分类号 密级

UDC

学 位 论 文

移动对象的时空轨迹相似性查询算法

|  |  |  |  |
| --- | --- | --- | --- |
| 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： | 2019年1月 | 答辩委员会主席： | 王大玲 |
| 评 阅 人： | 王斌、石祥滨 | | |

东 北 大 学

2018年12月

##### A Thesis in Computer System Architecture

**Spatio-Temporal Trajectory Similarity Query Algorithm of Moving Objects**

By Ding Guangwei

Supervisor: Professor Yang Xiaochun

**Northeastern University**

**December 2018**

摘 要

为了提高基于位置的服务的有效性和准确性，需要对移动对象的轨迹数据进行更好的分析，移动对象的时空轨迹相似性查询作为其中一项关键技术是本文研究的重点。

时空轨迹相似性查询的目的是在轨迹数据库获取与查询轨迹相似的轨迹数据集合。目前，国内外在轨迹相似性查询方面已经做了大量研究工作，一些工作致力于寻找通用的轨迹相似性查询算法，另一些致力于研究特性场景下的轨迹相似性查询，但是都存在一定的不足之处。本文对这些算法的不足之处进行分析后，提出了新的用于轨迹相似性查询的模型和算法，具体工作如下：

(1) 在分析了轨迹数据的时空特征之后，提出了时空归一化轨迹表示模型，解决了PTM等算法中时间维度融入相似性计算中出现的时空对应混乱问题。

(2)基于DTW算法和BDS算法提出了新的对应点匹配算法，使样本点可以很好地对齐到另一条轨迹上，并保证了匹配结果的时序性。

(3)提出了新的轨迹段距离的计算方法。轨迹段距离由时空距离和形状影响权值两部分组成。在时空距离计算中，提出了断点的概念，使计算结果受轨迹采样策略影响较小。在形状影响权值计算中，使用余弦距离与投影来量化轨迹段的形状相似性，提高轨迹段距离计算的准确性。

(4)基于上述工作设计更加准确的时空轨迹相似性查询算法。根据前面的工作可以计算出轨迹之间的距离，来支持对轨迹数据库的相似性查询，最终输出结果为轨迹数据库中与查询轨迹相似的轨迹数据。

实验使用了两个数据集，第一个是真实的汽车和行人的轨迹数据集，第二个是使用软件在北美路网上生成的数据集。实验主要探究算法中的参数对查询结果造成的影响，并与最新的研究成果进行了对比实验。在两个数据集上的实验结果表明，时空归一化轨迹表示模型是有效的，提出的时空轨迹相似性查询算法的准确性比前人的算法更高。

**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 order to improve the validity and accuracy of location-based services, it is necessary to analyze the trajectory data of moving objects, and the spatio-temporal trajectory similarity query of mobile objects is a key technology.

The purpose of the spatio-temporal trajectory similarity query is to obtain a trajectory data set similar to the query trajectory in the trajectory database. At present, a lot of work has been done on trajectory similarity query at home and abroad. Some work is devoted to the study of general trajectory similarity query algorithms, while others study the similarity query of trajectories in specific scenarios, but there are certain deficiencies. After analyzing the inadequacies of these algorithms, some new models and algorithms for trajectory similarity query are proposed. The specific work is as follows:

(1) After analyzing the spatio-temporal features of trajectory data, the spatio-temporal normalized trajectory representation model is proposed, which solves the problems in the integration of time dimension into similarity calculation in PTM and other algorithms.

(2) Based on DTW algorithm and BDS algorithm, a new corresponding point matching algorithm is proposed, so that the sample points can be well aligned to another trajectory and the timing of the matching results is guaranteed.

(3) A new method for calculating the distance of the trajectory segment is proposed. The trajectory segment distance consists of two parts: the spatio-temporal distance and the shape influence weight. In the calculation of spatio-temporal distance, the concept of breakpoint is proposed, which makes the calculation result less affected by the trajectory sampling strategy. In the shape influence weight calculation, the cosine distance and projection are used to quantify the shape similarity of the trajectory segment, so as to improve the accuracy of the trajectory segment distance calculation.

(4) A more accurate spatio-temporal trajectory similarity query algorithm is designed. According to the previous work, the distance between the trajectories can be calculated to support the similarity query algorithm, and the final output result is the trajectory data similar to the query trajectory in the trajectory database.

The experiment used two datasets, the first being a real car and pedestrian trajectory dataset, and the second being a dataset generated using software on a North American road network. The experiment mainly explores the influence of parameters in the algorithm on the query results, and compares the results with the latest research results. The experimental results on the two datasets show that the spatio-temporal normalized trajectory representation model is effective, and the accuracy of the proposed spatio-temporal trajectory similarity query algorithm is better than that of the previous ones.

**Key words:** location-based service; moving objects; spatio-temporal data; trajectory similarity

# 第1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[1-3]。

轨迹数据采样设备可以通过移动电话采集行人轨迹。通信公司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，并根据多次采集的数据去计算得到用户的平均移动速度和移动方向等。使用用户的移动数据，可以获得用户的大量个性化信息。如该用户经常去某家餐厅就餐，那么一些移动端应用可以按照用户爱好，为用户推荐餐厅。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。

采样设备还可以采集行车轨迹。很多出租车和私家车上都配置了车载GPS，车载GPS可以将汽车的轨迹上传到服务器，然后对大量出租车、私家车的轨迹数据进行挖掘，可以获得大量信息。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的路况[5-6]，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[7-9]，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[10]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、隐私保护和行为预测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[11-12]。

轨迹数据记载着移动对象在时间和空间中的移动历史，存储了空气、动物、车辆和人类的运动信息，在很多方面都有应用需求，这些应用都离不开高效的、准确的轨迹相似性查询算法，在移动对象的轨迹相似性查询算法中，轨迹距离计算函数是核心[13]。

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。国内外很多专家学者对轨迹相似性进行了深入的研究，提出了很多轨迹表示模型以及轨迹相似性查询方法。在空间上，有的研究基于欧式空间，有的基于路网。还有一些研究考虑时间维度，而另一些研究不考虑。本文考虑到欧式空间便于简化问题以及时间维度对轨迹相似性计算的意义重大，本文研究的相似性查询算法在欧式空间下，并且考虑时间对相似性查询的影响。

## 1.2研究内容

尽管在轨迹相似性查询方面已经有很多研究成果，但是仍然存在着很多问题。第一个问题是时空对应关系混乱，一些考虑时间因素的相似性查询算法在某些情况下会造成时空对应关系混乱，对于描述轨迹的时空相似性不够健壮。第二个问题是对应点匹配效果差，相似性算法的对应点匹配效果较差，会造成轨迹间距离计算不准确，影响最后的相似性查询结果。第三个问题是现有的一些相似性算法对采样策略比较敏感，不同的采样策略会采集到不同的轨迹数据，一些算法受轨迹数据的采样策略的影响较大，会直接影响对轨迹相似性的判断。为了解决上述三个问题，本文提出了一个新的轨迹表示模型和相似性查询算法，具体工作包括以下几个方面：

(1)提出了一个新的轨迹表示模型。通过对时间和空间维度的研究，寻找二者的关联，提出一个将时间维度与空间维度进行归一化的方法来构建轨迹表示模型。

(2)基于DTW算法和BDS算法中的对应点匹配思想，提出了一个新的对应点匹配算法，并解决了在时间和空间下引入DTW算法和BDS算法中对应点匹配方法存在的对齐效果不好以及匹配结果不保持时序性的问题。

(3)设计了一个更准确的相似性查询算法。使用一种全新的方法计算轨迹间的距离，包括时空距离和形状因素两个方面，并设计轨迹相似性查询算法，用于查找数据库中和查询轨迹最相似的子轨迹。

(4)算法实现及实验设计。实现时空轨迹相似性查询算法，并设计实验，使用两个轨迹数据集去验证算法的高效性和准确性。

## 1.3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章为相关工作，介绍了一些常用的轨迹表示模型以及以下现有的轨迹相似性度量函数，并分析其优缺点以及适用场景。

第3章分析了前人提出的轨迹表示模型和相似性度量函数存在的问题，与本文提出的模型和算法进行对比分析，并给出了问题定义。

第4章主要介绍了新的轨迹表示模型的设计，以及基于DTW算法和BDS算法的对应点匹配算法的设计。

第5章主要介绍了轨迹段时空距离以及形状影响权值的计算方法，提出了时空轨迹相似性查询算法。

第6章为实验部分。首先通过参数调节，研究本文提出的模型和算法中参数的影响，并与前人相关算法进行对比实验，验证本文提出的模型和算法的有效性和准确性。

第7章为总结与展望。首先总结本文工作，然后提出未来需要研究的方向。

# 第2章 相关工作

本章将介绍轨迹相似性查询的相关工作，包括四种轨迹表示模型和现有的轨迹相似性度量函数，并分析以上轨迹表示模型与相似性度量函数存在的优缺点及适用场景。

## 2.1轨迹表示模型

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，轨迹是通过对移动对象持续移动过程的采样获得的一系列离散点[7]，并且在不同的场景下，为了达到不同的目的，我们需要采用不同的轨迹表示形式。下面将介绍一些常用的轨迹表示模型。

### 2.1.1基于空间的轨迹表示模型(STR)

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Q = <, , …, >。在欧式空间下，点是一个坐标的形式，= <, >，表示经度，表示纬度。在路网空间下，可以将路网模型化为图的数据结构的形式，即Graph= (Vertex, Edge)，此时点表示的就是图Graph的顶点集合Vertex中的一个点。这种轨迹表示模型称为基于空间的轨迹表示模型(Space-based Trajectory Representation，STR)。

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算法将采样点映射到路网模型的顶点中[14]。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基于时间的轨迹表示模型(TTR)

轨迹信息有空间维度和时间维度，但是上面对轨迹的表述方法仅仅考虑了轨迹的空间维度，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，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。可以将含有时间信息的轨迹表示为Q = <(, ), (, ), …,(, )>，表示坐标或者顶点集合中的点，表示的是用户处于位置的时刻[15]。这种轨迹表示模型称为基于时间的轨迹表示模型(Time-based Trajectory Representation，TTR)。

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基于文本的轨迹表示模型(TETR)

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。可以表示为Q = <(, ), (, ), …,(, )>，其中表示坐标或者顶点集合中的点，表示位置的文本描述[16]。这种轨迹表示模型称为基于文本的轨迹表示模型(Text-based Trajectory Representation，TETR)

### 2.1.4基于网格的轨迹表示模型(CTR)

网格表示法也是一种常见的轨迹表示模型。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：Q={(cell, interval)}，其中cell是整个网格集合CELL中的一个网格，interval\_from和interval\_to分别表示进入和离开网格的时刻，(cell, interval)表示一个样本点[17]。这种轨迹表示模型称为基于网格的轨迹表示模型(Cell-based Trajectory Representation，CTR)。

## 2.2现有轨迹相似性度量函数

### 2.2.1欧氏距离(EU)

欧氏距离(Euclidean Distance，EU)[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。

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[17]，下面以求和的方式举例，并给出本文符号系统下的公式表示。给定两个一维的时间序列Q = <, , …, >，R = <, , …, >，序列Q和序列R的欧氏距离表达式如式2.1所示。欧式距离实际上是L*p*-norms在*p*=2情况下的一个特例。norms定义如式2.2所示。当*p*=1时，norms叫做曼哈顿距离。

(2.1)

norms()= (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[17]。

(2.3)

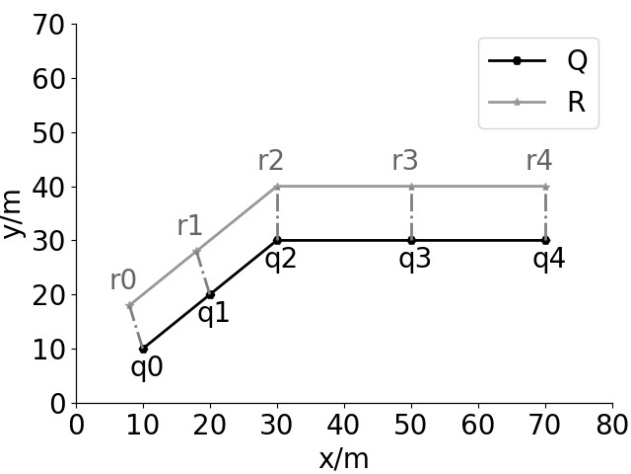


图2.1 欧氏距离

Fig. 2.1 Euclidean distance

欧氏距离的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 (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。Yi等人提出的动态时间弯曲(Dynamic Time Warping，DTW)[19-22]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。下面给出本文符号系统下的公式表示并根据公式对DTW算法进行分析。

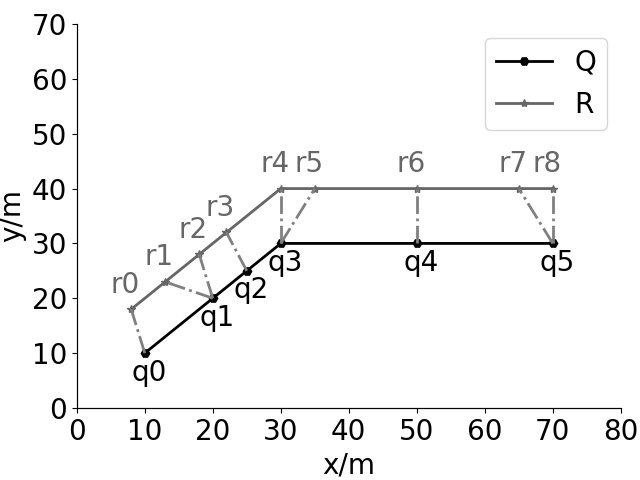


图2.2 动态时间规整

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Q和轨迹R的采样点的个数，即轨迹长度。Q.first和R.first表示轨迹Q和轨迹R的第一个采样点，而Q.rest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Q和R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。

(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，计算代价比欧氏距离大[18]。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 (LCSS)

顾名思义，最长公共子序列(Longest Common Subsequences，LCSS)[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。

(2.5)

其中：。

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，并且两段子轨迹的长度在一定的阈值以内，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，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，也需要的时间开销。

### 2.2.4编辑距离(EDR)

编辑距离(Edit Distance on Real Sequence，EDR)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[25]。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。

(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distedr得到的，如式2.8所示。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Q、R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。

(2.7)

(2.8)

### 2.2.5带真实惩罚的编辑距离(ERP)

带真实惩罚的编辑距离(Edit Distance with Real Penalty，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[26-27]。

(2.9)

(2.10)

### 2.2.6基于段的轨迹相似性计算(SDTW)

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2.3所示。



图2.3 不同采样方法造成的样本点错位

Fig. 2.3 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。由于传统轨迹相似性度量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[29]，表示为图2.4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。



图2.4 使用轨迹段面积表示轨迹距离

Fig. 2.4 Using trajectory area to represent trajectory distance

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2.5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2.5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，*p*1到seg2的距离和*p*2到seg1的距离来代表*p*1与*p*2之间的距离，如图2.6所示，其中为*p*1到轨迹S的点段距离。对应的距离公式如式2.11所示。



图2.5三角形面积替代阴影面积

Fig. 2.5 Triangle area instead of shadow area



图2.6用三角形的高代替三角形面积

Fig. 2.6 Replace triangle area with triangle height

(2.11)

其中，。

然后介绍预测距离。给定一组对应点和，时间戳分别为和，和不相等，如图2.7所示。



图2.7 预测距离

Fig. 2.7 Prediction distance

然后利用轨迹R在时刻的位置和与间的平均速度，来预测时刻的位置，得到，的预测距离计算公式如式2.12所示。

(2.12)

其中，。

那么这两个点的时间距离可以转化为时刻，轨迹R的位置到的距离。的预测距离计算公式如式2.13所示。

(2.13)

图2.8中，融合点段距离和预测距离，我们可以得到对应样本点和之间的时空距离公式，如式2.8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。

(2.14)



图2.8 利用夹角计算形状相似性

Fig. 2.8 Calculation of shape similarity based on included angles

利用样本点之间的时空距离，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，即段段距离，来计算轨迹段和轨迹段之间的形状相似程度，如式2.15所示。

(2.15)

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2.16所示，其中是两条轨迹段之间的夹角，和的计算公式如式2.17和式2.18所示。

(2.16)

(2.17)

(2.18)

前面得到了结合形状因素的轨迹段之间的时空距离，我们将其当做一个距离计算因素，代替DTW函数中使用的对应点之间距离，可以得到基于段的动态时间规整算法(Segment-based Dynamic Time Warping，SDTW)，如式2.19所示，其中R.first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，R.rest指的是出掉第一个轨迹段剩下的所有轨迹段。

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。

(2.19)

### 2.2.7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2.9所示。

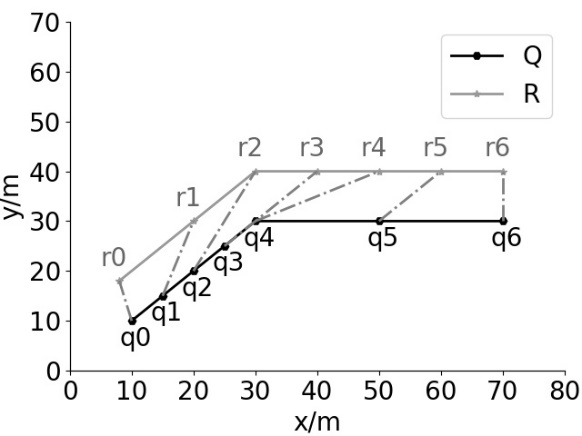


图2.9按时间匹配点对

Fig. 2.9 Match point by time

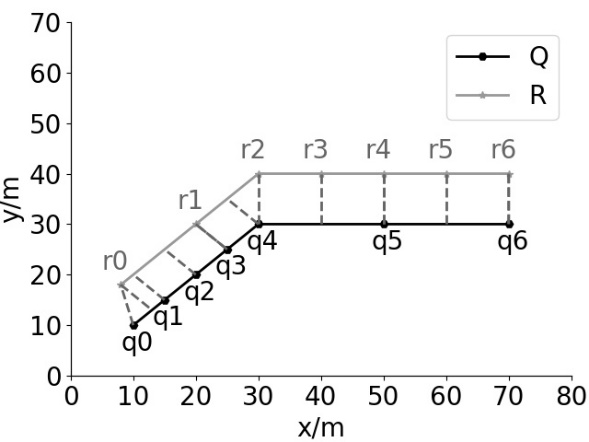


图2.10按点到轨迹段最小距离匹配

Fig. 2.10 Minimum distance matching from point to trajectory segment

因此Ta等人提出了双向映射相似性(Bi-directional Mapping Similarity，BDS)算法计算两条轨迹之间的相似程度[30]。与找轨迹的对应点的方法不同，BDS在计算轨迹Q和R的相似度时，通过累加Q的每个样本点到R的最短距离，如图2.10所示。下面根据我们的符号系统，给出BDS算法的公式。

这个最短距离的定义是将R上的所有样本点按照时间顺序连线，形成多条轨迹段，Q上的第*k*个样本点到这些轨迹段的最短距离就是Q上样本点到R的距离，其计算公式如式2.20所示。

(2.20)



图2.11点到轨迹段的距离

Fig. 2.11 Distance from point to trajectory segment

而Q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2.11所示。

BDS的计算公式如式2.21所示，其中是一个归一化的距离，最终结果是一个介于0到1之间的值，表示轨迹和轨迹之间的相似程度，值越大，表示两条轨迹越相似。

(2.21)

其中，。

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Q上第*k*个点到轨迹R的距离时，只需要计算点到以该点为中心的一定范围内的网格中存在的R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。

### 2.2.8空间网络中的个人轨迹匹配(PTM)

由于不同地点在不同用户心中的重要程度不同，所以为了使轨迹相似性查询结果更符合用户的预期，shang等人考虑到了这个问题，提出了个人轨迹匹配(Personalized Trajectory Matching，PTM)算法用于个人轨迹匹配[31]。

PTM算法使用TTR模型来表示轨迹。空间上的计算如公式2.22所示，是查询轨迹Q上的点和数据轨迹R上的点的空间距离，是和的空间距离影响因素，表示轨迹Q的第一个样本点的权值。然后基于LCSS算法[23-24]得到轨迹的空间相似性，如公式2.24所示。是和的时间差，是时间影响因素，计算如公式2.23和公式2.25所示。然后同样基于LCSS算法[23-24]得到时间相似性。最后使用参数将空间相似性和时间相似性结合，得到轨迹的时空相似性，如公式2.26所示。

(2.22)

(2.23)

(2.24)

(2.25)

(2.26)

该算法的优点是提出了样本点权值的概念，用于描述不同地点在不同用户那里的重要程度。但是该方法的缺点就是单独考虑时间和空间因素，在一些场景下可能会出现时间和空间上匹配的混乱，下个章节将详细说明这个问题。

### 2.2.9路网上的轨迹相似性计算

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。因此需要重新定义一个路网下的相似性函数[27]。



图2.12 真实路网

Fig. 2.12 Road network



图2.13 路网模型

Fig. 2.13 Model of road network

在计算相似性之前，需要将轨迹映射到路网上去，图2.12和图2.13来源于Yang等人的论文[27]，下面使用本文符号系统描述论文中的公式。给定移动对象的移动轨迹Q和R，轨迹格式为Q={(,,), (,,),…(,,)}，其中=(, )表示样本点坐标，表示对象在时刻的移动速度。我们用da(,R)表示从样本点到轨迹R的路网距离。不同条件下，对图中的轨迹进行相似性查询，我们有不同的距离函数[27]。

路网上Q和R的距离dN(Q, R)的计算公式如式2.27所示。其中，轨迹Q叫做查询轨迹，轨迹R叫做目标轨迹，m代表轨迹Q的样本点个数。

(2.27)

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，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2.28所示。

(2.28)

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SG代表当前道路最高限速，ds(, ri)代表轨迹Q上的第i个样本点与轨迹R上的样本点的速度之差，其中是轨迹B上到最近的样本点。带速度信息的轨迹距离公式如式2.29所示。

(2.29)

分析路况信息时，时间信息也是一个很重要的因素。我们令代表轨迹A上的第i个样本点与轨迹B上的第i个样本点的时间之差，其中是轨迹B上到最近的样本点。是轨迹Q的第一个点，是轨迹Q的最后一个点，是轨迹B上距最近的点，是轨迹B上距最近的点。带时间信息的轨迹距离公式如式2.30所示。

(2.30)

当结合权值、时间和空间来计算轨迹距离，我们有公式2.31，其中和分别代表对应的子计算方法的权值参数，并且两个。如果将速度信息加入公式，公式如式2.32所示。

(2.31)

(2.32)

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。

### 2.2.10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。分解条件就是用户的时间跨度，是一个时间阈值。分解后得到用户的小轨迹集，我们可以得到所有用户的小轨迹集。

然后两个小轨迹集合之间的重合程度来计算他们的相似程度。令代表用户的布尔型向量，的值如式2.33所示。

(2.33)

当比较用户Q和用户R之间的相似程度时，将和做与运算()，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性度量过程简单快速，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设置的好坏而变化，若设置过大，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。

## 2.4本章小结

本章主要介绍了四种常见的轨迹表示模型以及前人提出的一些轨迹相似性度量函数。有些相似性度量函数使用范围较为广泛，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，而另一些研究更加偏向于研究某个具体场景下的轨迹相似性。

# 第3章 问题分析及定义

第2章介绍了轨迹的四种轨迹表示模型以及一些相似性度量函数，本章将对本文提出的轨迹表示模型和相似性度量函数与前人的工作进行对比，分析各种方法的异同点，并提出问题定义。

## 3.1表示模型分析

在2.1节介绍了STR、TTR、TETR和CTR这四种轨迹表示模型，由于轨迹数据的来源是GPS采样数据，因此轨迹的表示模型大多基于样本点数据得到，例如STR、TTR和TETR这三种模型，而CTR模型以样本点所在网格表示该样本点，本质上也是使用的样本点来进行表示。因此这四种轨迹表示模型描述轨迹的最小粒度是相同的，都是由多个有限数目的样本点描述一条轨迹。这四种模型的区别在于样本点所包含的信息不同，如表格3.1所示，其中CTR模型中的时间信息为进入和离开网格的时间戳，而TTR模型中的时间信息为当前样本点的时间戳。TETR模型中虽然不包含时间信息，但是如果算法设计有需要，TETR模型支持对时间信息扩展。

当研究场景中同时涉及到时间和空间信息的时候，大部分研究会采用TTR来表示轨迹，而在轨迹时空相似性的研究中，时间和空间的相似性并不能单独考虑，TTR将时间维度与空间维度割裂开，给相似性度量方法的设计带来较大困难。在PTM算法中，时间相似性和空间相似性单独进行计算，最后使用二者线性组合的结果作为轨迹的时空相似性。这种将时间作为一个独立维度而不去结合空间信息的方法可能会导致时间与空间的混乱匹配，使最后结果没有任何意义，在下一章将给出详细案例及解释。

表3.1 轨迹表示模型数据信息表

Table 3.1 Trajectory representation model data information table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 表示模型 | 空间坐标 | 时间信息 | 语义信息 | 网格标号 | 时空坐标 |
| STR | ✓ | — | — | — | — |
| TTR | ✓ | ✓ | — | — | — |
| TETR | ✓ | — | ✓ | — | — |
| CTR | — | ✓ | — | ✓ | — |
| SNTR | — | — | — | — | ✓ |

为了解决TTR模型给相似性度量方法设计带来的困难，后面提出了时空归一化轨迹表示模型(Spatio-temporal Normalized Trajectory Representatio，SNTR)来解决这个问题。在SNTR模型的构建中，将时间和空间进行归一化处理，处理后得到的模型可以看做是一个三维时空，时间是三维时空的z轴，空间是三维时空的xoy平面。在SNTR模型下，两个点之间的距离包含了时间和空间上的距离，在相似性度量算法中可以统一计算轨迹的时空距离，不会再出现TTR模型可能导致的时空混乱的问题。在表3.1中列出了不同表示模型中最小粒度所包含的数据信息，其中“✓”表示具有该信息，“—”表示不具有该信息。

## 3.2度量函数分析

在相关工作中提到了很多轨迹相似性度量方法，下面将分析这些相似性度量方法与本文在后面提出了时空轨迹相似性查询算法(Spatio-tamporal Trajectory Similarity，STS)的异同点，如表3.2所示。

在各个算法中轨迹表示模型的选择会根据算法需要的信息而定。由于前五个算法仅考虑空间上的相似，不需要考虑时间信息，因此采用STR模型即可满足算法需要的信息。在SDTW需要使用时间计算平均速度，PTM算法考虑时空相似，因此采用TTR模型。本文提出的STS算法采用的是SNTR模型。

各个算法在空间选择上也有一定差异，除了PTM算法采用的是路网空间，其他算法都采用欧式空间。二者的区别是在路网中计算两点距离需要使用迪杰斯特拉算法进行计算，而欧式空间中可以之间计算两点的直线距离，因此路网和欧式空间的区别不是很大。

由于EU算法比较简单，要求参与相似性计算的两条轨迹样本点个数严格一致，并且样本点要按照时间顺序一一对应，导致该方法适用性和效果较差，适合一些精度要求不高的预处理工作。除了EU算法，其余算法都使用不同的对应点匹配方法寻找样本点的对应点，因此其他方法都允许进行相似性计算的两条轨迹的样本点数目不同。

时间信息引入轨迹相似性计算是一件比较有挑战的事情。本文认为时间信息与其它特征不同，时间上的相似需要和空间相似一同考虑。由于PTM算法采用的是TTR模型，在某些情况下，会产生时空对应关系混乱。STS算法采用的是SNTR模型，在对轨迹时空相似性的描述效果上比PTM算法更好。

轨迹数据由采样设备获取，采样策略不同会获取到不同的轨迹数据。针对不同采样策略获得的轨迹进行相似性计算，很多算法的计算结果受采样策略的影响较大，可能导致原本相似的轨迹，在某种采样策略下被算法认定为不相似，较多算法都没能解决该问题。在SDTW算法和STS算法中，主要考虑了轨迹段所包含的信息，减少对样本点的依赖，因此受采样策略的影响较小。在BDS算法中，由于使用了一种新颖的对应点匹配方法，可以获得更优的匹配结果。

不同相似性查询算法都需要进行轨迹的对应点匹配，轨迹距离和相似性需要基于这个中间结果计算得到，因此一个好的对应点匹配结果决定了最后的相似性计算结果的准确性和有效性。在BDS算法中，由于没有考虑时间信息，因此可能会出现对应点匹配结果时序混乱的情况。但是它的对应点匹配方法是一种十分有效的方法，因此STS算法使用BDS算法思想改进DTW算法中对应点匹配方法，得到了保持时序性的匹配结果，并且比DTW算法的对齐效果更好。

SDTW算法和STS算法考虑了轨迹的形状相似，这两种方法不同的地方在于SDTW算法使用了轨迹段的夹角，而STS算法采用轨迹段间的余弦距离[32]以及轨迹段的长度来描述形状相似，这样会比单纯考虑角度更加准确。

表3.2 相似性度量算法对比

Table 3.2 Comparison of similarity measure algorithms

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 相似性  算法 | 表示  模型 | 空间 | 不同样  本点数 | 时间  信息 | 抵抗采样  策略影响 | 匹配时序性 | 形状  信息 |
| EU | STR | 欧式 | — | — | — | ✓ | — |
| DTW | STR | 欧式 | ✓ | — | — | ✓ | — |
| LCSS | STR | 欧式 | ✓ | — | — | ✓ | — |
| EDR | STR | 欧式 | ✓ | — | — | ✓ | — |
| SDTW | TTR | 欧式 | ✓ | ✓ | ✓ | ✓ | ✓ |
| BDS | STR | 欧式 | ✓ | — | ✓ | — | — |
| PTM | TTR | 路网 | ✓ | ✓ | — | — | — |
| STS | SNTR | 欧式 | ✓ | ✓ | ✓ | ✓ | ✓ |

通过对各种相似性算法的纵向对比，得到了表3.2的内容，其中“✓”代表算法具有该优势或者功能，“—”表示不具有该优势或功能。与其他算法相比，STS算法有较多优势，后面将分章节介绍STS算法如何利用SNTR模型，实现这些优势。

## 3.3问题定义

轨迹数据中不但包含样本点的空间坐标信息，还包含时间信息，时间信息在分析轨迹相似性时是很重要的。

时间信息在自然科学的研究中是否重要。对候鸟迁徙的公共模式进行研究可以发现气候和环境的变化，候鸟只会在固定的季节迁徙，包含时间信息的轨迹相似性查询技术可以在候鸟迁徙轨迹中发现一些异常现象，比如一些候鸟在一个非迁徙的时间节点开始迁徙，或者一些候鸟停留在某个栖息地的时间的变化。因此时间信息在候鸟轨迹的相似性查询中有重要意义。

时间信息在分析交通情况时也有很大意义。如图3.1所示，查询轨迹Q是一辆匀速行驶的车辆的轨迹，数据轨迹R和S是另外两辆车的行车轨迹，其中轨迹R在点之前堵车了，而车辆C是匀速行驶的。如果仅从空间上判断，轨迹R与轨迹Q更相似。但是如果车辆A是在逃疑犯的车，警方为了获取目击证人，需要查询相似的轨迹。轨迹R在很长一段时间内被堵在点之前，因此不是希望得到的查询结果，反而与轨迹Q在空间距离上稍远一点的轨迹S是最希望得到的查询结果。因此在轨迹相似性查询中，时间信息是很重要的一个特征，需要考虑到相似性计算中。

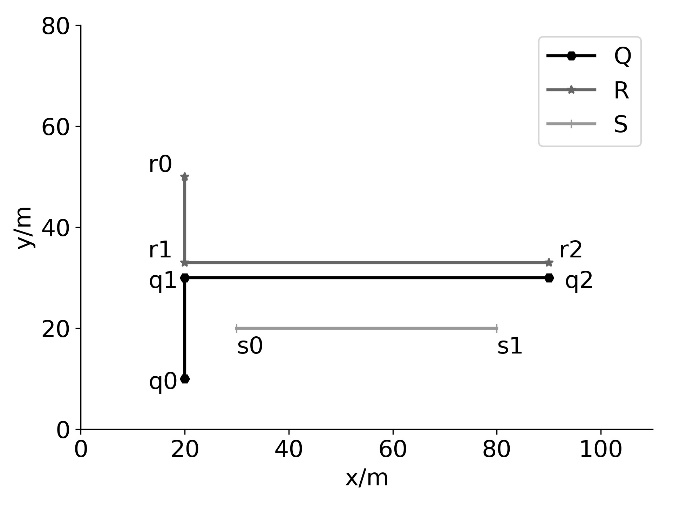


图3.1 轨迹相似性查询示例

Fig. 3.1 Example of trajectory similarity query

移动对象的真实路径包含了无穷多个点，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固定距离进行一次位置信息的记录，得到的信息叫样本点，因此最后得到的轨迹数据是由有限个采样点组成。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，采样点的空间信息包括横坐标x和纵坐标y。

样本点的格式为*p*=(*p.x*，*p.y*，*p.timestamp*)，样本点按照时间顺序组成轨迹T，T=<,,…,>。轨迹距离计算函数d(Q，R)是一个输入为Q和R，输出为Q和R之间的距离的函数。距离越小，代表相似度越大，两条轨迹越相似。轨迹相似性查询就是使用轨迹距离函数，在数据库中得到与查询轨迹距离小于某一阈值的轨迹数据。

**问题定义：**给定查询轨迹，以及一个存储轨迹的数据库，使用STS相似性查询算法得到数据库中与查询轨迹相似的轨迹。

本文称该问题为时空轨迹相似性查询问题。为了解决该问题，需要设计SNTR模型、对应点匹配算法、轨迹距离计算方法这三个部分，后面将对这三个部分详细介绍。本文使用到的符号如表3.1所示。

表3.3 符号定义

Table 3.3 The Symbol defination

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

## 3.4本章小结

本章首先分析了相关工作总提到的轨迹表示模型，从其包含的信息以及相似性算法设计的难度等角度进行了分析。然后分析了轨迹相似性度量函数，指出了各自的使用场景及异同点。在3.3节中给出了问题定义，阐述了本文主要解决的问题。

# 第4章 时空下对应点匹配算法

在第3章中提出了TTR模型存在的问题以及SNTR模型的优势，本章将在4.1节具体介绍SNTR模型的构建方法，并分析其如何解决TTR模型存在的问题。在4.2节介绍了对应点的相关概念，并提出一个基于DTW和BDS算法的对应点匹配算法。

## 4.1时空归一化轨迹表示模型

### 4.1.1存在的问题

之前有很多研究人员对时空轨迹相似性查询进行研究，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[31]。PTM算法的主要思想在相关工作中已经介绍，这里不再赘述，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4.1，图中可以看出在空间上轨迹R中至段和查询轨迹Q很相似，轨迹S中的至也和查询轨迹Q很相似，但是R和Q的距离更近一点。

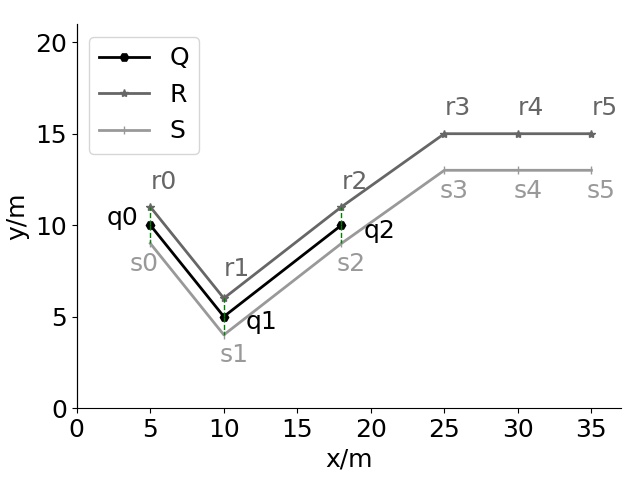


图4.1 轨迹在空间维度的情况

Fig. 4.1 Trajectory in spatio

三条轨迹的时间维度可以参考图4.2，该图纵轴代表从8点开始的分钟数，比如处的y值为15，代表处的时刻为8点15分。单独从时间维度上看，轨迹R在至内的时间戳和轨迹Q的更加吻合，而轨迹S在至内的时间戳和轨迹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，，因此Q和R的相似度更高。

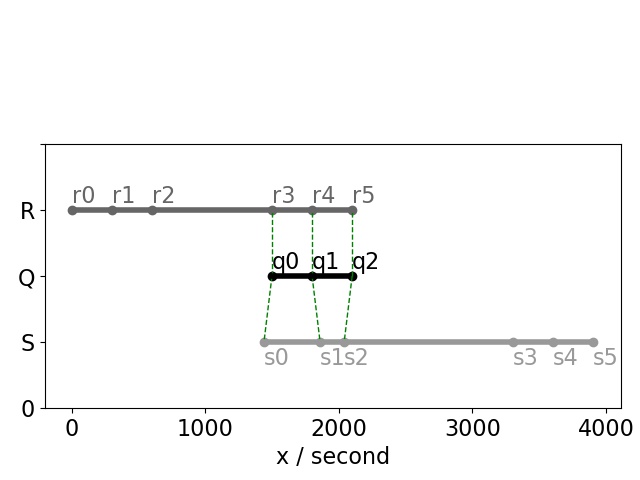


图4.2 轨迹在时间维度的情况

Fig. 4.2 Trajectory in temporal

PTM算法的相似性查询结果为轨迹R，这条轨迹的在空间上与Q相似，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和时间相似性，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到的距离，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。因此不能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。

### 4.1.2模型设计

根据第3章对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，时间的单位是秒，空间的单位为米，因此我们不能直接将时间纳入轨迹的时空距离的计算中。在图3.1描述的场景中，若因为堵车使轨迹R停留了一段时间，那么这段时间轨迹Q行驶的路程便是二者时间上造成的距离，因此需要以Q的速度作为时空转化因素，将时间维度向空间距离上做转化。

在上面那种对时间要求比较高的情况下，需要一个较大的数值，才能提升时间在最后相似性计算中的重要程度。如果应用场景不同，时空转化因素也会不同。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素可以取一个较小的值。

然后就可以使用该转化因素将时间t转化为空间中的z，如公式4.1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。

(4.1)

上面的方法就是时空归一化方法，得到了SNTR模型，可以将轨迹的时间和空间因素进行结合。时空归一化方法的具体伪代码如算法4.1所示。将查询轨迹和数据库中的所有轨迹trajectories依次遍历，对每一条轨迹数据进行时空归一化，最后会获得SNTR模型下的轨迹数据。

|  |  |
| --- | --- |
| **算法4.1 时空归一化** | |
| **输入**：原始轨迹数据trajectories，时空转化因素 | |
| **输出**：SNTR模型中的轨迹数据trajectories\_3d，时空转化因素 | |
| 1. | trajectories\_3d=[] |
| 2. | **for** trajectory |
| 3. |  |
| 4. | trajectories\_3d.add([]) |
| 5. | **end for** |
| 6. | **return** trajectorie\_3d |

使用SNTR模型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，点和的时空距离如公式4.2所示。后面如果没有指明，则默认表示点和的时空距离。将带入后，得公式4.3。

(4.2)

(4.3)

时空转化因素的作用是将时间转换为空间，而在其中使用轨迹平均速度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SNTR模型中，xyz三个维度的单位均为米。在SNTR模型中，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SNTR模型以及时空距离的定义之后，下面我们再看看4.1.1节中PTM算法未解决的问题。在这里我们令时空转化因素，再由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4.3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。

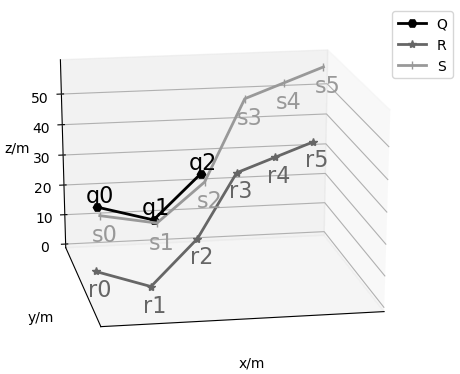


图4.3 轨迹在三维时空的情况

Fig. 4.3 Trajectory in Three dimensional spatio-temporal

## 4.2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，并改进前人算法中不适用于SNTR模型下的不足之处。

### 4.2.1存在的问题

首先介绍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4.4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。

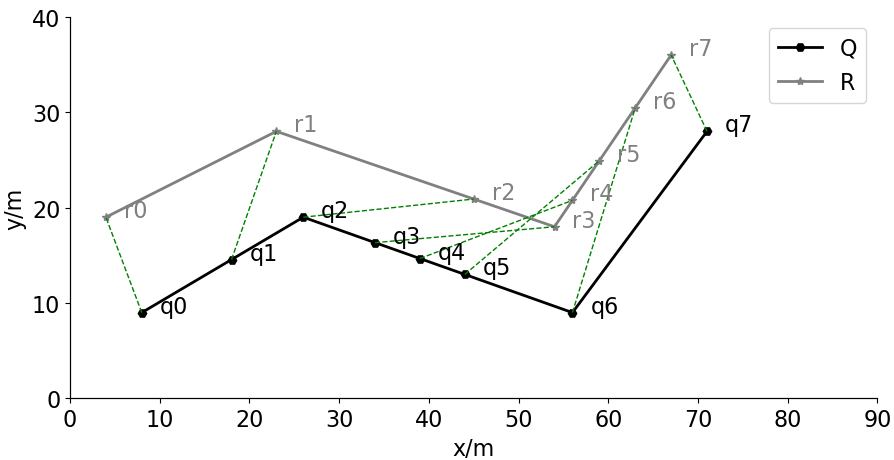


图4.4 按时间戳顺序寻找对应点

Fig. 4.4 Finding corresponding points in timestamp order

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4.5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。

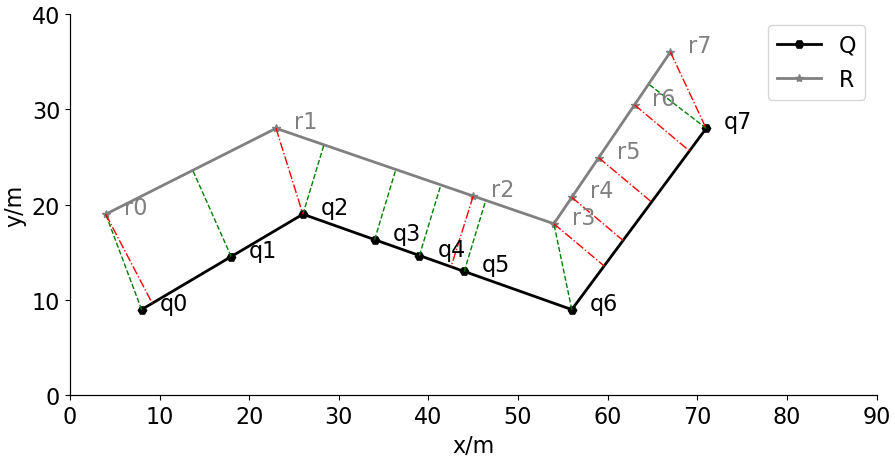


图4.5 按最近点寻找对应点

Fig. 4.5 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SNTR模型下，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在SNTR模型下直接使用BDS算法中的对应点匹配思想可能会出现的问题。

二维空间下的轨迹Q和R如图4.6所示，图4.7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4.7中可以看出，与有对应关系，与有对应关系。如果从匹配结果来看，轨迹R的至都与轨迹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，即至对应至，至对应至。因此BDS算法中，与轨迹R进行相似性计算的轨迹序列为<>，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。

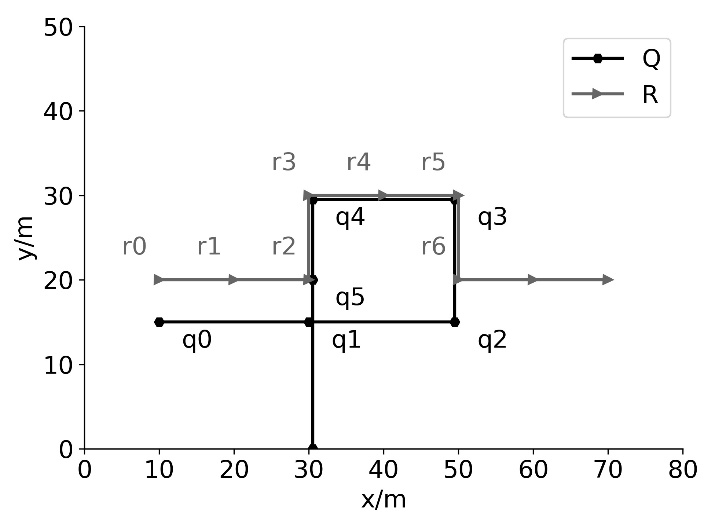


图4.6 二维空间下的轨迹

Fig. 4.6 Trajectories in two-dimensional spaio

在考虑SNTR模型下，BDS中的样本点匹配方法会导致时序错乱，因此不能直接用于解决SNTR模型下的样本点匹配问题。

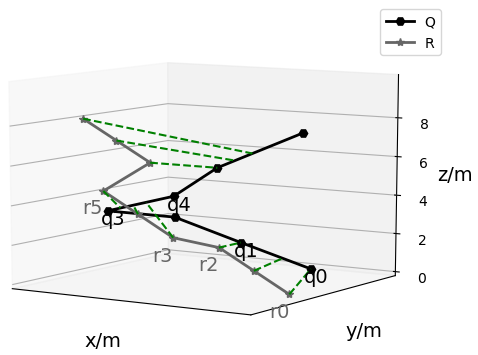


图4.7 BDS算法匹配结果

Fig. 4.7 Matching results of BDS algorithm

DTW[19]算法在寻找最优的对应点匹配策略时，采用了动态规划思想，按时间顺序从前往后匹配。在SNTR模型下使用DTW算法中的对应点匹配思想不会产生时序错乱问题。如果使用DTW算法解决刚才的问题，那么对应点匹配结果如图4.8所示。样本点匹配关系为：和对应，对应，对应，和对应，完全按照时间先后顺序来对应，在至处没有出现Q上对应点顺序的翻转，得到了一个相对较优的对应点匹配方案，可以用于解决在SNTR模型下使用BDS算法中对应点匹配思想可能造成的时序错乱的问题。但是由于DTW算法只允许一个样本点匹配到另一条轨迹的样本点上，对其效果较差，并且在SNTR模型下匹配策略受采样方法和样本点提取算法的影响较大，在SNTR模型下直接使用DTW算法中的对应点匹配思想不是一个最好的方法。

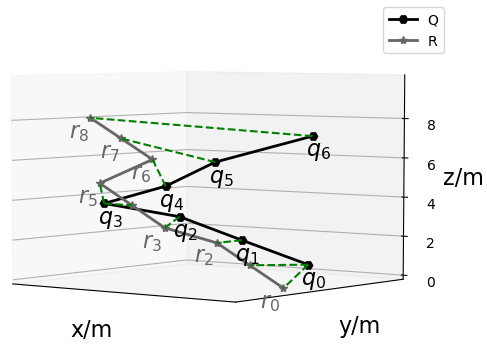


图4.8 DTW算法匹配结果

Fig. 4.8 Matching results of DTW algorithm

### 4.2.2算法设计

根据上面对DTW和BDS算法的分析，本节给出DTW和BDS结合的对应点匹配方法，保留二者优势，并解决将二者引入SNTR模型下存在的问题。

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，我们记的所有DTW对应点为DTW()，将DTW()中时间戳最早的记为DTW().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。假设的时间戳小于的时间戳，将使用BDS算法思想计算在间的对应点的过程记为BDS(,,)，这个称之为带上下界的BDS对应点匹配方法，时间戳较小的是BDS匹配方法的下界，是BDS匹配方法的上界。

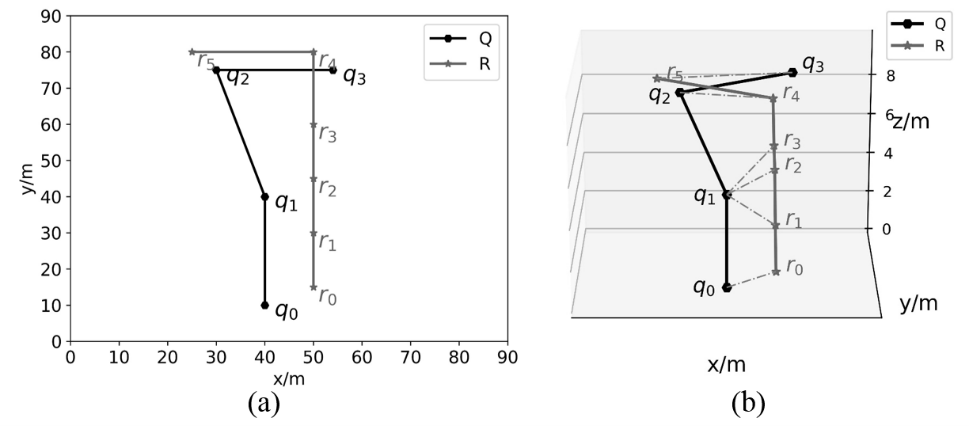


图4.9 轨迹示例

Fig. 4.9 Example of trajectories

我们使用图4.9为例进行讲解，图4.9(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4.9(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，将优化后的点记为DTW-BDS()。按照时间戳顺序，对R上所有点从前往后依次优化。

首先优化的对应点，在至DTW().first之间的子轨迹上求的BDS对应点，将其作为在轨迹Q上的DTW-BDS对应点，如公式4.4所示。如果恰好的第一个DTW对应点DTW().first就是，那么BDS()=。图4.10显示的是对应点的优化。

DTW-BDS()=BDS(,, DTW().first) (4.4)

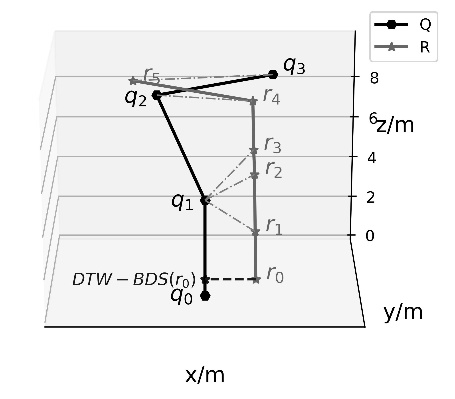


图4.10 的BDS优化结果

Fig. 4.10 BDS optimization result of

然后优化轨迹R中间的样本点的对应点。当是轨迹R中间的某个点，且在前面已经获得了的对应点DTW-BDS()，可以求在DTW-BDS()与DTW().first之间的BDS对应点，将其作为的DTW-BDS对应点，如公式4.5所示。图4.11(a)显示了、、对应点的优化。

DTW-BDS()=BDS(, DTW-BDS(), DTW().first) (4.5)

在获得了DTW-BDS()后，如果DTW-BDS()的时间戳比DTW().first大，此时需要重新计算的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()的时候，还没有求DTW-BDS()，因此使用DTW().first作为上界。当求出DTW-BDS()后，若发现DTW().first的时间戳小于DTW-BDS()的时间戳，的DTW-BDS对应点的上界应该要增大为DTW-BDS()，然后再次使用带上下界的BDS算法更新DTW-BDS()。若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4.11(a)所示，已经计算得到了DTW-BDS()，发现DTW-BDS()的时间戳大于DTW ()即的时间戳，此时需要进行前向更新操作。首先更新DTW-BDS()，将其上界定为DTW-BDS()，通过上下界的BDS对应点计算，得到了图4.11(b)中的DTW-BDS()。由于更新后的DTW-BDS()与更新前不是同一个点，需要以DTW-BDS()为上界更新DTW-BDS()，发现DTW-BDS()更新前后为同一个点，因此停止前向更新，最后结果如图4.11(b)所示。

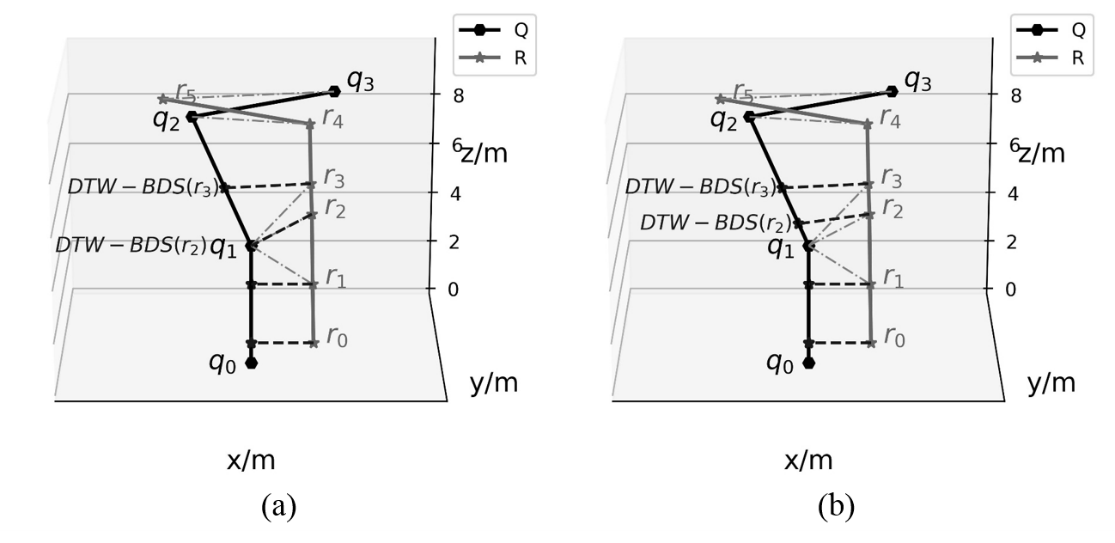


图4.11 中间对应点的优化

Fig. 4.11 Optimization of corresponding points in the middle

最后对R的最后一个点的对应点进行优化，求出在BDS()与轨迹Q的最后一个点之间的BDS对应点作为的DTW-BDS对应点，如公式4.6所示。最后完整的DTW-BDS算法得到的匹配结果如图4.12所示。

DTW-BDS()=BDS(, BDS(), ) (4.6)

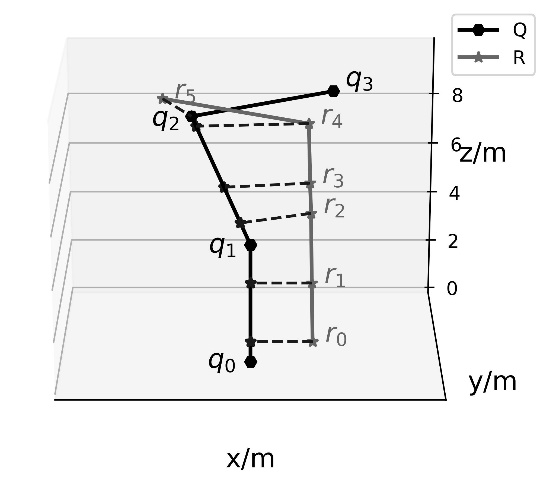


图4.12 DTW-BDS算法最终匹配结果

Fig. 4.12 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4.2是DTW-BDS对应点匹配算法的伪代码。

现在再看本文4.3.1节图4.7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寻找DTW-BDS对应点，可以得到如图4.13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。样本点原本匹配到了，通过局部优化，找到了距离和之间距离最近的点最为对应点，此时到DTW-BDS()的距离小于到DTW()的距离。除了之外，还有、、和不再对应到轨迹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、、和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。

|  |  |
| --- | --- |
| **算法4.2 DTW-BDS对应点匹配算法** | |
| **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 | |
| 1. | pair DTW算法匹配结果 |
| 2. | **for** each in R |
| 3. | **if** 是R的第一个样本点 |
| 4. | Q()= BDS(,, DTW().first) |
| 5. | **else if** 是R中间的样本点 |
| 6. | Q()= BDS(, DTW-BDS(), DTW().first) |
| 7. | **else** |
| 8. | Q()= BDS(, BDS(), ) |
| 9. | r = |
| 10. | q\_old = pair.getValue() |
| 11. | q\_new = Q() |
| 12. | pair (, Q()) |
| 13. | **while** q\_new.timestamp > q\_old.timestamp and r.pre != null |
| 14. | r = r.pre |
| 15. | q\_old = pair.getValue(r) |
| 16. | q\_new = BDS(r, DTW-BDS(r.pre), DTW-BDS(r.next)) |
| 17. | pair (r, q\_new) |
| 18. | **end while** |
| 19. | **end for** |

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，匹配到了，出现了时序错乱。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()与DTW()确定的范围内寻找的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。因此在SNTR模型下的DTW-BDS算法是行之有效的。

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，我们将在轨迹Q上的DTW-BDS对应点DTW-BDS()记为。

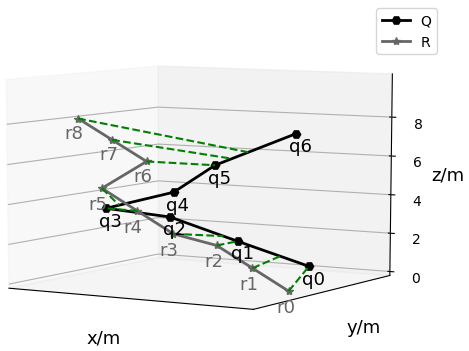


图4.13 DTW-BDS算法匹配结果

Fig. 4.13 Matching results of DTW-BDS algorithm

### 4.2.3轨迹段构建

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和，会在查询轨迹Q上分别获得他们的对应点和，那么轨迹段的对应轨迹段就是。轨迹段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，如图4.14所示。

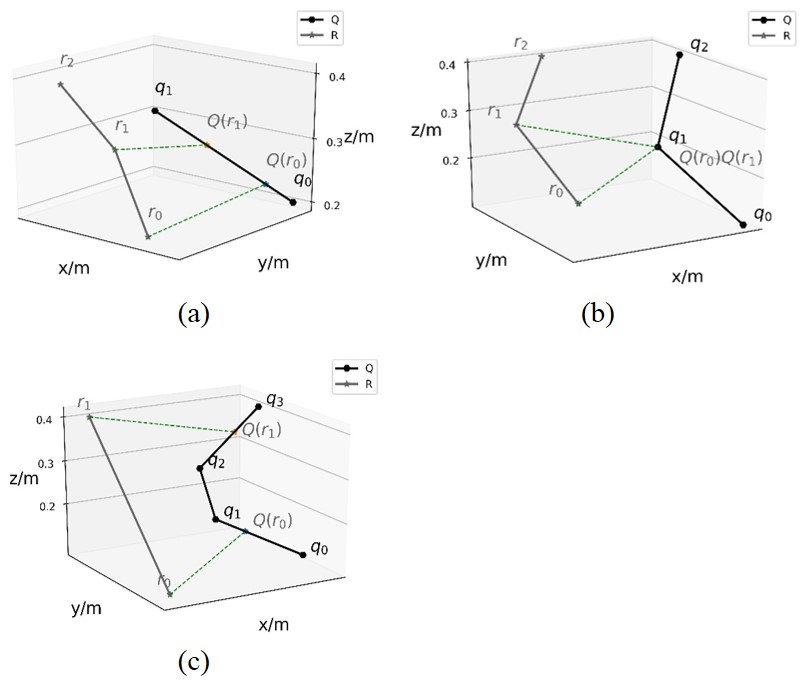


图4.14 对应轨迹段的三种情况

Fig. 4.14 Three cases corresponding to trajectory segments

第一种情况是和不是同一个点，并且除了和之外，轨迹段中不包含任何一个样本点，而和有可能是轨迹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4.14(a)所示，的对应点是轨迹Q的样本点，的对应点在样本点和之间。

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，和可能会对齐到Q上的同一个点，即和有可能是轨迹Q中的同一个样本点，如图4.14(b)所示，和的对应点和为同一个点，并且该点还是轨迹Q的样本点。

第三种情况是虽然和是两个连续的样本点，但是和中间隔着一个或多个样本点。图4.14(c)是第三种情况中的一个例子，和的对应点和中间夹着轨迹Q的样本点和。

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。

## 4.3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DTW算法和BDS算法各自的优点以及将这两个算法引入SNTR模型下可能会出现的问题，随后提出了SNTR模型下的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5章 基于轨迹段的轨迹相似性查询

在第4章中提出了SNTR模型和以及在SNTR模型下的DTW-BDS对应点匹配算法，并由对应点匹配结果得到对应轨迹段。本章将介绍轨迹段距离的计算方法，包括时空距离和形状影响因素两部分，由轨迹段距离可以计算得到轨迹距离。根据之前的所有工作，提出STS轨迹相似性查询算法。

## 5.1对应轨迹段的时空距离

### 5.1.1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SNTR模型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SNTR模型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5.1(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5.2(a)和图5.2(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。

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5.1(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。

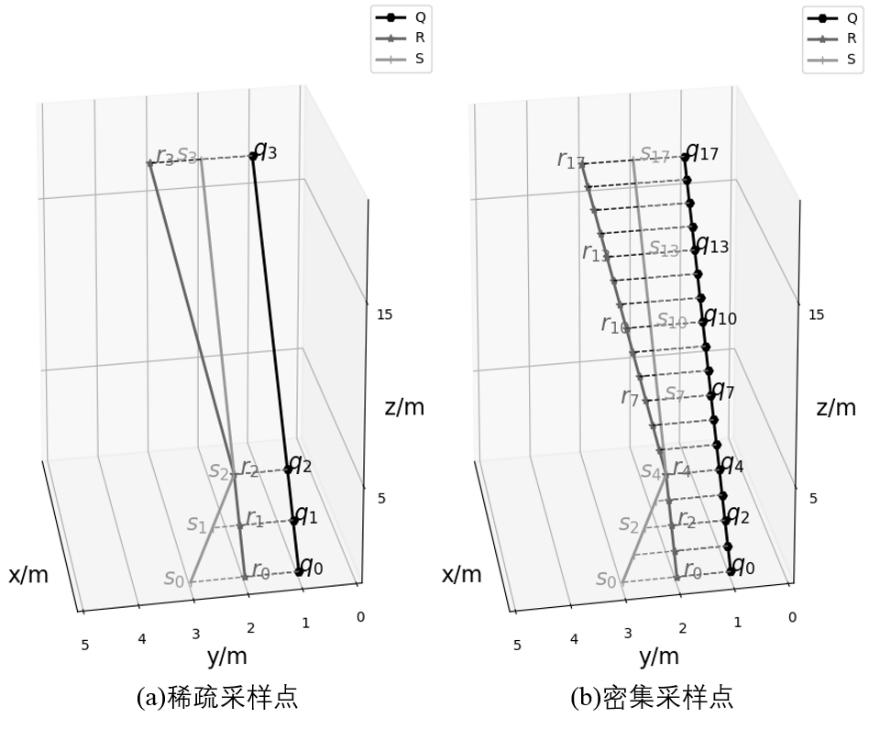


图5.1 DTW算法计算轨迹距离示例

Fig. 5.1 Example of DTW algorithm to calculate trajectory distance

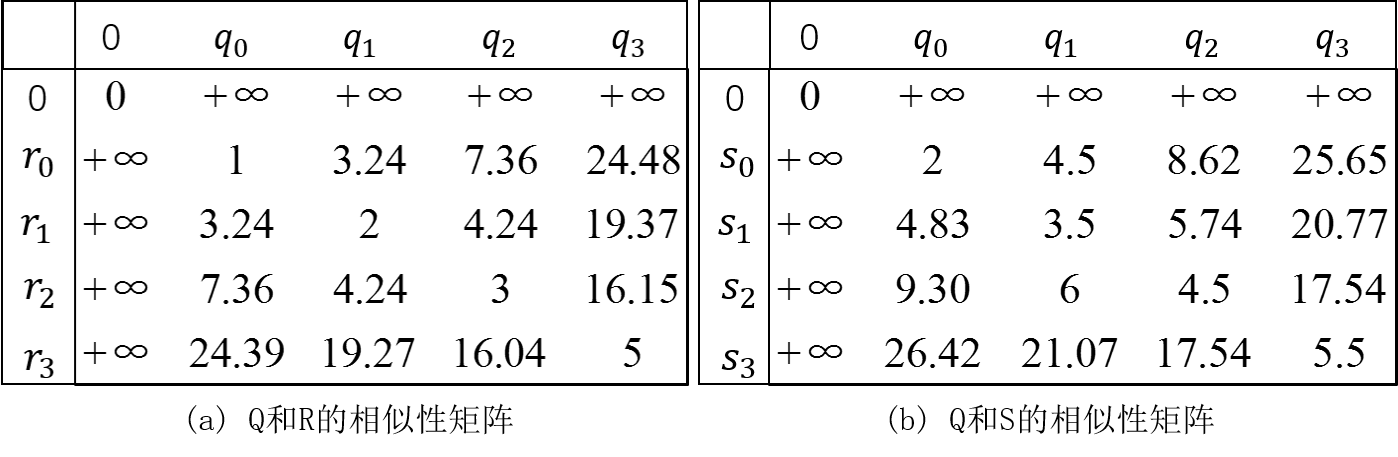


图5.2 DTW相似性矩阵

Fig. 5.2 DTW similarity matrix

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5.1(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。

### 5.1.2断点匹配

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将两个相邻样本点之间的轨迹段均匀分割，当阈值小于轨迹段长度时，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，由于轨迹段存在于SNTR模型，因此断点阈值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的大小。在轨迹段上，从样本点开始，每隔距离取一个断点，由于轨迹段的长度可能不是的整数倍，最后一个断点到的距离可能会小于，最终会获得d()条轨迹段，以及轨迹段上所有断点的集合BP()。如图5.3(a)所示，是查询轨迹R中的一条轨迹段，断点阈值将轨迹段划分为四小段，得到了六个断点，其中前三段一直到的长度均为，而末尾的小段的长度小于。

通过调整阈值的大小，将一条长轨迹段可以分隔为更短的轨迹段。而由5.1.1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。轨迹段是相邻样本点连接而成的一条直线段，但是其对应轨迹段存在图5.3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5.3(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，因此上所有断点的对应点都是。

对于图5.3(a)所示的第一种情况和图5.3(c)所示的第三种情况，虽然轨迹段上的所有断点在同一条直线上，但是其对应轨迹段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和之间，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。

下面介绍上所有断点的匹配过程。首先将和其对应轨迹段视作完整的轨迹，作为DTW-BDS算法的输入，并将中的所有断点以及和视为样本点。算法输出为的和的对应点以及上所有断点的对应点，保存所有断点的对应点，并将断点的对应点记作。

### 5.1.3时空距离计算

本小节介绍轨迹段间时空距离的计算。由于对应的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。

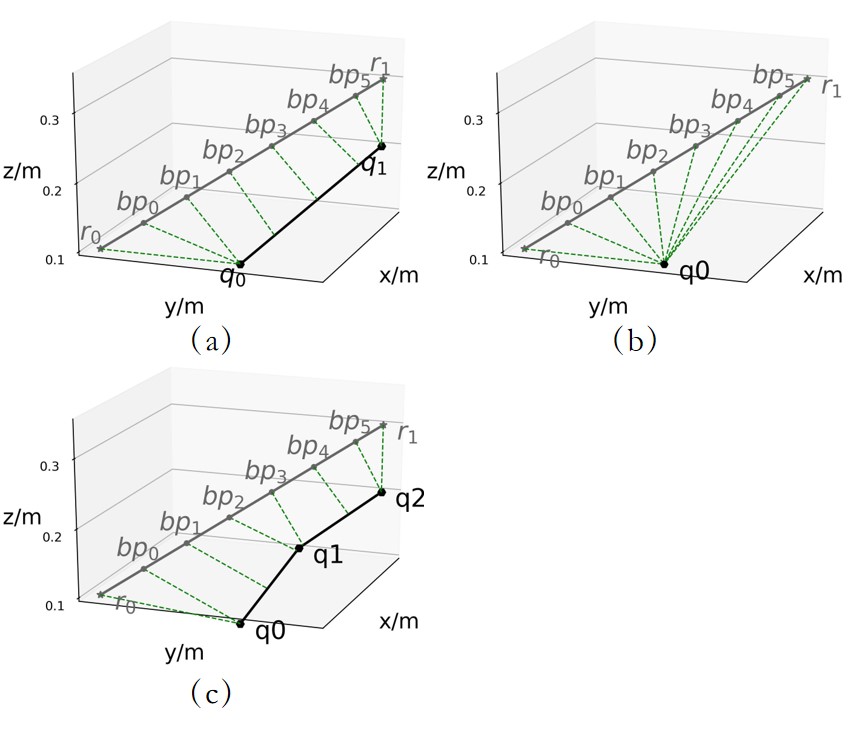


图5.3 不同对应轨迹段中断点的对应点

Fig. 5.3 Corresponding points of break points in different corresponding trajectory segments

先讨论第一种情况，即图5.3(a)中所展示的轨迹段是一条直线段。首先根据断点阈值计算得到上的所有断点，然后根据上一节中给出的方法，使用DTW-BDS样本点匹配算法，将和作为输入，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，得到和。

下面的问题就是如何将轨迹段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。如果趋向于无穷小，那么会在轨迹段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5.4(a)所示。

如果将增大一点，就变成图5.4(b)中的情况，因为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，与断点阈值成正比。

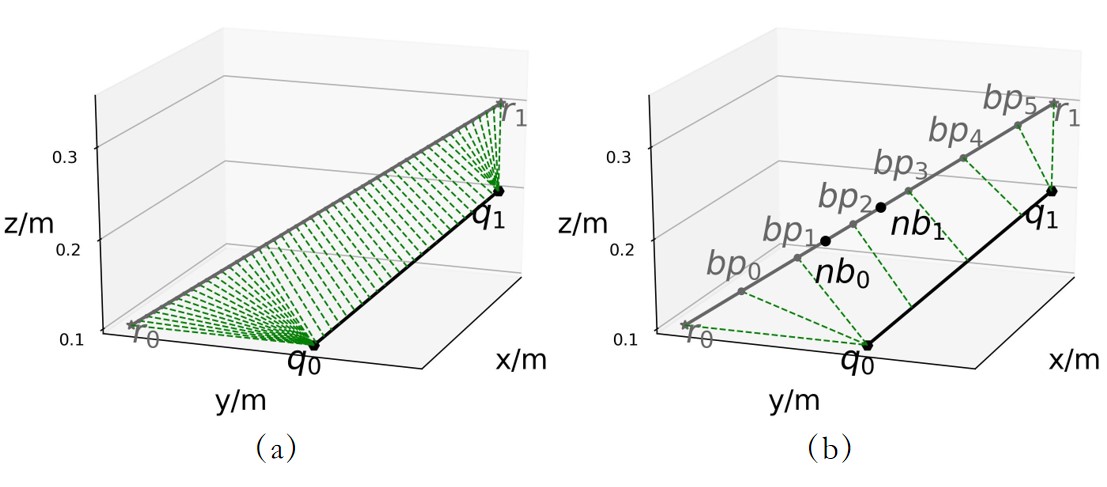


图5.4 断点代表的段

Fig. 5.4 Segment represented by breakpoint

在图5.4(b)中，可以使用代替从到间的所有消失的断点，与分别是断点到相邻断点的中点。断点代表的消失断点数目与的长度成正比，将称为断点代表的段。左端点代表的段的长度为，断点到代表的段的长度均为断点阈值，代表的段的长度为，右端点代表的段的长度为。可以同时消去所有代表段的长度中的，可以得到的权值为1/2，到的权值均为1，权值为，权值为。

(5.1)

其中，为断点的个数。

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到对应轨迹段的距离公式，如公式5.1所示。

当出现第二种空间分布情况，即与为同一个点，如图5.3(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，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。算法5.1是对应轨迹段时空距离计算的伪代码。

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。

|  |  |
| --- | --- |
| **算法5.1 轨迹段时空距离计算** | |
| **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 | |
| **输出**：数据轨迹所有轨迹段到对应轨迹段的时空距离dss\_list | |
| 1. | **for** |
| 2. | **if** |
| 3. | 获取轨迹段上所有断点 |
| 4. | 获取和的对应点和 |
| 5. | 计算所有断点的对应点 |
| 6. | d=，， |
| 7. | w=[1/2, 1, 1,…,1, , ] |
| 8. | **else if** |
| 9. | d=， |
| 10. |  |
| 11. | dss \_list 🡨 |
| 12. | **end for** |
| 13. | **return** dss \_list |

## 5.2对应轨迹段的形状影响因素

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。

### 5.2.1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5.5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，余弦距离的相关定义参考了别人的文章[32]。三维空间中向量和向量的余弦距离[32]如公式5.2所示，其中和表示向量模长。在轨迹段余弦距离中，的取值范围为[0, ]，的取值范围为[-1,1]。

, (5.2)

如果把上面的公式做一下变换，两边同时乘以向量的模长，如公式5.3所示，得到的是映射后的的长度，为在方向上的映射。

(5.3)

由于余弦距离在中是一个单调递减的函数，因此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。

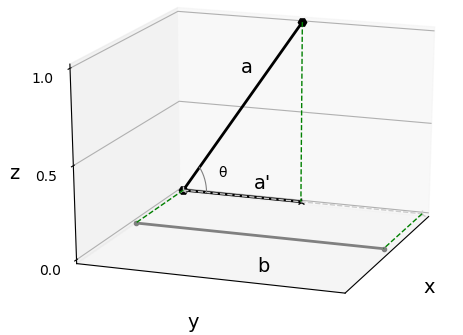


图5.5 异面直线的夹角

Fig. 5.5 Angle between different straight lines

### 5.2.2轨迹段形状相似性

在图5.6(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，但是投影后的的长度小于，反映了夹角越小，形状越相似。在图5.6(b)中，OA和OB与x轴的夹角相同，但是OA更长，导致的长度大于，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5.6 影响形状相似性的两个因素

Fig. 5.6 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，我们可以使用表示轨迹段与其对应的直线轨迹段的形状相似性。

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，即比的长度大很多倍时，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，为映射到方向上的距离，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，我们将的长度规定为轨迹段形状相似性的上限，即形状相似性允许的最大激励不得超过轨迹段的长度，再大的部分我们认为是无效部分。

根据上面的讨论，这里给出与对应轨迹段的轨迹段形状相似性的计算公式，如公式5.4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，二者相似性越高。

(5.4)

(5.5)

|  |  |  |
| --- | --- | --- |
|  | =+  + | (5.6) |

上面讨论了轨迹段的对应轨迹段是直线段的情况，我们还需要对另外两种情况进行讨论。

当出现图5.2(b)中的情况，即和是轨迹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5.7所示，点处的方向为。

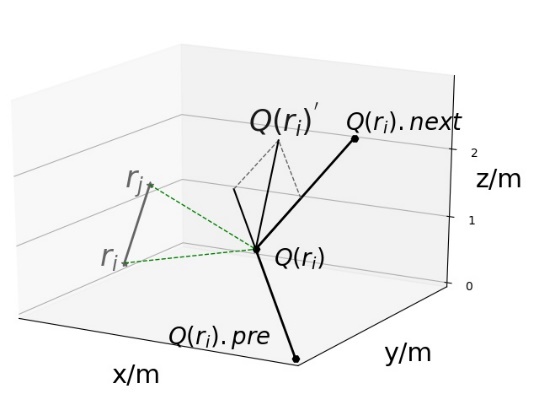


图5.7 的方向

Fig. 5.7 The direction of

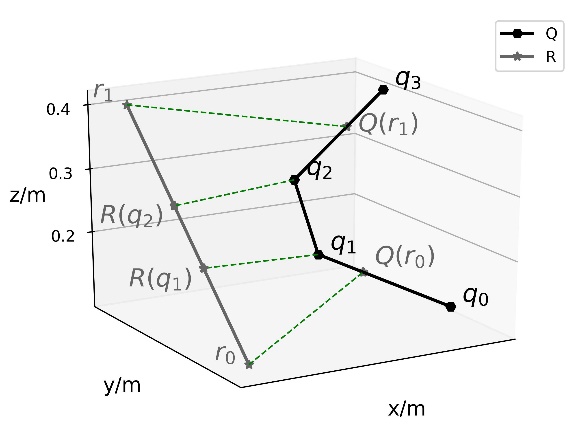


图5.8 第三种轨迹段的形状相似性计算示例

Fig. 5.8 Example of calculating the shape similarity of third trajectory segments

计算具体的值时，首先需要找到轨迹中与相邻的两个样本点，分别记为和，是处的向量，的值为前后两个轨迹段的单位向量的和，此时的方向就是前后两个方向的均值，如公式5.5所示，表示向量方向上的单位向量。结合公式5.5可以看出当和的夹角是一个锐角时，由于受限于为0，所以轨迹段形状相似性为0，当和处的夹角是一个钝角时，轨迹段形状相似性小于0。

当出现图5.2(c)中的情况，即和轨迹段中间包含一个或多个样本点，此时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5.8中，Q()和Q()中间间隔着和，此时需要使用DTW-BDS样本点匹配算法，输入轨迹段为Q() Q()和，将和视作样本点，寻找轨迹R上的对应点R()和R()。然后将、和视为三个独立的轨迹段，其对应轨迹段分别为、和。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，得到轨迹段和之间的轨迹段相似距离，如公式5.6所示，其中，至是与之间间隔的样本点，且m<n。

|  |  |
| --- | --- |
| **算法5.2 轨迹段形状性计算** | |
| **输入**：样本点，及其对应点, | |
| **输出**：与的形状相似性sim\_shape | |
| 1. | **if** 和间无样本点或者二者为同一个点 |
| 2. |  |
| 3. | **return** |
| 4. | **else** |
| 5. | **for** 中的每条轨迹段 |
| 6. | R() = DTW-BDS() |
| 7. | R() = DTW-BDS() |
| 8. | sim\_shape += sim\_shape\_calculate(R(), R(), , ) |
| 9. | **end for** |
| 10. | **return** sim\_shape |

轨迹段形状相似性计算的伪代码如算法5.2所示，其作用是计算对应轨迹段之间的形状相似性，便于后面计算形状影响权值。轨迹段相似性计算的伪代码如下，算法中将分别对三种轨迹段情况进行讨论，当对应轨迹段之间没有样本点，或者二者为同一个样本点时，处理情况是一样的，当对应轨迹段之间包含多个样本点时，这时需要单独进行计算。

### 5.2.3轨迹段的影响权值

为了描述轨迹段之间形状相似性为轨迹距离带来的影响，这里提出了基于sigmoid函数s(x)的形状影响权值的概念。公式5.7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，将变量映射到0与1之间，函数形状如图5.9所示。

(5.7)



图5.9 sigmoid函数图像

Fig. 5.9 Sigmoid function image

轨迹段形状相似性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权值表示的是形状的差异程度，即形状越不相似，形状影响权值越大。但由于形状相似性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权值。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，由于形状影响权值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权值的计算公式，如公式5.8所示, 的函数图像如图5.10所示。其中形状敏感度参数可以调节轨迹相似性对轨迹段形状的敏感程度，越小，表示轨迹相似性对形状因素越敏感。

(5.8)

其中，。



图5.10 函数图像

Fig. 5.10 function image

下面给出所有轨迹段之间的形状影响权值计算的伪代码，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权值。

|  |  |
| --- | --- |
| **算法5.3 轨迹段形状影响权值计算** | |
| **输入**：查询轨迹Q，数据轨迹R，R的对应点对pair，形状影响权值的权重 | |
| **输出**：数据轨迹所有轨迹段到对应轨迹段的形状影响权值I\_shape \_list | |
| 1. | **for** in R |
| 2. | sim\_shape 计算与)的形状相似性 |
| 3. | I\_shape= |
| 4. | I\_shape \_list.add(I\_shape) |
| 5. | **end for** |
| 6. | **return** I\_shape \_list |

## 5.3轨迹相似性查询算法

### 5.3.1轨迹段间距离

通过对对应轨迹段形状相似性和时空距离的讨论，得到了形状影响权值以及对应轨迹段间的三维时空距离。下面我们给出对应轨迹段距离的计算公式，如公式5.9所示。与轨迹形状影响权值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。

|  |  |  |
| --- | --- | --- |
|  | = | (5.9) |

### 5.3.2算法设计

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进行轨迹之间的距离计算。轨迹间的距离d(Q,R)的计算方法是计算轨迹R上的所有轨迹段到对应轨迹段的距离之和，如公式5.10所示。

|  |  |  |
| --- | --- | --- |
|  |  | (5.10) |

算法5.4给出了STS算法的伪代码。在STS算法的查询过程中，第一步是使用时空转换因素构建SNTR模型，将查询者发送的查询轨迹和数据库中的数据轨迹转化为SNTR模型下的数据。第二步是对数据轨迹集合中的每一条数据轨迹R进行遍历，首先通过DTW-BDS对应点匹配算法，获取数据轨迹R中的每一个样本点在查询轨迹Q中的对应点，在获得对应点的同时也获得了对应轨迹段。为了计算R与Q之间的距离，需要计算所有对应轨迹段距离之和，而轨迹段之间的距离计算包括两个部分：时空距离和形状影响因素。在时空距离计算中，需要进行断点划分，并通过断点阈值确定断点权重，通过DTW-BDS对应点匹配算法，获取断点的对应点，轨迹段的时空距离由断点和轨迹段端点到对应点的距离以及权重计算得到。在形状影响因素的计算中，需要使用余弦距离[32]和投影去量化轨迹段的形状相似程度，获得形状影响权值。由轨迹段的时空距离和形状影响权值可以计算得到轨迹段距离。通过累加所有轨迹段距离，就得到了轨迹间的距离。在得到所有轨迹距离之后，判断该距离是否小于给定的轨迹距离阈值，如果小于该阈值，则认为R与Q是相似轨迹。最后对所有数据轨迹遍历完成后，将所有相似的轨迹集合返回给查询者。

|  |  |
| --- | --- |
| **算法5.4 时空轨迹相似性查询** | |
| **输入**：查询轨迹Q，数据轨迹集合，轨迹距离阈值 | |
| **输出**：与Q相似的轨迹集和T\_result | |
| 1. | 时空归一化 |
| 2. | for RT |
| 3. | 对Q和R进行DTW-BDS对应点匹配 |
| 4. | for R |
| 5. | d(Q,R) += |
| 6. | if d(Q,R)> |
| 7. | R加入T\_result |
| 8. | **return** T\_result |

在STS算法中，使用了SNTR模型允许将时间距离和空间距离进行统一计算，解决了使用TTR模型可能造成的时空混乱问题。使用DTW-BDS对应点匹配算法保证了样本点的匹配效果，利用DTW算法的思想保持匹配结果的时序性，利用BDS算法提升样本点的对齐效果。然后在计算轨迹段距离的时候，考虑了轨迹段之间的时空距离和形状影响因素，使用断点减少采样策略对距离计算结果的影响，并量化轨迹段的形状相似性，提升查询结果的准确性。通过上述分析，在理论上SNTR模型和STS算法是有效和准确的。

## 5.4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，本章首先提出轨迹段时空距离以及轨迹段形状影响权值的计算方法，然后将二者结合得到轨迹段距离。接着使用轨迹段距离引出了轨迹距离的计算方法。最后根据上面所有工作，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6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SNTR模型和STS相似性查询算法的有效性，以及与一些最新的研究成果做比较。

## 6.1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[33-35]，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在北美路网North America Road Network上合成的轨迹数据集，均匀分布在路网上，以下简称NARN。轨迹最短的时间跨度为380秒，包含轨迹数目为1.5万条，每条轨迹中样本点个数为10到150个。

由于人每天的活动大部分以一天为周期，因此如果遇到轨迹数据跨越了一天的情况，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[36]，相关定义参考了别人的文章。查准率P(Precision)如公式6.1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[36]。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[36]。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。

(6.1)

查全率R(Recall)如公式6.2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[36]。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。

(6.2)

## 6.2参数的影响

本文提出的STS算法中包含如下参数，划分断点的断点距离阈值η，形状敏感度参数，局部相似性计算结果长度限制参数，以及子轨迹到查询轨迹的距离阈值。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。

### 6.2.1时空转化因素对查询结果的影响

在SNTR模型中，时空转化因素是模型中唯一的一个参数，该参数的大小会对生成的时空数据造成影响，可能会对相似性查询算法的准确性带来一定影响。因此这里使用了不同大小的，在两个数据集上分别进行了实验。

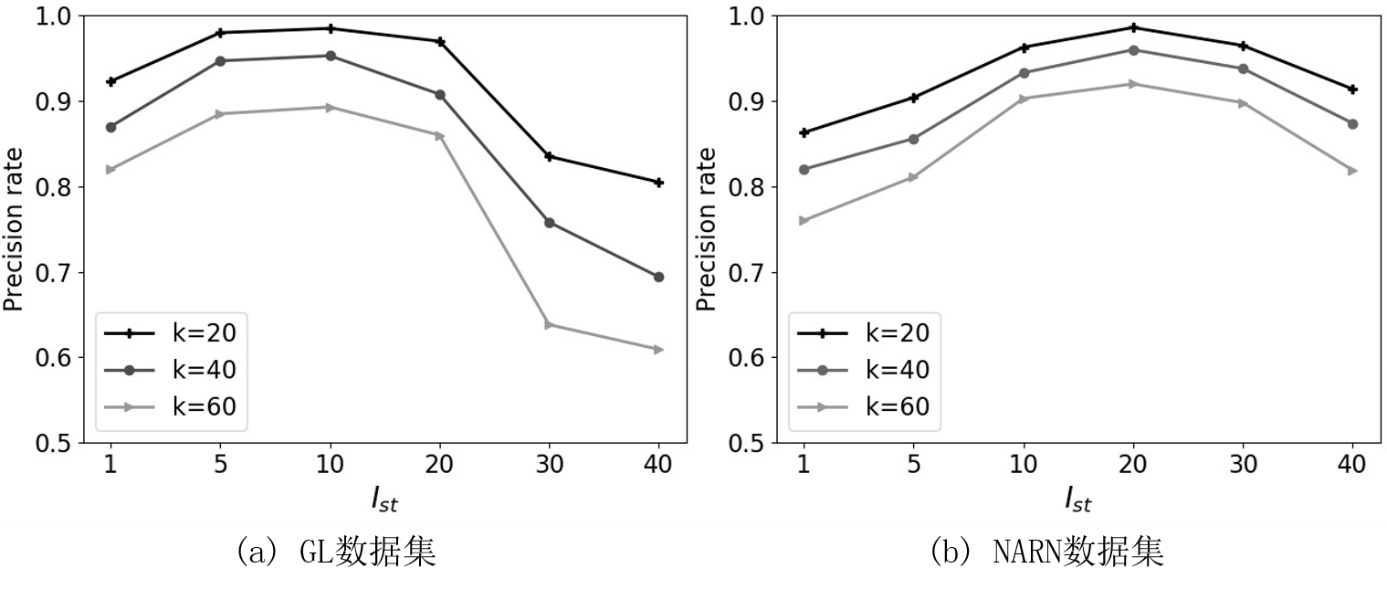


图6.1 对查准率的影响

Fig. 6.1 Effect of on precision ratio

实验结果如图6.1所示，在GL和NARN数据集中显示的趋势是近似的，在不断变大的过程中，查准率会由小变大，达到峰值后再减小，但是两个数据集峰值处的的值是不一样大的，变化趋势的快慢也不相同，因为GL数据集是在北京道路上采集的，由于车流和人流量较大，平均速度较慢，因此在图6.1(a)中，在1到20范围内，算法表现较好。而NARN数据集是软件生成的模拟轨迹数据集，生成时设定的平均车速较快，在图6.1(b)中，在10到40范围内，算法表现较好。

### 6.2.2断点距离阈值对查询结果的影响

断点距离阈值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，越小，划分出的断点越多。本实验中取值为5米至200米。而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。

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6.2所示。从图中可以看出，图6.2(a)的GL数据集和图6.2(b)中的NARN数据集相似性查询结果的查准率都随着的变大而减小。

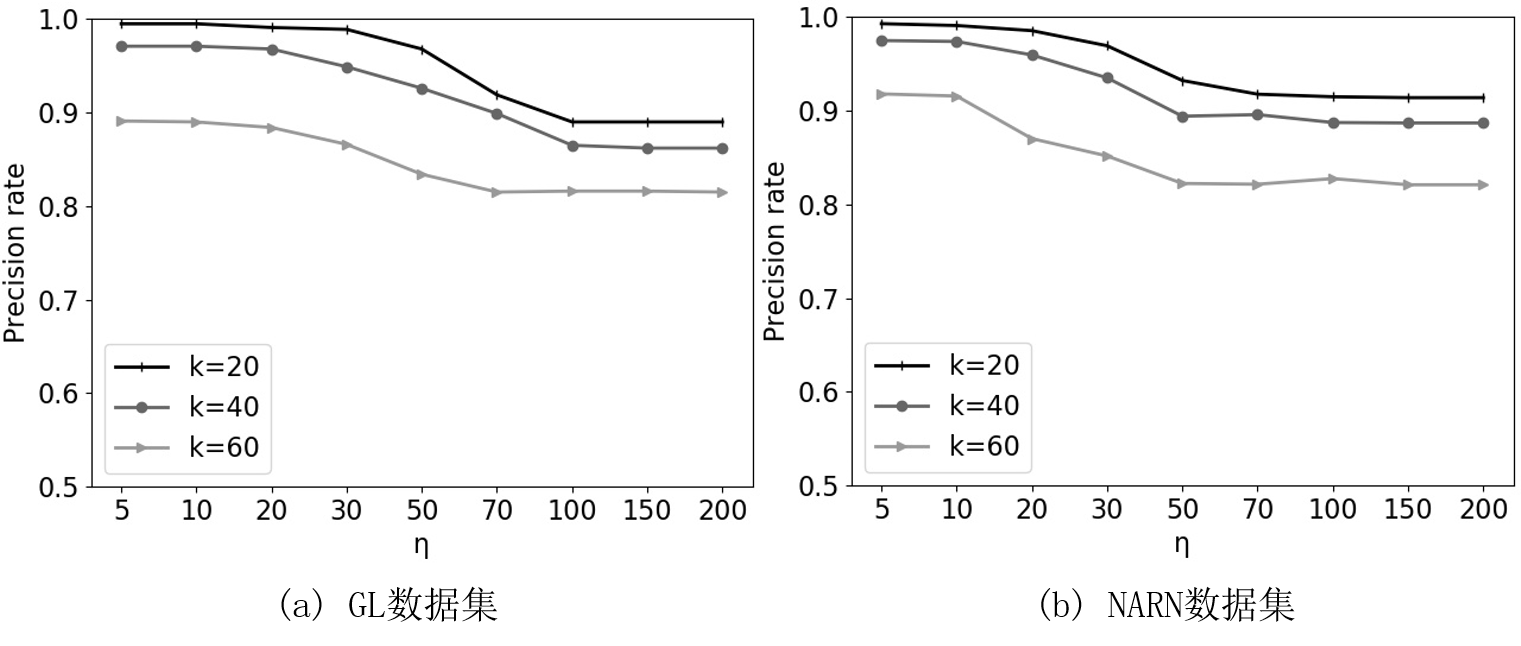


图6.2 对查准率的影响

Fig. 6.2 Effect of on precision ratio

在GL数据集中，在5到20的时候，查准率较高，因为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。而随着在20到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。最后在在70到200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NARN数据集上，虽然数据不同，但是查准率变化情况类似。

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，所以在同样的情况下，返回的轨迹条数越多，查准率会越低。

### 6.2.3形状敏感度参数对查询结果的影响

形状敏感度参数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，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的大小对最终查准率Precision的影响程度，如图6.3所示。

从上面两个数据集的实验结果中我们可以看出，形状敏感度参数时，查准率较低。当在1.5到2.5之间，查准率在该区间内有一个较高值，当继续增加，查准率又会慢慢降低，直至趋于稳定。

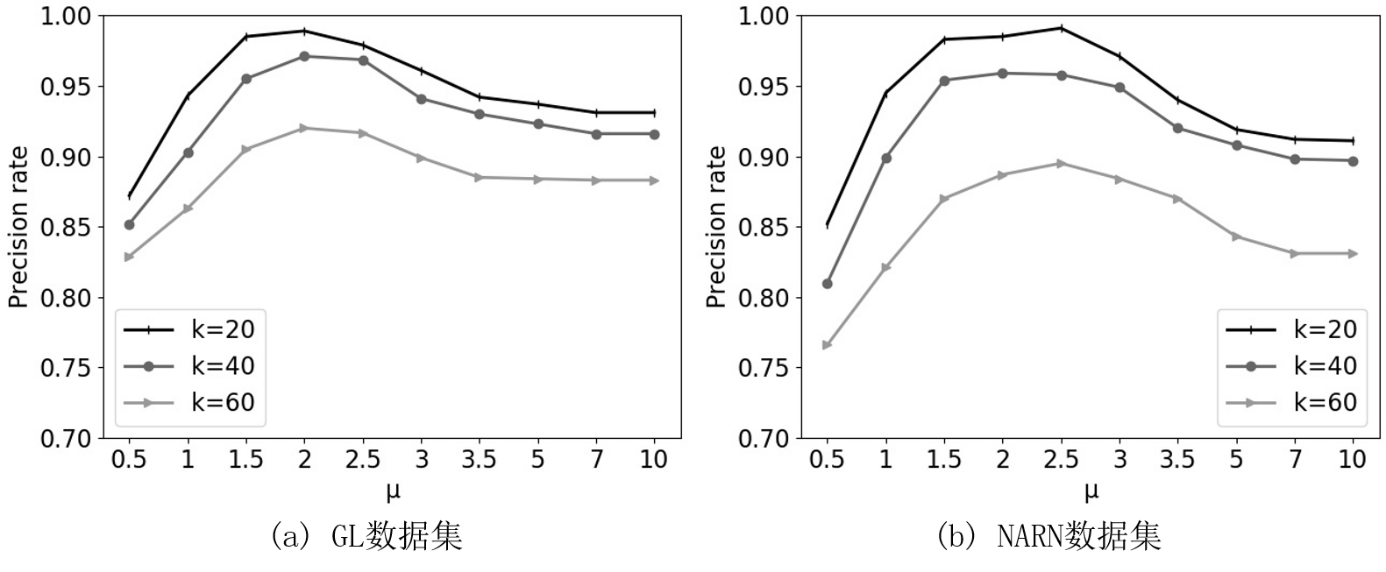


图6.3 对查准率的影响

Fig. 6.3 Effect of on precision ratio

下面分析一下为什么会出现这个情况。当=0.5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。当增大到2左右，形状相似性因素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。

### 6.2.4轨迹距离阈值对查询结果的影响

本文中的算法使用距离阈值筛选出所有距离查询轨迹小于的数据轨迹作为查询结果，也就是说距离阈值是可以作为轨迹相似与否的标准。当轨迹间距离小于等于时，认为轨迹相似，当轨迹间距离大于时，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。当我们研究对查询结果的影响时，可以使用查准率Precision和查全率Recall来共同监控查询结果。



图6.4 对查准率和查全率的影响

Fig. 6.4 Effect of on precision and recall rate

实验结果如图6.4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，随的增大，查全率Recall和查准率Precision呈现相反的增长趋势。随着不断增大，查询结果中，查全率Recall在不断上升，在=1000之后逐渐趋近于1，增大可以减小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Precision在不断下降，因为降低标准会增大FP，减小TP的比重。当时，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Recall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，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，可以取一个较大的值，大于1000即可。如果想综合考虑上面两个要求，可以取在800到1000之间，此时查准率和查全率均处于较高水平。

## 6.3与最新研究成果的对比实验

### 6.3.1不同规模的数据轨迹对算法的影响

在本节的实验中使用了SDTW算法[27]、PTM算法[31]和本文的提出的STS相似性查询算法做比较。在相关工作中介绍了SDTW算法和PTM算法的轨迹相似性的计算过程，并且这两个算法在相似性的定义中均包含了时间上的相似性，和本文的STS算法考虑的维度大体一致，因此将这两个算法与本文的STS算法进行对比实验，验证在不同规模的数据集上这三个算法的表现。

在SDTW算法中需要设置时间权重*t*和参数，时间权重*t*的作用是调节时间维度在最终结果中的重要性，的作用是调节算法中轨迹形状相似性的重要性，取*t*在到10之间以及在1到10之间的最大的查准率作为SDTW算法的效果。PTM算法需要指定阈值和来作为对应点之间的最大的空间距离和时间差的阈值，我们取值范围在5到20之间，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，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，断点断点阈值取10到20之间，形状敏感度参数取值在1到2之间，长度限制参数取值在到之间，通过不同参数组合，获得最大的查准率。

为了验证三个算法在不同规模下的轨迹数据上的表现，将数据轨迹划分为三组，第一组轨迹数量为5000条，第二组轨迹数量为10000条，第三组轨迹数量为15000条。最后使用与查询轨迹距离最小的前40条作为查询结果。

在两个数据集上的实验结果的查准率如图6.5所示。从实验结果中我们可以看出， 虽然SDTW算法、PTM算法和STS算法均考虑了空间维度和时间维度，但是效果差别较大。效果最差的算法是PTM算法，因为在PTM算法中将轨迹的时间相似性和空间相似性线性结合得到轨迹的时空相似性，可能会在一些相似性查询时出现时空混乱的情况，因此PTM算法的效果稍差一点。SDTW算法中在数据集较少的情况下，表现较好，但是数据量大了之后，查准率降低地比较严重。本文提出的STS算法在不同规模下表现都相对较好，在两个数据集中的查准率均在88%至97%之内，受查询轨迹的规模的变化产生的影响较小，稳定性较好。

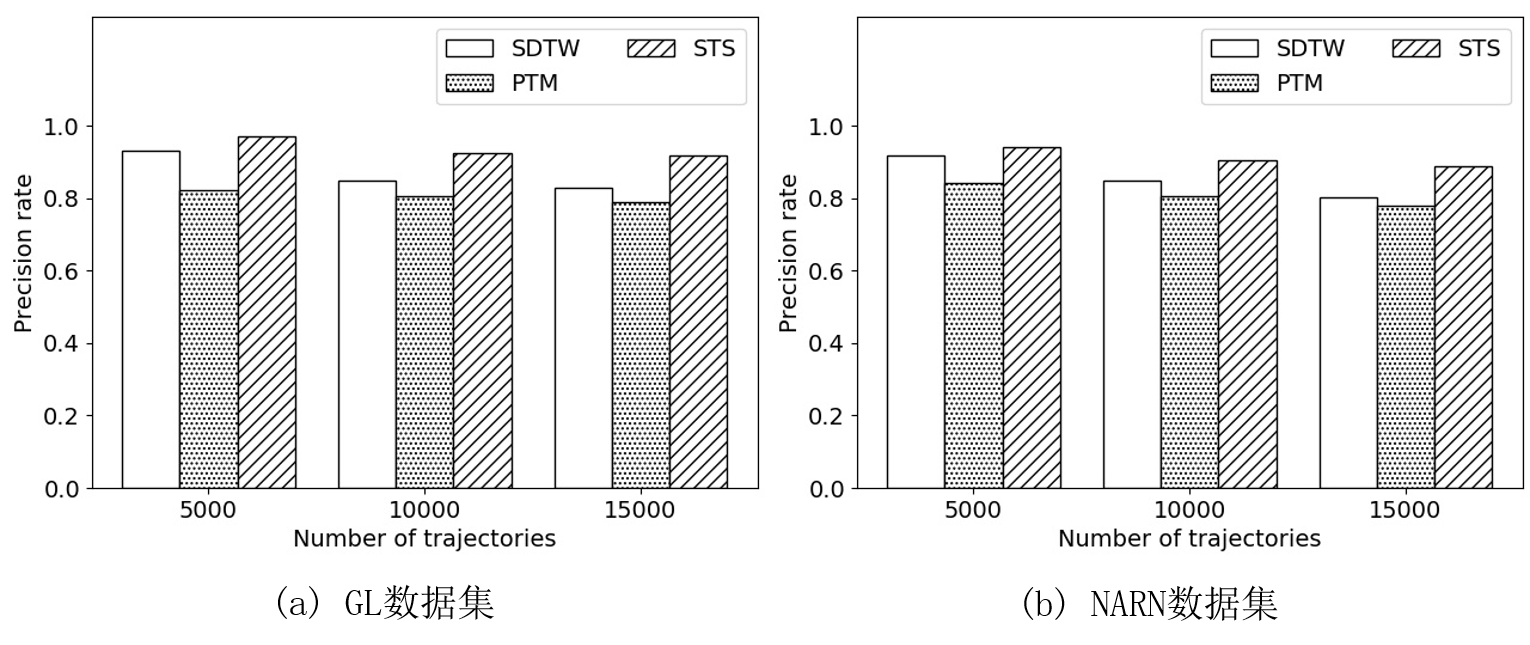


图6.5查询轨迹长度对不同算法查准率的影响

Fig. 6.5 Influence of query trajectory length on precision of different algorithms

### 6.3.2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。噪音使用的是均匀分布的随机数，噪音率来表示不同程度的噪音，的取值范围在0.1到1。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。

实验结果如图6.6所示，可以看出这三个算法在两个数据集上对噪声的抵抗能力有明显的差异。在SDTW算法中，由于使用了DTW算法中的对应点匹配方法，因此抗噪音干扰的能力较强，从实验结果中也能明显反应这个优点，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STS算法使用了DTW对应点匹配的优势，还结合断点去描述每一条轨迹段，提高了抗噪声干扰的能力，查准率相对较高。

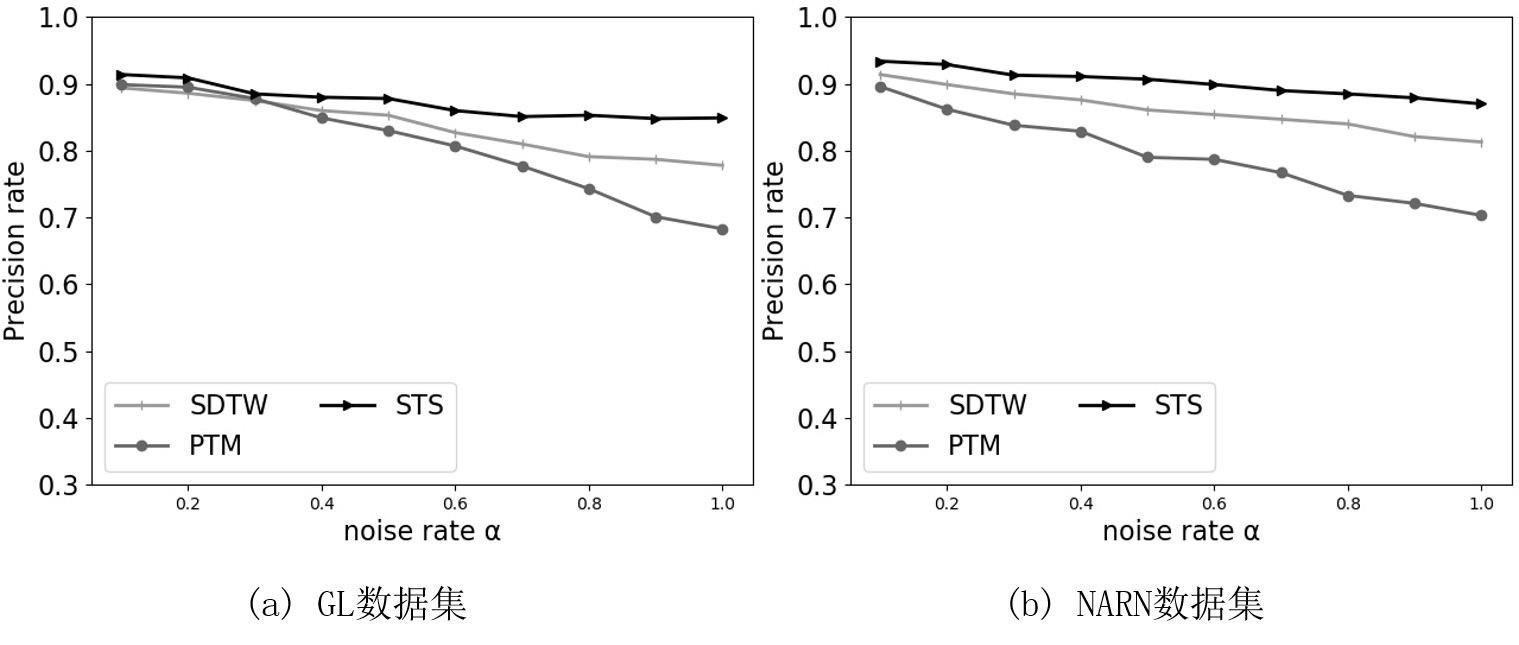


图6.6 噪音率对不同算法查准率的影响

Fig. 6.6 Influence of noise rate on precision of different algorithms

## 6.4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7章 总结与展望

## 7.1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SNTR模型的概念，使用时空转换因素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。算法解决了在SNTR模型下，直接使用DTW算法和BDS算法对应点匹配思想可能造成的对齐效果不好以及时序错乱的问题。

(3)本文提出了一个轨迹段三维空间距离的计算方法。利用断点阈值，在轨迹每隔距离便取一个点，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权值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函数获得形状影响权值，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权值的计算，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。

## 7.2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。