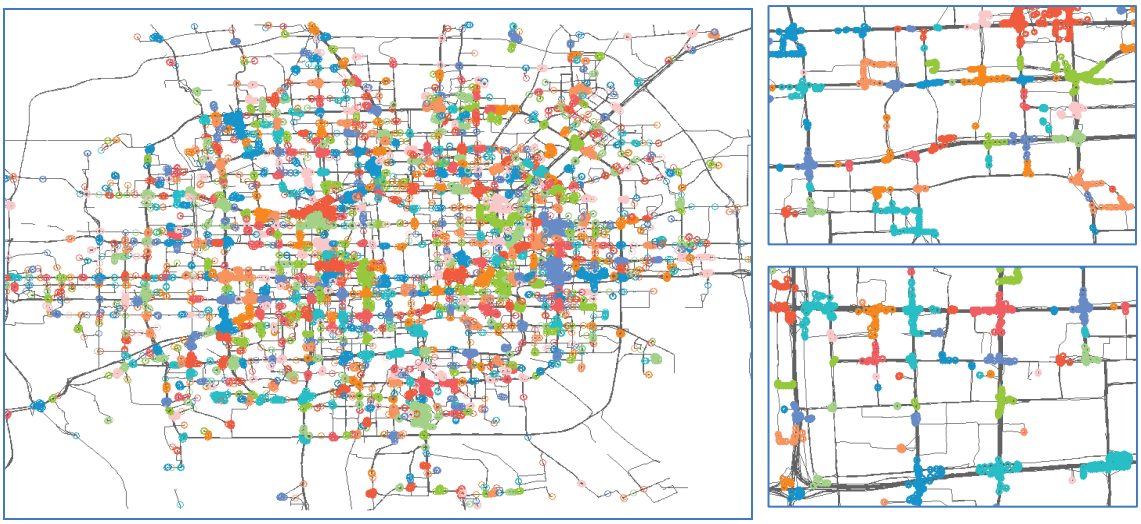

图 9 (网络版彩图) 聚类结果展示

Figure 9 (Color online) Visualization of clustering results

表 4 不同轨迹数目下特征点提取准确率

Table 4 Feature extraction accuracy under different number of trajectories

No. of trajectories	1000	2000	3000	4000	5000	6000	7000	8000
Accuracy (%)	81.3	83.4	84.1	84.8	80.9	76.3	75.8	76.1

7.4 轨迹特征提取有效性分析

图 9 展示了在大规模轨迹数据集上角度变化点提取算法的提取效果, 将计算出的包含角度变化点的特征点候选集合分成不同的簇, 均匀分布在路网中, 并使用不同深浅进行区分. 可以发现, 特征点大多集中在路网的路口附近, 这与预期相符, 也就是说算法可以准确提取出轨迹角度变化的特征点, 作为轨迹转换及轨迹预测的重要条件. 此外, 算法不依赖路网数据, 当实际路网发生改变时, 算法可以根据新增的轨迹数据进行实时修改, 准确反映路网信息, 避免了由于路网数据陈旧导致的偏差, 因此对于在此基础上进行的轨迹转换和轨迹预测具有良好的支持. 不依赖路网数据同时可以降低计算复杂度, 节约存储空间. 对于一条完整的轨迹, 可以使用算法提取的特征点序列进行表示, 从而方便轨迹存储和预测.

应用本文所提算法提取的特征点大多是路口的转弯点, 如果能够准确提取这些转弯点, 对移动对象位置精准预测具有重要的意义. 为了验证转弯处特征点提取的准确性, 本小节观察随着轨迹数据增加特征点提取算法正确识别出的轨迹点占所有轨迹点的百分比, 实验结果如表 4 所示.

通过表 4 可以发现: 随着轨迹数量增加, 开始正确识别的转弯点的准确性增大, 当轨迹数量达到一定规模 (大于 4000 条) 的时候, 特征点提取准确率呈现下降趋势并趋于稳定, 整体预测准确性大于 75%, 在智能交通领域被认为是较好的预测结果. 主要原因在于当轨迹数据增大到一定的规模, 轨迹特征模式趋于稳定, 应用本文所提算法能够识别出大多数的运动模式, 当轨迹数量持续增加, 由于 GPS 数据本身受到噪声数据的干扰, 准确率保持在一定的水平, 不会持续增加.

8 结束语

针对大规模移动对象轨迹数据, 本文提出了一种新的轨迹特征提取算法, 为基于机器学习的轨迹大数据挖掘提供支持. 本文提出一种轨迹点空间索引结构 GeoHashTree, 通过计算角度变化点, 结合轨迹聚类算法对提取点进行聚类. 将空间索引技术和轨迹聚类方法相结合, 提高了轨迹特征提取的效率. 大量真实轨迹数据集上的实验结果表明基于 GeoHashTree 空间编码结构的轨迹聚类算法在提升时间性能的同时保证了结果的准确性. 在大规模数据集上, 轨迹特征提取方法能够准确找到角度变化点. 此外, 特征提取结果可视化结果证明了本文所提方法的有效性. 未来工作主要包括: (1) 将轨迹特征点提取的结果应用于基于机器学习的轨迹预测算法中, 提高预测的准确性; (2) 将所提算法应用于基于位置的社交数据, 分析个体用户的出行模式, 对用户行为进行聚类分析, 提取群体的出行特征.

参考文献

1 Qiao S J, Han N, Zhu W, et al. TraPlan: an effective three-in-one trajectory prediction model in transportation networks. IEEE Trans Intel Transp Syst, 2015, 16: 1188–1198

2 Zheng Y. Trajectory data mining: an overview. ACM Trans Intel Syst Technol, 2015, 6: 1–41

3 Bao J, Zheng Y, Wilkie D, et al. Recommendations in location-based social networks: a survey. Geoinformatica, 2015, 19: 525–565

4 Zheng K, Zheng Y, Yuan J, et al. Online discovery of gathering patterns over trajectories. IEEE Trans Knowl Data Eng, 2014, 26: 1974–1988

5 Li X, Zhang J S, Ma J, et al. Feature extraction algorithm in consideration of the trend changing of track. J Comput Aid Des Comput Graph, 2016, 28: 1341–1349 [李翔, 张江水, 马健, 等. 顾及轨迹趋势变化的特征提取算法. 计算机辅助设计与图形学学报, 2016, 28: 1341–1349]

6 Yuan J, Zheng Y, Xie X, et al. Discovering urban functional zones using latent activity trajectories. IEEE Trans Knowl Data Eng, 2015, 27: 712–725

7 Qiao S, Shen D, Wang X, et al. A self-adaptive parameter selection trajectory prediction approach via hidden markov models. IEEE Trans Intel Transp Syst, 2015, 16: 284–296

8 Yuan J, Zheng Y, Xie X, et al. An interactive-voting based map matching algorithm. In: Proceedings of the 11th International Conference on Mobile Data Management, Kansas City, 2010. 43–52

9 Shan Z, Wu H, Sun W, et al. COBWEB: a robust map update system using GPS trajectories. In: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Osaka, 2015. 927–937

10 Guo L, Li H W, Zhang Z J, et al. Geometry matching method for transportation road network data based on projection. Geometry Inf Sci Wuhan Univ, 2013, 38: 1113–1117 [郭黎, 李宏伟, 张泽建, 等. 道路网信息投影匹配方法研究. 武汉大学学报: 信息科学版, 2013, 38: 1113–1117]

11 Zhao D B, Liu X M, Guo L. Real time map matching algorithm of floating car in support of spatial grid index. J Comput Aid Des Comput Graph, 2014, 26: 1550–1556 [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: 1550–1556]

12 Feng W, Han C. A novel approach for trajectory feature representation and anomalous trajectory detection. In: Proceedings of the 18th International Conference on Information Fusion, Washington, 2015. 1093–1099

13 Zabalza J, Ren J, Zheng J, et al. Novel two-dimensional singular spectrum analysis for effective feature extraction and data classification in hyperspectral imaging. IEEE Trans Geosci Remote Sens, 2015, 53: 4418–4433

14 Sauer F, Yu H, Ma K. Trajectory-based flow feature tracking in joint particle/volume datasets. IEEE Trans Visual Comput Graph, 2014, 20: 2565–2574

15 Chen C, Zhang D, Li N, et al. B-Planner: planning bidirectional night bus routes using large-scale taxi GPS traces. IEEE Trans Intel Transp Syst, 2014, 15: 1451–1465

16 Zhang D, Li N, Zhou Z H, et al. iBAT: detecting anomalous taxi trajectories from GPS traces. In: Proceedings of the 13th International Conference on Ubiquitous Computing, Beijing, 2011. 99–108

17 Chen C, Zhang D, Ma X, et al. Crowddeliver: planning city-wide package delivery paths leveraging the crowd of taxis.

IEEE Trans Intel Transp Syst, 2017, 18: 1478–1496
18 Jin A, Cheng C Q, Song S H, et al. Regional query of area data based on Geohash. Geogr Geo-Inform Sci, 2013, 29: 31–35 [金安, 程承旗, 宋树华, 等. 基于 Geohash 的面数据区域查询. 地理与地理信息科学, 2013, 29: 31–35]
19 Qiao S, Tang C, Jin H, et al. PutMode: prediction of uncertain trajectories in moving objects databases. Appl Intel, 2010, 33: 370–386
20 Yuan J, Zheng Y, Xie X, et al. T-Drive: enhancing driving directions with taxi drivers' intelligence. IEEE Trans Knowl Data Eng, 2013, 25: 220–232

A trajectory feature extraction approach based on spatial coding technique

Shaojie QIAO[1], Nan HAN[2], Tianrui LI[3], Xi XIONG[1],
Chang-an YUAN[4], Jiangtao HUANG[4*] & Xiaoteng WANG[3]

1. *School of Cybersecurity, Chengdu University of Information Technology, Chengdu* 610225*, China*;
2. *School of Management, Chengdu University of Information Technology, Chengdu* 610103*, China*;
3. *School of Information Science and Technology, Southwest Jiaotong University, Chengdu* 611756*, China*;
4. *School of Computer and Information Engineering, Guangxi Teachers Education University, Nanning* 541004*, China*
* Corresponding author. E-mail: hjt@gxtc.edu.cn

Abstract GPS data often have position deviations in precision and are apt to be affected by noise; hence, it is essential to extract features from trajectories before performing large-scale data mining. A GeoHash-based spatial coding technique called GeoHashTree was used to index spatiotemporal trajectory points in order to improve the efficiency of nearest-neighbor search. The GeoHashTree was applied in trajectory clustering and an improved density-based clustering algorithm was proposed to reduce the time complexity of nearest-neighbor search from $O(n^2)$ to $O(n \log n)$. After extracting trajectory points with changing angles, the proposed clustering approach was employed to achieve deep-level feature extraction on trajectory points with changing angles, which aims to accurately identify feature points. Extensive experiments are conducted on real GPS data and the results demonstrate that the proposed trajectory-clustering algorithm based on the GeoHashTree spatial index structure can improve time performance by an average of 90.89% as well as guarantee the accuracy of clustering compared with the traditional clustering method. The visualization results show that the trajectory feature extraction approach can effectively find trajectory points with changing angles and discover a varying types of feature points from large-scale data sets. In addition, the proposed approach does not depend on road network data and can dynamically update with new incoming trajectory data as road networks change in real time.

Keywords trajectory, big data, coding methods, clustering analysis, feature extraction

Shaojie QIAO was born in 1981. He received his B.S. and Ph.D. degrees from Sichuan University, Chengdu, in 2004 and 2009, respectively. He is currently a professor at the School of Cybersecurity, Chengdu University of Information Technology. His research interests include trajectory prediction and machine learning. He is a senior member of CCF and a member of IEEE.

Nan HAN was born in 1984. She received her Ph.D. degree from Chengdu University of Traditional Chinese Medicine, Chengdu, in 2012. She is currently a lecturer at the School of Management, Chengdu University of Information Technology. Her research interests include trajectory prediction and machine learning.

Tianrui LI received his Ph.D. degree from Southwest Jiaotong University, Chengdu, in 2002. He is currently a professor at the School of Information Science and Technology, Southwest Jiaotong University. His research interests include data mining, knowledge discovery, granular computing, and rough sets.

Xi XIONG received his Ph.D. degree from Sichuan University, Chengdu, in 2013. He currently is a lecturer at the School of Cybersecurity, Chengdu University of Information Technology. His research interests include social computing and machine learning.