分类号_____ 密级 _____

UDC _____

学 位 论 文

移动对象的时空轨迹相似性算法研究

作 者 姓 名 ： 丁光伟

指 导 教 师 ： 杨晓春 教授

东北大学计算机科学与工程学院

申请学位级别： 硕士 学 科 类 别 ： 工学

学科专业名称： 计算机系统结构

论文提交日期： 2018 年 12 月 论 文 答 辩 日 期 ： 2018 年 12 月

学位授予日期： 答辩委员会主席：

评 阅 人：

东 北 大 学

2018 年 12 月

A Thesis in Computer Software and Theory

Research on Spatio-Temporal Trajectory Similarity Algorithm of Moving Objects

By Ding Guangwei

Supervisor: Professor Yang Xiaochun

Northeastern University

December 2017

独创性声明

本人声明，所呈交的学位论文是在导师的指导下完成的。论文中取得的研究成果除加以标注和致谢的地方外，不包含其他人已经发表或撰写过的研究成果，也不包括本人为获得其他学位而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：

日　　期：

学位论文版权使用授权书

本学位论文作者和指导教师完全了解东北大学有关保留、使用学位论文的规定：即学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人同意东北大学可以将学位论文的全部或部分内容编入有关数据库进行检索、交流。

作者和导师同意网上交流的时间为作者获得学位后：

半年 □　　一年□　　一年半□　　两年□

学位论文作者签名：　　导师签名：

签字日期：　　签字日期：

摘　要

近年来，随着移动设备和 GPS 的普及，产生了大量的时空轨迹数据，对路网中车辆和行人轨迹的追踪越来越普遍，因此基于位置的服务得到了迅速发展。为了对移动对象的轨迹数据进行更好的分析，产生了移动对象轨迹相似性查询技术，可以应用于行为模式分析、轨迹预测和路径推荐等方面，为人们的生活提供便利。

目前，在轨迹相似性查询方面已经产生了大量研究。有很多研究致力于寻找一个通用的相似性计算函数，或者研究某一个特定场景下最合适的相似性计算函数，相似性函数的结果代表两条轨迹之间的相似程度。而相似性查询技术可以通过相似性计算函数从数据库中的海量轨迹数据中得到与查询轨迹最相似的轨迹。

本文分析了轨迹数据的时空特征，提出了可以将时间维度转化为空间维度的时空转化因子，得到了三维时空，解决了 PTM 等算法中没有将时间维度很好融入相似性计算中的问题。随后，本文结合 DTW 算法和 BDS 算法在对应点匹配中的思想，提出 DTW-BDS 算法，解决了原始算法中对样本点依赖严重和匹配结果时序错乱的问题，达到了获取一个较优的样本点匹配方案的目的。在得到样本点匹配方案之后，生成了对应轨迹段。为了计算对应轨迹段的距离，本文提出断点的概念，解决了 DTW 算法计算轨迹间距离时对采样策略过于敏感的缺点，还提出了对应轨迹段之间的形状影响因子，用于描述形状对相似性的影响。根据对应轨迹段距离提出了局部轨迹相似性查询算法。

最后，本文在两个真实的车辆和行人的轨迹数据集上进行了大量实验，研究了参数对算法的影响，并与最新研究进行对比，验证了本文提出算法在查询相似轨迹上的准确性。

关键词：基于位置的服务；移动对象；时空数据；轨迹相似性

Abstract

In recent years, with the popularization of mobile devices and GPS, a large amount of trajectory data has been generated, and the tracking of vehicles and pedestrian trajectories in road networks has become more and more popular, so location-based service has been rapidly developed. In order to analyze the trajectory data of moving objects better, the trajectory similarity query technology of moving objects is generated, which can be applied to behavior pattern analysis, trajectory prediction and path recommendation, etc., to facilitate people's lives.

At present, the academic community has produced a lot of research results on trajectory similarity query. There are many studies devoted to finding a general similarity calculation function, or studying the most suitable similarity calculation function in a particular scene. The result of the similarity function represents the degree of similarity between the two trajectories. The similarity query technique can obtain the trajectory most similar to the query trajectory from the massive trajectory data in the database through the similarity calculation function.

In this paper, the spatio-temporal characteristics of trajectory data are analyzed. The spatio-temporal transformation factor which can transform the time dimension into spatial dimension is proposed. The three-dimensional spatio-temporal is obtained, which solves the problem that the time dimension is not well integrated into the similarity calculation in PTM and other algorithms. Subsequently, this paper combines the idea of DTW algorithm and BDS algorithm in corresponding point matching, and proposes DTW-BDS algorithm, which solves the problem that the original algorithm relies heavily on sample points and the timing of the matching result is disordered, and achieves a better sample point. Match the purpose of the program. After the sample point matching scheme is obtained, the corresponding track segment is generated. In order to calculate the distance of the corresponding trajectory segment, this paper proposes the concept of breakpoint, which solves the shortcomings of the DTW algorithm which is too sensitive to the sampling strategy when calculating the distance between the trajectories. It also proposes the shape influence factor between the corresponding trajectory segments, which is used to describe the shape pair. The impact of similarity. A local trajectory similarity query algorithm is proposed based on the corresponding trajectory segment distance.

Finally, this thesis carried out a lot of experiments on two real vehicle and pedestrian trajectory datasets, studied the influence of parameters on the algorithm,

and compared with the latest research, verified the accuracy of the proposed algorithm in querying similar trajectories.

Keywords: location-based service; moving objects; spatio-temporal data; trajectory similarity

目录

目录

第一章 绪论

1.1 研究背景及意义

近年来，随着移动设备和 GPS 的不断发展，人们已经可以很轻松地获取移动物体的地理位置信息，为了可以更好地利用这些信息，将需要使用一些技术手段去对这些信息进行处理，而更好地处理前提是需要更多的数据去支持算法的运行，从而又带动了地理位置的采集，形成一个良性循环。一个按时间先后顺序排列的地理位置信息便组成了一条轨迹数据，轨迹数据已变成了位置大数据时代的最重要的数据来源之一[1][2][3]。

数据采集中获取到的数据有很多种类。比如用户手持移动电话，通信公司就会根据手机信号和信号发射基站的位置去确定用户的具体位置，根据一个制定好的采样策略，收集用户所在位置的经纬度、当前时间，并根据多次采集数据去计算得到用户的平均移动速度和移动方向等。根据用户的移动数据，可以获取到大量用户的个性化信息，比如该用户经常去某家餐厅就餐，那么一些手机客户端可以根据用户喜好，为用户推荐类似口味的餐厅。根据用户一周内频繁出现的场所，为用户推荐周边的美食、娱乐场所，或者推荐相同兴趣爱好的好友[4]。还可以根据用户的移动速度的变化，判断用户在某段路程里打了出租车，可以为用户推荐上车周边更好打车的地点。让用户可以不刻意得去记录自己的日常行为，仅仅被记录下行为轨迹，便可以获得个性化的推荐。

除了用户轨迹，现在很多出租车和私家车上都安装了车载 GPS，通过车载 GPS 的工作可以将汽车每日的移动轨迹上传到服务器，然后通过对一个城市大量出租车、私家车的轨迹分析，可以得到很多信息。比如通过分析一天的轨迹信息中道路上车辆行驶速度，可以得到该城市每日早高峰晚高峰大约会出现在什么时间段，建议不赶时间的司机错峰行驶。还可以通过实时轨迹数据得到当前时间道路的车流量，判断该条道路在该时刻的拥堵状况及预测未来一小时内的路况[6][7][8]，道路拥堵信息可以在广播频道里司机进行实时指导路线，或者在手机的出行 app 里动态展示，为司机挑选相对通畅的道路。

除了对用户位置的信息采集，对出租车移动路线信息的采集之外，还有对野生动物行为轨迹的采集来研究其生活习惯以及迁徙路线[9][10]，军事领域对地方目标轨迹的实时监测以实现精准打击，对飓风移动路径数据的采集来预测气候[5]和预防自然灾害等等。随着数据采集设备的改良和采集方式的优化，各个领域都产生了海量的轨迹数据。所以对轨迹数据的分析利用变得十分重要。人们为了发掘海量数据中隐藏的价值，得到丰富的数据特征空间以及用户轨迹的规律性信息，开发了聚类分析[10]、隐私保护[11]和行为预测[12]等一系列的应用技

术，而这些技术的实现都得益于移动对象轨迹的相似性查询技术的发展。

轨迹数据展示了移动对象的时空动态，以数据的形式存储了空气、动物、车辆和人类的运动信息，在预测风暴移动、研究动物迁徙、规划城市建设和提供出行路线等方面有着重要的应用。而这些应用都需要轨迹数据库提供一个高效的轨迹相似性查询功能，在移动对象的轨迹相似性查询中，相似性计算函数是核心。本文将在后面给出轨迹相似性查询以及相似性计算函数的定义。

当前该领域主要围绕两个方面的问题进行研究，一是研究合适的相似性计算函数，二是研究高效的检索机制。选择一个合适的相似性计算函数和利用函数制定高效的检索机制至关重要，这些因素同时决定了查询方法的好坏。比如有时候我们无需对采样得到的整段轨迹计算与其他轨迹的相似度，只需要对一小段子轨迹选取合适的函数进行相似性计算即可，这样就可以在一定程度上减少运算时间，并获得相对而言更重要的轨迹相似性信息，因为相似的那段比不相似的那段更有价值。因此在面对不同场景时我们需要根据具体情况采用合适的相似性计算方法。

大多数对轨迹相似性的查询研究和常用的一些轨迹相似性计算函数，比如动态时间规整算法，最长公共子序列算法和编辑距离算法一般针对的是完整轨迹，最后计算结果得到的是两条完整轨迹的距离或者表达轨迹相似程度的数值。比如给定一条查询轨迹 Q，一条数据库中的数据轨迹 R，通过相似性计算函数得到的是轨迹 Q 和轨迹 R 从起始位置到终止位置所有样本点按照时间顺序连成轨迹段的相似性。但是实际情况下，轨迹数据库中的单条轨迹在很大部分轨迹段上与查询轨迹 Q 并不相似，但是有一小部分，比如有三分之一的部分和 Q 在时间和空间上都很接近的，那么这三分之一的轨迹的重要程度远大于另外三分之二的轨迹，但是之前整段轨迹比较相似性的一个缺点就是其余三分之二的不相似的轨迹段容易掩盖掉这三分之一的特征，因此我们需要额外使用一个方法，将这最相似的三分之一的轨迹段找出来。

国内外很多专家学者对轨迹相似性进行了深入的研究，使用了不同的空间网络、不同的轨迹格式表示以及不同的维度企图去找到一种更好地方法去表示出轨迹之间的相似程度。但是由于研究的问题会涉及到具体的场景，由于大家研究的问题不尽相同，所以研究出了很多的相似性表示方法。首先空间网络有欧式空间下和路网下的轨迹相似性，这两种研究场景最大的区别在于欧式空间不考虑道路对移动的限制，可以使用一条直线距离去衡量任意两个地点的距离，而由于路网空间对人员和车辆的限制，两个地点之间的距离必须使用真实的道路距离去衡量。还有轨迹格式会根据研究的问题不同而选取不同的格式，比如在欧式空间下采用网格表示轨迹，还有不考虑时间因素的情况下，轨迹点信息中仅包含经纬度的信息，如果考虑时间、空间、速度和运动方向的话，轨

迹点的格式就会同时包含经纬度、时间戳、瞬时运动速度以及运动的方向。而本文考虑到欧式空间对于研究的便捷性以及计算的高效性，所以不考虑道路交通情况，而是研究欧式空间下包含时间、空间、运动方向等信息的轨迹相似性。

1.2 本文研究内容

尽管在轨迹相似性方向上已经有很多研究成果，但是上文中提到的两个问题，几乎没有一个很好的解决方案。第一个问题是轨迹的时间距离在与空间距离结合的时候，普遍使用的参数结合的方法不能赋予参数一个明确的含义。第二个问题是以往的相似性函数忽略了局部相似的情况。为解决以上两个问题，本文采用了三维时空去结合时间和空间维度，并提出基于三维时空的相似性计算方法。具体包括以下几个方面：

（1）三维时空的定义。本文基于统计学知识和欧式空间构造出了三维时空，并阐述了三维时空的特点，以及如何解决之前的研究工作中存在的问题，可以更好地研究轨迹的相似程度。

（2）在三维时空的基础上，给出各种距离的定义，包括点与点之间的时空距离，轨迹段与轨迹段之间的时空距离，以及轨迹段在形状上的距离。三维时空中的距离包含了空间维度和时间维度信息，更好地表示空间中的轨迹段的相似程度。

（3）基于三维时空的局部轨迹相似性算法的研究。研究的核心问题是如何利用三维时空的特性，在很长的数据轨迹中找出与查询轨迹最相似的部分。设计了局部轨迹相似性查询算法，找出最相似部分轨迹并表示数据轨迹与查询轨迹的相似程度。

（4）算法实现及实验设计。实现局部轨迹相似性查询算法，并设计实验，使用真实轨迹数据集去验证算法的高效性和准确性。

1.3 本文组织结构

本文的组织结构如下：

第 1 章为绪论，介绍了轨迹相似性查询技术及其相关背景知识。

第 2 章为相关理论与关键技术，介绍了一些常用的轨迹表示方法，需要根据不同场景选择不同的表示方法。然后介绍了现有的轨迹相似性查询以及相似性计算函数，并分析其优缺点以及适用背景。最后介绍了一些空间索引结构，加速轨迹查询速度。

第 3 章主要介绍了本文为了结合时间和空间维度，采用三维时空来计算轨迹在时间和空间上的距离，并根据轨迹的对应点得到对应轨迹段，然后对已有相似性查询方法中的对应点匹配方法做出优化。

第 4 章主要介绍了根据找出对应轨迹段之后，提出了两个衡量轨迹段之间距离的指标，在获得轨迹段距离的基础上提出了查找与查询轨迹最相似的子轨迹的算法。

第 5 章为实验部分。通过算法参数调整以及与前任算法的对比实验，验证本文提出的算法的有效性。

第 6 章为总结与展望。首先对本文工作作出总结，然后提出未来需要研究的方向。

第二章 相关理论与关键技术

本章主要介绍一些轨迹相似性相关理论与关键技术，包括轨迹的表示方法、现有的一些轨迹相似性查询算法以及一些加速轨迹相似性查询的索引技术。

2.1 轨迹表示方法

移动对象在移动过程中，我们对其按既定的规则进行采样，会获取一系列采样点，这些采样点再按照时间先后进行排序，就会大致还原移动对象的移动过程。实际情况中，轨迹是移动对象的连续移动过程中采样得到的一系列离散的位置序列。并且在不同的场景下，为了达到不同的目的，我们需要采用不用的轨迹表示形式。下面将介绍一些常用的轨迹表示方法。

2.1.1 仅基于位置信息的轨迹表示方法

一般情况下，轨迹可以表示为一系列包含信息的点组成的有序集合，即 $T = <p_1, p_2, \ldots, p_n>$。在欧式空间下，点 p_i 是一个坐标的形式，$p_i = <lon_i, lat_i>$，lon_i 表示经度，lat_i 表示纬度。在路网空间下，我们可以将路网模型化为图的数据结构的形式，即 $G = (V, E)$，此时点 p_i 表示的就是图 G 的顶点集合 V 中的一个点。其实在真实的路网环境下，采样得到的点也是用坐标形式表示的，我们会使用 map-matching 算法[14]将采样点映射到路网模型的顶点中。单纯地记录轨迹位置信息的优点是简单方便，没有太多的数据冗余，并且可以使用多种简单的相似性度量方法来计算轨迹相似性，比如 DTW、LCSS 和 EDR，这些相似性度量方法针对的都是仅包含空间信息的轨迹，简单高效。

2.1.2 基于时间信息的轨迹表示方法

我们都知道，轨迹信息有空间尺度和时间尺度，但是上面对轨迹的表述方法仅仅考虑了轨迹的空间尺度，没有考虑到移动物体在采样点空间位置下的时间信息。在研究用户的移动模型或者预测用户下一时刻的位置等情况下，为了更准确地得到研究结果，我们一般会在记录用户位置信息的同时，记录下用户处于该位置的时间信息。我们可以将含有时间信息的轨迹表示为 $T = <(p_1, t_1), (p_2, t_2), \ldots,(p_n, t_n)>$，$p_i$ 表示坐标或者顶点集合中的点，t_i 表示的是用户处于 p_i 位置的时刻[15]。记录下轨迹的时间信息比单纯记录空间位置的轨迹应用地更广泛，不但可以更加详细的描述原始轨迹，还可以使用一些对时间信息敏感的轨

迹相似性度量方法对轨迹进行相似性度量，获得更为准确和有效的相似性结果。

2.1.3 基于文本信息的轨迹表示方法

在某种特殊情况下，我们可能无需考虑轨迹的时间信息，甚至空间信息也不是首要考虑的，但是我们需要好好利用轨迹的文本信息。比如我们要开发一个推荐系统，通过研究用户的个人偏好和个性化的要求，然后给出合理的推荐方案。在生成推荐方案时，为了结合用户的偏好和要求，我们需要考虑轨迹的文本特性，由此产生了基于文本特性的轨迹信息。基于文本特性的轨迹可以表示为 $T = <(p_1, k_1), (p_2, k_2), \ldots,(p_n, k_n)>$，$p_i$ 表示坐标或者顶点集合中的点，k_i 表示 pi 位置的文本描述[16]。

2.1.4 基于网格的轨迹表示方法

网格表示法也是一种常见的轨迹表示方法。将图平面按照一定规则划分成网格，采样点的 id 用所在网格的 id 表示，同时记录下该采样点进入所在网格和离开所在网格的时刻。网格表示法的一般表示形式为：$TR=\{(c, I)|c\in C, I=(t_{in}, t_{out})\}$，其中 c 代表整个网格集合 C 中的一个网格，$t_{in}$ 和 t_{out} 分别表示进入和离开网格的时刻。

2. 2 现有轨迹相似性计算方法

2.2.1 欧氏距离（EU）

欧氏距离[17][18]的计算首先要得到两条轨迹的对应点，按照时间先后，一一对应，如图 2.1 所示。

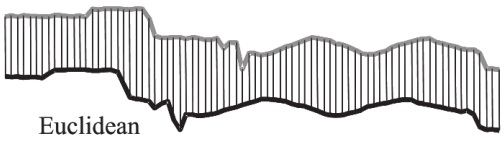


图 2.1 欧氏距离

Fig. 2.1 Euclidean distance

然后将所有对应点的欧氏距离进行综合处理，可以求和、求均值、取中值等，下面以求和的方式举例。给定两个一维的时间序列$A = \{a_1, a_2, \ldots a_n\}$，$B = \{b_1, b_2, \ldots b_n\}$，序列 A 和序列 B 的欧氏距离表达式如式 2.1 所示。欧式距离实际上是 Lp-norms 在 p=2 情况下的一个特例。Lp-norms 定义如式 2.2 所示。当 p=1 时，L1-norms 叫做曼哈顿距离。

$$EU(A,B)=\sqrt{\sum_{i=1}^{n}(a_i-b_i)^2} \tag{2.1}$$

$$l_p norms(A,B)=\sqrt[p]{\sum_{i=1}^{n}(a_i-b_i)^p} \tag{2.2}$$

欧氏距离有很多优点，比如计算简单，长度为 n 的两条轨迹，可以在$O(n)$时间内计算出它们的相似度，而且它满足三角不等式，如式 2.3 所示，运用三角不等式可以计算两条轨迹之间的距离下限，从而可以进行高效的轨迹查询。

$$\forall A,B,C\in T, EU(A,C)+EU(B,C)\geq EU(A,B) \tag{2.3}$$

虽然欧氏距离计算十分简单，时间复杂度低，但是缺点也是显而易见的。第一，使用欧氏距离的前提就是两条轨迹必须要拥有相等的长度，因为欧氏距离的公式决定了两条轨迹必须使用相对应的点来进行计算二维距离。第二，欧式距离不能处理局部时间偏移，局部时间偏移是指由于采样策略或对象移动速度的不同，轨迹上的样本点不能在时间上一一对应，在另一条轨迹上的对应点可能是一段时间之前或者一段时间之后的。第三，使用欧氏距离进行相似性计算容易受到噪声的影响，因为在欧氏距离的计算中，轨迹中的每个点对应到另一条轨迹上的点，如果有噪声点，那么噪声点对最后结果会产生一定的影响，带来更大的距离。随着数据量的变大和研究的深入，我们现在一般用此函数对轨迹数据进行预处理，利用其时间代价低的优点，起到一个初步筛选的作用。

2.2.2 动态时间弯曲距离（DTW）

由于样本点采集设备的误差等原因，两条轨迹数据的样本点在时间上不能一一对应，会产生局部时间偏移的问题，只有将轨迹在时间维度上进行拉伸之后才能进行有效的相似性计算。DTW[17][19][20][21][22]将计算两条轨迹中最小对应距离之和，而不是按照时间关系一一对应，如图 2.2 所示。

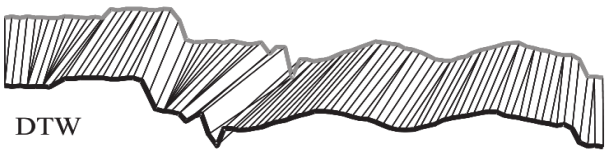


图 2.2 动态时间规整

Fig. 2.2 Dynamic time warping

DTW 的二维空间上的计算公式如式 2.4 所示。其中 m 和 n 分别表示轨迹 A 和轨迹 B 的采样点的个数，即轨迹长度。Head()函数表示轨迹的第一个采样点。d(Head(A),Head(B))表示轨迹 A 和轨迹 B 的第一个采样点之间的欧氏距离。而 Rest()函数表示轨迹除去第一个点剩余的部分。动态时间弯曲距离的公式是用递归定义的，公式的含义是轨迹 A 和 B 的第一个采样点之间的欧氏距离加上轨迹剩余部分的最小的一个 DTW 值，直到轨迹剩余部分长度为零。下面所有公式中涉及到的 Head()和 Rest()函数和动态时间弯曲距离中的 Head()和

Rest()函数意义相同。

$$DTW(A,B)=\begin{cases}0, if\ \ n=0\ \ and\ \ m=0\\ \infty, if\ \ n=0\ \ or\ \ m=0\\ d\big(Head(A),Head(B)\big)+min\begin{cases}DTW\big(A,Rest(B)\big)\\ DTW(Rest(A),B)\\ DTW\big(Rest(A),Rest(B)\big)\end{cases}, \text{otherwise}\end{cases} \tag{2.4}$$

由于动态时间弯曲距离可以通过复制某些点来解决局部时间偏移的问题，弥补了欧氏距离只能处理等长的轨迹数据的缺点，所以应用范围比欧氏距离更广。但是动态时间弯曲的时间复杂度是$O(mn)$，计算代价比欧氏距离大。此外，与欧氏距离一样，计算动态时间弯曲距离时，每一个点都会被强制性找出其对应点，所以也会产生噪声干扰的问题。

2.2.3 最长公共子序列距离（LCSS）

顾名思义，最长公共子序列距离[23][24]计算的是两条轨迹中最长的公共子序列的长度，以此来表示两条轨迹的相似度，计算公式如公式 2.5 所示。

$$\text{LCSS(A, B)}=\begin{cases}0, if\ \ n=0\ \ or\ \ m=0\\ LCSS\big(Rest(A),Rest(B)\big)+1,\\ \quad if\ \ d\big(Head(A),Head(B)\big)\le\varepsilon, and\ \ |n-m|<\delta\\ \max\{LCSS(Rest(A),B),LCSS(A,Rest(B)\},otherwise\end{cases} \tag{2.5}$$

其中：$\text{subcost}=\begin{cases}0, & if\ \ d\big(Head(A),Head(B)\big)\le\varepsilon\\ 1, & otherwise\end{cases}$

实际上，最长公共子序列距离表示的并不是空间距离，而是“得分”，两条轨迹的得分越高，表示它们相似度就越高。由计算公式可知，在递归过程中，每当两条子轨迹的第一个采样点的欧氏距离小于一个阈值ε，并且两段子轨迹的长度在一定的阈值δ以内，就认为这两个点是匹配的，可以给当前结果加一分，继续取二者的子轨迹进行递归，否则就取子轨迹组合中最大的得分，直到子轨迹的长度为零。

相比较前面介绍的两种函数而言，最长公共子序列距离可以有效地避免噪声的干扰。因为噪声点对应到另一条轨迹上时，距离会大于阈值ε，噪声点将不会匹配上另一条轨迹上的点，从而排除了噪声点的干扰。在时间复杂度上，最长公共子序列距离和动态时间弯曲距离一样，也需要$O(mn)$的时间开销。

2.2.4 编辑距离（EDR）

EDR[25]的核心思想是从字符串领域借鉴来的。为了判断两个字符串之间的相似程度，根据对其中一个字符串做增加、删除和修改操作，其中删除一个字符串中的字符可看做是在另一个字符串的增加字符。增加字符的操作是为了使两个字符串序列长度相等，我们把增加的字符叫做间隙元素（gap）。两个字符

串之间的距离如式 2.6 所示。

$$dist(r_i, s_i) = \begin{cases} 0 & if\ \ r_i = s_i \\ 1 & if\ \ r_i \text{ or } s_i\ \ is\ \ a\ \ gap \\ 1 & otherwise \end{cases} \tag{2.6}$$

然而时间序列中的元素是实数，有时候不会像字符那样完全相等，所以当两个实数之差小于阈值δ时，我们就认为这两个实数相等，因此时间序列中元素之间的距离如式 2.7 所示。EDR 是基于时间序列中元素的距离 $dist_{edr}$ 得到的，如式 2.8 所示，其中序列 R 和 S 的长度分别是 m 和 n，Rest（R）和 Rest(S)是序列 R 和 S 除第一个元素以外的剩余元素。EDR 能够处理时间序列偏移的能力就是由于当 r1 和 s1 不相等时，取值为 R、S 和其剩余部分相结合 EDR 的最小值，从而匹配了最合适的点对。

$$dist_{edr}(r_i, s_i) = \begin{cases} 0 & if\ \ |r_i - s_i| \le \delta \\ 1 & if\ \ r_i \text{ or } s_i\ \ is\ \ a\ \ gap \\ 1 & otherwise \end{cases} \tag{2.7}$$

$$\text{EDR(R, S)} = \begin{cases} n & if\ \ m = 0 \\ m & if\ \ n = 0 \\ EDR\big(Rest(R), Rest(S)\big) & if\ \ dist_{edr}(r1, s1) = 0 \\ \min \begin{cases} EDR\big(Rest(R), Rest(S)\big) + dist_{edr}(r1, s1) \\ EDR(Rest(R), S) + dist_{edr}(r1, gap) \\ EDR\big(R, Rest(S)\big) + dist_{edr}(gap, s1) \end{cases} & otherwise \end{cases} \tag{2.8}$$

2.2.5 带真实惩罚的编辑距离（ERP）

ERP[25]是 L1-norms 和 EDR 的一个结合，在计算两个元素之间距离的时候，当遇到两个非间隙元素时采用元素间真实的 L1-norms 距离而不是 0，当其中有一个元素是间隙元素时，利用一个常数 g 来参与 L1-norms 距离计算，因此 ERP 的计算结果中包含了两条轨迹之间真实的距离。ERP 中两个序列中元素的距离表示如式 2.9 所示。基于 disterp 的 ERP 计算公式如式 2.10 所示，类似于 EDR 的计算方法，当序列 R 和 S 长度均不为 0 时，ERP 将计算 R、S 与其剩余部分结合的 ERP 最小值，因此 ERP 同样可以处理局部时间偏移。

$$dist_{erp}(r_i, s_i) = \begin{cases} |r_i - s_i|\ \ ,if\ \ r_i\ \ ,s_i\ \ not\ \ gaps \\ |r_i - g|\ \ ,if\ \ s_i\ \ is\ \ a\ \ gap \\ |s_i - g|\ \ ,if\ \ r_i\ \ is\ \ a\ \ gap \end{cases} \tag{2.9}$$

$$\text{ERP(R, S)} = \begin{cases} \sum_1^n |s_i - g|, if\ \ m = 0 \\ \sum_1^m |r_i - g|, if\ \ n = 0 \\ \min \begin{cases} ERP\big(Rest(R), Rest(S)\big) + dist_{erp}(r1, s1) \\ ERP(Rest(R), S) + dist_{erp}(r1, gap) \\ ERP\big(R, Rest(S)\big) + dist_{erp}(gap, s1) \end{cases} & ,otherwise \end{cases} \tag{2.10}$$

2.2.6 路网上的轨迹相似性查询

当将一段轨迹展示在路网中时，考虑到实际道路情况，点与点之间不一定有直线道路相连，使用二维空间的 Lp-norms 即欧氏距离来计算两条轨迹的相似度和实际相似情况可能相差较大。因此我们需要重新定义一个适用于路网的相似性函数[27]。

在计算相似性之前，需要将轨迹映射到路网上去，如图 2.3 和图 2.4 所示。给定移动对象 a 和 b 的移动轨迹 T_a 和 T_b，轨迹格式为 $T=\{(l_1, v_1, t_1), (l_2, v_2, t_2),\ldots(l_m, v_m, t_m)\}$，其中 $l_i=(lg_i, la_i)$表示样本点坐标，vi 表示对象在 ti 时刻的移动速度。我们用 $d_a(lai,Tb)$表示从样本点 l_{ai} 到轨迹 T_b 的路网距离。l_{ai} 到 T_b 路网距离指，T_b 上距离 l_{ai} 最近的点到 l_{ai} 的路径距离。用 D_G 表示路网 G 的直径。在不同条件下，对图 G 中的轨迹进行相似性查询，我们有不同的距离函数。

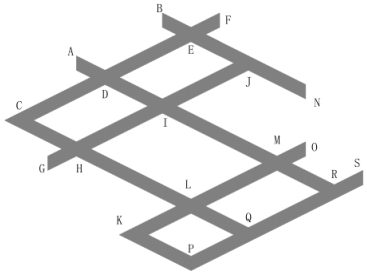


图 2.3 真实路网

Fig. 2.3 Road network

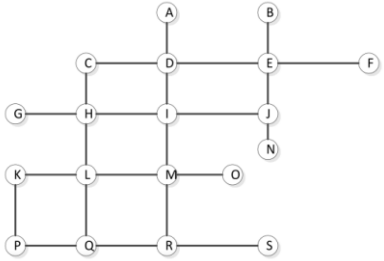


图 2.4 路网模型

Fig. 2.4 Model of road network

路网上 Ta 和 Tb 的距离 dN(Ta, Tb)的计算公式如式 2.14 所示。其中，轨迹 Ta 叫做查询轨迹，轨迹 Tb 叫做目标轨迹，m 代表轨迹 Ta 的样本点个数。

$$d_N(T_a,T_b)=\frac{1}{m}\sum_{i=1}^{m}\frac{d_a(l_{ai},T_b)}{D_G} \tag{2.14}$$

当查询用户有对兴趣点的查询需求时，式中的样本点将会拥有权值 wai，wai 的大小代表样本点在相似性查询操作时不同的重要性。带权值的轨迹距离公式如式 2.15 所示。

$$d_{NW}(T_a,T_b)=\frac{1}{m}\sum_{i=1}^{m}\frac{w_{ai}d_a(l_{ai},T_b)}{D_G} \tag{2.15}$$

当需要对路况信息或者交通拥堵信息进行分析时，轨迹的实时速度信息就

十分重要了。令 S_G 代表当前道路最高限速，ds(lai,lbi)代表轨迹 A 上的第 i 个样本点 lai 与轨迹 B 上的第 i 个样本点 lbi 的速度之差，其中 lbi 是轨迹 B 上到 lai 最近的样本点。带速度信息的轨迹距离公式如式 2.16 所示。

$$d_{NWS} = \frac{1}{m}\sum_{i=1}^{m} \frac{w_{ai} d_a(l_{ai},T_b)}{D_G} \frac{d_s(l_{ai},l_{bi})}{S_G} \tag{2.16}$$

分析路况信息时，时间信息也是一个很重要的因素。我们令$|d_t(l_{ai},l_{bi})|$代表轨迹 A 上的第 i 个样本点 lai 与轨迹 B 上的第 i 个样本点 lbi 的时间之差，其中 lbi 是轨迹 B 上到 lai 最近的样本点。la1 是轨迹 Ta 的第一个点，lam 是轨迹 Ta 的最后一个点，lb1 是轨迹 B 上距 la1 最近的点，lbm 是轨迹 B 上距 lam 最近的点。带时间信息的轨迹距离公式如式 2.17 所示。

$$d_r = \frac{1}{m}\sum_{i=1}^{m} \frac{|d_t(l_{ai},l_{bi})|}{max\{|d_t(l_{am},l_{b1})|,|d_t(l_{a1},l_{bm})|,|d_t(l_{am},l_{a1})|,|d_t(l_{bm},l_{b1})|\}} \tag{2.17}$$

当结合权值、时间和空间来计算轨迹距离，我们有公式 2.18，其中w_{NW}和w_r分别代表对应的子计算方法的权值参数，并且两个$w_{NW}+w_r=1$。如果将速度信息加入公式，公式如式 2.19 所示。

$$d_{NWT} = d_{NW}(T_a,T_b) + w_r d_T \tag{2.18}$$

$$d_c = w_{NWS} d_{NWS}(T_a,T_b) + w_r d_r \tag{2.19}$$

上述五种情况的基本思想是首先找出对应点，给定查询轨迹上的点，它的对应点是目标轨迹上到该点路网距离最近的点。然后考虑每一组点对之间距离、权值、速度和时间占总体的比例，最终得出以上公式。优点是并没有采取按照时间来匹配对应点的想法，计算结果可能更符合原始相似情况，并且计算思路简单、清晰，扩展方便。缺点是样本点权值的选取没有给出一个具体的方法，可能会使计算效果不好。

2.2.7 城市运输系统中基于段的轨迹相似性计算

轨迹点是根据一个给定的采样方法获取的，不同的采样方法给轨迹相似性计算带来了很大的影响。比如两条完全相同的轨迹，但是由于采样开始时间不同，就造成了样本点的错位，如图 2.5 所示。

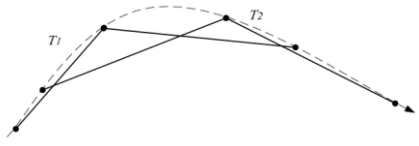


图 2.5 不同采样方法造成的样本点错位

Fig. 2.5 Dislocation of sample points caused by different sampling methods

传统的相似性计算方法，比如 DTW 就没有考虑这个问题。LCSS 忽略了轨迹的空间距离，EDR 没有考虑到轨迹的形状因素。由于传统轨迹相似度计算方法计算结果的不准确，所以 Mao 等人提出了基于段的轨迹相似性计算方法[28]。

首先介绍点段距离的概念。点段距离是两条轨迹对应点之间的特殊距离，表示为图 2.6 中的阴影面积，由样本点 R、S 以及各自前后样本点的中点连接而成。由于不规则阴影面积的计算比较复杂，而阴影部分正比于图 2.7 中两个虚线三角形的面积之和，所以将阴影部分面积的计算转换为图 2.7 中三角形面积的计算。但是当 seg1 和 seg2 很长时，效果并不好，所以用三角形的高，p1 到 seg2 的距离和 p2 到 seg1 的距离来代表 p1 与 p2 之间的距离，如图 2.8 所示，其中$dist_{ps}$为 p1 到轨迹 S 的点段距离。对应的距离公式如式 2.20 所示。

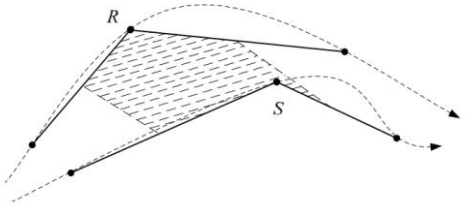


图 2.6 使用轨迹段面积表示轨迹距离

Fig. 2.6 Using trajectory area to represent track distance

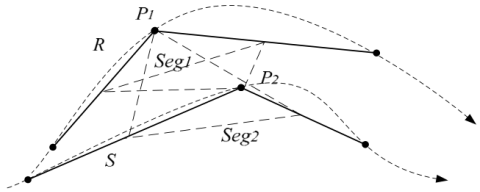


图 2.7 三角形面积替代阴影面积

Fig. 2.7 Triangle area instead of shadow area

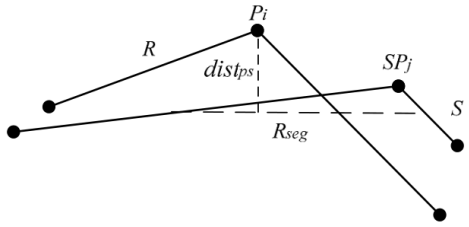


图 2.8 用三角形的高代替三角形面积

Fig. 2.8 Replace triangle area with triangle height

$$dist_{ps}(P_1, R_{seg}) = \begin{cases} \sqrt{(x_i - x_{mid1})^2 + (y_i - y_{mid1})^2} & if \ \ r \le 0 \\ \sqrt{(x_i - x_{mid2})^2 + (y_i - y_{mid2})^2} & if \ \ r \ge L_{seg} \\ \sqrt{(x_i - dx)^2 + (y_{mid1} - dy)^2} & otherwise \end{cases} \tag{2.20}$$

其中，$\begin{cases} (x_{mid1}, y_{mid1}) = ((x_{j-1} + x_j)/2, (y_{j-1} + y_j)/2) \\ (x_{mid2}, y_{mid2}) = ((x_j + x_{j+1})/2, (y_j + y_{j+1})/2) \end{cases}$

然后介绍预测距离。给定一组对应点P_i和SP_j，时间戳分别为 ti 和 tj，ti 和

tj 不相等，令时间戳大的点SP_j等于 A，时间戳小的点P_i等于 B，如图 2.9 所示。

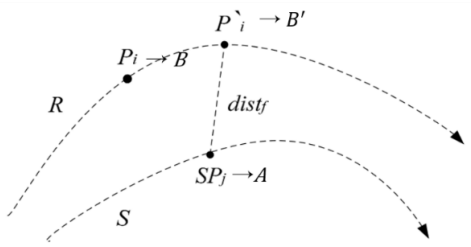


图 2.9 预测 B 在与 A 相同时刻的位置B′

Fig. 2.9 Predict the location of B named B' at the same time as A

然后利用轨迹 R 在 ti-1 时刻的位置和 ti-1 与 ti 间的平均速度，来预测 tj 时刻的位置，得到P_i'，$P_i' = (x_{B'}, y_{B'})$的预测距离计算公式如式 2.21 所示。

$$\begin{cases} x_{B'} = x_{i-1} + v_i^x(t_B + \Delta t - t_{i-1}) \\ y_{B'} = y_{i-1} + v_i^y(t_B + \Delta t - t_{i-1}) \end{cases} \tag{2.21}$$

其中，$\begin{cases} v_i^x = (x_i - x_{i-1})/\Delta t_i \\ v_i^y = (y_i - y_{i-1})/\Delta t_j \end{cases}$

那么这两个点的时间距离可以转化为 tj 时刻，轨迹 R 的位置P_i'到SP_j的距离，如图 2.9 所示。P_i'的预测距离计算公式如式 2.22 所示。

$$dist_t(A, B) = dist(A, B') \tag{2.22}$$

图 2.10 中，融合点段距离$dist_P(P_i, SP_j)$和预测距离$dist_t(P_i, SP_j)$，我们可以得到对应样本点P_i和SP_j之间的时空距离公式，如式 2.23 所示，其中 t 是时间距离的敏感参数，值越大表示时间距离越重要。

$$dist_{st}(P_i, SP_j) = dist_P(P_i, SP_j) + t \times dist_t(P_i, SP_j) \tag{2.23}$$

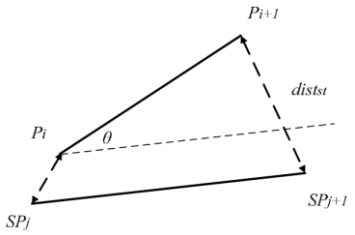


图 2.10 利用夹角计算形状相似性

Fig. 2.10 Calculation of shape similarity based on included angles

利用样本点之间的时空距离$dist_{st}(P_i, SP_j)$，可以得到该样本点和后一个样本点形成的轨迹段之间的距离$dist_{st}(S_i, SS_j)$，即段段距离，来计算轨迹段S_i和

轨迹SS_j之间的形状相似程度，如式 2.24 所示，其中S_i是P_i与P_{i+1}之间的轨迹段，SS_j是SP_i与SP_{i+1}之间的轨迹段。

$$dist_{st}(S_i, SS_j) = dist_{st}(P_i, SP_j) + dist_{st}(P_{i+1}, SP_{j+1}) \tag{2.24}$$

在判断两条轨迹的相似程度的时候，形状上的相似也十分重要。结合形状因素的段段距离如式 2.25 所示，其中θ是两条轨迹段之间的夹角，θ和$f(\theta)$的计算公式如式 2.26 和式 2.27 所示。

$$dist_s(S_i, SS_j) = f(\theta) dist_{st}(S_i, SS_j) \tag{2.25}$$

$$\theta = |arctan2(y_{i+1} - y_i, x_{i+1} - x_i) - arctan2(y_{j+1} - y_j, x_{j+1} - x_j)| \tag{2.26}$$

$$f(\theta) = \frac{dist_{smid}(S_i, SS_j)}{dist_{max}(R,S)} \times (\omega + \theta) \tag{2.27}$$

前面得到了结合形状因素的轨迹段之间的时空距离$dist_s(S_i, SS_j)$，我们将其当做一个距离计算因子，代替 DTW 函数中使用的对应点之间距离，可以得到式（28），其中$Head(R)$指的是轨迹 R 的第一个样本点和第二个样本点之间的轨迹段，$Rest(R)$指的是出掉第一个轨迹段剩下的所有轨迹段。

$$SDTW(R,S) = \begin{cases} 0, & if\ \ n = 0\ \ and\ \ m = 0 \\ \infty, & if\ \ n = 0\ \ or\ \ m = 0 \\ dist_s(Head(R), Head(S)) + \min \begin{cases} SDTW(T, Rest(S)) \\ SDTW(S, Rest(T)) \\ SDTW(Rest(T), Rest(S)) \end{cases} \end{cases} \tag{2.28}$$

该方法主要优点有三个：

(1)改进 DTW 函数，采用轨迹段到轨迹段的距离来代替点到点的距离计算，可以减少对轨迹采样方法的敏感程度。

(2)利用预测的方法，对于一个点对，预测时间戳靠前的点在下一刻的位置，使得两个点时间戳相同，将时间距离转换为空间距离，考虑到相似性计算中去。

(3)将形状因素加入到相似性计算中，提高形状相似性方面的精度。

2.2.8 简化的轨迹相似性计算

由于利用轨迹相似性可以进行未来某一时刻的位置预测，Liu 等人首先提出了基于社会传染理论的位置预测算法[29]，然后提出了一个简化的轨迹相似性计算方法来找出带预测用户的相似用户组，以此支撑社会传染理论的运行，

根据用户在某地活动消耗的的时间比仅仅路过该地的时间长，并减少相似性计算的复杂度，我们将完整的轨迹分解成小轨迹。分解条件就是用户u_i的时间跨度$t_i > T_{cut}$，　T_{cut}是一个时间阈值。分解后得到用户u_i的小轨迹集$Tr_i = \{trace_1, trace_2, \dots trace_n | trace_i \in \mathrm{L}, 1 \le \mathrm{i} \le \mathrm{n}\}$。我们可以得到所有用户的小轨

迹集$Tr_{total} = Tr_1 \cup Tr_2 \cup \ldots \cup Tr_n$。

然后两个小轨迹集之间的重合程度来计算他们的相似程度。令$V_i = \{v_1, v_2, \ldots v_m\}$代表用户$u_i$的布尔型向量，$v_i$的值如式 2.29 所示。

$$v_i = \begin{cases} 1, & ith\ trace \in\ Tr_i \\ 1, & otherwise \end{cases} \tag{2.29}$$

当比较用户u_i和用户u_j之间的相似程度时，将v_i和v_j做与运算（$v_i \& v_j$），取 1 的个数作为轨迹之间的相似度值。

这个方法的优点是相似度计算过程简单快速，没有过多地求轨迹之间的空间距离以及考虑时间、速度等因素。缺点就是计算结果可能随阈值T_{cut}设置的好坏而变化，若T_{cut}设置过大，导致小轨迹过长，会让两条相似的轨迹重合的小轨迹变少，从而不能反映真实的相似程度，精度不够。

2.2.9 基于签名的轨迹相似性计算(BDS)

当前的轨迹相似性函数很大程度依赖两条轨迹的对应样本点，然后计算对应样本点之间包含各种信息的距离，但是由于采样频率或者物体移动速度不同，样本点很可能不能一一对应，如图 2.11 所示。因此 Ta 等人提出了 BDS(Bi-directional mapping similarity)算法[30]计算两条轨迹之间的相似程度。与找轨迹的对应点的方法不同，BDS 在计算轨迹 Ti 和 Tj 的相似度时，通过累加 Ti 的每个样本点到 Tj 的最短距离，如图 2.12 所示。

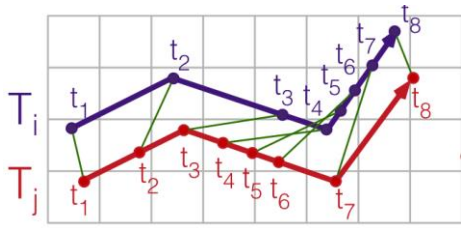


图 2.11 按时间匹配点对

Fig. 2.11 Match point by time

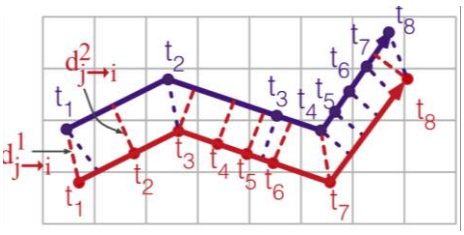


图 2.12 按点到轨迹段最小距离匹配

Fig. 2.12 Minimum distance matching from point to track segment

这个最短距离的定义是将 Tj 上的所有样本点按照时间顺序连线，形成 Tj-1 条轨迹段，Ti 上的第 k 个样本点到这 Tj-1 条轨迹段的最短距离就是 Ti 上样本点到 Tj 的距离$Dist_{PT}(p_i^k, T_j)$，其计算公式如式 2.30 所示。

$$Dist_{PT}(p_i^k, T_j) = min_{l \in T_j} Dist_{PL}(p_i^k, l) \tag{2.30}$$

而 Ti 上样本点到轨迹段的最短距离分为两种情况，如果点到轨迹段的垂线与轨迹段相交，距离就是垂线的长度，否则就是样本点到轨迹段里自己最近的端点的距离，如图 2.13 所示。

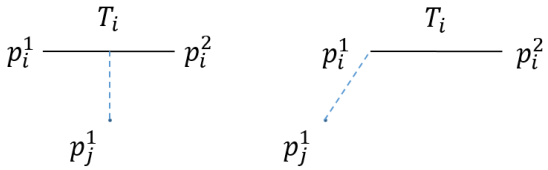


图 2.13 点到轨迹段的距离

Fig. 2.13 Distance from point to track segment

BDS 的计算公式如式 2.31 所示，其中$d_{i\to j}^k$是一个归一化的距离，最终结果$SIM(T_i, T_j)$是一个介于 0 到 1 之间的值，表示轨迹T_i和轨迹T_j之间的相似程度，值越大，表示两条轨迹越相似。

$$SIM(T_i, T_j) = 1 - \frac{\sum_{k=1}^{|T_i|} d_{i\to j}^k + \sum_{k=1}^{|T_i|} d_{j\to i}^k}{|T_i| + |T_j|} \tag{2.31}$$

其中，$d_{i\to j}^k = \begin{cases} \frac{Dist_{PT}(p_i^k, T_j)}{D_{max}} & if\ \ Dist_{PT}(p_i^k, T_j) \le D_{max} \\ +\infty & if\ \ Dist_{PT}(p_i^k, T_j) \le D_{max} \end{cases}$

该相似性函数的计算量很大，对于两条轨迹，要计算出每一个样本点到另一条轨迹所有段的最短距离。为了弥补这个缺点，使用该方法之前需要利用网格作为轨迹签名，先进行一次筛选，然后在计算轨迹T_i上第 k 个点p_i^k到轨迹T_j的距离时，只需要计算点p_i^k到以该点为中心的一定范围内的网格中存在的T_j轨迹段的距离，极大程度地减少了计算次数。

这个相似性计算方法方法的优点是不用硬性地将两条轨迹中的点进行配对，一个样本点的对应点可能在另一条轨迹段中两个样本点之间，将两条相似的轨迹更好地进行吻合。缺点就是由于仅考虑了两条轨迹的空间位置，没有考虑其他信息，比如时间和速度信息，因此仅适用于比较两条道路的相似性，不能完整地反映移动对象的详细信息。

2.4 本章小结

本章主要介绍了不同的轨迹的表示方法以及为解决不同问题而提出来的一些轨迹相似性计算方法。有些相似性使用范围较为广泛，比如 DTW、LCSS 和 EDR，但是这些算法不针对某个具体场景或具体问题，并没有研究具体情况下

该引入哪些特征去描述轨迹或者该怎么去优化轨迹间的相似性表示。因此现在轨迹相似性的研究更加偏向于研究某个具体场景下的轨迹相似性，比如城市运输系统中基于段的相似性查询，和路网上的轨迹相似性查询等。

第三章 三维时空下对应点匹配算法

3.1 问题定义

3.1.1 轨迹相似性查询相关概念

人们会使用采样设备记录下移动对象的位置信息，位置信息中包含时间和空间信息，采样得到的数据就是包含时间和空间信息的时空数据。采样设备对某一个对象连续采样得到的位置信息按照时间先后排序得到的有序序列叫做轨迹数据。

实际上移动对象的真实路径包含无穷多个点，这无穷个点共同组成移动对象的一段连续的移动路线，但是 GPS 的采样策略一般是每隔一定时间或者移动对象每移动一段距离进行一次位置信息的记录，因此最后得到的轨迹数据是由有限个位置点组成。为了突出研究的问题，简化其他计算步骤，本文在空间上使用欧式空间，由于原始轨迹数据中的空间信息使用经纬度表示，我们在数据预处理时将经纬度转化为欧式空间下的坐标 x 和 y。下面介绍与轨迹数据相似性相关概念的定义。

定义 3.1 样本点。由采样设备获得移动对象在某一时刻下的地理位置信息，包括横坐标 x、纵坐标 y 和时间信息 timestamp，这些信息组成一个样本点 p，或者叫采样点，$\mathrm{p} = (p.x, p.y, p.timestamp)$。

定义 3.2 轨迹。通过采样设备获得了一个移动对象一段时间内的样本点，这些样本点以时间顺序组成的序列 T 叫做轨迹，其中$\mathrm{T} =< p_1, p_2, p_3 \ldots p_n >$。

定义 3.3 轨迹相似性计算函数。给定两条轨迹数据 Q、R 和一个轨迹相似性计算函数$sim(Q, R)$，输出一个可以表示轨迹之间的相似度的值，或者输出距离，距离越小，代表相似度越大，两条轨迹越相似。

定义 3.4 基于阈值的轨迹相似性查询。给定查询轨迹 Q，轨迹数据库 DB，以及一个相似性阈值η，输出 DB 中的轨迹集合T，使得$\forall \mathrm{R} \in T$，有$sim(Q, R) \geq \eta$，并且$\forall \mathrm{S} \in DB \ \ and \ \ S \subsetneq T$，有$sim(Q, S) < \eta$。

3.1.2 轨迹相似性问题定义

数据库中存储的采样设备采集的轨迹数据通常是一个用户或者一辆汽车一天的移动路线的记录，描述了一个移动对象在很长一段时间内的移动情况。如果是出租车轨迹，可能这辆车一天之内来来回回在城市大小景点、酒店中间要转好几圈，而我们输入的查询轨迹有时会比较短，查询轨迹只需要起点、终点

加几个转折点就可以确定，下面以一个例子说明这种情况中存在的问题。为简化起见，我们使用两条长度稍短的轨迹来说明。

假设一个小偷行窃之后驾车逃跑，失主开车在后面追。从 A 点开始追，经过 B 点之后，在 C 点拦下了小偷的车辆，如图所示。但是车上没有发现丢失物品，怀疑是小偷中途把物品扔下车了，但是中途由于路况复杂，跟车较远，失主的行车记录仪没有拍到扔东西的过程。为了找到证据，失主希望通过小偷的行车轨迹，在轨迹数据库中查找有相似轨迹的车辆，查看是否有车辆的行车记录仪拍摄到小偷抛弃物品的过程。

假如数据库中存在两条轨迹 R 和 S，如图 3.1 所示。其中轨迹 R 在 8 点的时候从r_0开往r_7处，到达r_7点为 8 点 30 分，途径 A 点和 B 点。轨迹 S 在 8 点 10 分从s_0点开至s_1点，到达s_1点为 8 点 25 分。查询轨迹 Q 即用户的行车轨迹，是 8 点 15 分从 A 点开始，途径 B 点，到 8 点 25 分结束于 C 点。查询要求是希望得到和查询轨迹最接近的轨迹。

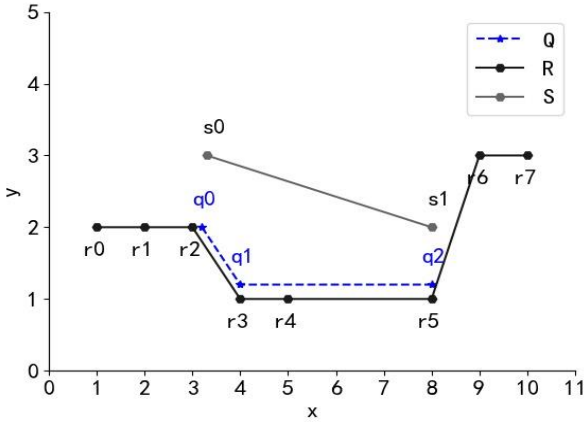


图 3.1 轨迹相似性查询示例

Fig. 3.1 Distance from point to track segment

根据轨迹 R 和 S 的文字描述以及可视化的轨迹展示，发现轨迹 R 中从r_3到r_5这段子轨迹很符合查询要求。但是如果使用之前的轨迹相似性查询方法，会直接计算整条数据轨迹 R 和查询轨迹 Q 的相似性，其结果就是轨迹 R 中轨迹 Q 相距较远的样本点如r_0到r_2、r_6到r_7会对加大轨迹间的距离，掩盖了轨迹 R 中r_3到r_5子轨迹段和查询轨迹的相似性，使得与 Q 相距较远的轨迹 S 成为了搜索结果，很明显 S 不是我们希望得到的查询结果。

为了解决上述问题，下面给出欧式空间下子轨迹的定义。

定义 3.5 子轨迹。给定一条轨迹数据T $=< p_1, p_2, p_3 \ldots p_n >$，T 的子轨迹$T(p_i, p_j)$是 T 的一部分，其中$T(p_i, p_j) =< p_i, p_{i+1}, p_{i+2} \ldots p_j >$，i<=j，且$T(p_i, p_j)$包含了轨迹数据 T 从$p_i$到$p_j$的所有样本点。

问题定义：给定一条查询轨迹 Q，以及一个存储轨迹的数据库，在考虑时间、空间、轨迹方向和形状的情况下，查询得到数据库中与 Q 相似的轨迹或子轨迹。

本文称该问题为轨迹相似性查询问题。为了解决该问题，首先需要制定一个适用于本场景的轨迹相似性计算方法，其中需要包含对时间差、空间距离以及轨迹形状的考虑。然后利用该相似性查询算法，去轨迹数据库中进行查询，会涉及到设计索引来过滤掉其余轨迹并加速计算过程和查询过程。

本文使用到的符号如表 3.1 所示。

表 3.1 符号定义

Table 3.1 The Symbol defination

名称	描述
Q	查询轨迹
R，S	数据库中的数据轨迹
q_i, r_i, s_i	对应轨迹上的第 i 个点
Q(r_i)	样本点r_i在轨迹 Q 上的对应点
q_i.pre	轨迹 Q 中点q_i的前一个样本点
q_i.next	轨迹 Q 中点q_i的后一个样本点
DTW(r_i)	r_i的所有 DTW 对应点
DTW(r_i).first	DTW(r_i)中时间戳最小的点
BDS(r_i)	r_i的 BDS 对应点
DTW-BDS(r_i)	r_i的 DTW-BDS 对应点

3. 2 时空归一化

3.1.1 PTM 算法中时空维度结合存在的问题

之前有很多研究人员对时空轨迹相似性进行研究，在处理时间和空间的关系上，大多采取将二者分开计算的做法，分别计算相似程度或者距离，然后给出权值将二者进行结合。最典型的方法就是下面的 PTM 算法[34]（Personalized trajectory matching）。

下面介绍 PTM 算法的主要思想。在空间维度上，令查询轨迹 Q 上的点q_i和数据轨迹 R 上的点r_j的空间距离为$sd(q_i, r_j)$，这里的空间距离计算的是路网上的距离，基于此空间距离，获得q_i和r_j的空间距离影响因子$I_s(q_i, r_j)$，如公式 3.1 所示。距离越大，$I_s(q_i, r_j)$越小，当距离大到一定程度时，$I_s(q_i, r_j)$为 0。然后借用 LCSS 的思想，基于轨迹上所有样本点的位置关系以及样本点的权重得到空间相似性$S_{sim}(Q, R)$，如公式 3.3 所示，其中权重$Q.head.w$是用户对 Q 的

第一个样本点的重视程度，也是文章的创新点之一。在时间维度上，查询轨迹 Q 上的点q_i和数据轨迹 R 上的点r_j的时间差为$|q_i.t - r_j.t|$，时间影响因子$I_t(q_i, r_j)$的计算如公式 3.2 和公式 3.4 所示。然后基于样本点的时间关系得到时间相似性$T_{sim}(Q,R)$，其计算方法与空间相似性类似。最后使用参数λ将空间相似性和时间相似性结合，得到轨迹的时空相似性$ST_{sim}(Q,R)$，如公式 3.5 所示。

$$I_s(q_i, r_j) = \begin{cases} 0 \ , if \ \ sd(q_i.p, r_j.p) > \varepsilon_s \\ e^{-sd(q_i.p, r_j.p)} \ , otherwise \end{cases} \tag{3.1}$$

$$I_t(q_i, r_j) = \begin{cases} 0 \ , if \ \ |q_i.t - r_j.t| > \varepsilon_t \\ e^{-|q_i.t - r_j.t|} \ , otherwise \end{cases} \tag{3.2}$$

$$S_{sim}(Q,R) = max\begin{cases} Q.head.w \times I_s(Q.head, R.head) + S_{sim}(Q.tail, R) \\ S_{sim}(Q, R.tail) \end{cases} \tag{3.3}$$

$$T_{sim}(Q,R) = max\begin{cases} Q.head.w \times I_t(Q.head, R.head) + T_{sim}(Q.tail, R) \\ T_{sim}(Q, R.tail) \end{cases} \tag{3.4}$$

$$ST_{sim}(Q,R) = \lambda S_{sim}(Q,R) + (1-\lambda) T_{sim}(Q,R) \tag{3.5}$$

下面使用一个例子来解释 PTM 算法的计算过程。为了减少时间和空间在数值上的影响，这里使用的时间和空间的数值尽量接近。并且为了简化运算，突出主要问题，这里使用欧氏距离代替论文中使用的路网距离。

假设我们数据库中有数据轨迹 R 和 S，并给定了一条查询轨迹 Q。三条轨迹的空间维度可以参考图 3.2，图中可以看出在空间上轨迹 R 中r_0至r_2段和查询轨迹 Q 很相似，轨迹 S 中的s_0至s_2也和查询轨迹 Q 很相似，但是 R 和 Q 的距离更近一点。三条轨迹的时间维度可以参考图 3.3，该图纵轴代表从 8 点开始的分钟数，比如q_0处的 y 值为 15，代表q_0处的时刻为 8 点 15 分。单独从时间维度上看，轨迹 R 在r_3至r_5内的时间戳和轨迹 Q 的更加吻合，而轨迹 S 在s_0至s_2内的时间戳和轨迹 Q 稍有偏差。PTM 算法的参数设置如下，所有样本点权重相等，时间和空间的权重相等。计算得到$ST_{sim}(Q,R) = 2.61$，$ST_{sim}(Q,S) = 1.10$，因此 Q 和 R 的相似度更高。

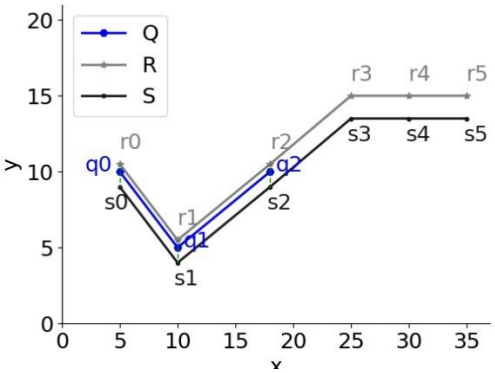


图 3.2 轨迹在空间维度的情况

Fig. 3.2 Trajectory in spatio

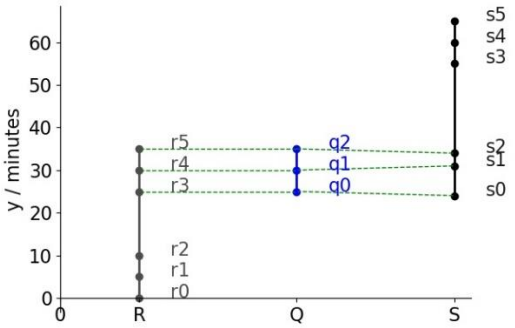


图 3.3 轨迹在时间维度的情况

Fig. 3.3 Trajectory in temporal

PTM 算法的相似性查询结果为轨迹 R，这条轨迹的r_0r_2在空间上与 Q 相似，r_3r_5在时间上与 Q 相似。这里引出了一个问题就是，在 PTM 算法中使用 LCSS 算法的基本思想计算得到的空间相似性S_{sim}和时间相似性T_{sim}，能否通过线性组合得到我们需要的时空相似性。轨迹的时空相似性指的是同时考虑时间维度和空间维度下轨迹的相似程度。

在这个例子中，和 R 相比，显然 S 是最优解。如果使用问题定义中的情景来解释，即 Q、R 和 S 均为三条车辆的轨迹，Q 和 R 车辆在同一时刻最接近的情况下是q_0到r_3的距离，R 车辆不可能有机会拍摄到 Q 车辆。相比之下，S 车辆在同一时刻下，距离 Q 车辆很近，很有可能拍摄到 Q 车辆中司机的行为，S 车辆是我们想查询得到的结果。因此不可用通过两条轨迹的空间相似性和时间相似性的线性组合得到轨迹的时空相似性。

3.1.2 时空归一化方法

根据前面对 PTM 算法的分析，得知轨迹的时空相似性不能简单由时间相似性和空间相似性的线性组合得到，在计算轨迹相似性的时候，需要将时间维度与空间维度统一考虑，不能单独计算各自维度的相似性。

由于时间和空间属于两个维度，我们不能直接将时间纳入时空距离的计算中。若场景是问题定义中的寻找到一辆能拍摄小偷驾驶车辆的车，假设摄像头能拍摄清楚的最远距离为 200 米，且以小偷车辆的平均车速行驶，20 秒钟行驶 200 米，空间上落后 200 米的效果等同于时间上落后 20 秒的效果，均达不到拍摄不到小偷的车辆的要求，而距离等于时间乘以速度，距离与时间之间存在一个正比关系，因此可以将速度看做一个时空转化因子I_{st}，将时间维度向空间距离上做转化。I_{st}等于 200 米除以 20 秒，为 10 米每秒。

如果应用场景不同，时空转化因子也会不同。刚才的场景是需要车辆进行实时追踪，还有种情况是使用相似性查询为未来时刻服务。比如想找到相似轨迹搭顺风车，那么空间上的间隔要求小于 200 米，因为不想多走路，时间上可能在 15 分钟之内都能接受，那么这里的时空转化因子I_{st}等于 0.22 米每秒。

然后就可以使用该转化因子I_{st}将时间 t 转化为空间中的 z，如公式 3.6 所示。这样结合二维的欧式空间的 x 和 y 两个维度，就可以将时间和空间归一化为一个三维的空间，其中 x 轴和 y 轴表示的是原本的空间的两个维度，z 轴表示的是由时间转化过来的维度。

$$\mathrm{z} = I_{st} \times \mathrm{t} \tag{3.6}$$

上述方法即为本文提出的时空归一化方法，可以将轨迹的时间和空间因素进行结合，得到的空间可以称为三维时空。时空归一化算法的具体伪代码如算法所示。

算法 3.1 时空归一化 normalization(trajectorie, I_{st})

输入：原始轨迹数据 trajectories，时空转化因子I_{st}

输出：三维时空中的轨迹数据 trajectories_3d，时空转化因子I_{st}

```
1.  trajectories_3d=[]
2.  for p_i ∈ trajectory
3.      z = I_st × p_i.timestamp
4.      trajectories_3d.add([p_i.x, p_i.y, z])
5.  end for
6.  return trajectorie_3d
```

使用该方法可以直接获得同一对样本点上的时间和空间上的差异，我们统称为时空距离。时空距离的定义与普通三维欧式空间的定义完全相同，点v_1和v_2的时空距离$\mathrm{d}(v_1, v_2)$如公式 3.7 所示。后面如果没有指明，则默认$\mathrm{d}(v_1, v_2)$表示点v_1和v_2的时空距离。将$v_i.z = I_{st} \times v_i.t$带入后，得公式 3.8。

$$\mathrm{d}(v_1, v_2) = \sqrt{(v_1.x - v_2.x)^2 + (v_1.y - v_2.y)^2 + (v_1.z - v_2.z)^2} \tag{3.7}$$

$$\mathrm{d}(v_1, v_2) = \sqrt{(v_1.x - v_2.x)^2 + (v_1.y - v_2.y)^2 + I_{st}^2(v_1.t - v_2.t)^2} \tag{3.8}$$

时空转化因子I_{st}的作用是将时间转换为空间，而在其中使用轨迹平均速度

v_{avg}的原因是希望将时间上的差距近似的转化以一个平均速度，在该时间差距内移动的平均空间距离。其中不论时间的单位是什么，平均速度的单位随时间单位变化而变化，如果时间最小单位是分钟，那么速度单位就是米每分钟，如果时间最小单位是秒，那么速度单位就是米每秒，最后转化为空间的单位都是米，即三维时空中，xyz 三个维度的单位均为米。在三维时空中，并没有改变原本空间上的两个维度，因此原本只考虑空间距离的相似性算法仍然适用于归一化后的空间。

给出了三维时空以及时空距离的定义之后，下面我们再看看 3.1.1 节中 PTM 算法未解决的问题。在这里我们令时空转化因子$I_{st} = 0.92$，再由I_{st}可得到每个样本点对应的 z 值。转化后映射到三维时空如图 3.4 所示，在引入了由时间转化的维度之后，就可以很明显地看出，在时空维度下与查询轨迹 Q 最接近的是轨迹 S。因此，在三维时空下进行轨迹时空相似性查询可以有效解决 PTM 算法中时间相似性与空间相似性线性结合存在的问题。

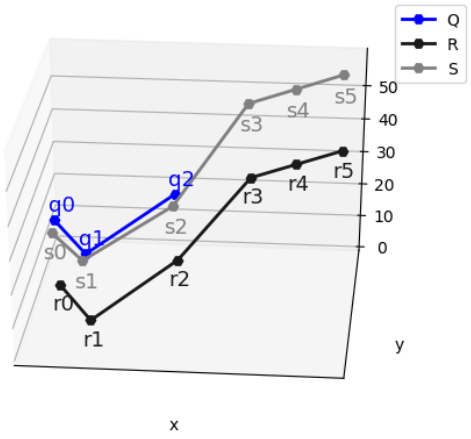


图 3.4 轨迹在三维时空的情况

Fig. 3.4 Trajectory in Three dimensional spatio-temporal

3. 3 对应点匹配算法

首先介绍一下对应点的概念，在两条轨迹 Q 和 R 的相似性计算中，若算法中使用轨迹 R 上的点 r 到轨迹 Q 上点 q 的距离代表点 r 到轨迹 Q 的距离，那么点 q 就是点 r 的对应点。不同的相似性算法，对应点有不同要求，有些算法的对应点必须是样本点，而有些算法中允许对应点为样本点之间直线段上的点。

本节主要分析之前的研究人员提出的相似性算法中的对应点匹配存在的优势与不足，以及本文中使用的对应点匹配算法如何解决这些缺点。

3.3.1BDS 算法对应点匹配分析

本小节主要介绍 Na T 等学者提出的 BDS 算法的对应点匹配方法[30]。二维空间中，给定一条查询轨迹 Q 和一条数据轨迹 R，首先找到轨迹 Q 的所有样本点到轨迹 R 上的对应点，以及轨迹 R 的所有样本点在轨迹 Q 上的对应点。BDS 算法中对应点不要求是样本点，运行一个样本点匹配到另一条轨迹中两个样本点之间的直线段上的某个点。

由于采样策略的原因，可能会导致样本点的空间分布很不均匀，如图 3.5 所示。如果使用轨迹相似性计算中的欧氏距离算法，即采用样本点一一对应的关系的话，会导致样本点匹配的时候错位很严重，不仅前后不同时间的样本点匹配错误，更会由此导致整体距离变大，不能准确的描述原本空间、时间和形状都很接近的轨迹的相似程度。

如果采用 BDS 算法寻找对应点，则会直接将样本点匹配到另一条轨迹中距离自己最近的点，如图 3.6 所示。BDS 算法的优点在于不考虑一个点的对应点必须要是样本点，减少了由于样本点位置和序列带来的限制，会更好地描述样本点之间的匹配情况。

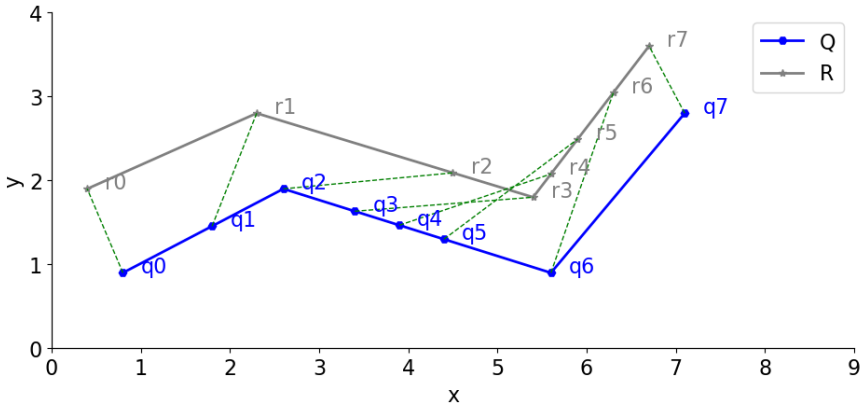


图 3.5 按时间戳顺序寻找对应点

Fig. 3.5 Finding corresponding points in timestamp order

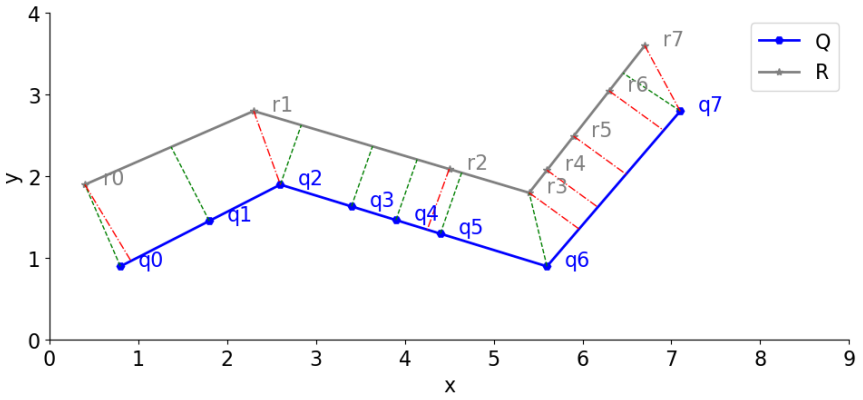


图 3.6 按最近点寻找对应点

Fig. 3.6 Finding corresponding points according to the nearest point

BDS 算法使用了一种很好的样本点匹配方法，但是在三维时空下，BDS 没有需要考虑两条轨迹上所有对应关系的时间先后顺序。下面使用一个例子来说明将 BDS 算法引入三维时空中存在的问题。

二维空间下的轨迹 Q 和 R 如图 3.7 所示，图 3.8 展示了加上时间维度的三维时空下的 Q 和 R，并在图中展示出了 BDS 算法得到的所有对应关系。图 3.8 中可以看出，r_0r_2与q_0q_1有对应关系，r_3r_5与q_3q_4有对应关系。如果从匹配结果来看，轨迹 R 的r_0至r_6都与轨迹 Q 很相似，但是中间发生了移动方向的翻转，导致了 BDS 算法对应点匹配结果中时序的错位，即r_0至r_2对应q_0至q_1，r_3至r_5对应q_4至q_3。因此 BDS 算法中，与轨迹 R 进行相似性计算的轨迹序列为$<q_0, q_1, q_2, q_4, q_3, q_5>$，而不是我们输入的查询轨迹 Q，因为轨迹数据是一个时间有序的序列，计算样本点全都相同，但是二者并不是同一条轨迹。

在考虑时间和空间的情况下，BDS 中的样本点匹配方法会导致时序错位，因此不能直接用于解决三维时空下的样本点匹配问题。

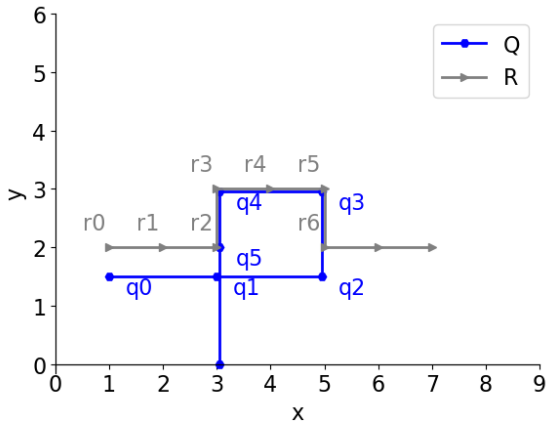


图 3.7 二维空间下的轨迹

Fig. 3.7 Trajectories in two-dimensional spaio

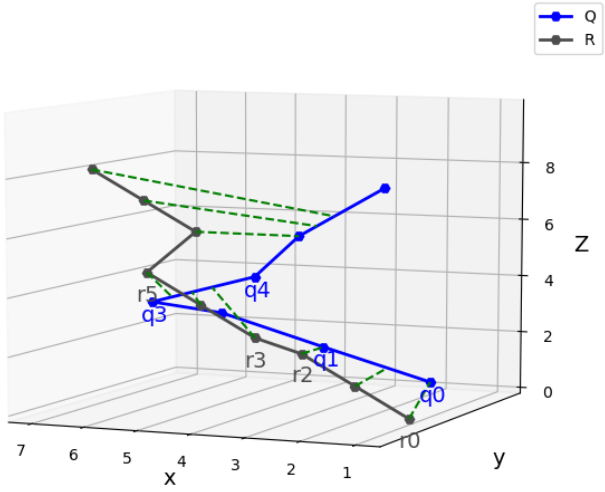


图 3.8 BDS 算法匹配结果

Fig. 3.8 Matching results of BDS algorithm

3.3.2DTW 算法对应点匹配分析

DTW[19]的计算公式如式 2.4 所示，在寻找最优的匹配策略时，DTW 采用了动态规划思想，按时间顺序从前往后匹配，不会产生 BDS 算法中的时序错位问题，并且允许进行一对多的匹配，因此对应点匹配效果优于 EU 算法[18]。

如果使用 DTW 解决刚才的问题，那么其对应关系如图 3.8 所示。样本点匹配关系为：r_0和r_1对应q_0，r_2对应q_1，r_3对应q_2，r_4和r_5对应q_3，完全按照时间先后顺序来对应，在r_3至r_5处没有出现 Q 上对应点顺序的翻转，解决了 BDS 中时序错位的问题，得到了一个相对较优的对应点匹配方案。但是由于 DTW 只允许一个样本点匹配到另一条轨迹的样本点上，因此匹配策略受采样方法和样本点提取算法的影响较大，还存在优化的空间。

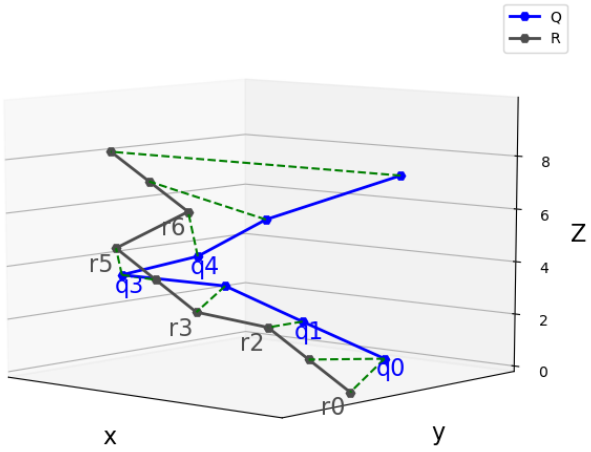


图 3.9 DTW 算法匹配结果

Fig. 3.9 Matching results of DTW algorithm

3.3.3DTW-BDS 对应点匹配算法

根据上面对 DTW 和 BDS 算法的分析，发现二者在对应点匹配中可以优劣势互补，因此本节给出 DTW 和 BDS 结合的对应点匹配方法，保留二者优势，并解决二者存在的问题。

假设轨迹 R 的样本点个数为 n，轨迹 Q 的样本点个数为 m，并且我们已经获得了所有 DTW 对应点对集合，由于 DTW 中对应点允许一对多，我们记r_i的所有 DTW 对应点为 DTW(r_i)，将 DTW(r_i)中时间戳最早的记为 DTW(r_i).first。

此外，基于 BDS 算法中对应点匹配思想，本文提出一个带上下界的 BDS 对应点匹配方法的概念。假设q_a的时间戳小于q_b的时间戳，将使用 BDS 算法思想计算r_i在$q_a q_b$间的对应点的过程记为 BDS(r_i，q_a，q_b)，这个称之为带上下界的 BDS 对应点匹配方法，时间戳较小的q_a是 BDS 匹配方法的下界，q_b是 BDS

匹配方法的上界。

我们使用图 3.10 为例进行讲解，图 3.10(a)显示的是轨迹 Q 和 R 在二维空间中的情况，使用 DTW 算法进行对应点匹配后，得到图 3.10(b)中的结果。在获得了轨迹 R 所有样本点的 DTW 对应点之后，使用 BDS 中对应点匹配的思想对其进行局部优化，将优化后r_i的点记为 DTW-BDS(r_i)。按照时间戳顺序，对 R 上所有点从前往后依次优化。

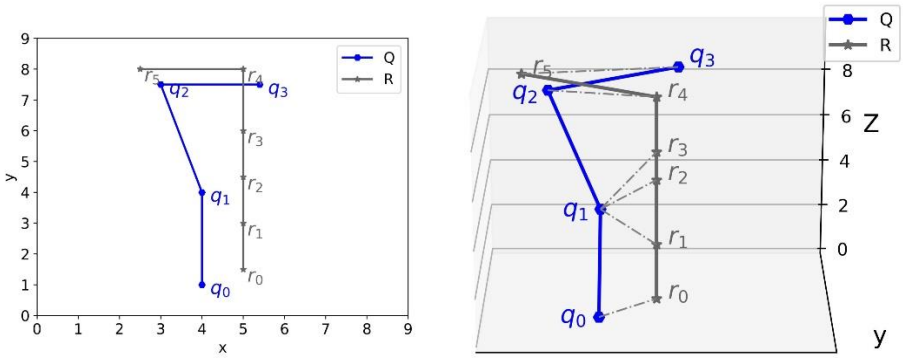


图 3.10 轨迹示例

Fig. 3.10 Example of trajectories

首先优化r_0的对应点，在q_0至 DTW(r_i).first 之间的子轨迹上求r_0的 BDS 对应点，将其作为r_0在轨迹 Q 上的 DTW-BDS 对应点，如公式 3.9 所示。如果恰好r_1的第一个 DTW 对应点 DTW(r_1).first 就是q_0，那么 BDS(r_0)=q_0。图 3.11 显示的是r_0对应点的优化。

$$\text{DTW-BDS}(r_0)=\text{BDS}(r_0,\ q_0,\ \text{DTW}(r_1).\text{first}) \tag{3.9}$$

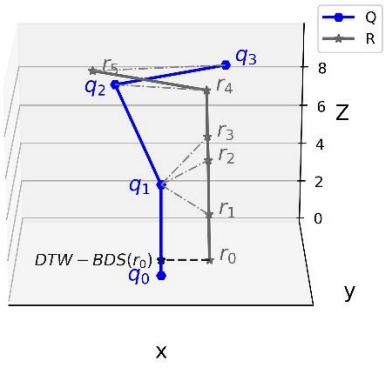


图 3.11 r_0的 BDS 优化结果

Fig. 3.11 BDS optimization result of r_0

然后优化轨迹 R 中间的样本点的对应点。当r_i是轨迹 R 中间的某个点，且在前面已经获得了r_{i-1}的对应点 DTW-BDS(r_{i-1})，可以求r_i在 DTW-BDS(r_{i-1})与 DTW(r_{i+1}).first 之间的 BDS 对应点，将其作为r_i的 DTW-BDS 对应点，如公式 3.10 所示。图 3.12(a)显示了r_1、r_2、r_3对应点的优化。

DTW-BDS(r_i)=BDS(r_i, DTW-BDS(r_{i-1}), DTW(r_{i+1}).first) (3.10)

在获得了 DTW-BDS(r_i)后，如果 DTW-BDS(r_i)的时间戳比 DTW(r_i).first 大，此时需要重新计算r_{i-1}的 DTW-BDS 对应点，本文将这个过程叫做前向更新。由于样本点的 DTW-BDS 对应点是按照样本点时间戳顺序依次计算，在求 DTW-BDS(r_{i-1})的时候，还没有求 DTW-BDS(r_i)，因此使用 DTW(r_{i+1}).first 作为上界。当求出 DTW-BDS(r_i)后，若发现 DTW(r_i).first 的时间戳小于 DTW-BDS(r_i)的时间戳，r_{i-1}的 DTW-BDS 对应点的上界应该要增大为 DTW-BDS(r_i)，然后再次使用带上下界的 BDS 算法更新 DTW-BDS(r_{i-1})。若r_{i-1}更新后的对应点与更新前不是同一个点，还需要循环更新前一个点，一直更新到某个点的对应点在更新前后相同为止，或一直更新到第一个点。

如图 3.12(a)所示，已经计算得到了 DTW-BDS(r_3)，发现 DTW-BDS(r_3)的时间戳大于 DTW (r_3)即q_1的时间戳，此时需要进行前向更新操作。首先更新 DTW-BDS(r_2)，将其上界定为 DTW-BDS(r_3)，通过上下界的 BDS 对应点计算，得到了图 3.12(b)中的 DTW-BDS(r_2)。由于更新后的 DTW-BDS(r_2)与更新前不是同一个点，需要以 DTW-BDS(r_2)为上界更新 DTW-BDS(r_1)，发现 DTW-BDS(r_1)更新前后为同一个点，因此停止前向更新，最后结果如图 3.12(b)所示。

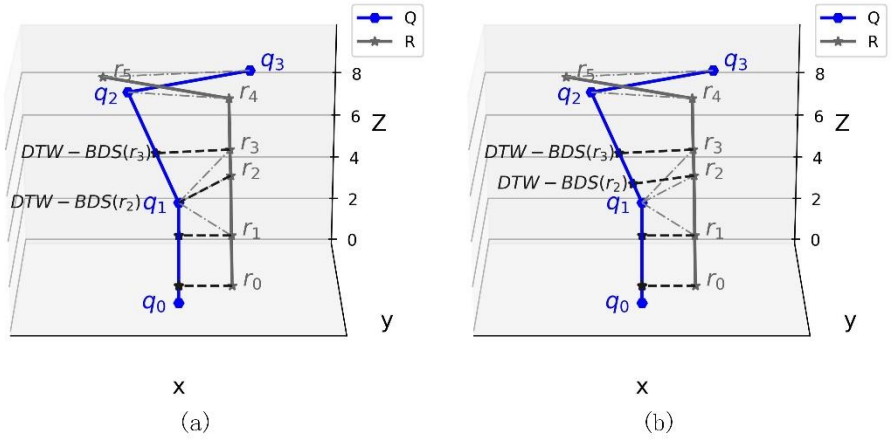


图 3.12 中间对应点的优化

Fig. 3.12 Optimization of corresponding points in the middle

最后对 R 的最后一个点r_{n-1}的对应点进行优化，求出r_{n-1}在 BDS(r_{i-1})与轨迹 Q 的最后一个点q_m之间的 BDS 对应点作为r_{n-1}的 DTW-BDS 对应点，如公式 3.11 所示。最后完整的 DTW-BDS 算法得到的匹配结果如图 3.13 所示。

DTW-BDS(r_{n-1})=BDS(r_{n-1}, BDS(r_{i-1}), q_m) (3.11)

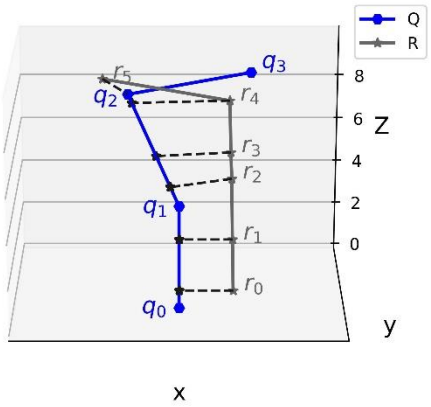


图 3.13 DTW-BDS 算法最终匹配结果

Fig. 3.13 Final match result of DTW-BDS algorithm

通过对轨迹 R 的所有 DTW 对应点按照时间戳顺序使用 BDS 匹配算法做局部优化，可以获得比 DTW 和 BDS 更优的匹配方案。算法 3.2 是 DTW-BDS 对应点匹配算法的伪代码。

算法 3.2 DTW-BDS 对应点匹配算法 DTW_BDS_match(Q,R)

输入：查询轨迹 Q，数据轨迹 R

输出：R 在 Q 上的 DTW-BDS 对应点对 pair

```
pair ← DTW 算法匹配结果
for each r_i in R
    if r_i 是 R 的第一个点
        Q(r_i)= BDS(r_0, q_0, DTW(r_1).first)
    else if r_i 是 R 中间的点
        Q(r_i)= BDS(r_i, DTW-BDS(r_{i-1}), DTW(r_{i+1}).first)
    else
        Q(r_i)= BDS(r_{n-1}, BDS(r_{i-1}), q_m)
    r = r_i
    q_old = pair.getValue(r_i)
    q_new = Q(r_i)
    pair ← (r_i, Q(r_i))
    while q_new.timestamp > q_old.timestamp and r.pre != null
        r = r.pre
        q_old = pair.getValue(r)
        q_new = BDS(r, DTW-BDS(r.pre), DTW-BDS(r.next))
        pair ← (r, q_new)
    end while
end for
```

现在再看本文 3.3.1 节图 3.7 的例子，如果使用 DTW-BDS 对应点匹配算法，先获得轨迹 R 中所有样本点的 DTW 对应点，再按照时间戳顺序依次为每个样本点r_i寻找 DTW-BDS 对应点，可以得到如图 3.14 所示的优化结果。DTW-BDS 算法带来了两个好处。

第一个好处是优化了 DTW 的样本点匹配方法。样本点r_1原本匹配到了q_0，通过局部优化，找到了距离q_0和q_1之间距离r_1最近的点最为对应点，此时r_1到 DTW-BDS(r_1)的距离小于r_1到 DTW(r_1)的距离。除了r_1之外，还有r_3、r_5、r_7和r_8不再对应到轨迹 Q 的样本点上了，而是借助 BDS 算法中的思想，寻找轨迹 Q 上与各自最近的点作为对应点，进一步减小了对应点之间的距离。而r_0、r_2、r_4和r_6的对应点在经过优化之后仍然为原来的 DTW 对应点。因此使用这个方法摆脱了 DTW 算法中要求的对应点只能对应到另一条轨迹上样本点的限制，减小了对采样策略和样本点提取算法的敏感程度，进一步优化了 DTW 匹配结果，减小了对应点整体上的距离，从而使轨迹间的对应关系更准确。

第二个好处虽然使用 BDS 算法中对应点匹配的思想，但是保证了匹配结果的时序性。在 BDS 算法中，r_3r_5匹配到了q_4q_3，出现了时序错位。但在 DTW-BDS 算法中，先使用 DTW 算法确定了最终对应点可能的范围，再对轨迹 R 中每个样本点按照时间戳顺序，在 DTW-BDS(r_{i-1})与 DTW(r_{i+1})确定的范围内寻找r_i的 BDS 对应点，因此不会出现时序错乱的问题，保证匹配结果的时序性。因此 DTW-BDS 算法是行之有效的。

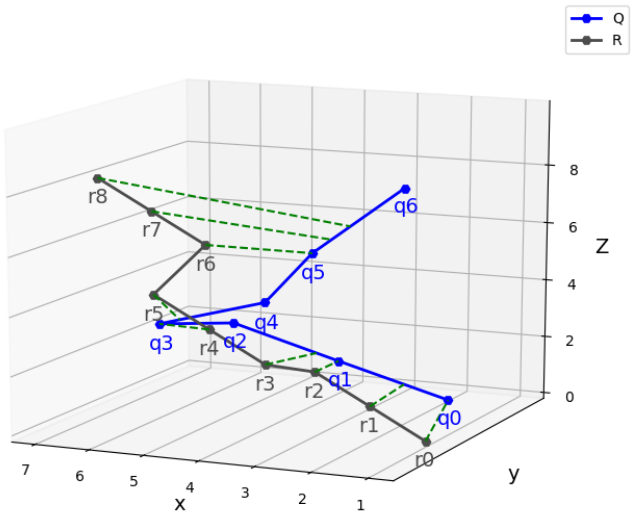


图 3.14 DTW-BDS 算法匹配结果

Fig. 3.14 Matching results of DTW-BDS algorithm

由于本文后面使用到的对应点均为 DTW-BDS 算法寻找到的对应点，为方便起见，我们将r_i在轨迹 Q 上的 DTW-BDS 对应点 DTW-BDS(r_i)记为$Q(r_i)$。

3.3.4 对应轨迹段

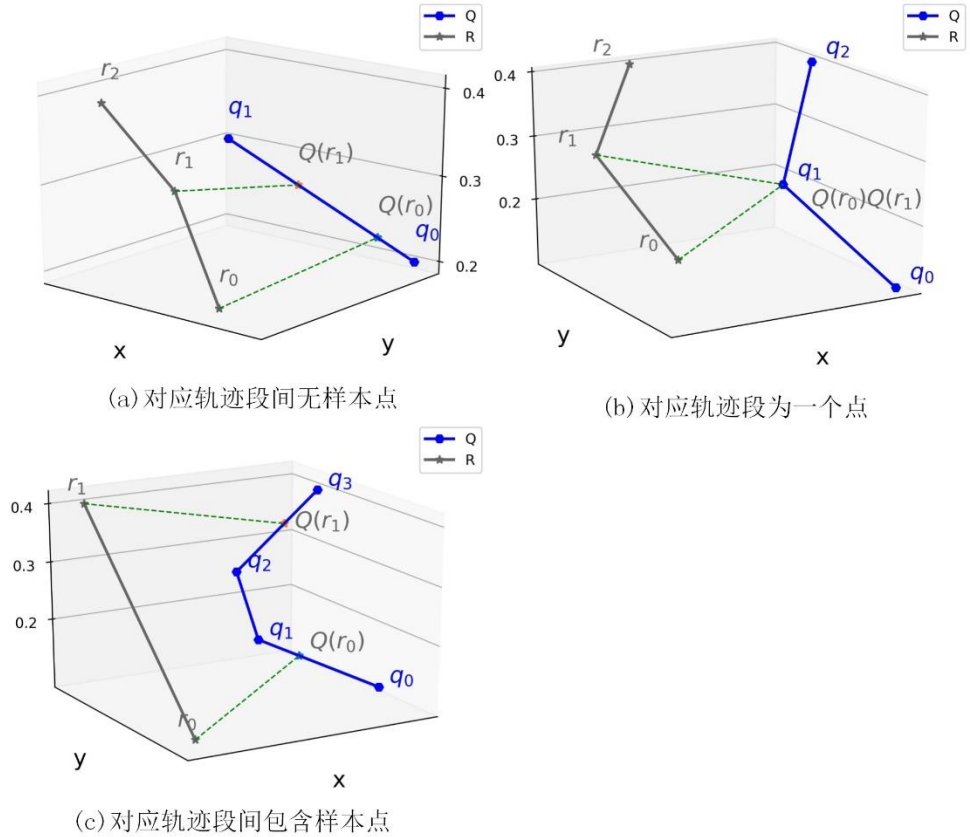
在获得对应样本点的基础上，我们可以得到对应轨迹段的概念。数据轨迹 R 上的连续两个样本点r_i和r_{i+1}，会在查询轨迹 Q 上分别获得他们的对应点$Q(r_i)$和$Q(r_{i+1})$，那么轨迹段r_ir_{i+1}的对应轨迹段就是$Q(r_i)Q(r_{i+1})$。轨迹段 r_i r_(i+1)的对应轨迹段会出现多种情况，后面对每种情况的对应轨迹段的处理方

式不尽相同，下面给出对应轨迹段之间的几种分布情况。

第一种情况是$Q(r_i)$和$Q(r_{i+1})$不是同一个点，并且除了$Q(r_i)$和$Q(r_{i+1})$之外，$Q(r_i)Q(r_{i+1})$轨迹段中不包含任何一个样本点，而$Q(r_i)$和$Q(r_{i+1})$有可能是轨迹 Q 的样本点，也有可能是轨迹 Q 中某两个样本点连线上的点。如图 3.15(a)所示，r_1的对应点$Q(r_1)$是轨迹 Q 的样本点q_1，r_2的对应点$Q(r_2)$在样本点q_1和q_2之间。

第二种情况是由于数据轨迹 Q 移动速度和移动距离等因素，r_i和r_{i+1}可能会对齐到 Q 上的同一个点，即$Q(r_i)$和$Q(r_{i+1})$有可能是轨迹 Q 中的同一个样本点，如图 3.15(b)所示，r_1和r_2的对应点$Q(r_1)$和$Q(r_2)$为同一个点，并且该点还是轨迹 Q 的样本点。

第三种情况是虽然r_i和r_{i+1}是两个连续的样本点，但是$Q(r_i)$和$Q(r_{i+1})$中间隔着一个或多个样本点。图 3.15(c)是第三种情况中的一个例子，r_1和r_2的对应点$Q(r_1)$和$Q(r_2)$中间夹着轨迹 Q 的样本点q_1和q_2。



(a) 对应轨迹段间无样本点

(b) 对应轨迹段为一个点

(c) 对应轨迹段间包含样本点

图 3.15 对应轨迹段的三种情况

Fig. 3.15 Three cases corresponding to track segments

以上就是对应轨迹段之间存在的三种时空分布情况，本文后面的内容将根据不同的分布情况来确定不同情况的轨迹段之间距离的计算方法。

3.4 本章小结

本章首先介绍了本篇论文的问题定义以及相关应用场景。第二节中指出前人对轨迹相似性的研究在时空维度结合上存在的问题，提出了时空归一化方法，由二维欧式空间和一维时间构造出三维时空来解决时空结合的问题。在第三节中分析了三维时空下 DTW 算法和 BDS 算法中对应点匹配存在的优点与缺点，本文将两个算法中对应点匹配的思想相结合，提出了三维时空下的 DTW-BDS 对应点匹配算法，用于计算得到样本点在另一条轨迹上的对应点。在得到数据轨迹 R 上样本点在查询轨迹 Q 上的对应点之后，我们给出了对应轨迹段的概念，并分析了可能出现的集中对应轨迹段的情况。

第四章 时空轨迹相似性算法

第三章中提出了结合时间维度与空间维度的三维时空，然后通过算法得到三维空间下的 DTW-BDS 算法进行对应点匹配，并由对应点得到的对应轨迹段。本章将介绍对应轨迹段之间的时空距离以及形状相似性，最后获得对应轨迹段的距离。然后基于轨迹段距离得到轨迹相似性算法。

在三维时空中，数据轨迹段$r_i r_{i+1}$和对应的查询轨迹段$Q(r_i)Q(r_{i+1})$之间的相似性由两部分组成，第一部分是轨迹段之间的时空距离，第二部分是轨迹段间形状上的相似性。下面依次对这两部分进行讲解。

4.1 对应轨迹段的时空距离

4.1.1DTW 算法空间距离计算中存在的问题

前人对轨迹在空间距离上的计算大多从对应样本点的距离入手，最典型的是 DTW 算法[19]。DTW 算法中是使用动态规划思想先找到最优的对应点匹配，然后计算所有对应点的距离之和，作为轨迹之间的距离。该方法的好处是通过允许对应点间的“一对多”，改进了欧氏距离算法[17][18]中样本点“一对一”的限制，解决了局部时间偏移的问题。不足之处在于 DTW 无论是在对应点匹配过程中，还是在计算轨迹间距离时均完全依赖样本点进行计算，计算结果对采样策略比较敏感，可能不同的采样策略会造成完全相反的相似性计算结果。

本文的三维时空中包含的是二维空间和一维时间，其距离计算方法与三维空间中的距离计算完全相同。因此下面给出使用 DTW 算法计算三维时空中两条轨迹距离的例子。

给定三个物体的运动轨迹，包括查询轨迹 Q 和数据轨迹 R 和 S，得到了如图 4.1(a)所示的时空轨迹数据。这是三条比较简单的轨迹，该采样设备通过分布较为稀疏的样本点，表达了三个物体的移动路径。使用 DTW 算法分别对查询轨迹 Q 和数据轨迹 R 以及 S 进行相似性计算，得到的 DTW 相似性矩阵如图 4.2(a)和图 4.2(b)所示。根据 DTW 距离矩阵，我们可以得到查询轨迹 Q 和数据轨迹 R 的 DTW 距离为 5，Q 和数据轨迹 S 的 DTW 距离为 5.5，因此，本次计算结果显示 Q 和 R 的距离更小，相似度更高。

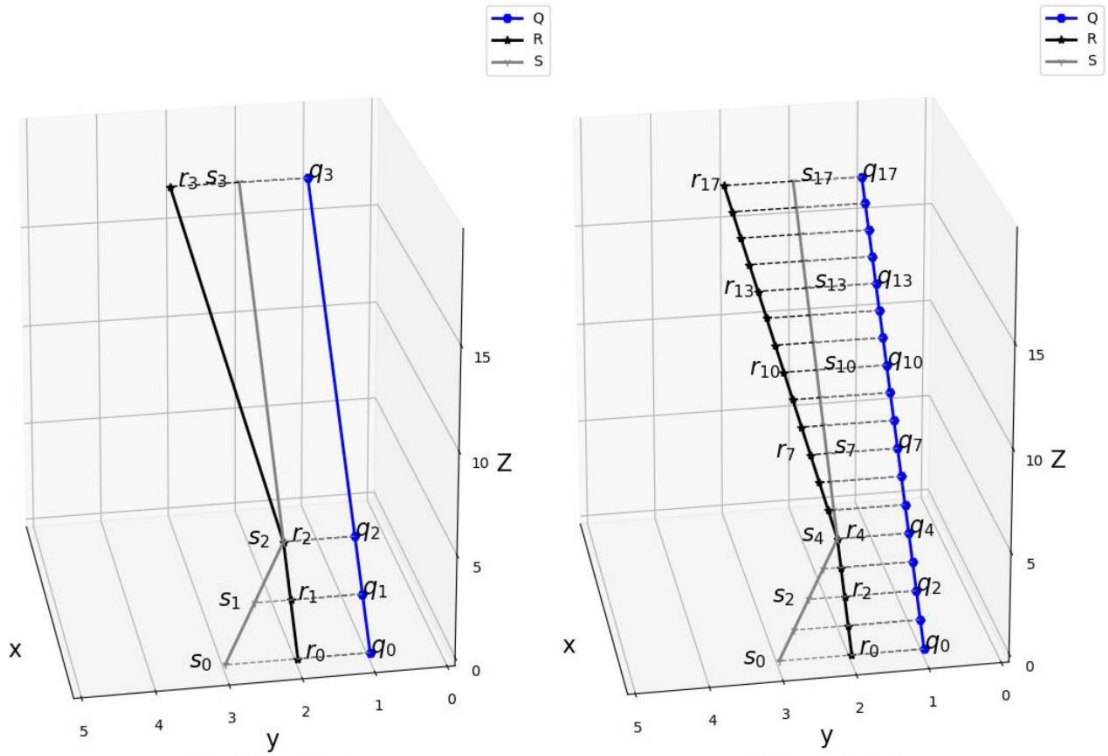


图 4.1 DTW 算法计算轨迹距离示例

Fig. 4.1 Example of DTW algorithm to calculate trajectory distance

	0	q_0	q_1	q_2	q_3
0	0	+∞	+∞	+∞	+∞
q_0	+∞	1	3.24	7.36	24.48
q_1	+∞	3.24	2	4.24	19.37
q_2	+∞	7.36	4.24	3	16.15
q_3	+∞	24.39	19.27	16.04	5

(a) Q和R的相似性矩阵

	0	q_0	q_1	q_2	q_3
0	0	+∞	+∞	+∞	+∞
q_0	+∞	2	4.5	8.62	25.65
q_1	+∞	4.83	3.5	5.74	20.77
q_2	+∞	9.30	6	4.5	17.54
q_3	+∞	26.42	21.07	17.54	5.5

(b) Q和S的相似性矩阵

图 4.2 DTW 相似性矩阵

Fig. 4.2 DTW similarity matrix

如果使用另外一套采样策略不同的采样设备，采集刚才三个移动对象的轨迹，可能会得到样本点较密集的轨迹数据，如图 4.1(b)所示。虽然样本点比刚才的轨迹数据更多，但是可以从坐标上看出，描述的还是刚才三个物体的运动，包括起点终点和转折点。再使用 DTW 算法，分别计算出 Q 和 R 的距离，以及 Q 和 S 的距离。得到查询轨迹 Q 和数据轨迹 R 的 DTW 距离为 25，查询轨迹 Q 和数据轨迹 S 的 DTW 距离为 20.5。本次计算结果显示，Q 和 S 的距离更小，相似程度更高。

上面的例子表明，DTW 算法对采样策略的敏感程度较高，对于使用不同采样策略得到的轨迹数据进行计算，可能会得到不同的结论。根本原因在于 DTW 算法对样本点过于依赖，忽略了采样策略造成的影响。

移动对象的原始移动轨迹应该是一条连续的曲线。假设使用一个采样频率无穷大的设备对移动对象的轨迹进行采样，就会得到无穷多个采样点，这些采样点的集合就是移动对象原始的移动曲线，而获取到的轨迹数据中有限个样本点连接而成的折线只是对原始移动轨迹的一个近似表示。因此采样频率越高，在轨迹相似性计算中提供的信息越多，越有助于还原原始轨迹的时空特征。图4.1(b)中的数据反映的信息更多，DTW 算法的计算结果相对更准确，因此实际上 Q 和 S 更相似。

DTW 算法在计算采样频率较低的数据时可能会产生较大误差，DTW 算法在计算的时候只考虑了每一个样本点的时空信息，忽略了样本点之间的轨迹段的时空信息，会造成大量信息损失。

4.1.2 断点及其对应点

基于以上讨论，本文提出断点（break point）的概念。使用一个固定的断点阈值η将两个相邻样本点之间的轨迹段均匀分割，当阈值η小于轨迹段长度时，轨迹段会被分割为多段，将分割轨迹段的点称为断点。断点是轨迹段上的点，同一条轨迹段上的断点与断点之间距离均为η，由于轨迹段存在于三维时空，因此断点阈值η描述的是断点间的时空距离。

一条轨迹段中断点的个数取决于轨迹段长度以及η的大小。在轨迹段$r_i r_{i+1}$上，从样本点r_i开始，每隔η距离取一个断点bp_k，由于轨迹段的长度可能不是η的整数倍，最后一个断点到r_{i+1}的距离可能会小于η，最终会获得$\lceil d(r_i, r_{i+1})/\eta \rceil$条轨迹段，以及轨迹段$r_i r_{i+1}$上所有断点的集合 $BP(r_i r_{i+1})$。如图 4.3(a)所示，$r_0 r_1$是查询轨迹 R 中的一条轨迹段，断点阈值η将轨迹段$r_0 r_1$划分为四小段，得到了六个断点，其中前三段$r_0 bp_0$一直到$bp_4 bp_5$的长度均为η，而末尾的小段$bp_5 r_1$的长度小于η。

通过调整阈值η的大小，将一条长轨迹段$r_i r_{i+1}$可以分隔为更短的轨迹段。而由 4.1.1 节分析得知，通过计算采样率较高的轨迹的信息，可以更准确地描述原始长轨迹段$r_i r_{i+1}$与其对应轨迹段间的距离。断点的作用就是通过人为增加采样点，更细粒度地考虑轨迹段$r_i r_{i+1}$的时空特征，从而使相似性计算结果更准确。

为了得到对应轨迹段的时空距离，需要获得断点在另一条轨迹上的对应点。轨迹段$r_i r_{i+1}$是相邻样本点连接而成的一条直线段，但是其对应轨迹段$Q(r_i)Q(r_{i+1})$存在图 4.3 所示的三种情况，这三种情况下断点的对应点的求法不完全相同。

对于图 4.3(b)所示的第二种情况，对应轨迹段是一个点，由于时空数据的样本点匹配关系要有时序性，因此$r_i r_{i+1}$上所有断点的对应点都是$Q(r_i)$。

对于图 4.3(a)所示的第一种情况和图 4.3(c)所示的第三种情况，虽然轨迹段$r_i r_{i+1}$上的所有断点在同一条直线上，但是其对应轨迹段$Q(r_i)Q(r_{i+1})$可能存在复杂的移动情况，如果直接使用 BDS 算法会导致匹配结果时序错乱。也不能直接使用 DTW 算法进行样本点匹配，因为断点存在于两个样本点r_i和r_{i+1}之间，寻找的是断点到对应轨迹段的对应点，若对应轨迹段中样本点个数极少，DTW 算法的匹配结果受样本点限制很大，不能达到寻找断点的对应点的目的。因此采用本文前面提出的 DTW-BDS 对应点匹配算法，可以避免时序错乱或者受样本点限制等缺陷。

下面介绍$r_i r_{i+1}$上所有断点的匹配过程。首先将$r_i r_{i+1}$和其对应轨迹段$Q(r_i)Q(r_{i+1})$视作完整的轨迹，作为 DTW-BDS 算法的输入，并将$r_i r_{i+1}$中的所有断点以及$Q(r_i)$和$Q(r_{i+1})$视为样本点。算法输出为的r_i和r_{i+1}的对应点以及$r_i r_{i+1}$上所有断点的对应点，保存所有断点的对应点，并将断点bp_k的对应点记作$Q(bp_k)$。

4.1.3 轨迹段间的时空距离

本小节介绍轨迹段间时空距离的计算。由于$r_i r_{i+1}$对应的$Q(r_i)Q(r_{i+1})$存在三种空间分布情况，在计算其时空距离时需要分别进行考虑。

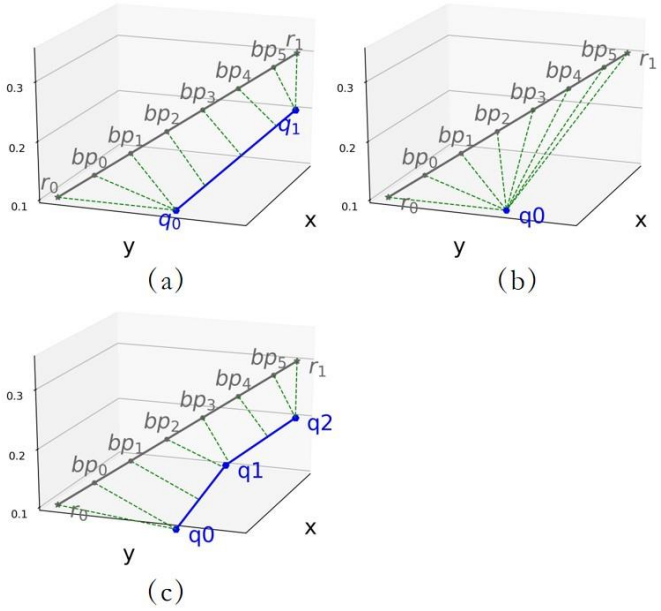


图 4.3 不同对应轨迹段中断点的对应点

Fig. 4.3 Corresponding points of break points in different corresponding track segments

先讨论第一种情况，即图 4.3(a)中所展示的轨迹段$Q(r_i)Q(r_{i+1})$是一条直线段。首先根据断点阈值η计算得到$r_i r_{i+1}$上的所有断点，然后根据上一节中给出的方法，使用 DTW-BDS 样本点匹配算法，将$r_i r_{i+1}$和$Q(r_i)Q(r_{i+1})$作为输入，找到所有断点的对应点。我们可以计算每一个断点与其对应点之间的时空距

离，得到$d(r_i, Q(r_i))$和$d(bp_k, Q(bp_k))$。

下面的问题就是如何将轨迹段$r_i r_{i+1}$上得到的这些距离转换成一个可以表示轨迹段间距离的数值。如果η趋向于无穷小，那么会在轨迹段$r_i r_{i+1}$上得到无穷多个断点，相邻断点之间的间距趋向于无穷小，相邻断点到对应点的距离近似相等，如图 4.4(a)所示。

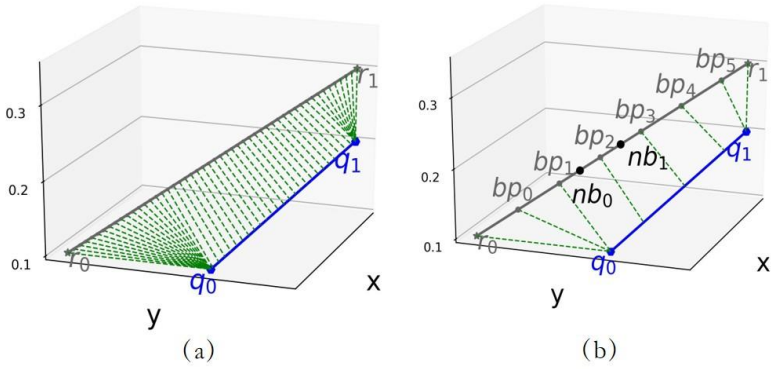


图 4.4 断点代表的段

Fig. 4.4 Segment represented by breakpoint

如果将η增大一点，就变成图 4.4(b)中的情况，因为η增大而消失了很多断点，但是由于剩下断点到对应点的距离近似等于与其相邻的消失断点的到对应点的距离，所以在一定精度范围内，可以使用剩下断点代替消失断点，而每个断点能代替的对应点的个数，与断点阈值η成正比。

在图 4.4(b)中，可以使用bp_2代替从nb_0到nb_1间的所有消失的断点，nb_0与nb_1分别是断点bp_2到相邻断点的中点。断点bp_2代表的消失断点数目与$nb_0 nb_1$的长度成正比，将$nb_0 nb_1$称为断点bp_2代表的段。左端点r_0代表的段的长度为$\eta/2$，断点bp_0到bp_4代表的段的长度均为断点阈值η，bp_5代表的段的长度为$[\eta + \mathrm{d}(bp_5, r_1)]/2$，右端点$r_1$代表的段的长度为$\mathrm{d}(bp_5, r_1)/2$。可以同时消去所有代表段的长度中的η，可以得到$r_0$的权值为 1/2，$bp_0$到$bp_4$的权值均为 1，$bp_5$权值为$[\eta + \mathrm{d}(bp_5, r_1)]/2\eta$，$r_1$权值为$\mathrm{d}(bp_5, r_1)/2\eta$。

根据端点和断点到对应点的距离以及各自权值，下面给出第一种情况下对应轨迹段$r_i r_{i+1}$到对应轨迹段$Q(r_i)Q(r_{i+1})$的距离公式，如公式 4.1 所示。

$$d_{st}\left(r_i r_{i+1}, Q(r_i)Q(r_{i+1})\right) = \begin{cases} \frac{1}{2} d(r_i, Q(r_i)) + \sum_{j=0}^{n-2} d\left(bp_j, Q(bp_j)\right) \\ + \frac{1}{2}\left(1 + \frac{d(bp_{n-1}, r_{i+1})}{\eta}\right) d\left(bp_{n-1}, \mathrm{Q}(bp_{n-1})\right) \\ + \frac{d(bp_{n-1}, r_{i+1})}{2\eta} d(r_{i+1}, Q(r_{i+1})) \ , if \ \ n \geq 1 \\ \frac{d(r_i, r_{i+1})}{2\eta} \times [d(r_i, Q(r_i)) + d(r_{i+1}, Q(r_{i+1}))] \ , if \ \ n = 1 \end{cases} \tag{4.1}$$

其中$\mathrm{n} = \lfloor d(r_i, r_{i+1})/\eta \rfloor$，为断点的个数。

当出现第二种空间分布情况，即$Q(r_i)$与$Q(r_{i+1})$为同一个点，如图 4.3(b)所示，依然可以使用上述方法，只不过这里每个端点和断点的对应点都是$Q(r_i)$，

省去了求断点对应点的步骤。对应轨迹段第三种空间分布情况只是对应轨迹段为折线段，与第一种情况下的轨迹段时空距离的求法没有区别。

下面给出对应轨迹段时空距离计算的伪代码。

算法 4.1 轨迹段时空距离计算 spatio_temporal_distance_calculate(Q, R, pair, η)

输入：数据轨迹 R，查询轨迹 Q，R 的对应点对 pair，断点距离阈值η

输出：数据轨迹所有轨迹段$r_i r_{i+1}$到对应轨迹段的时空距离 dss_list

1. **for** $r_i r_{i+1} \in$ R
2. **if** $d(r_i, r_{i+1}) > \eta$
3. 使用参数η切分轨迹段$r_i r_{i+1}$，获得所有断点
4. 从 pair 中获取$Q(r_i)$和$Q(r_{i+1})$
5. 计算断点对应点$Q(bp_j) = BDS(bp_j, Q(r_i), Q(r_{i+1}))$
6. d=$d(r_i, Q(r_i))$，$d(bp_j, Q(bp_j))_{j=0,n-1}$，$d(r_{i+1}, Q(r_{i+1}))$
7. w=[1/2, 1, 1…1, $\frac{1}{2}\left(1+\frac{d(bp_{n-1}, r_{i+1})}{\eta}\right)$, $\frac{d(bp_{n-1}, r_{i+1})}{2\eta}$]
8. **else if** $d(r_i, r_{i+1}) \le \eta$
9. d=$d(r_i, Q(r_i))$，$d(r_{i+1}, Q(r_{i+1}))$
10. w = $[\frac{d(r_i, r_{i+1})}{2\eta}, \frac{d(r_i, r_{i+1})}{2\eta}]$
11. dss _list ← $\sum d * w$
12. **end for**
13. **return** dss _list

本文提出的计算对应轨迹段之间的距离的方法解决了本节开始提出的 DTW 中存在的问题。DTW 中的问题根源在于依赖轨迹数据中保留的采样点来计算轨迹间距离过于依赖，完全忽视了样本点间隔给轨迹间距离带来的影响，如果使用不同的采样策略或者不同的兴趣点转折点提取方法，会对最后的相似性结果产生很大影响，甚至产生截然相反的结论，比如前面的例子。而本文中的方法不仅仅会考虑轨迹中对应点之间的距离，还考虑到了相邻样本点之间的轨迹段的长度带来的影响。提出了断点的概念，让查询轨迹段包含更多点，断点的个数与轨迹段长度呈正相关，然后考虑轨迹段端点和所有断点到其对应点的距离，虽然仍然是求对应点之间的距离，但是断点在设置的时候，考虑到了轨迹段的因素。这种方法更多的包含了轨迹段的信息在里面，最后的计算结果也会更加准确。

4.2 基于余弦距离的形状影响因子

轨迹段形状相似性和轨迹段之间的夹角以及轨迹段的长度相关，轨迹段之间夹角越小的情况下，两条轨迹段长度越长，那么轨迹段形状越相似。下面引入余弦距离来计算轨迹段之间的形状上的相似程度，将从夹角和轨迹段长度两

方面进行讨论。

4.2.1 余弦距离

三维时空中，两条轨迹段可能存在异面的情况，如图 4.5 所示。这里需要利用三维空间中的向量去求两个异面线段的夹角的余弦距离。三维空间中向量$\vec{a}$和向量$\vec{b}$的余弦距离[35][36]如公式 4.2 所示，其中$|\vec{a}|$和$|\vec{b}|$表示向量模长。在轨迹段余弦距离中，θ的取值范围为[0, π]，cos(θ)的取值范围为[-1,1]。

$$\cos(\theta) = \frac{\vec{a}*\vec{b}}{|\vec{a}|*|\vec{b}|}, \theta \in [0, \pi] \tag{4.2}$$

如果把上面的公式做一下变换，两边同时乘以向量$\vec{a}$的模长$|\vec{a}|$，如公式 4.3 所示，得到的是映射后的a'的长度，a'为$\vec{a}$在$\vec{b}$方向上的映射。

$$|a'| = |\vec{a}|\cos(\theta) = \frac{\vec{a}*\vec{b}}{|\vec{b}|} \tag{4.3}$$

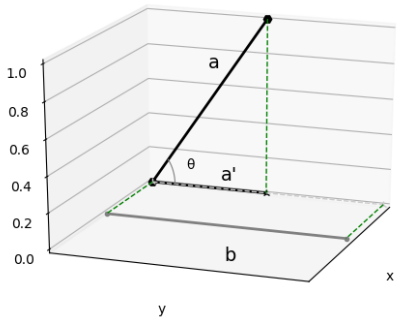


图 4.5 异面直线的夹角

Fig. 4.5 Angle between different straight lines

由于余弦距离$\cos(\theta)$在$\theta \in [0, \pi]$中是一个单调递减的函数，因此$\cos(\theta)$的值随着两条轨迹间夹角变大而减小，可以用来衡量两条轨迹段夹角的差异，而夹角的差异代表着方向和形状的差异，因此可以使用余弦距离将两条轨迹段在形状上的差异性进行量化。

4.2.2 轨迹段形状相似性

在图 4.6(a)中，OA 与 x 轴的夹角大于 OB 与 x 轴的夹角，虽然 OA 和 OB 长度相同，但是投影后的OA'的长度小于OB'，反映了夹角越小，形状越相似。在图 4.6(b)中，OA 和 OB 与 x 轴的夹角相同，但是 OA 更长，导致OA'的长度大于OB'，可以表示出夹角相同的情况下，长度越长，形状越相似。因此两条轨迹段的形状相似与夹角和轨迹段长度相关。

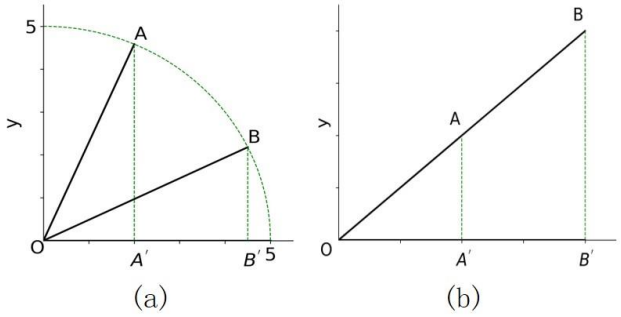


图 4.6 影响形状相似性的两个因素

Fig. 4.6 Two factors affecting shape similarity

我们首先仍然假设两条轨迹段均为直线段，即如图所示的第一种情况。在获得对应轨迹段之间的夹角和轨迹段的长度后，我们可以使用$d(r_i, r_{i+1}) \times \cos(\overrightarrow{r_i r_{i+1}}, \overrightarrow{Q(r_i)Q(r_{i+1})})$表示轨迹段$r_i r_{i+1}$与其对应的直线轨迹段$Q(r_i)Q(r_{i+1})$的形状相似性。

在轨迹段之间夹角很小的情况下，在一定程度上，数据轨迹段$r_i r_{i+1}$的长度越长，代表两条轨迹越相似。但是如果超过了这个程度，即$r_i r_{i+1}$比$Q(r_i)Q(r_{i+1})$的长度大很多倍时，对应轨迹段之间的相似性如果再随着轨迹段$r_i r_{i+1}$长度的增长而增加，就不符合我们对轨迹形状上的相似性的要求了，因为在轨迹段形状相似性的要求是，在小锐角的情况下，两条轨迹段的长度越长越相似。因此不能仅根据一条轨迹段的长度边长去延伸。也就是说对数据轨迹段长度的激励需要有一个限制。

而依据余弦距离的几何意义，$d(r_i, r_{i+1}) \times \cos(\overrightarrow{r_i r_{i+1}}, \overrightarrow{Q(r_i)Q(r_{i+1})})$为$\overrightarrow{r_i r_{i+1}}$映射到$\overrightarrow{Q(r_i)Q(r_{i+1})}$方向上的距离，这个距离带有方向性，如果夹角为钝角，距离就是负数。使用余弦距离投影后，如果$r_i r_{i+1}$距离特别长而对应轨迹段的距离特别短，投影结果不能反映轨迹段之间长度的差异，为了抑制这种情况，在轨迹段形状相似性计算时，我们将$Q(r_i)Q(r_{i+1})$的长度$d(Q(r_i), Q(r_{i+1}))$规定为轨迹段形状相似性的上限，即形状相似性$sim_{shape}(r_i r_{i+1}, Q(r_i)Q(r_{i+1}))$允许的最大激励不得超过轨迹段$Q(r_i)Q(r_{i+1})$的长度，再大的部分我们认为是无效部分。

根据上面的讨论，这里给出$r_i r_{i+1}$与对应轨迹段$Q(r_i)Q(r_{i+1})$的轨迹段形状相似性$sim_{shape}(r_i r_{i+1}, Q(r_i)Q(r_{i+1}))$的计算公式，如公式 4.4 所示。由公式可以看出，轨迹段之间夹角为一个小锐角时，两条轨迹段长度越长，二者相似距离越大。

$$sim_{shape}(r_i r_{i+1}, Q(r_i)Q(r_{i+1})) = min\begin{Bmatrix} d(r_i, r_{i+1}) * \cos(\overrightarrow{r_i r_{i+1}}, \overrightarrow{Q(r_i)Q(r_{i+1})}) \\ d(Q(r_i), Q(r_{i+1})) \end{Bmatrix} \quad (4.4)$$

其中，$\varepsilon \in (0,1)$。

上面讨论了轨迹段$r_i r_{i+1}$的对应轨迹段$Q(r_i)Q(r_{i+1})$是直线段的情况，我们还需要对另外两种情况进行讨论。

当出现图 4.2(b)中的情况，即$Q(r_i)$和$Q(r_{i+1})$是轨迹 Q 中的同一个样本点，由于一个点无法获得其方向，而考虑到实际情况中，一个转折点可以看做是前面一段轨迹到后面一段轨迹的过渡，因此可以将该点前后两端轨迹的平均方向看做该点处的方向，如图 4.7 所示，点$Q(r_i)$处的方向为$Q(r_i)Q(r_i)'$。

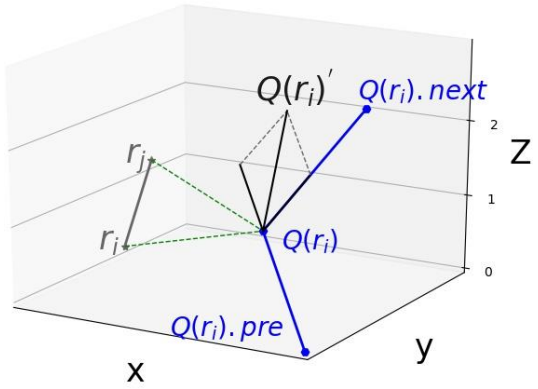


图 4.7 $Q(r_i)$的方向

Fig. 4.7 The direction of $Q(r_i)$

计算具体的值时，首先需要找到轨迹中与$Q(r_i)$相邻的两个样本点，分别记为$Q(r_i).pre$和$Q(r_i).next$，$\overrightarrow{Q(r_i)}$是$Q(r_i)$处的向量，$\overrightarrow{Q(r_i)}$的值为前后两个轨迹段的单位向量的和，此时$\overrightarrow{Q(r_i)}$的方向就是前后两个方向的均值，如公式 4.5 所示，$\frac{\overrightarrow{Q(r_i).pre\ \ Q(r_i)}}{\left|\overrightarrow{Q(r_i).pre\ \ Q(r_i)}\right|}$表示向量$\overrightarrow{Q(r_i).pre\ \ Q(r_i)}$方向上的单位向量。结合公式 4.4 可以看出当$\overrightarrow{r_i r_{i+1}}$和$\overrightarrow{Q(r_i)}$的夹角是一个锐角时，由于受限于$d\left(Q(r_i), Q(r_{i+1})\right)$为 0，所以轨迹段形状相似性$sim_{shape}$为 0，当$\overrightarrow{r_i r_{i+1}}$和$\overrightarrow{Q(r_i)}$处的夹角是一个钝角时，轨迹段形状相似性$sim_{shape}$小于 0。

$$\overrightarrow{Q(r_i)} = \frac{\overrightarrow{Q(r_i).pre\ \ Q(r_i)}}{\left|\overrightarrow{Q(r_i).pre\ \ Q(r_i)}\right|} + \frac{\overrightarrow{Q(r_i)\ \ Q(r_i).next}}{\left|\overrightarrow{Q(r_i)\ \ Q(r_i).next}\right|} \tag{4.5}$$

当出现图 4.2(c)中的情况，即$Q(r_i)$和$Q(r_{i+1})$轨迹段中间包含一个或多个样本点，此时$Q(r_i)Q(r_{i+1})$可能不是一条直线段，不能使用上面的公式直接进行计算。在图 4.8 中，Q(r_0)和 Q(r_1)中间间隔着q_1和q_2，此时需要使用 DTW-BDS 样本点匹配算法，输入轨迹段为 Q(r_0) Q(r_1)和$r_0 r_1$，将q_1和q_2视作样本点，寻找轨迹 R 上的对应点 R(q_1)和 R(q_2)。然后将r_0R(q_1)、R(q_1)R(q_2)和R(q_2)r_1视为三个独立的轨迹段，其对应轨迹段分别为$Q(r_0)q_1$、$q_1 q_2$和$q_2 Q(r_1)$。此时所有对应轨

迹段均为直线段，计算对应轨迹段的轨迹形状相似性。最后，将每一段独立的对应轨迹段之间的轨迹段形状相似性相加，得到轨迹段$r_i r_{i+1}$和$Q(r_i)Q(r_{i+1})$之间的轨迹段相似距离，如公式 4.6 所示，其中，q_m至q_n是$Q(r_i)$与$Q(r_j)$之间间隔的样本点，m<n。

$$sim_{shape}(r_i r_{i+1}, Q(r_i)Q(r_{i+1}))=sim_{shape}(r_i R(q_m), Q(r_i)q_m)+ \sum_{l=m}^{n-1} sim_{shape}(R(q_l)R(q_{l+1}), q_l q_{l+1})+sim_{shape}(R(q_n)r_{i+1}, q_n Q(r_{i+1})) \quad (4.7)$$

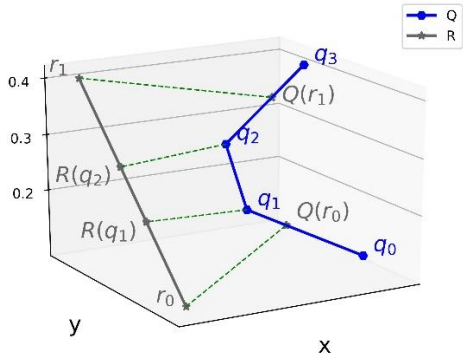


图 4.8 第三种轨迹段的形状相似性计算示例

Fig. 4.8 Example of calculating the shape similarity of third track segments

轨迹段形状相似性计算的伪代码如算法 4.2 所示。sim_shape_calculate 的作用是计算对应轨迹段之间的形状相似性。轨迹段相似性计算的伪代码如下。

算法 4.2 轨迹段形状性计算 sim_shape_calculate(r_i, r_{i+1}, $Q(r_i)$, $Q(r_{i+1})$)

输入：样本点r_i，r_{i+1}及其对应点$Q(r_i)$，$Q(r_{i+1})$

输出：轨迹段$r_i r_{i+1}$到对应轨迹段$Q(r_i)$，$Q(r_{i+1})$的形状相似性 sim_shape

```
if Q(r_i)和Q(r_{i+1})间无样本点或者二者为同一个点
    sim_shape = min(0, d(r_i, r_{i+1}) * [cos(r_i r_{i+1}, Q(r_i)Q(r_{i+1}))])
    return sim_shape
else
    for Q(r_i)Q(r_{i+1})中的每条轨迹段v_x v_{x+1}
        R(v_x) = DTW-BDS(v_x)
        R(v_{x+1}) = DTW-BDS(v_{x+1})
        sim_shape += sim_shape_calculate(R(v_x), R(v_{x+1}), v_x, v_{x+1})
    end for
return sim_shape
```

4.2.3 轨迹段形状影响因子

为了描述轨迹段之间形状相似性sim_{shape}为轨迹距离带来的影响，这里提出了基于 sigmoid 函数 s(x)的形状影响因子的概念。公式 4.7 是 sigmoid 函数 s(x)的表达式，该函数的定义域为全体实数，值域在 0 到 1 之间，是机器学习中很

常见的一个阈值函数，可以将变量映射 0,1 之间，函数形状如图 4.9 所示。

$$s(x) = \frac{1}{1+e^{-x}} \tag{4.7}$$

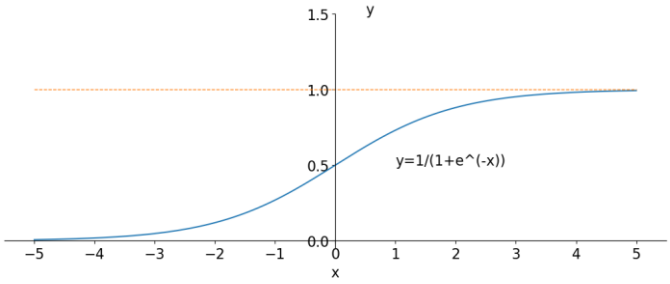


图 4.9 sigmoid 函数图像

Fig. 4.9 Sigmoid function image

轨迹段形状相似性sim_{shape}的取值范围在负无穷到正无穷，我们可以利用 sigmoid 函数的特性，将轨迹段形状相似性映射在 0,1 之间。但是我们希望最后的形状影响因子表示的是形状的差异程度，即形状越不相似，形状影响因子越大。但由于形状相似性sim_{shape}越大表示的是形状越相似，我们需要一个单调递减的函数来表示轨迹段的形状影响因子。因此我们使用 s(x)关于 y 轴对称的函数来作为阈值函数。此外，由于形状影响因子对轨迹相似性起到的是一个影响作用，我们需要一个参数去调节这个影响的大小。下面给出基于 sigmoid 函数的关于 y 轴对称的函数的轨迹段形状影响因子I_{shape}的计算公式，如公式 4.8 所示，I_{shape}的函数图像如图 4.10 所示。其中形状敏感度参数μ可以调节轨迹相似性对轨迹段形状的敏感程度，μ越小，表示轨迹相似性对形状因素越敏感。

$$I_{shape}\left(r_i r_{i+1}, Q(r_i)Q(r_{i+1})\right) = \frac{1}{1+e^{sim_{shape}\left(r_i r_{i+1}, Q(r_i)Q(r_{i+1})\right)}} + \mu \tag{4.8}$$

其中，$\mu \in [0, +\infty)$。

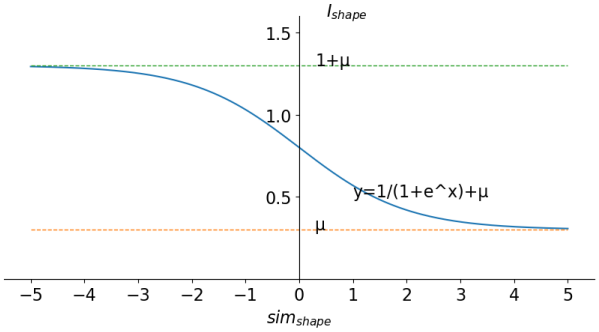


图 4.10 I_{shape}函数图像

Fig. 4.10 I_{shape} function image

下面给出所有轨迹段之间的形状影响因子计算的伪代码，如算法所示。这里调用前面的 sim_shape_calculate 算法计算所有轨迹间的相似性，最终获得所有轨迹段间的形状影响因子。

算法 4.3 轨迹段形状影响因子计算 I_shape_calculate(Q,R,pair,μ)

输入：查询轨迹 Q，数据轨迹 R，R 的对应点对 pair，形状影响因子的权重μ

输出：数据轨迹所有轨迹段到对应轨迹段的形状影响因子 I_shape _list

1. **for** $r_i r_{i+1}$ in R
2. sim_shape= sim_shape_calculate(r_i, r_{i+1}, $Q(r_i)$, $Q(r_{i+1})$)
3. I_shape=$(1+e^{sim_{shape}(r_i r_{i+1}, Q(r_i)Q(r_{i+1}))})^{-1}+\mu$
4. I_shape _list.add(I_shape)
5. **end for**
6. **return** I_shape _list

4. 3 三维时空下的轨迹相似性查询

4.3.1 对应轨迹段距离

通过对对应轨迹段形状相似性和时空距离的讨论，得到了形状影响因子I_{shape}以及对应轨迹段间的三维时空距离d_{st}。下面我们给出对应轨迹段距离$d_{segment}$的计算公式，如公式 4.10 所示。$d_{segment}$与轨迹形状影响因子和轨迹时空距离呈正相关，轨迹段形状越相似，轨迹段的时空距离越小，表示轨迹段距离越小。

$$d_{segment}(r_i r_{i+1}, Q(r_i)Q(r_{i+1}))=I_{shape}(r_i r_{i+1}, Q(r_i)Q(r_{i+1})) \times d_{st}(r_i r_{i+1}, Q(r_i)Q(r_{i+1})) \tag{4.10}$$

4.3.2 有效子轨迹

给出了轨迹段距离的计算公式后，可以计算得到所有数据轨迹段到其对应的查询轨迹段的距离，我们需要基于轨迹段做相似性计算。

轨迹相似性查询的目的是找出在空间和时间上与查询轨迹接近的数据轨迹。而很多情况下，数据轨迹的长度与查询轨迹的长度不相同。如图 4.11 所示，查询轨迹 Q 的总长度小于数据轨迹 R 的总长度，大于数据轨迹 S 的总长度。如果直接使用完整轨迹之间的距离来衡量轨迹的相似程度，那么$r_0 r_2$、$r_5 r_7$首尾两段会加大轨迹 R 与 Q 的时空距离，拉低了整体的相似性，不利于发现$r_2 r_5$部分与查询轨迹的相似性。

因此在相似性计算的时候需要利用对应点匹配结果，将长轨迹的首尾部分进行剪枝操作。首先找到数据轨迹中第一个与邻居样本点的对应点不同的样本点，称为有效首节点，类似地可以找到最后一个与邻居样本点不同的样本点，称为有效尾节点，有效首尾节点连接成的子轨迹称为轨迹 R 的有效子轨迹

R_{effect}。若R_{effect}的长度$R_{effect}.length$与 Q 的总长度$Q.length$接近，就可以使用R_{effect}到轨迹 Q 的距离作为 R 与 Q 之间的距离。这样可以从长轨迹中挖掘出与查询轨迹最相似的部分。若发现有效子轨迹的长度远小于 Q 的总长度，则需要将有效部分往首尾两端扩张，直至有效子轨迹的长度与 Q 长度接近。使用 L-rate(Q, R_{effect})表示 R 有效部分与 Q 的长度比值，如公式所示，当 L-rate(Q, R_{effect})> ε时，可以停止扩张，其中ε为子轨迹长度限制参数。

$$\text{L-rate(Q, } R_{effect}\text{)}=\frac{R_{effect}.length}{Q.length} \tag{4.11}$$

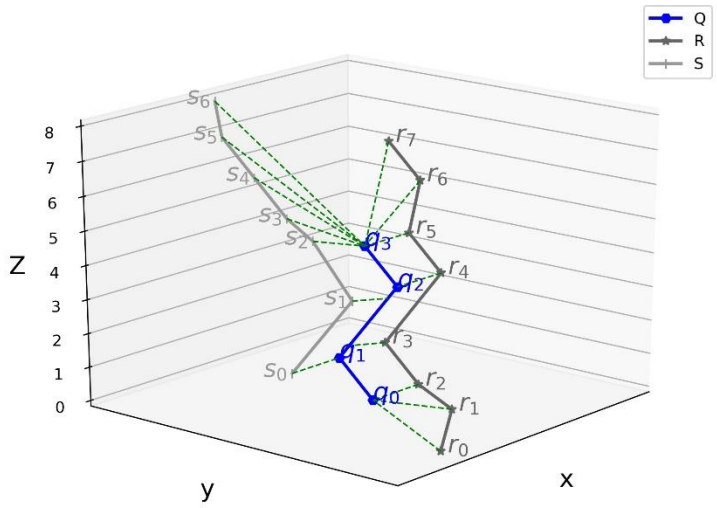


图 4.11 有效子轨迹示例

Fig. 4.11 Example of effective subtrajectry

在图 4.11 中，按照上述方法查找到轨迹 R，发现 Q(r_2)与 Q(r_3)不是同一个点，因此 R 的有效首节点为r_2，同理 R 的有效尾节点为r_5，R 的有效子轨迹R_{effect}为r_2r_5。对应轨迹 S，其有效子轨迹S_{effect}为s_0s_2，但是发现S_{effect}的长度小于轨迹 Q 的长度，此时需要长度限制参数ε发挥作用，当ε取 1 时，表示有效子轨迹长度必须大于 Q 的长度，因此需要将s_2s_4纳入有效子轨迹S_{effect}，最后获得的S_{effect}为s_0s_4。下面给出有效子轨迹查找算法的伪代码。

算法 4.4 有效子轨迹查找算法 effect_trajectory_finding(Q, R, pair, ε)

输入：查询轨迹 Q，数据轨迹 R，R 的对应点对 pair，长度限制参数ε

输出：轨迹 R 的有效子轨迹R_{effect}

1. 根据 pair 获得 R 的有效首节点r_{start}和有效尾节点r_{end}
2. R_{effect}=$r_{start}r_{end}$

3. **while** L-rate(Q, R_{effect}) < ε and R_{effect}!=R
4. **if** r_{start}.pre 存在
5. R_{effect} ← r_{start}.pre
6. r_{start} = r_{start}.pre
7. **if** r_{end}.next 存在 and L-rate(Q, R_{effect}) < ε
8. R_{effect} ← r_{end}.next
9. r_{end} = r_{end}.next
10. **end while**
11. **return** R_{effect}

4.3.3 轨迹相似性查询算法

在得到轨迹 R 的有效子轨迹R_{effect}后，我们计算R_{effect}中所有轨迹段与其对应轨迹段的距离$d_{segment}$之和，作为轨迹 R 与轨迹 Q 之间的距离。

$$d(Q,R)=\sum_{r_i r_{i+1}\in R_{effect}} d_{segment}\left(r_i r_{i+1}, Q(r_i)Q(r_{i+1})\right) \tag{4.12}$$

下面是完整的计算查询轨迹 Q 和数据轨迹 R 相似性算法的伪代码。

算法 4.5 轨迹相似性计算 similarity_calculate(Q, R, ε, η, μ, ε)

输入：查询轨迹 Q，数据轨迹 R，激励参数ε，断点切分阈值η，形状权重μ，长度限制参数ε

输出：轨迹 R 的有效子轨迹R_{effect}，Q 和 R 的距离$d(Q,R)$

1. pair ← 计算 Q 和 R 的 DTW-BDS 对应点
2. 查找 R 的有效子轨迹R_{effect}
3. I_shape _list ← R_{effect}与 Q 对应轨迹段的形状影响因子
4. d_st_list ← R_{effect}与 Q 对应轨迹段的时空距离
5. d_segment_list ← dss_list × I_shape _list
6. 计算$d(Q,R)$
7. **return** R_{effect}, $d(Q,R)$

下面介绍一下整体的查询过程，首先给定时空转换因子I_{st}，对数据库中所有轨迹数据采用 normalization 算法，将查询轨迹 Q 和数据库中的数据轨迹转化到三维时空。然后对数据库中每一条数据轨迹都与 Q 进行轨迹相似性计算。在 Q 和 R 的相似性计算中，首先通过 DTW_BDS 样本点匹配算法获取对应样本点，然后查找 R 的有效子轨迹R_{effect}，接着计算R_{effect}中每一条轨迹段到对应轨迹段的距离，最后相加得到的就是轨迹 R 与 Q 的距离。判断该距离是否小于给定的轨迹距离阈值δ，如果小于该阈值，那么该保留子轨迹，加入到查询结果中去，否则丢弃。

4.4 本章小结

为了在数据轨迹中获得与查询轨迹最相似的子轨迹，本章首先提出了对应轨迹段之间的时空距离d_{spatio}以及对应轨迹段形状影响因子I_{shape}，二者分别描述了三维空间中轨迹段之间的时空距离以及形状上的相似性对轨迹段相似性的影响，然后将二者结合得到轨迹段的距离$d_{segment}$。接着通过样本点匹配，提出了有效子轨迹的概念，将有效子轨迹中的所有轨迹段到查询轨迹的距离之和，作为数据轨迹和查询轨迹的距离。最后根据上面所有算法，给出了轨迹相似性查询的整个过程，可以得到数据库中与查询轨迹最相似的轨迹。

第五章 实验设计与分析

本章主要介绍通过一系列轨迹相似性查询实验来验证本文所提出的算法的性能，以及与一些最新的相似性算法做比较。

5.1 实验环境与数据集

本章实验使用的计算机和相应软件如表 6.1 所示。对本文提出的所有算法均采用 python 语言实现，所用 PC 机的操作系统为 64 位的 Windows 7，集成开发环境为 JetBrains PyCharm。

表 6.1 实验环境
Table 6.1 Experimental setting

类别	描述
CPU	Intel (R) Core(TM) i7-6700 3.40 GHz
硬盘	8.00 GB
内存	1T
操作系统	Microsoft Windows 7(64 位)
IDE	JetBrains PyCharm
编程语言	python
相关开发包	numpy，matplotlib，mpl_toolkits

实验中采用的第一个数据集是微软亚洲研究院在 GeoLife 项目中采集的真实的北京市 182 个志愿者的日常移动轨迹数据集[31][32][33]，以下简称 GL。轨迹数据的收集时间长达 5 年，共有 17621 条轨迹数据，总距离为 1251654 千米，采样策略为每 2-5 秒进行一次采样或者每隔 5-10 米进行一次采样。实验中使用的第二个数据集是北美路网的轨迹数据集 North America Road Network，以下简称 NARN。轨迹最短的时间跨度为 380 秒，包含轨迹数目为 2 万条，每条轨迹中样本点个数为 20 到 100 个。

由于 GL 数据集中的每一条数据都是一个移动对象一天内的移动轨迹，因此单条轨迹很长，时间跨度较大。本文提出的相似性算法可以很准确的找出长轨迹中与查询轨迹相似的部分，但是之前的相似性计算方法计算的是整条轨迹的与查询轨迹的距离，这会导致空间上差异太大，不便于比较。因此本文将轨迹数据按照时间段进行划分，从 0 点开始，每 6 小时作为一个时间段，如果一条轨迹跨越多个时间段，则按时间段划分为多条轨迹。然后在设计实验的时候，查询轨迹的时间跨度不会跨过每个时间段，这样就解决了数据集中单条轨迹太长的问题。因为本算法计算的是数据轨迹与查询轨迹最相似部分的距离，这样做可以尽量减小其他算法在查找局部轨迹相似能力方面与本算法的差异。

由于人每天的活动大部分开始于早晨，停止于夜晚，以一天为周期，因此我们数据集中的时间采用 24 小时时间制，时间的单位是秒，时间戳数值是从 0 时到当前时刻的秒数，如果遇到轨迹数据跨越了 24 一天的分界线 0 时，就将轨迹数据切分为两条数据，因此所有轨迹中的样本点的时间戳会按照时间顺序不断增大。

每次实验都使用 50 次独立的查询轨迹进行查询，分别获取轨迹相似性算法计算得到的最初结果，即轨迹间的距离。然后根据不同实验需求，进行不同的处理。如果是求最接近的 top-k 条轨迹，则返回距离最小的 k 条轨迹作为查询结果。如果是给定了一个距离阈值作为相似与不相似的分界，则返回小于距离阈值的轨迹作为查询结果。

实验中使用到的评价指标有查准率和查全率。查准率 P（Precision）如公式 5.1 所示。其中 TP 代表正例被归为正例，即原本相似的轨迹在相似性查询结果当中出现。FP 代表反例被归为正例，即原本是不相似的轨迹，但是却出现在查询结果中。我们将查询结果与已经事先标上的正确类别做比对，可以得出每一次查询的 TP 和 FP。实验中使用的查询轨迹属于不同的类别，因此我们给出多次查询的查准率P_mul，如公式 5.2 所示，其中 n 代表总的查询次数，i 代表每一次查询，表示查询出所有正确分类的数目除以所有查询结果数目之和。

$$P = \frac{TP}{TP+FP} \tag{5.1}$$

$$P\_mul = \frac{\sum_i^n TP_i}{\sum_i^n TP_i+\sum_i^n FP_i} \tag{5.2}$$

查全率 R（Recall）如公式 5.3 所示，其中 FN 表示正例被归为反例，即原本相似的轨迹，却没有出现在查询结果中。查全率 R 与查准率 P 不同的地方在于查全率 R 的分母是所有与查询轨迹相似的轨迹数目，来量化查询结果有没有覆盖所有真实结果。和 P_mul 的定义类似，我们给出多次查询的查全率 R_mul 的定义，如公式 5.4 所示。

$$R = \frac{TP}{TP+FN} \tag{5.3}$$

$$R\_mul = \frac{\sum_i^n TP_i}{\sum_i^n TP_i+\sum_i^n FN_i} \tag{5.4}$$

5. 2 参数的影响

本文提出的 STS 算法中包含如下参数，划分断点的距离阈值η，形状敏感度参数μ，局部相似性计算结果长度限制参数ε，以及子轨迹到查询轨迹的距离阈值δ。针对这些参数，本文设计了下面的实验，去研究参数的变化给查询结果带来的影响。

5.2.1 距离阈值η对查询结果的影响

距离阈值η用于划分数据轨迹中每条轨迹段上断点，轨迹段长度相同的情况下，η越小，划分出的断点越多。本实验中η取值为 1 米至 500 米。而η的不同会导致最后的轨迹之间距离值的不同，轨迹间距离越小代表越相似，这里使用每次距离计算中 top-k 的轨迹作为查询结果，k 分别取 20、40 和 60。

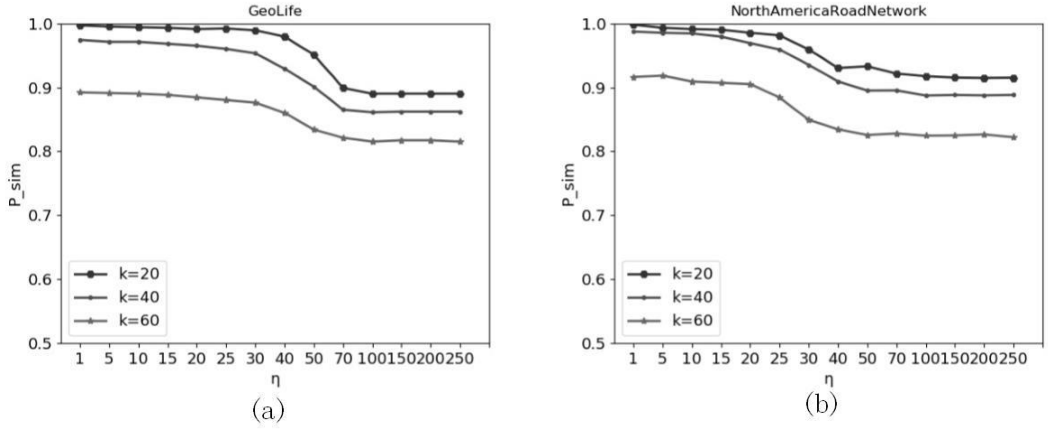


图 5.1 η对查准率的影响

Fig. 5.1 Effect of η on precision ratio

通过对 GL 和 NA 两个数据集的实验，得出了不同数据集上的查准率，如图 5.1 所示。从图中可以看出，图 5.1(a)的 GL 数据集和图 5.1(b)中的 NARN 数据集相似性查询结果的查准率都随着η的变大而减小。

在 GL 数据集中，$\eta \in [1,30]$的时候，查准率较高，因为η较小会导致轨迹段中的断点数目较多，能更好地表示出轨迹段的空间情况。而随着$\eta \in [30,70]$之间不断变大，查准率在不断降低，即查询结果中实际不相似轨迹的比例在增大。最后在$\eta \in [70,250]$的情况下，轨迹段中的断点数目很少，接近于 0，断点的作用就变得很小，接近于没有断点作用下算法的查准率情况。在该实验中可以看出，断点对查准率的提升起到了很大作用。在 NA 数据集上，查准率变化情况类似，但是由于不同数据集的轨迹段平均长度不同，所以导致在 NA 数据集中，$\eta \in [20,40]$时查准率下降很严重，其他区域趋于平缓，当$\eta \in [1,20]$时有着不错的查询效果。

由于每条查询轨迹包含的与其相似的轨迹条数是一定的，如果在 top-k 查询中增大 k，即 k 由 20 增大到 40 和 60，必然会导致一些不相似的轨迹掺杂进来，所以在同样η的情况下，返回的轨迹条数越多，查准率会越低。

5.2.2 形状敏感度参数μ对查询结果的影响

形状敏感度参数μ的取值范围为全体正实数，可以调节相似性计算结果中对

两条轨迹形状差异的敏感程度，μ越小会使查询结果对形状越敏感，让形状不相似的轨迹段对计算结果造成的影响更大，以过滤掉形状不相似的轨迹段，但是如果取值过小，会导致最终计算结果中，形状因素占有更大比重，从而忽略了三维空间距离的影响。因此实验中需要验证μ的大小对最终查准率 P_mul 的影响程度，如图 5.2 所示。

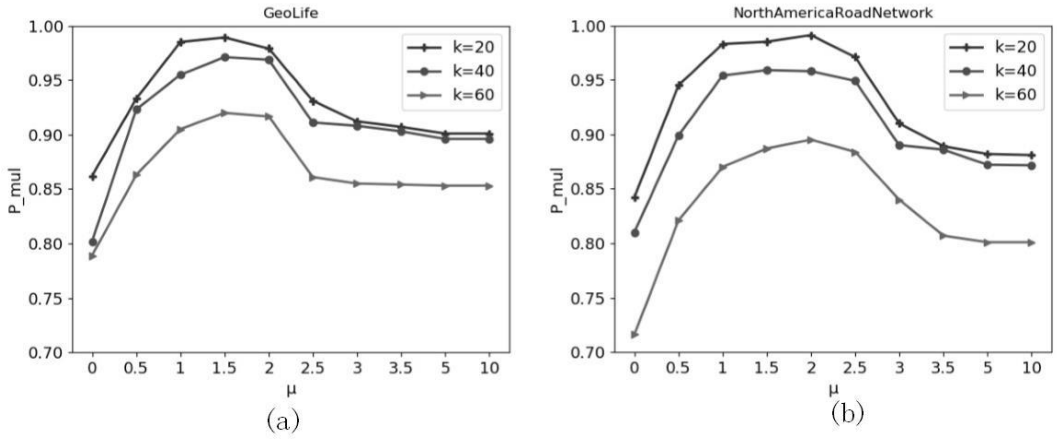


图 5.2 μ对查准率的影响

Fig. 5.2 Effect of μ on precision ratio

从上面两个数据集的实验结果中我们可以看出，形状敏感度参数$\mu \in [0,0.5]$时，查准率较低。当$\mu \in [1,2]$，查准率在该区间内有一个较高值，当$\mu > 2$继续增加，查准率又会慢慢降低，直至趋于稳定。

下面分析一下为什么会出现这个情况。当μ=0 时，此时计算结果对形状差异的敏感程度最大，相对而言减小了轨迹时空距离带来的影响，结果显示此时查准率最低，表明了移动对象轨迹相似性计算不能仅仅考虑形状上的相似，时空维度的距离对轨迹是否相似也有很大的影响。当μ增大到 1 到 2 时，形状相似性因子在一定程度上减小了形状差异给最终结果带来的影响，同时时空距离也能表达出对应的信息，此时算法在两个数据集上都有较不错的表现。当μ继续增大到 3.5 以上时，计算结果中几乎忽略形状上的差异造成的影响，此时的查准率较低，说明轨迹相似性也不能仅仅只考虑时间和空间维度，轨迹形状上的相似性也是一个影响的因素。但是在轨迹相似性查询中，时间和空间因素还是起主要影响作用，不能减小或忽视其作用。因此形状敏感度参数过大或者过小，都会对查询结果造成不好的影响。

5.2.3 长度限制参数ε对查询结果的影响

长度限制参数ε是 L-rate(Q, R_{effect})的下限，其作用是保证有效子轨迹的长度必须要与查询轨迹长度接近，如果太小，查询结果中便会包含很多只与查询

轨迹的一部分相似的轨迹，如果太大，则查询结果会过多地受有效部分之外的轨迹段影响，结果不准确。因此ε需要在 1 附近取值。

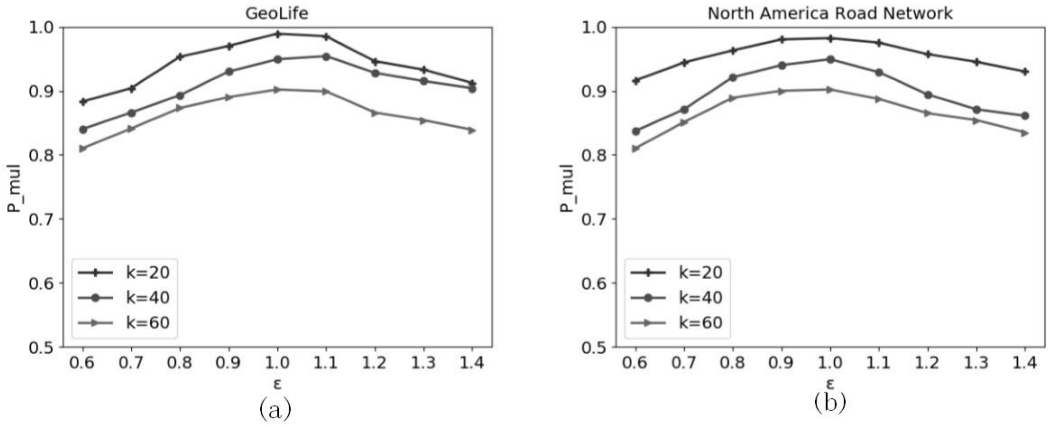


图 5.3 ε对查准率的影响

Fig. 5.3 Effect of ε on precision ratio

实验结果如图 5.3 所示，由实验可以看出，在ε ∈[0.9, 1.1]区间内，相似性查询结果的查准率 P_mul 较大。当ε小于该区间，由于对子轨迹长度的限制过小，如果ε取 0.6，仅考虑了与查询轨迹 60%的部分的相似程度，因此计算结果中有很多是错误结果。而如果ε较大，比如ε取 1.4，则代表查询轨迹的有效子轨迹长度要超出数据轨迹 40%，数据轨迹中真正与查询轨迹完全匹配的只有接近查询轨迹长度的部分轨迹段，计算中包含太多多余的轨迹段，会拉低相似性，导致查准率降低。

因此参数ε应该在 1 附近取值，发挥有效子轨迹的作用，提升查准率。

5.2.4 距离阈值δ对查询结果的影响

本文中的算法使用距离阈值δ筛选出所有距离查询轨迹小于δ的数据轨迹作为查询结果，也就是说距离阈值δ是可以作为轨迹相似与否的标准。当轨迹间距离小于等于δ时，认为轨迹相似，当轨迹间距离大于δ时，认为轨迹不相似，而不再像之前的实验那样使用计算结果的 top-k 作为查询结果。当我们研究δ对查询结果的影响时，可以使用查准率 P_mul 和查全率 R_mul 来共同监控查询结果。

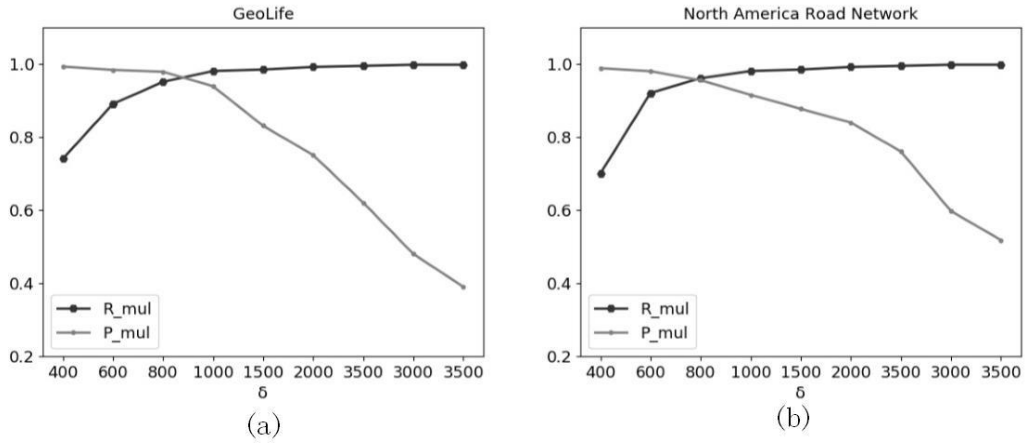


图 5.4 δ对查准率和查全率的影响

Fig. 5.4 Effect of ε on precision and recall rate

实验结果如图 5.4 所示，根据对两个数据集的实验，得到了以上的实验结果，两个数据集上都反应了同一个规律，随δ的增大，查全率 R_mul 和查准率 P_mul 呈现相反的增长趋势。随着δ不断增大，查询结果中，查全率 R_mul 在不断上升，在δ=1000 之后逐渐趋近于 1，增大δ可以减小 FN，增大 TP，因此可以提高查全率。但这不是一个好现象，因为不断增大δ意味着相似的标准在不断降低，让一些原本不相似的，被排除在查询结果之外的轨迹被判断为相似轨迹，作为查询结果返回，导致的就是查准率 P_mul 在不断下降，因为降低标准会增大 FP，减小 TP 的比重。当δ = 400时，此时对相似性的要求较为严格，因此此时纳入查询结果中的轨迹真正与查询轨迹相似的比例很大，但是这个带来的不足就是由于标准的提高，使得一些距离稍远，但是仍然相似的轨迹被排除在外，因此此时查全率 R_mul 较低。

根据以上分析，如果查询需求是想获得与查询轨迹相似度很高的轨迹，但是不要求太多数量，δ可以取一个相对较小的值，在 400 至 600 之间均可。如果查询需求是想更多的获得与查询轨迹相似的数据轨迹，可以取一个较大的δ值，大于 1000 即可。如果想综合考虑上面两个要求，可以取δ在 800 到 1000 之间，此时查准率和查全率均处于较高水平。

5. 3 与最新研究成果的对比实验

5.3.1 查询轨迹长度对不同算法的影响

在本节的实验中使用了 DTW 算法、SDTW 算法、PTM 算法和本文的提出的轨迹局部相似性查询算法（STS）做比较。

DTW 算法中的距离采用二维的欧式空间距离。在 SDTW 算法中需要设置

参数ω，ω可以对 SDTW 算法中对轨迹形状相似性的权重进行调节，取ω在 1 到 10 之间的最大的查准率作为 SDTW 算法的效果。PTM 算法需要指定阈值ε_s和ε_t来作为对应点之间的最大的空间距离和时间差的阈值，我们ε_s取值范围在 5 到 20 之间，ε_t取值范围在 10 到 30 之间，此外还需要设置 PTM 算法中调节时间和空间权重大小的参数λ，λ在 0.3 至 0.7 之间取值，通过以上调整，获取 PTM 算法最大的查准率作为 PTM 算法的效率。STS 算法中，断点距离阈值η取 10 到 20 之间，形状敏感度参数μ取值在 1 到 2 之间，长度限制参数ε取值在10^{-7}到10^{-4}之间，通过不同参数组合，获得最大的查准率。

为了验证本文 STS 算法在查找局部轨迹相似性上的优点，根据查询轨迹的总长度 L(Q)将查询轨迹数据分为三堆，第一堆轨迹长度在 1 千米左右，第二堆轨迹的长度在 5 千米左右，第三堆轨迹的长度在 10 千米左右，每堆中的查询轨迹长度 L(Q)与各自标准长度差距在 200 米以内。使用不同长度的查询轨迹，用于研究不同算法对不同长度查询轨迹的处理效果。使用以上算法计算出各自距离或相似性之后，这里采用 top-40 的轨迹作为查询结果。

在两个数据集上的实验结果的查准率如图 5.5 所示。从实验结果中我们可以看出，DTW 算法的查准率随着查询轨迹长度的变化在 50%到 80%之间波动，相对其他算法而言，查准率较低，原因是 DTW 算法中计算距离时并没有考虑到时间因素，因此查准率较低。SDTW 算法和 PTM 算法均考虑了时间因素，但是 PTM 算法中采用了轨迹整体程度上的时间相似性和空间相似性相结合，因此会导致有一定误差出现，效果比 SDTW 算法稍差一点，并且这两个算法随轨迹长度变化，查准率也发生较大变化。STS 算法在不同的查询轨迹长度上的表现都很好，在两个数据集的不同长度的查询轨迹上的查准率 P_mul 均在 95%左右，没有因为查询轨迹的长度的变化而影响查准率，稳定性较好。

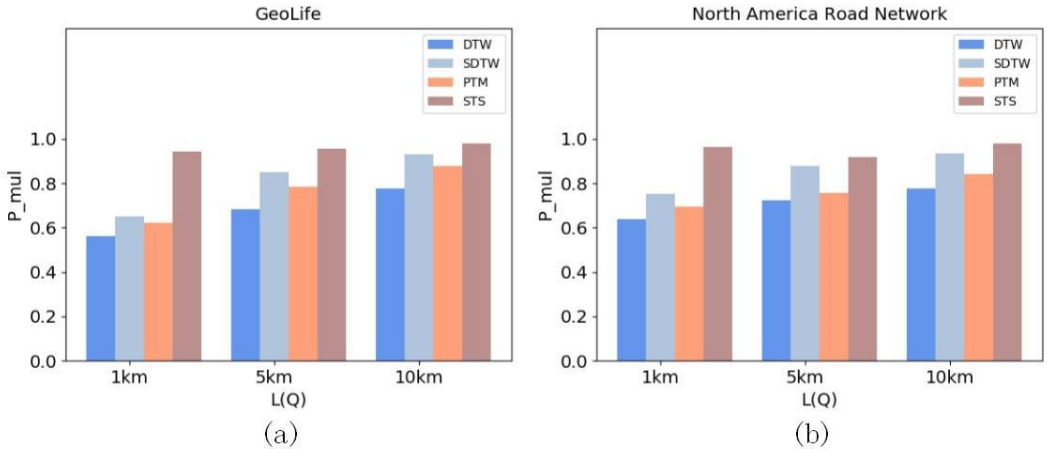


图 5.5 查询轨迹长度对不同算法查准率的影响

Fig. 5.5 Influence of query trajectory length on precision of different algorithms

5.3.2 三维时空的有效性的研究

为了研究本文在第三章提出的三维时空在结合时间和空间的方法上的有效性，这里使用 DTW 算法作为参照，将采用二维欧式空间距离的 DTW 算法，引入三维时空距离的 DTW 算法以及 STS 算法来进行对比，我们将采用二维欧式空间距离的 DTW 算法记为 DTW-2d，引入三维时空距离的 DTW 算法记为 DTW-3d。时空转化因子I_{st}取值为数据集的平均速度。为了减小查询轨迹长度对 DTW 算法的影响，这里的查询轨迹只使用长度大于 7 千米的查询轨迹，使用 top-40 的轨迹作为查询结果，最后使用查准率 P_mul 作为算法评价指标。

实验结果如图 5.6 所示，从实验结果中可以看出，无论是在 GL 数据集还是 NARN 数据集上，引入了三维时空距离的 DTW-3d 的表现明显要比 DTW-2d 好很多，考虑了时间因素之后，DTW 算法的查准率有了一个大约 10%左右的提升。因此本文提出的三维时空是有效的。

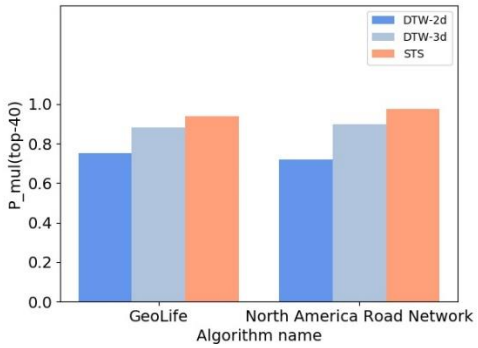


图 5.6 对比验证三维时空有效性

Fig. 5.6 Comparison of three dimensional spatio-temporal validity

5.3.3 噪音对不同算法的影响

为了研究以上提到的相似性算法的健壮性，本节实验使用加入噪音的轨迹数据来研究算法对噪音的抗干扰能力。噪音使用的是[-100,100]之间的随机数，随机数的分布是均匀分布。这里使用噪音率α来表示不同程度的噪音，α的取值范围在 0.1 到 1，最后叠加到数据上的噪音就是$[-100\alpha, 100\alpha]$的均匀分布的随机数。将数据集中的每一个数都叠加一个随机噪音，获得噪音数据。然后使用 DTW 算法、SDTW 算法、PTM 算法和 STS 算法进行相似性计算，取计算结果的 top-40 作为查询结果，使用查准率作为评价指标。

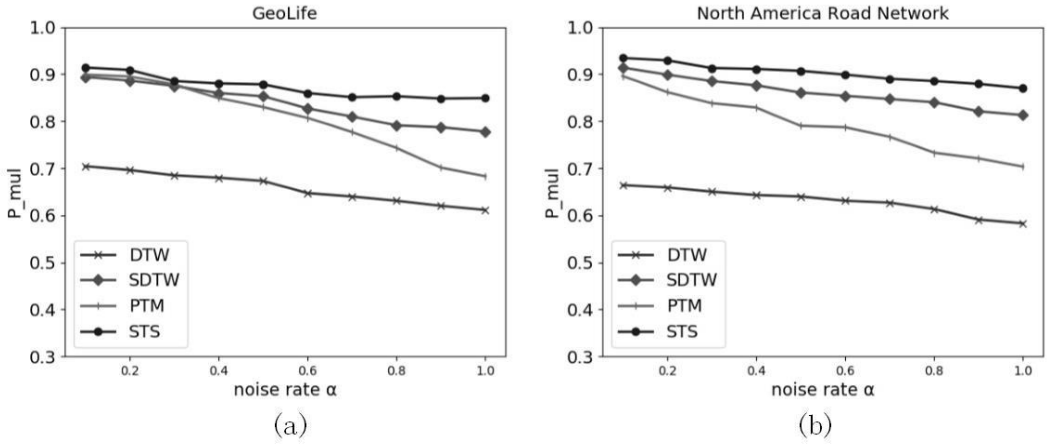


图 5.7 噪音率对不同算法查准率的影响

Fig. 5.7 Influence of noise rate on precision of different algorithms

实验结果如图 5.7 所示，可以看出四个相似性计算函数在两个数据集上对噪声的抵抗能力。由于 DTW 算法没有考虑时间因素，查准率仍然较低，而 DTW 算法的对应点匹配效果好，所以 DTW 算法的抗噪声干扰能力比较好，噪音率α从 0.1 变化到 1，虽然查准率不是很高，但是查准率波动较小。而考虑了时间因素的 SDTW 算法的表现便能明显反应出来 DTW 算法中抗干扰的优点，不但查准率较高，而且波动较小。PTM 算法的表现相对于 SDTW 算法就要差一点， PTM 算法的查准率随噪音率α的增大产生了较大的变化，在 GL 数据集中，前面较小的噪音率情况下，PTM 算法的表现与 SDTW 算法不相上下，均在 80%到 90%之间波动，但是随着噪音率的增大至 0.5 以上，PTM 算法的表现较差，查准率下降地越来越快，而在 NARN 数据集中，PTM 算法的查准率在一开始便呈现出较快的下降趋势，表现出了抗噪声干扰能力较低，因为其对应点匹配的时候没有考虑到时序问题，所以在噪声环境中，对应点匹配可能出现严重的时序错乱，从而导致相似性计算结果产生较大误差。在噪声环境中表现最好的是本文提出的 STS 算法，在噪音率α从 0.1 变化到 1 中，在 GL 数据集中的下降幅度为 7%，在 NARN 数据集中的下降幅度为 6%，表现出较强的稳定性和较高的抗干扰能力，因为其不但融入了 DTW 对应点匹配的优势，还结合了断点去描述每一条轨迹段，因此不但抗噪声干扰能力强，而且有着较高的查准率。

5.4 本章小结

本章为了研究本文提出的 STS 算法的性能，在 GL 和 NARN 两个数据集进行了一系列实验。首先研究了 STS 算法中的各个参数会对查询结果造成的影响，并分析了产生不同效果的原因。然后使用前人的提出的 DTW、SDTW、

PTM 算法与 STS 算法进行对比，验证了 STS 算法在某些方面的优势并分析其原因，说明了 STS 算法是一个有效的、抗干扰能力较强的相似性查询算法。

第六章 总结与展望

6.1 本文总结

随着车载 GPS 设备的大量应用以及智能手机的普及，每时每刻都会产生大量的轨迹数据，其中包含丰富的信息，比如个人的轨迹数据中包含个人日常作息习惯、常用交通工具、饮食习惯、休闲娱乐场所等等，车辆的轨迹数据中包含了车辆的行为模式，启动频率、驾驶时间、频繁行驶的路段等等。为了更好地研究人类和车辆的行为模式，我们需要使用轨迹相似性计算方法来挖掘出轨迹数据中隐藏的信息，可以通过频繁出入的休闲娱乐场所给没太多时间交友的上班族推荐有共同兴趣爱好的好友[4]，可以通过商家的访问人次，为游客推荐很多本地人经常去美食店铺，可以通过车辆日常行驶路线，给一些上班族推荐拼车。可以看出如果能利用好轨迹数据中隐藏的信息，将会给人们的生活、工作带来巨大的便利。

然而，现有的轨迹相似性算法还存在着一定的不足，比如对采样策略高度敏感，在抗噪声方面不够健壮，对时间信息的处理不够完善等等。为了解决上面提到的问题，本文提出了 STS 算法用于进行轨迹相似性查询，在保证查询结果的准确性的基础上，还具有较好的抗噪声干扰能力。本文主要贡献有以下几点：

（1）为了将时间维度融入轨迹距离的计算中，本文提出了三维时空的概念，使用时空转换因子I_{st}将一维的时间与二维的欧式空间相结合，让转化后的时间变为三维时空的 z 轴，将欧式空间作为三维空间的 xoy 平面，达到了将时间距离和空间距离在三维空间中统一计算的目的，并解决了前人工作中将时间与空间分开考虑会导致查询轨迹时序混乱的问题。

（2）本文结合 DTW 算法和 BDS 算法中对应点匹配的优势和缺点，提出了一个新的对应点匹配算法：DTW-BDS 对应点匹配算法，该算法将二者寻找对应点的思想结合，先使用 DTW 算法从全局角度寻找对应点，再使用 BDS 算法中的思想，将寻找到的对应点进行局部调整，获得最优对应点。不但可以获得比 DTW 更优的对应点，还不会发生 BDS 算法中时序混乱的问题。

（3）本文提出了一个轨迹段三维空间距离的计算方法。利用距离阈值η，在轨迹每隔η距离便取一个点，称之为断点，然后使用轨迹段上所有断点以及轨迹段的两个端点到对应轨迹段的距离的加权和作为轨迹段之间的距离，这样计算可以解决 DTW 算法计算轨迹相似性存在的矛盾。

（4）本文提出了形状影响因子I_{shape}的概念去量化形状对轨迹相似性造成

的影响，结合了欧氏距离和几何中投影的概念，以及限制了最大激励为查询轨迹段的长度，然后使用 sigmoid 函数获得形状影响因子，与形状相似呈负相关的关系。

（5）结合 DTW-BDS 对应点匹配算法、轨迹段的三维空间距离计算方法以及轨迹段的形状影响因子的计算，提出了两条轨迹之间距离的计算方法。该方法在获得最优对应点的基础上，计算对应轨迹段之间的距离，然后在数据轨迹中获取距离查询最短的子轨迹作为局部相似性查询结果，子轨迹到查询轨迹的距离作为数据轨迹到查询轨迹的距离。然后根据轨迹局部相似性算法提出了对数据库进行轨迹相似性查询算法。

本文提出了一个全新的相似性计算方法，通过获取局部最相似的子轨迹来计算轨迹之间的距离，并在相似性计算中解决了前人工作中出现的一些问题。最后通过实验，验证了算法在轨迹相似性查询中的有效性。

6.2 工作展望

本文的主要工作是提出了一个轨迹相似性算法，使用该算法可以计算出两条轨迹之间的距离，距离越小代表轨迹越相似。而由于研究时间的原因，本文的工作还有待进一步深入研究。主要包括以下几个方面：

（1）在考虑轨迹相似时，处于本文研究背景的需要以及采样设备获取的采样信息，本文只考虑了轨迹数据的空间信息、时间信息，以及由空间信息得到的轨迹方向信息，暂未考虑其他因素。如果需要将轨迹相似性算法应用到其他场景，可能需要增加更多的特征去描述一条轨迹，并且在相似性计算的时候将所有特征都考虑进去。

（2）本文的相似性计算只考虑了行人与出租车等地面移动对象的轨迹，因此空间上采用了二维的欧式空间，如果需要计算鸟类或者飞行器等可以飞行物体的移动轨迹，则需要将二维欧式空间衍生为三维欧式空间，再加上时间维度，那么就是一个四维的时空数据，在四维时空下研究飞行物体的移动轨迹相似性，也是一个很重要并且值得研究的方向。

参考文献

[1] 刘经南, 方媛, 郭迟,等. 位置大数据的分析处理研究进展[J]. 武汉大学学报(信息科学版), 2014, 39(4):379-385.

[2] 陆锋, 张恒才. 大数据与广义 GIS[J]. 武汉大学学报(信息科学版), 2014, 39(6):645-654.

[3] 刘经南, 方媛, 郭迟,等. 位置大数据的分析处理研究进展[J]. 武汉大学学报(信息科学版), 2014, 39(4):379-385.

[4] Li Q, Zheng Y, Xie X, et al. Mining user similarity based on location history[C]// ACM Sigspatial International Conference on Advances in Geographic Information Systems. ACM, 2008:1-10.

[5] Sefidmazgi M G, Sayemuzzaman M, Homaifar A. Non-stationary Time Series Clustering with Application to Climate Systems[M]// Advance Trends in Soft Computing. Springer International Publishing, 2014:55-63.

[6] Karimi H A, Liu X. A predictive location model for location-based services[C]// ACM International Symposium on Advances in Geographic Information Systems. ACM, 2003:126-133.

[7] 陆锋, 郑年波, 段滢滢,等. 出行信息服务关键技术研究进展与问题探讨[J]. 中国图象图形学报, 2009, 14(7):1219-1229.

[8] Jeung H, Liu Q, Shen H T, et al. A Hybrid Prediction Model for Moving Objects[C]// IEEE, International Conference on Data Engineering. IEEE, 2008:70-79.

[9] Gurarie E, Andrews RD, Laidre KL. A novel method for identifying behavioural changes in animal movement data[J]. Ecology Letters, 2010, 12(5):395-408.

[10] Roberts, Guilford, Rezek, et al. Positional entropy during pigeon homing I: application of Bayesian latent state modelling.[J]. Journal of Theoretical Biology, 2004, 227(1):25-38.

[11] Chih-Chieh Hung, Wen-Chih Peng, Wang-Chien Lee. Clustering and aggregating clues of trajectories for mining trajectory patterns and routes [J]. VLDB, 2015, (24): 169-192.

[12] Sheng Gao, Jianfeng Ma1, Weisong Shi, Guoxing Zhan. LTPPM: a location and trajectory privacy protection mechanism in participatory sensing [J]. Wireless communication and mobile computing, 2015, (15):155-169.

[13] Zelei Liu, Liang Hu, Chunyi Wu, Yan Ding, Jia Zhao. A novel trajectory similarity–based approach for location prediction [J]. International Journal of Distributed Sensor Networks, 2016, (11):113-126.

[14] Jin C, Qian W, Zhou A, Analysis and management of streaming data: a survey[J]. Journal of Software, 2014, 15(8): 1172-1181.

[15] Florian Damerow, Stefan Klingelschmitt and Julian Eggert. Spatio-Temporal Trajectory

Similarity and its Application to Predicting Lack of Interaction in Traffic Situations[C]. International Conference on Intelligent Transportation Systems, Windsor Oceanico Hotel, Rio de Janeiro, Brazil, November 1-4, 2016.

[16] Bolong Zheng, Nicholas Jing Yuan, Kai Zheng, Xing Xie, Shazia Sadiq and Xiaofang Zhou. Approximate Keyword Search in Semantic Trajectory Database[C]. International Conferenc on Data Engineering, Seoul, South Korea, 2015.

[17] 龚旭东. 轨迹数据相似性查询及其应用研究[D]. 中国科学技术大学, 2015.

[18] Liu X Y, Zhou Y M. Fast Subsequence Matching in Time-series Database[J]. Journal of Chinese Computer Systems, 2008.

[19] Yi B K, Jagadish H V, Faloutsos C. Efficient retrieval of similar time sequences under time warping[C]// International Conference on Data Engineering, 1998. Proceedings. IEEE, 1998:201-208.

[20] Kim S W, Park S, Chu W W. An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases[C]// International Conference on Data Engineering. IEEE Computer Society, 2001:607.

[21] Keogh E. Exact indexing of dynamic time warping[C]// International Conference on Very Large Data Bases. VLDB Endowment, 2002:406-417.

[22] Yi B K, Jagadish H V, Faloutsos C. Efficient Retrieval of Similar Time Sequences Under Time Warping[C]// Fourteenth International Conference on Data Engineering. IEEE Computer Society, 1998:201-208.

[23] Boreczky J S, Rowe L A. Comparison of Video Shot Boundary Detection Techniques[J]. Journal of Electronic Imaging, 1996, 2670(2):32-8.

[24] Vlachos M, Gunopoulos D, Kollios G. Discovering Similar Multidimensional Trajectories[C] International Conference on Data Engineering. IEEE Computer Society, 2002:673.

[25] Chen L, Ng R. On the marriage of Lp-norms and edit distance[C]// Thirtieth International Conference on Very Large Data Bases. VLDB Endowment, 2004:792-803.

[26] Lee S L, Chun S J, Kim D H, et al. Similarity Search for Multidimensional Data Sequences[C]// International Conference on Data Engineering, 2000. Proceedings. IEEE, 2000:599-608.

[27] Yang N, Zheng J, Liu Q, et al. A Novel Trajectory Similarity Evaluation Method in VANETs [J]. International Journal of Multimedia and Ubiquitous Engineering, 2014. 183-192.

[28] Mao Y, Zhong H, Xiao X, et al. A Segment-Based Trajectory Similarity Measure in the Urban Transportation Systems [J]. Sensors 2017. 524-540.

[29] Liu Z, Hu L, Wu C, et al. A novel trajectory Similarity-Based approach for location prediction[J]. International Journal of Distributed Sensor Networks, 2016, 12(11).

[30] Na T, Li G, Xie Y, et al. Signature-Based Trajectory Similarity Join[J]. IEEE Transactions on Knowledge & Data Engineering, 2017, 29(4):870-883.

[31] Yu Zheng, Like Liu, Longhao Wang, Xing Xie. Learning Transportation Modes from Raw GPS Data for Geographic Application on the Web, In Proceedings of International conference on World Wild Web (WWW 2008), Beijing, China. ACM Press: 247-256

[32] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie. Understanding Mobility Based on GPS Data. In Proceedings of ACM conference on Ubiquitous Computing (UbiComp 2008), Seoul, Korea. ACM Press: 312–321.

[33] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, Wei-Ying Ma. Understanding transportation modes based on GPS data for Web applications. ACM Transaction on the Web. Volume 4, Issue 1, January, 2010. pp. 1-36.

[34] Shang S, Ding R, Zheng K, et al. Personalized trajectory matching in spatial networks[J]. Vldb Journal, 2014, 23(3):449-468.

[35] Kumar C S , George K K , Ramachandran K I , et al. Weighted Cosine Distance Features for Speaker Verification[C]// India Conference. IEEE, 2016.

[36] 郭江利. 基于余弦角距离加权复杂网络的小规模人群行为分类[D]. 2017.

致谢

时光如匆匆流水，两年半的之间转眼便逝去了。在读研的两年半时间里，我在科研、专业技能和生活上都成长了很多。时光的流逝带来的是知识和能力的提升，所以我最后作为在校学生的研究生期间十分充实。

首先感谢我的导师杨晓春老师。杨老师在学术上有很高的造诣，品行上也是一位值得学习的老师。在平时的学习与组会中，杨老师总是在安静地听完我的观点与想法，然后悉心地指出其中的不足之处，然后与我讨论可以从哪方面进行改进。在这个过程中，我不但学到了这个问题的解决方法，还学到了当下次面对一个全新的问题时，我该从哪方面入手，该考虑哪些因素。从杨老师那里我学习到的是解决问题的方法，而不是某一个问题的答案。

感谢我的老师王斌老师。作为一个工程系的老师，王老师带领我们实验室完成了很多工程项目，在不断地实践中，我们获得了锻炼，能力得到了提升，思维方式也得到了改变，为我们以后的学习和工作奠定了良好的基础。

感谢实验室的同学们，感谢我们时序小组的孙学磊、王琦，在项目搭建和二次开发时做出了很大贡献，感谢隐私保护组的王雷霞、刘旺媛、李莉，在我进行科研创新时给予了很多技术上的帮助，感谢 VR 组的张瑞麒、王晓琼、朱莹、张鑫，在完成本篇论文时给了我很多鼓励和帮助。感谢我们实验室的邱涛、孙晶、崔宁宁师兄，感谢韩雨童、张青博师姐，感谢赵征、张洪佳师弟，感谢实验室所有老师同学给我的帮助和支持。

最后，还需要由衷的感谢我的父母、家人和我的女朋友，有你们在我读研期间对我的关心和理解，我才能顺利的完成研究生的学业。毕业之后，我将告别学生时代，正式踏入社会，我还需要在你们的支持下学习去融入这个社会，我也会更加努力，来回报你们对我的关爱！

攻硕期间的科研成果及获奖情况

参加的项目：

- 国家自然科学基金项目：溯源驱动的弱可用性轨迹数据管理关键技术(61572122)，2016.1-2019.12。
- 国家自然科学基金通用技术基础研究联合基金：面向社交网络中虚拟身份的实体识别技术（U173610072），2018.1-2020.12。

获奖情况：

- 2016～2017 学年，获东北大学研究生一等奖学金
- 2017～2018 学年，获研究生国家奖学金
- 2017～2018 学年，获东北大学研究生一等奖学金
- 2017~2018 学年，获“东北大学优秀研究生”荣誉称号
- 2018 年 10 月，获东北大学研究生一等奖学金