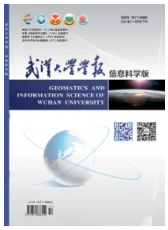

武汉大学学报(信息科学版)
Geomatics and Information Science of Wuhan University
ISSN 1671-8860,CN 42-1676/TN

《武汉大学学报(信息科学版)》网络首发论文

题目：时空轨迹多层级相似子段匹配方法
作者：郭宁，熊伟，欧阳雪，杨岸然，吴烨，陈荦，景宁
DOI：10.13203/j.whugis20200170
收稿日期：2020-04-17
网络首发日期：2021-04-13
引用格式：郭宁，熊伟，欧阳雪，杨岸然，吴烨，陈荦，景宁．时空轨迹多层级相似子段匹配方法．武汉大学学报(信息科学版).
https://doi.org/10.13203/j.whugis20200170

DOI:10.13203/j.whugis20200170

引用格式：

郭宁,熊伟,欧阳雪,等. 时空轨迹多层级相似子段匹配方法[J].武汉大学学报·信息科学版.DOI:10.13203/j.whugis20200170（GUO Ning，XIONG Wei，OUYANG Xue，et al. Multi-level Similarity Sub-segment Matching Method for Spatiotemporal Trajectory [J].Geomatics and Information Science of Wuhan University. DOI:10.13203/j.whugis20200170）

时空轨迹多层级相似子段匹配方法

郭宁[1,2] 熊伟[2] 欧阳雪[2] 杨岸然[2] 吴烨[2] 陈荦[2] 景宁[2]

1 军事科学院战争研究院，北京，100091

2 国防科技大学电子科学学院，湖南 长沙，410073

摘 要：轨迹子段匹配是轨迹数据挖掘的重要手段，针对其计算复杂度较高、受噪声影响大的问题，本文提出一种融合了自适应希尔伯特地理网格编码的多层级轨迹编码树结构，在可接受的建树代价下，形成了从轨迹整段到最小片段的层次化组织形式和子段从属关系表达结构，并在轨迹片段编码树的基础上设计了相似子段匹配算法，将复杂的空间计算转化为空间编码的字符串前缀匹配操作，极大地降低轨迹子段匹配的计算复杂度。实际轨迹数据上的实验表明，在不影响匹配准确率的前提下，本文提出的子段匹配方法的效率相比于基于经典距离的相似性度量方法获得了超过一个数量级的性能提升。

关键词：子段匹配；相似性；多层级；编码树；轨迹分段

中图分类号：P208　　**文献标识码**：A

随着各类位置传感器、定位芯片的广泛安装，具有时空属性的数据出现在日常生产生活和经济文化活动的各个领域中。时空轨迹，即记录移动对象随着时间推移而发生的位置变化情况的数据，作为最为典型的时空数据类型，数据量呈爆炸式增长态势，其中蕴含的信息反映出移动对象不同的运动规律和行为模式，具有可观的挖掘价值。

轨迹相似性分析是轨迹模式挖掘的重要基础，在实际生活中的应用丰富，扩展性强。例如，社区人员的相似行为匹配与挖掘、基于相似出行模式的用户推荐、目标异常移动监测、船舶出航习惯聚类、基于相似历史轨迹的交通调度等。利用人员的移动、出行轨迹模式计算也可在预估和防治传染病传播中发挥巨大作用，例如更加精准地筛查高危人群、划定传播范围，更加高效地控制病毒扩散和减缓蔓延趋势。

真实移动对象移动速度不一，移动方式多样，产生的轨迹形态以及在不同的采样策略下得到的离散有序轨迹点分布也复杂多变，传统的轨迹相似性计算一般直接采用时间翘曲或拉伸的方式匹配采样点，基于点距离计算轨迹段间的关系。而轨迹时空形态中蕴含的时空共现、时空聚集等模式往往只存在于轨迹的部分片段，衡量整体轨迹的相似性难以发现这些子段模式。本文从面向轨迹点的传统视角转为面向轨迹段，提出了一种基于编码树结构的轨迹相似子段匹配方法，具体贡献如下：

(1) 结合改进的希尔伯特地理编码，设计了一种多层级轨迹编码树结构，形成从轨迹整段到最小片段的层次化组织，实现了从轨迹粗粒度到细粒度的统一表达；

收稿日期：2020-04-17
项目资助：国家自然科学基金（41971362，41871284）

第一作者：郭宁，博士，研究方向为时空数据建模与分析、空间数据库与GIS。guoning10@nudt.edu.cn
通讯作者：熊伟，副教授。xiongwei@nudt.edu.cn

(2) 在多层级编码树的基础上设计了轨迹相似子段匹配方法，将复杂度高的空间计算转换为字符串前缀匹配操作，显著提高了子段匹配效率。

本文结构安排如下：第 1 节介绍了轨迹子段匹配与聚类方法的研究现状；第 2 节介绍了轨迹片段编码树的结构和各层级子段的编码方法；第 3 节提出了依托子段编码树的相似子段筛选算法；第 4 节使用真实轨迹数据和人工合成数据对提出的轨迹子段匹配算法的效率和准确性进行了实验验证；第 5 节对本文内容进行了总结。

1 研究现状

轨迹的子段匹配是轨迹数据挖掘领域的经典问题，也是轨迹聚类的主要手段。轨迹的分段会直接影响子段匹配的结果，传统做法一般是遍历两条轨迹中的所有可能的子段，再采用特定的距离或相似性度量筛选出最接近的子段对，效率低耗时长，不具有实用性。多年来领域内学者提出了众多不同的提高子段匹配效率和准确率的方法。

韩国Sunchon大学的Lim等人在2007年基于DTW[1]距离提出一种使用有限时间翘曲的子轨迹匹配算法，减小了子段过滤的候选集，实现了高效的子轨迹相似查询[2]。荷兰埃因霍温工业大学的Buchin等人在2011年采用Fréchet距离寻找轨迹相似子段，证明了寻找最长相似子段是NP-complete问题，并利用自由空间图[3]提出了近似计算方法，降低了计算复杂度[4]。为了在非等间隔采样的轨迹中寻找相似子轨迹，Buchin在同年发表的论文中又设计了对应时间平均欧氏距离的高效算法，将时间复杂度从平方降低到线性，并面向有时间漂移的轨迹提出了$1+\varepsilon$近似算法[5]。在其他相似性度量方法方面，香港科技大学的谢敏在2014年提出基于编辑距离的轨迹片段相似性度量EDS，并将其应用于轨迹子段相似性查询中[6]。北京交通大学的陈彦俊在2014年提出考虑子轨迹多维运动特征的距离度量，并以此为基础改进了只考虑运动方向变化的轨迹分段算法[7]。巴西圣卡塔琳娜联邦大学的Furtado等人在2016年通过加权归一化操作设计了一种计算多维度语义轨迹相似性的模型，并验证了该模型对轨迹不同维度出现的噪声和空隙具有更好的鲁棒性[8]。河海大学的毛莺池在2017年结合DTW算法提出了基于轨迹分段的动态时间翘曲距离度量SDTW，将点与点的距离替换为点和段之间的距离，具体方法为计算点与段两端点连接而包围的面积。此度量同时考虑了轨迹形状和时间距离，精度更高、对噪声鲁棒性更强[9]。

较有代表性的最新研究成果是韩国高等科技学院的Yoo在2019年提出的基于Hausdorff距离的时空子片段匹配算法MaTIS[10]，也是本文的主要对比方法。MaTIS算法在实现过程中先使用子段最小包围框(MBR)建立的R树索引以进行快速的剪枝，通过分割索引、相似计算和缝合重建三个步骤实现轨迹子片段的最长匹配，相比于谢敏提出的EDS度量[6]大幅提高了准确率，同时也具有较好的扩展性[10]，可应用于不同采样条件的轨迹数据中。

可以看出，现有轨迹子段相似性匹配方法大多是经典轨迹距离或相似性度量(例如DTW[1]、LCSS[11]、ERP[12]、EDR[13]、Hausdorff和Fréchet距离等)在轨迹分段基础上的扩展，这些方法虽然部分考虑了轨迹点间的前后顺序特性，但忽略了轨迹采样点之间的空隙，割裂了轨迹点之间的连续性，且未能兼顾轨迹多粒度特性，且在计算效率方面有很大的提升空间。为解决以上不足，本文从轨迹段的全新角度出发，以相邻采样点间的片段为基本单位，设计了轨迹多层级编码树，充分反映轨迹段的时序特性和轨迹的多粒度性质，实现了基于段的编码和匹配方法。

2 轨迹编码树

2.1 相关概念

现有关于移动对象与轨迹的索引结构基本以分离的轨迹点为依据对R树、KD树等经典索引的改进[14-16]，要在轨迹子段匹配中实现从面向点到面向段的思路转变，还需要对本文方法涉及的一些术语进行简单介绍，相关概念在图1中进行了标识。

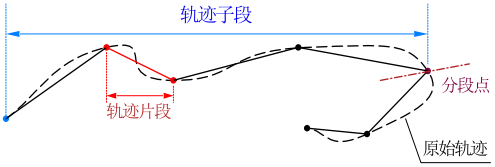

图 1 轨迹分段相关概念

Fig.1 Related Concepts

原始/真实轨迹：一定时间段内移动对象在欧几里得空间中连续运动形成的曲线，可以用连续时间函数精确描述其形态。

轨迹点序列：由传感器按固定或随机的频率记录移动对象的空间位置，作为真实轨迹数据的采样，通常用一串包含空间和时间信息的位置序列来表示。

轨迹片段：由相邻两个轨迹采样点构成的最小时空片段。

轨迹子段：将轨迹序列分割为多个分段，分段依据为轨迹形态特征或语义信息。每个分段包含多个连续轨迹采样点。

2.2 轨迹编码树构建方法

地理网格编码可以用一个完备的格网模型及其编码规则描述地球表面的任意一个位置，不同编码技术的区别在于格网划分方式和码元设置规范。其中Geohash编码具有简洁的规则、精度可控的位置描述能力和良好的体系兼容性，在地理信息网格组织、空间数据网格索引等方面具有广泛应用[17-18]。Geohash编码通过将全球经纬度范围不断二分，将地理空间不断缩小逼近目标点，同时用0、1作为两部分空间的代码，逐步将精确的经纬度二维坐标转换为一个一维的字符串[19]，并可以通过Base32等编码方式进行表示。为了提高Geohash编码的空间局部性，我们使用Hilbert空间填充曲线顺序作为网格遍历和编码的依据，其前两个两层级的网格划分示例如图2。

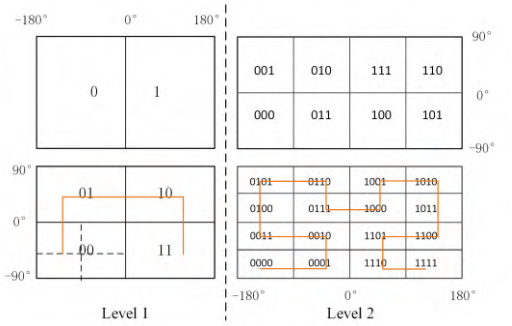

图 2 Hilbert Geohash编码

Fig.2 Hilbert Geohash Coding

使用分层级的自适应Geohash编码可以描述轨迹的多粒度特性，其中编码层级是由轨迹分段的尺度大小自适应计算得到，只要使得当前层级的网格大小刚好不小于该分段的外包框范围即可。而编码依据为不同粒度轨迹分段最小包围框的中心点坐标。具体编码流程见图3。

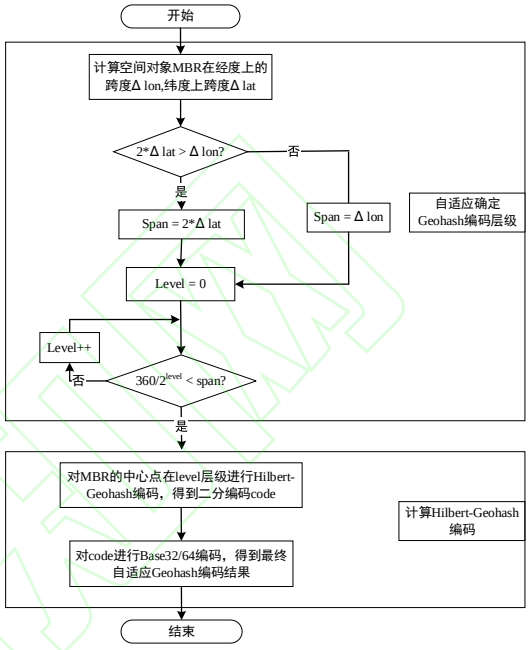

图 3 自适应地理网格编码流程

Fig.3 Flow chart of Adaptive Geo-grid Encoding

采用以上编码方法，即可形成定位精度与地理实体覆盖空间范围相匹配的自适应希尔伯特Geohash空间网格编码，下文简称为AHG(Adaptive Hilbert-Geohash)编码。不同层级轨迹分段的AHG编码十分适合构建上下层级具有空间从属关系的树结构，且树的父子、兄弟关系都具有明确的空间关系映射。所以可以通过多层级轨迹片段编码树结构来表达轨迹的多粒度特性。

构建多层级轨迹编码树的基本步骤为：以整条轨迹的自适应空间网格编码为根结点，自上而下地进行多层级分段和编码，构成编码树的中间结点，最底层结点由相邻两轨迹点构成时空片段的编码组成，各结点编码根据分段的从属关系设置父子结点链接，最终形成编码长度与分段数量同步扩展的多粒度片段编码树。一个典型三层级编码树结构如图4所示。

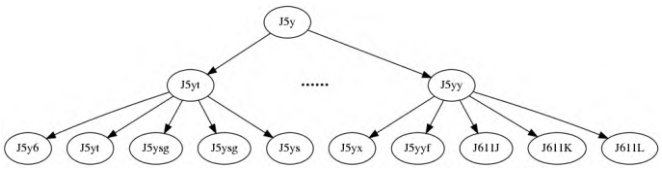

图 4 三层级轨迹编码树结构

Fig.4 A 3-layer Trajectory Code Tree

轨迹的分段可以根据实际数据特点和应用需求采用不同的方式，典型方法有等间隔点分段、等距离分段、拐点分段等。算法1为均匀等间隔点分段方式构建轨迹编码树的伪代码，其中自适应Hilbert-Geohash编码即为图3流程图描述的方法，用函数*AHG*表示，单条轨迹为*trajectory*，轨迹分段为*segment*，最小片段为*fragment*。

算法 1: 构建轨迹编码树

输入: 一条时空轨迹 (traj)，轨迹分段间隔 (step)
输出: 轨迹对应的编码树 (codetree)

```
linestring = geometry.Linestring(traj)
/* 对轨迹全段进行自适应Hilbert-Geohash编码 */
trajCode = AHG(linestring)
codetree = tree()
/* 以轨迹编码建立编码树根结点 */
rootNode = codetree.Node(name=trajCode)
/* 对轨迹进行均匀分段 */
For i from 0 to int(len(traj)/step) do
    segment = traj[i*step: (i+1)*step]
    /* 对分段编码 */
    segCode = AHG(geometry.Linestring(segment))
    /*建立分段结点插入树中，父结点为根结点*/
    segNode=codetree.Node(name=segCode,
parent=rootNode)
    /*对当前轨迹分段包含的最小片段进行编码*/
    For j from 0 to len(segment) - 2 do
        fragment=segment[j:j+1]
        fragCode=AHG(geometry.Linestring(fragm
ent))
    /* 建立最小片段叶子结点并插入树中，父结点
为所属分段 */
        fragNode = codetree.Node(name=fragCode,
parent=segNode)
Return codetree
```

2.3 轨迹编码树的特性

空间编码树的结构与原轨迹具体的对应关系如下：

根结点：整体轨迹的AHG编码；

枝结点(中间层级的结点)：轨迹分段/子轨迹的AHG编码；

叶子结点：轨迹片段的AHG编码；

子树：子段/子轨迹；

结点或子树的父子关系：轨迹片段或分段的从属关系；

结点或子树的左右兄弟关系：轨迹片段或分段的前后相邻关系。

根据实际应用场景和实际轨迹数据特点，也可以构建多于三层的编码树，即在轨迹根节点和最小时空片段叶子节点之间，建立两层以上的中间结点，涉及到更多粒度的轨迹分段，可以由粗到细更加精细地描述轨迹多层级的特性，适用于长度较长、形态变化较为复杂的轨迹。但更多层级轨迹编码树的存储结构更复杂，构建耗时也更高，故本文主要以三层轨迹编码树为例进行相似子段匹配的分析和探讨。

不同粒度网格的尺度也间接反映轨迹多层次的不确定性，例如，当轨迹分段粒度较细时，自适应网格尺度也较小，网格序列可以基本反映出轨迹的形态走势，对应的编码精度也较高，基于该编码树可以直接得到较为精确的分析结果，但细粒度下构建编码树的时间与空间代价也较大；当轨迹分段粒度较粗时，自适应网格尺度较大，网格的不确定性范围也比较大，对应的编码精度较低，但该粒度下构建编码树的时间与空间代价小，后续基于编码树的分析计算也较快，可以快速得到粗略的分析结果。所以需要根据轨迹的来源、采样特点、形态特点、长度分布等因素，对分段粒度大小进行权衡选择，得到性能和效率的平衡。

3 轨迹相似子段匹配

轨迹编码树的相邻结点可以反映轨迹片段时序和空间临近关系，可以把复杂的点统计或大量点距离的计算转化为简单的字符串编码前缀匹配。如图5所示，两个轨迹分段的空间编码相同，代表对应轨迹段从属的网格相同，属于相似子段。而不同轨迹编码树间任意一对结点编码的相同前缀长度代表着轨迹段的相似程度，这个性质给轨迹

子段相似性匹配问题带来新的思路。

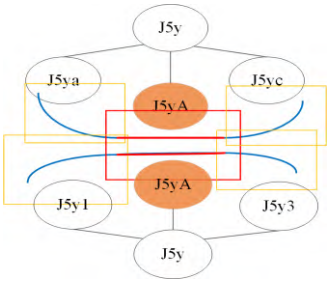

图 5 相似轨迹子段对应的编码关系

Fig.5 Code of Matched Sub-trajectories

3.1 问题定义

在实际轨迹的比较中，对整条轨迹进行相似性计算可以得到一个相似值，但真实移动对象的运动轨迹的往往长短不一，轨迹往往具有局部的相似性和整体的不相似性。现实的需求也常常不需要得到整体的相似度，而是需要寻找两条轨迹中相似的某一部分。

计算轨迹相似子段的步骤为：对两条轨迹分别进行分段，然后计算两个集合中两两分段的相似性，得到相似性最高的一对或几对分段，即为两条轨迹的相似分段。分段的对应关系不必按照分段在原轨迹中的次序。故将轨迹子段相似匹配问题定义如下：

输入数据：两条时空轨迹(*traj1*, *traj2*)，轨迹分段粒度(采用等间隔点分段时即为间隔点数*step*)；

目标输出：两轨迹包含的相似子段；

约束条件：显著性检验p值小于α，α通常取值为0.05。

3.2 相似子段计算与显著性检验

利用第2节提出的编码树结构，可以将相似字段的寻找转换为简单的字符串前缀匹配操作。具体步骤如下：

建树：对两条轨迹进行分段，分别建立轨迹编码树。

遍历子段组合：遍历不同的子段组合，通过对应的编码子树计算子段对的相似度。计算每一对子段的相似度分为初始判断和最小片段相似值计算两个步骤：

初始相似度判断：为减小候选子段数量，首先可以根据子段编码的粗粒度相似度进行预剪枝，将编码差别较大的子段排除出候选集。记当前两个子段的在编码树中的编码为*code1*与*code2*，求取其相同前缀为*commonPrefix*，则子段初始相似度为相同前缀长度与两子段编码平均长度的比值，即：

$$init\_similarity = \frac{2 \times len(commonPrefix)}{len(code1) + len(code2)}$$

在其大于设定的粗相似度阈值时再继续进行下一步的细粒度片段的相似度计算，否则可以直接排除当前子段对为相似子段。

最小片段的相似值计算：在编码树中取出两个子段的子结点，也就是子段包含的片段集合，按顺序组合两个集合中的元素，得到一组片段对及其对应的编码对，使用相同的公式计算其相似度，并进行加和与平均计算，即得到片段粒度的相似值，作为当前子段的最终相似度。

排序筛选：对符合条件的子段对的相似度进行排序，获取大于阈值的子段对，并进行去重、合并等操作，得到相似子段匹配备选结果。

显著性检验：对备选匹配子段进行显著性检验，得到统计显著的相似子段。

轨迹相似子段计算伪代码见算法2，其中*build_codetree*即为算法1描述的构建轨迹编码树的算法函数，*product*函数功能为形成分别取自两个集合中所有可能元素对的列表，*zip*函数功能为按顺序分别取两个集合的元素形成元素对的列表。

算法 2: 筛选轨迹相似子段

```
输入：两条轨迹 (traj1, traj2)，分段粒度 (step)，子段相似阈值 (threshold)
输出：匹配子段对 (matchPair)
codetree1 = build_codetree(traj1, step)
codetree1 = build_codetree(traj2, step)
matchPair = []
/* 遍历子段所有组合 */
For (segment1, segment2) in product (codetree1.getNode(depth=1), codetree2.getNode(depth=1)) do
    segCode1 = segment1.name
    segCode2 = segment2.name
    segCommonPrefix = commonPrefix(segCode1, segCode2)
    /* 子段粒度的相似度判断 */
    If len (2segCommonPrefix > threshold (len (segCode1) + len(segCode2)) then
        segSimValue = 0  // 初始化子段相似值
        /*片段粒度的相似值计算*/
        For (fragment1, fragment2) in zip (segment1.children, segment2.children) do
            fragCode1 = segment1.name
```

```
            fragCode2 = segment2.name
            fragCommonPrefix=commonPrefix(fragCode1, fragCode2)
            /* 片段相似值计算 */
            fragSimValue=2len(fragCommonPrefix) / (len(fragCode1)+len(fragCode2))
            segSimValue += fragSimValue
        /* 最小片段粒度的相似性判断 */
        If segSimValue > thresholdstep then
            matchPair.append((segment1, segment2))
Return matchPair
```

得到子段匹配候选集后，采用最常见的T-test检验方法对结果进行显著性检验。检验的零假设H_0为：轨迹子段对是不相关的。如果T-test检验得到的子轨迹的相关性的p值小于α，则拒绝原假设，α的取值可根据应用场景对误判和漏判的不同要求进行设定，一般可取0.05、0.01、0.005等值，α取值越小，对误判容忍程度越小，即显著性要求越高。最后统计子段匹配算法找出的结果中通过显著性检验的子段对数，计算与结果总数的比值，可以作为算法的准确率衡量指标。

3.3 复杂度分析

代表轨迹分段编码的字符串在内存中都是以字节(*Byte*)形式保存的，在算法具体实现过程中，可以将字符串前缀匹配操作转换为内存单元的位计算，能够大幅提高字符串匹配效率，复杂度仅为常数级。所以本文提出的子段匹配方法的复杂度主要在于构建轨迹编码树和遍历子段组合的过程。

在构建轨迹编码树的复杂度方面，对单个轨迹段进行自适应网格编码为常数复杂度。记单条轨迹长度，即包含的最小时空片段数为N，故当以$step$为轨迹第二层级分段粒度时，建立三层级编码树的复杂度为$O(1+N/step+N)$。

而在遍历子段组合与筛选相似子段过程中，可能存在的组合数为$O(N/step \cdot (N/step-1)/2)$，由于子段相似性计算是基于字符串前缀匹配的简单操作，故计算所有子段对相似度的复杂度为$O(N/step \cdot (N/step-1)/2) = O(N^2/step^2 - N/step)$，顺序遍历单个子段对所包含的最小时空片段对并计算其编码相似度的复杂度为$O(step)$，所以最小片段粒度的匹配复杂度为$O(N/step \cdot (N/step)/2 \cdot step) = O(N^2/step - N)$。另外，相邻子段判断、子段合并操作在轨迹编码树结构基础上仅为常数级复杂度。

将上述几个步骤复杂度进行加和，得到使用轨迹编码树寻找相似子段的总体复杂度为$O(N/step + N + N^2/step^2 - N/step + N^2/step - N) = O(N^2/step^2 + N^2/step)$。当分段粒度step为固定常数时，算法总体复杂度为$O(N^2)$。

4. 实验

本节设计数据实验对提出的轨迹编码树的性能进行验证，具体目的包括：

(1) 衡量分段粒度、轨迹长度等因素对建立轨迹编码树耗时的影响。

(2) 验证使用轨迹编码树寻找轨迹相似子段的有效性；

(3) 比较基于轨迹编码树的子段匹配方法与经典距离算法和最新文献方法[10]的效率、准确性；

4.1 实验数据与环境设置

实验同时使用真实轨迹数据和人工合成的轨迹数据，具体描述如下:

(1) 真实轨迹数据

Marine Cadastre:美国海事局自动识别系统(Automatic Identification System, AIS)记录的船舶轨迹数据，包含了美国邻近海域上超过150 000艘船舶的航行轨迹，为典型的无约束轨迹，轨迹点的采样频率为2~10秒。

T-Drive:旧金山市出租车的真实行驶轨迹，仅包含采样点的位置数据，使用的数据样本包含超过20 000条出租车轨迹，每条轨迹有100~500个位置采样点，为典型的有路网约束的轨迹，轨迹点的采样频率为10~30秒。

(2) 人工合成数据

从真实轨迹中提取子段加入固定偏移或者随机噪声生成的具有一定相似性的轨迹，其中固定偏移和随机噪声大小的选取与原轨迹段的长度成正比，以达到扰动程度相对一致的目的。

实验硬件环境为Intel® Core™ i7-4710MQ CPU @ 2.50 GHz，内存16 GB，操作系统为Ubuntu 18.04 LTS，实验软件由Python3.6程序编写。

为了评估所提出方法的各项性能，选取了一些经典距离度量和最新方法作为对比基准，具体如下：

(1) 基于经典轨迹距离的相似度量：Fréchet距离, LCSS[11]；

(2) 最新文献方法：MaTIS方法[10]。

4.2 轨迹编码树构建效率

本实验采用等间隔点分段方法测试编码树的构建效率。为探究轨迹长度与建树耗时的关系，首先固定轨迹分段参数为10，即每隔10个轨迹点进行分段，再分别使用AIS船舶数据和T-Drive出租车数据建立三层级轨迹编码树，记录建树和在硬盘持久化的耗时。随着选取的轨迹长度的增加，建树耗时情况如图6所示，其中vessel代表船舶轨迹的建树耗时，cab代表出租车轨迹的建树耗时。可以看出，两种轨迹类型的建树耗时均与轨迹长度呈线性函数关系，单条轨迹的建树时间在毫秒量级。

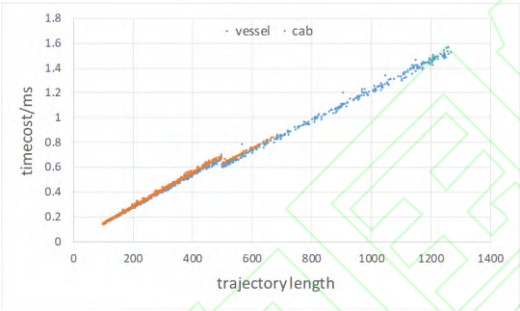

图 6 不同长度轨迹构建编码树耗时

Fig.6 Code Tree Building Time of Different Length

取1 000条轨迹数据，采用不同大小的分段粒度建立编码树，耗时结果如图7所示。可以明显看出，随着分段粒度的增加，两种轨迹数据集建树的耗时呈下降趋势。这是因为分段长度越大，单个轨迹对应的分段数越少，编码树结点总数和进行空间编码和构建整树的耗时也就越少。

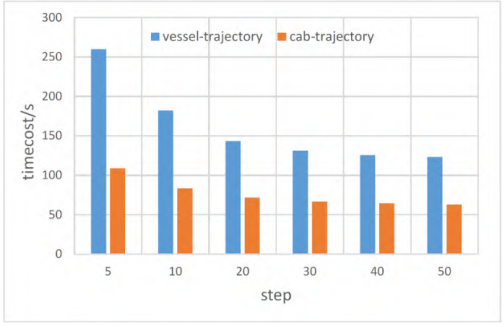

图 7 构建不同粒度的轨迹编码树耗时

Fig.7 Code Tree Building Time of Different Granulate

4.3 子段匹配效率与准确率

构造相同的子段匹配计算任务，使用统一的标准衡量本文提出的编码树匹配方法与其他经典方法的性能，对单个测试项和准确率计算步骤设计如下：

对一条轨迹(*Traj*)按照一定的粒度(作为实验变量)进行分段后，随机选取某一子段(*Origin-Sub*)作为测试数据，对其包含的所有轨迹点使用固定方向的固定长度偏移或者随机方向的随机噪声进行扰动，生成测试子段(*Test-Sub*)，加入的偏移和噪声大小为原始字段长度的5%，是原始轨迹子段对应的正确匹配结果；

使用不同的相似性度量方法寻找原轨迹与该测试子段相似性最高的子段作为匹配结果。如果匹配结果恰好为噪声扰动前的子段(*Origin-Sub*)，则该测试结果为命中。

从真实轨迹数据集中选取1 000条长度较长的轨迹，以便于使用不同粒度进行分段得到足够数量的子段，进行上述测试，结果命中数量与测试轨迹数量的比值即为该子段匹配方法的准确率。

在生成测试子段过程中，使用的固定偏移方向为东北(即直接对轨迹点经纬度使用加法)，偏移大小为该子段长度的2.5%；使用的随机噪声符合正态分布，其均值为0，标准差为该子段长度的5%。而候选集的大小控制了匹配过滤操作的精度，为达到本实验中采用的候选集大小为3，即保留原轨迹中与测试子段相似度排在前3的子段，作为相似子段候选集进行后续的判断。

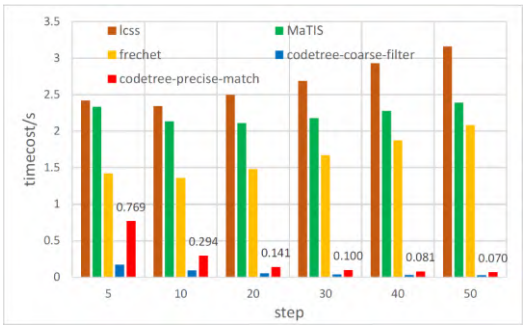

图 8 加入固定偏移的子段匹配耗时

Fig.8 Sub-trajectory Match Timecost with Fixed Offsets

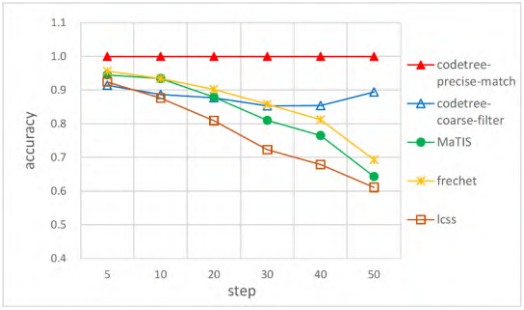

图 9 加入固定偏移的子段匹配准确率

Fig.9 Sub-trajectory Match Accuracy with Fixed Offsets

使用不同的分段粒度，匹配加入固定偏移的子段的耗时和准确率结果如图8、9所示。可以看出，在效率方面，随着分段粒度增大，LCSS、Fréchet距离方法的总耗时呈缓慢上升趋势，这是由于计算单对子段的经典距离相似性的复杂度为$O(step \cdot log(step))$(子段长度等于分段粒度)，需要进行$N/step$对子段的相似性计算(N为未分段原始轨迹长度)，所以总复杂度为$O(N \cdot log(step))$。而MaTIS方法单个相似性测试的复杂度为$O(|Q|N)$[10]，其中Q为测试子段的长度，即为本实验中的分段长度$step$，所以总复杂度为$O(N)$，所以总耗时呈水平不变趋势，实验耗时结果符合预计。而编码树方法的匹配耗时相较于其他方法都有了超过一个数量级的提升，且分段粒度越大，效率优势越大，这是因为基于编码树进行轨迹相似性计算只需要对其对应层级的结点编码进行字符串的前缀匹配操作，且转化为位计算后效率极高，分段粒度变大，子段对应的AHG编码长度缩短，需要判断的子段对数量也随之降低，所以总耗时以接近线性的速度下降。

在准确率方面，蓝色空心三角图例标识的结果是使用轨迹编码树进行子段匹配的准确率，可以看出在分段粒度较小时，即每隔5或10个轨迹点进行分段时，编码树的准确率落后于基于Fréchet距离的相似性匹配方法和MaTIS方法，甚至稍低于准确率下降最快的最小公共子序列法，这是因为三种对比方法的计算过程都使用了轨迹点间的精确欧氏距离作为度量基础，在分段较小时可以更好地衡量子段的相似性。但在分段粒度较大时，编码树方法的准确率与经典方法持平甚至高于经典距离方法，例如分段粒度为50时就大幅领先另外三种对比方法。这是因为轨迹编码树的中间层结点是轨迹分段的地理网格编码，在空间形态上是用一个矩形网格代表该子段，子段粒度越大，这种表示方法忽略的细节就越多，以至于减弱了对子段加入偏移的差异性地表达，在计算相似性时就会出现相似性较高的情况，所以编码树匹配方法在分段粒度较大时更容易根据测试子段筛选出原始子段，可以达到较高的准确率。

介于分段粒度较小时编码树方法的准确率并无明显优势，为了进一步说明编码树方法的性能，在实验中对编码树的子段匹配结果继续进行精确匹配计算，具体做法为在子段粗过滤结果集中，使用轨迹编码树最底层的叶子结点，即轨迹最小片段的MBR端点的精确距离计算出片段距离，并进行加和计算得到子段的综合距离，筛选出距离最小的子段作为匹配结果。这种精确计算的方法可以使子段匹配结果达到100%的准确率，而粗过滤和精匹配的总耗时，如图8中红色柱体标识结果，仍然明显低于经典距离方法，且分段粒度越大，效率优势越大。

相比于固定方向上的固定偏移，随机方向的噪声对轨迹相似性的影响更为显著，且更符合实际场景。在对轨迹子段加入随机小噪声后，使用不同的分段粒度进行子段匹配的耗时与准确率结果如图10和图11所示。

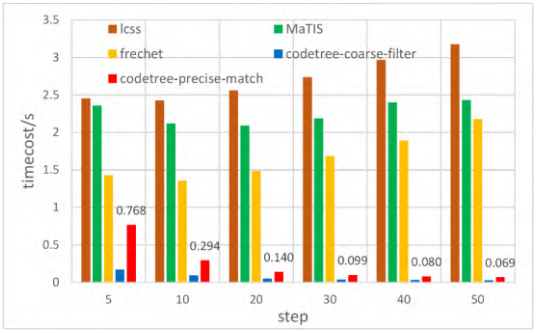

图 10 加入随机噪声的子段匹配耗时

Fig.10 Sub-trajectory Match Timecost with Random Noise

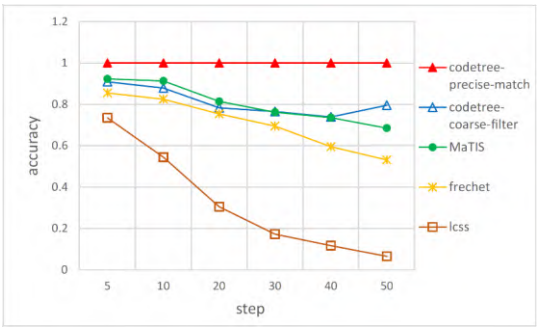

图 11 加入随机噪声的子段匹配准确率

Fig.11 Sub-trajectory Match Accuracy with Random Noise

同样可以看出，在计算效率方面，编码树方法的子段匹配耗时相较于其他方法有超过一个数量级的提升。但在不进行精确匹配操作时，分段粒度较小的编码树的准确率仍然逊色于MaTIS方法，仅在分段粒度较大(间隔50点)时超过MaTIS。所以同样对子段匹配粗过滤结果使用最底层的最小片段结点进行精确匹配计算，得到精确结果，即匹配准确率达到100%。如图10中红色柱体所示，相比于仅进行粗过滤，精确匹配的耗时有所增加，但与其他方法的粗过滤耗时仍具有显著优势，且分段粒度越小，优势越明显。

5 结论

本文结合轨迹的多粒度特性，从面向轨迹点转为面向轨迹段的新型视角，基于改进的自适应希尔伯特网格编码方法提出一种多层级轨迹编码树结构，在可接受的建树代价下，实现了从轨迹粗粒度到细粒度的层次化组织。并在此编码树的基础上设计了轨迹相似子段匹配算法，将复杂的空间计算转化为空间编码间的字符串匹配操作，极大地降低了轨迹相似子段匹配的计算复杂度。在不影响匹配准确率的前提下，效率性能取得显著提升。本文提出的方法为挖掘分析多层次精细化轨迹模式、扩展多样化轨迹相似性应用场景提供了效率保证。由于本文提出的子段匹配方法使用了空间填充曲线编码，具有不可避免的编码突变性，需要进行二次筛查才能得到完备的结果，如何结合R树、KD树等传统完备的空间索引进一步提高准确性是下一步研究方向。

多层级轨迹片段编码树结构具有很好的数据挖掘和应用开发价值，可以给许多经典轨迹数据挖掘问题提供新的解决思路，如轨迹模式的聚类与分类问题(根据粗相似路线划分人群迁移模式或野生动物迁徙群体、社区人群出行模式匹配与服务推荐、舰船航行的热点海域挖掘与筛选等)、地图匹配问题(将汽车或行人不同精度的轨迹路线匹配至城市道路网)等。在更多的新型轨迹相似模式分析场景中，如轨迹整体相似性、局部相似度以及不同层级间相似度的关系等，轨迹编码树结构的灵活扩展和应用都值得进一步深入研究。

参 考 文 献（References）

[1] Keogh E, Ratanamahatana C A. Exact indexing of dynamic time warping [J]. *Knowledge and information systems*. 2005, 7 (3): 358–386

[2] Lim E-C, Shim C-B. Similarity search algorithm for efficient sub-trajectory matching in moving databases [C]. In International Conference on Computational Science. 2007: 821–828.

[3] Godau M. A Natural Metric for Curves —Computing the Distance for Polygonal Chains and Approximation Algorithms [C]. In Symposium on Theoretical Aspects of Computer Science. 1991: 127–136.

[4] Buchin K, Buchin M, Gudmundsson J, et al. Detecting commuting patterns by clustering subtrajectories [J]. *International Journal of Computational Geometry & Applications*. 2011, 21 (03): 253–282.

[5] Buchin K, Buchin M, Van Kreveld M, et al. Finding long and similar parts of trajectories [J]. *Computational Geometry*. 2011, 44 (9): 465–476.

[6] Xie M. EDS: a segment-based distance measure for sub-trajectory similarity search [C]. In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. 2014: 1609–1610.

[7] Chen Y, Shen H, Tian H. Clustering subtrajectories of moving objects based on a distance metric with multi-dimensional weights [C]. In 2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming. 2014: 203–208.

[8] Furtado A S, Kopanaki D, Alvares L O, et al. Multidimensional similarity measuring for semantic trajectories [J]. *Transactions in GIS*. 2016, 20 (2): 280–298.

[9] Mao Y, Zhong H, Xiao X, et al. A segment-based trajectory similarity measure in the urban transportation systems [J]. *Sensors*. 2017, 17 (3): 524.

[10] Yoo J-J, Loh W-K, Whang K-Y. Indexable sub-trajectory matching using multi-segment approximation: a partition-and-stitch framework [J]. *The Journal of Supercomputing*. 2019: 1–29.

[11] Vlachos M, Kollios G, Gunopulos D. Discovering Similar Multidimensional Trajectories [J]. *Proc. of the 18th Int. Conf. on Data Engineering*, 2002.

[12] Chen L, Ng R. On the marriage of lp-norms and edit distance [C]. In Proceedings of the Thirtieth International Conference on Very Large Data Bases, Volume 30. 2004: 792–803.

[13] Lei C, Özsu M T, Oria V. Robust and fast similarity search for moving object trajectories [C]. In ACM SIGMOD International Conference on Management of Data. 2005.

[14] Ye Xiaoping, Guo huan, Tang Yong, et al. Index of Mobile Data Based on Phase Points Analysis [J]. Chinese Journal of Computers. 2011, 34(2): 256-274. (叶小平, 郭欢, 汤庸, et al. 基于相点分析的移动数据索引技术[J]. 计算机学报, 2011, 34(2): 256-274.)

[15] Chen Zhaoying, PM-tree: An Index of Mobile Objects Based on Spatiotemporal Phase Point [D]. South China Normal University. 2015. (陈钊滢. 时空相点移动对象数据索: PM-tree[D]. 广东:华南师范大学, 2015.)

[16] Wang Fei, Pang Yue, Zhou Xiangdong, et al. A Method of Track Index for Similarity Search [J]. Computer Applications and Software. 2017. 034(11): 1-5, 63. (王飞, 庞悦, 周向东等. 一种面向相似查询的轨迹索引方法[J]. 计算机应用与软件. 2017. 034(11): 1-5,63.)

[17] Hong Van L, Takasu A. An efficient distributed index for geospatial databases [C]. In Database and Expert Systems Applications. 2015: 28–42.

[18] Lu N, Cheng C, Jin A, et al. An index and retrieval method of spatial data based on GeoSOT global discrete grid system [C]. In 2013 IEEE International Geoscience and Remote Sensing Symposium-IGARSS. July 2013: 4519–4522.

[19] Morton G M. A computer oriented geodetic data base and a new technique in file sequencing [J]. 1966.

Multi-level Similarity Sub-segment Matching Method for Spatiotemporal Trajectory

GUO Ning[1,2] *XIONG Wei*[2] *OUYANG Xue*[2] *YANG Anran*[2] *WU Ye*[2] *CHEN Luo*[2] *JING Ning*[2]

1 Institute of War, Academy of Military Sciences, Beijing 100091, China

2 College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, China

Abstract: Addressing the problem of high complexity of trajectory sub-segment similarity matching and result sensitivity of trajectory noise, this paper proposes a multi-level trajectory code tree structure that integrates adaptive Hilbert geographic grid coding. A hierarchical organizational form and sub-segment subordinate relationship expression structure are formed from the entire segment of the trajectory to the smallest segment, and a sub-segment similarity matching algorithm is designed on the basis of the trajectory segment code tree to transform complex spatial calculation into string matching operation, which greatly reduces the computational complexity of similar matching of sub-segments. Experiments on actual trajectory data show that the efficiency of the proposed method achieved more than an order of magnitude improvement over the classical distance-based similarity measurement method without affecting accuracy.

Keywords: sub-segment match; similarity; multi-level; code tree; trajectory segmentation

First Author: GUO Ning, PhD, specializes in spatiotemporal data modeling and analysis, spatial database and GIS. E-mail: guoning10@nudt.edu.cn

Corresponding author: XIONG Wei, associate professor. E-mail: xiongwei@nudt.edu.cn

Foundation Support: The National Natural Science Foundation of China (41971362, 41871284)