高时效性下的轨迹局部相似性查询算法的设计与实现

目录

[第一章 绪论 4](#_Toc528826484)

[1.1研究背景及意义 4](#_Toc528826485)

[1.2本文研究内容 7](#_Toc528826486)

[1.3本文组织结构 8](#_Toc528826487)

[第二章 相关理论与关键技术 9](#_Toc528826488)

[2.1轨迹表示方法 9](#_Toc528826489)

[2.1.1仅基于位置信息的轨迹表示方法 9](#_Toc528826490)

[2.1.2基于时间信息的轨迹表示方法 9](#_Toc528826491)

[2.1.3基于文本信息的轨迹表示方法 10](#_Toc528826492)

[2.1.4基于网格的轨迹表示方法 10](#_Toc528826493)

[2.2现有轨迹相似性计算方法 10](#_Toc528826494)

[2.2.1欧氏距离（EU） 10](#_Toc528826495)

[2.2.2动态时间弯曲距离（DTW） 11](#_Toc528826496)

[2.2.3最长公共子序列距离（LCSS） 12](#_Toc528826497)

[2.2.4编辑距离（EDR） 13](#_Toc528826498)

[2.2.5带真实惩罚的编辑距离（ERP） 13](#_Toc528826499)

[2.2.6路网上的轨迹相似性查询 14](#_Toc528826500)

[2.2.7城市运输系统中基于段的轨迹相似性计算 16](#_Toc528826501)

[2.2.8简化的轨迹相似性计算 19](#_Toc528826502)

[2.2.9基于签名的轨迹相似性计算 20](#_Toc528826503)

[2.3 高效的轨迹索引技术 21](#_Toc528826504)

[2.3.1 R树索引 21](#_Toc528826505)

[2.3.2 Voronoi图 22](#_Toc528826506)

[2.4本章小结 22](#_Toc528826507)

[第三章 三维时空下轨迹段的相似性 24](#_Toc528826508)

[3.1问题定义 24](#_Toc528826509)

[3.1.1轨迹相似性查询相关概念 24](#_Toc528826510)

[3.1.2轨迹局部相似性问题定义 25](#_Toc528826511)

[【问题】加上符号定义 27](#_Toc528826512)

[3.2时空归一化 27](#_Toc528826513)

[3.1.1 PTM算法中时空维度结合存在的问题 27](#_Toc528826514)

[3.1.2时空归一化方法 29](#_Toc528826515)

[3.3对应点和对应轨迹段 32](#_Toc528826516)

[3.3.1BDS算法找对应点的优势与不足 32](#_Toc528826517)

[3.3.2DTW算法找对应点的优势与不足 34](#_Toc528826518)

[3.3.3DTW-BDS对应点匹配算法 35](#_Toc528826519)

[3.3.4对应轨迹段 38](#_Toc528826520)

[3.4本章小结 39](#_Toc528826521)

[第四章 局部轨迹相似性算法 40](#_Toc528826522)

[4.1对应轨迹段的时空距离 40](#_Toc528826523)

[4.1.1DTW算法计算二维空间距离及其存在的问题 40](#_Toc528826524)

[4.1.2断点 43](#_Toc528826525)

[4.1.3轨迹段三维时空距离 44](#_Toc528826526)

[4.2对应轨迹段形状相似性 47](#_Toc528826527)

[4.2.1余弦距离 47](#_Toc528826528)

[4.2.2轨迹段形状影响因子 48](#_Toc528826529)

[4.2.3形状相似性与SSD计算方法的比较 52](#_Toc528826530)

[4.3三维时空下的轨迹局部相似性查询 54](#_Toc528826531)

[4.3.1对应轨迹段距离 54](#_Toc528826532)

[4.3.2轨迹局部相似性计算 54](#_Toc528826533)

[【问题】缺少一个贯穿始终的例子 56](#_Toc528826534)

[4.4本章小结 56](#_Toc528826535)

[第五章 实验设计与分析 57](#_Toc528826536)

[5.1实验环境与数据集 57](#_Toc528826537)

[5.2算法中参数对结果的影响 59](#_Toc528826538)

[5.2.1距离阈值对查询结果的影响 59](#_Toc528826539)

[5.2.2形状因素权重对查询结果的影响 60](#_Toc528826540)

[5.2.3长度限制参数对查询结果的影响 61](#_Toc528826541)

[5.2.4距离阈值对查询结果的影响 62](#_Toc528826542)

[5.3对比实验 62](#_Toc528826543)

[第六章 总结与展望 64](#_Toc528826544)

# 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。

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。或者根据用户一周内频繁出现的场所，为用户推荐周边的美食或娱乐场所。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，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。

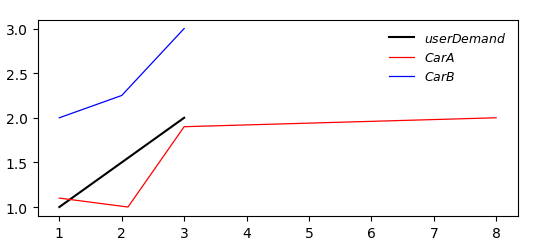
除了对用户位置的信息采集，对出租车移动路线信息的采集之外，还有对野生动物行为轨迹的采集，军事领域对地方目标轨迹的实时监测以实现精准打击，对飓风移动路径数据的采集来预报和预防自然灾害等等。随着数据采集设备的改良和采集方式的优化，各个领域都产生了海量的轨迹数据。所以对轨迹数据的分析利用变得十分重要。人们为了发掘海量数据中隐藏的价值，得到丰富的数据特征空间以及用户轨迹的规律性信息，开发了聚类分析、隐私保护和行为预测等一系列的应用技术，而这些技术的实现都得益于移动对象轨迹的相似性查询技术的发展。

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，比如三分之一的长度和Q是很重合的，有的时候这三分之一的轨迹包含的信息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。

举一个例子来说明这个问题。比如为用户推荐顺风车，用户需要的路线是顺风车CarA预计行驶轨迹的三分之一，另外一辆顺风车CarB的预计行驶轨迹与用户需要的路线很接近，但是大于CarA与用户接近的那部分轨迹距离，那么最好的情况应该是为用户推荐CarA。但是如果使用之前轨迹相似性算法计算轨迹全段的相似程度，可能出现由于CarA的轨迹很大部分与用户轨迹不相似，导致掩盖了相似的也是用户最需要的那部分结果，而为用户推荐了CarB的结果。这并不是使用相似性算法的本意，而计算完整轨迹的相似性计算没法解决这个问题。此外，顺风车的推荐还需要考虑一个时间因素，如果下图中CarA和CarB的轨迹是一小时之前，而用户发出呼叫顺风车的请求是在现在，那么任何一辆车都不能推荐给用户。针对这种时效性的问题，还必须考虑轨迹之间的时间距离。之前的许多研究都涉及到了时间方面的因素，比如将两点之间时间戳的差值作为时间距离，最后使用一个系数和空间距离结合起来，得到一个轨迹间的相似程度。但是大多数情况是根据算法研究人员经验值去确定参数值，并且不同规模的轨迹可能还需要调整参数的大小，参数的确定很难给出一个明确的意义。



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间隔距离，以及轨迹段在形状上的余弦距离。三维时空中的距离可以包含构造三维时空中的每一个维度表达的信息，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，使用微软提供的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

# 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 = <p1, p2, …, pn>。在欧式空间下，点pi是一个坐标的形式，pi = <loni, lati>，loni表示经度，lati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pi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 = <(p1, t1), (p2, t2), …,(pn, tn)>，pi表示坐标或者顶点集合中的点，ti表示的是用户处于pi位置的时刻。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 = <(p1, k1), (p2, k2), …,(pn, kn)>，pi表示坐标或者顶点集合中的点，ki表示pi位置的文本描述。

### 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)|cC, I=(tin, tout)}，其中c代表整个网格集合C中的一个网格，tin和tout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 欧氏距离一对一的对应关系

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，，序列A和序列B的欧氏距离表达式如式(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）所示。当p=1时，L1-norms叫做曼哈顿距离。

(1)

(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（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。

(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。动态时间弯曲距离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 DTW一对多的对应关系

DTW的二维空间上的计算公式如式(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。

(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。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(5)所示。

(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）

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。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（6）所示。

(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（7）所示。EDR是基于时间序列中元素的距离distedr得到的，如式（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。

(7)

(8)

### 2.2.5带真实惩罚的编辑距离（ERP）

ERP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（9）所示。基于disterp的ERP计算公式如式（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。

(9)

(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。

在计算相似性之前，需要将轨迹映射到路网上去，如图2.5所示。给定移动对象a和b的移动轨迹Ta和T b，轨迹格式为T={(l1, v1, t1), (l2, v2, t2),…(lm, vm, tm)}，其中li=(lgi, lai)表示样本点坐标，vi表示对象在ti时刻的移动速度。我们用da(lai,Tb)表示从样本点lai到轨迹Tb的路网距离。lai到Tb路网距离指，Tb上距离lai最近的点到lai的路径距离。用DG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2.5 将道路转化为路网

路网上Ta和Tb的距离dN(Ta, Tb)的计算公式如式（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。

(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（15）所示。

(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SG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（16）所示。

(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（17）所示。

(17)

当结合权值、时间和空间来计算轨迹距离，我们有公式（18），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（19）所示。

(18)

(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2.6所示。



图2.6 不同采样方法造成的样本点错位

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。

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2.7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2.8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2.8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2.9所示，其中为p1到轨迹S的点段距离。对应的距离公式如式（20）所示。

图2.7 使用轨迹段面积代替点点距离 图2.8三角形面积正比于阴影面积



图2.9 使用三角形的高代替三角形面积

(20)

其中，

然后介绍预测距离。给定一组对应点和，时间戳分别为ti和tj，ti和tj不相等，令时间戳大的点等于A，时间戳小的点等于B，如图2.10所示。



图2.10 预测B在与A相同时刻的位置

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，的预测距离计算公式如式（21）所示。

(21)

其中，

那么这两个点的时间距离可以转化为tj时刻，轨迹R的位置到的距离，如图2.10所示。的预测距离计算公式如式（22）所示。

(22)

图2.11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（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。

(23)



图2.11 利用夹角计算形状相似性

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之间的形状相似程度，如式（24）所示，其中是与之间的轨迹段，是与之间的轨迹段。

(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（25）所示，其中是两条轨迹段之间的夹角，和的计算公式如式（26）和式（27）所示。

(25)

(26)

(27)

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，指的是出掉第一个轨迹段剩下的所有轨迹段。

(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，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（29）所示。

(29)

当比较用户和用户之间的相似程度时，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。

### 2.2.9基于签名的轨迹相似性计算

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2.12所示。因此Ta等人提出了BDS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2.13所示。

图2.12按时间匹配点对 图2.13按点到轨迹段最小距离匹配

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，其计算公式如式（30）所示。

(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2.14所示。



图2.14点到轨迹段的距离

BDS的计算公式如式（31）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。

(31)

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，然后在计算轨迹上第k个点到轨迹的距离时，只需要计算点以该点为中心的一定范围内的网格中存在的轨迹段的距离，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。

## 2.3 高效的轨迹索引技术

### 2.3.1 R树索引

R树是一种针对空间数据建立的树形索引，其核心思想是将每片需要建立索引的区域使用一个外接矩形框起来，作为树的叶子节点，即叶子节点包含着指向该片区域的指针。然后在建树的过程中，矩形之间允许多层嵌套，每个矩形都是一个节点，外层的矩形是被嵌套的矩形的父节点，最终构造成一颗高度平衡树。解决了传统数据库对空间数据检索效率低下的问题。

由于R树需要构建成一颗高度的平衡树，所以有一个构建规则。假设M是树中所有节点允许包含的最大的条目数，m是节点包含的最小条目数，通常情况下m=M/2，此时R树的构建规则如下：

（1）所有叶子节点包含m至M个记录索引。如果一个叶子节点是树的根节点，那么它包含的记录数可以小于m。

（2）叶子中存储的指针指向的记录，这些记录均为可以覆盖其代表空间中区域的最小外包矩形。

（3）一个非根且非叶子的节点，其包含的孩子数目在m到M之间。

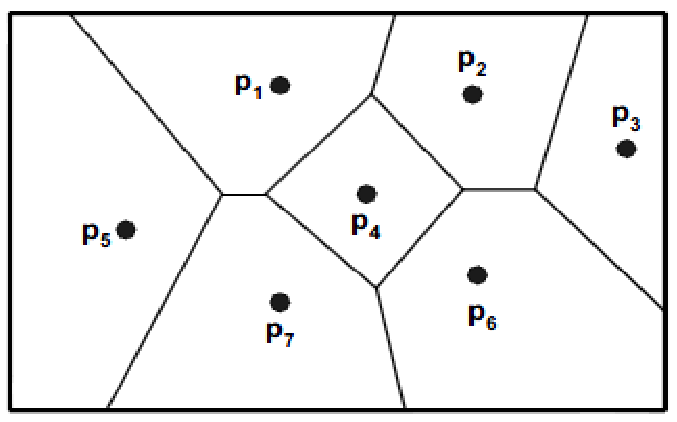
（4）对于一个非叶子节点，节点中的记录是节点所有孩子节点代表区域的最小外包矩形。

（5）R树的所有叶子节点均在同一层。

由于以上特性，相比传统数据库，R树可以很快检索空间数据。但是R树允许外包范围重叠，所以使得一个区域可以对应多条搜索路径，严重影响了R树性能，使空间搜索效率降低。因此R+树和R\*树和三维R树被提出来解决搜索区域的重叠问题。

### 2.3.2 Voronoi图

Voronoi图根据空间中的点集的分布，将一个欧式空间划分为多个凸多边形，凸多边形的每一条边都是一个点与周围点的垂直平分线，该点叫图多边形的质心。每个凸多边形都有一个质心，任意一个凸多边形内的点到其他凸多边形质心的欧式距离都小于到本凸多边形质心的欧氏距离，如图所示。后来有学者对Voronoi图进行了进一步研究，根据发生点和生成面的不同，提出了广义Voronoi图，其中包括有序k阶Voronoi图，最远点Voronoi图，权重Voronoi图等。Voronoi图作为空间数据索引中的一类重要索引结构，解决了很多传统数据库不能解决的问题。



## 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。此外，本章还介绍了加速空间数据搜索的几个应用比较广泛的索引结构。

# 第三章 三维时空下轨迹段的相似性

## 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一些位置信息，而位置信息中包含了空间和时间属性。现实世界虽然是一个三维的空间，但是人和车辆的移动可以看做是在二维平面中的移动，并且在轨迹相似性查询中，关于移动对象在二维空间中的移动轨迹的研究最为广泛，因此在空间上本文将问题放在二维空间下研究。除了空间信息外，本文中的位置信息还涉及到时间信息。考虑到人的户外活动一般是以天为单位，因此我们时间信息中不需要包含年月日等信息，只需要用到包含时分秒的时刻信息。所以位置点包含的信息有经度、纬度和时间戳。

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。由于本文为了突出研究问题的核心，简化其他计算步骤，因此本文的空间维度使用欧式空间，由于原始轨迹数据中的信息包含经纬度，我们默认在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，包括空间信息和时间信息，这些信息组成一个样本点p，或者叫采样点，。

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。

定义3-3（轨迹相似性计算函数）输入两条轨迹数据Q和R，使用轨迹相似性计算函数进行计算，输出一个实数值表示轨迹之间的相似程度，该值越大，两条轨迹越相似。

定义3-3（轨迹相似性查询）给定一条查询轨迹Q，以及一个存储了轨迹数据的数据库DB，输出DB中的一条轨迹，使得，有。

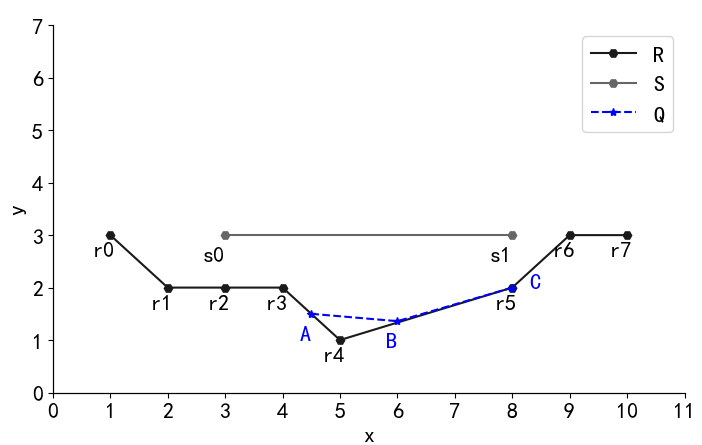
### 3.1.2轨迹局部相似性问题定义

上面定义了一些轨迹相似性计算中用到的一些基本概念，下面再介绍本文中需要用到的一些其他条件。本文在实验中首先使用了兴趣点和转折点提取算法，将能描述轨迹形状的点以及移动对象的兴趣点提取出来，进行进一步分析。因此如果不加解释，本文中使用的样本点全部是经过提取算法处理后的点，而不是原始轨迹数据。

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，如图所示。其中轨迹R在8点的时候从开往处，到达点为8点30分，途径A点和B点。轨迹S在8点10分从点开至点，到达点为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。



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中从到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到、到会对加大轨迹间的距离，掩盖了轨迹R中到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3-4（子轨迹）给定一条轨迹数据，T的子轨迹是T的一部分，其中，i<=j，且包含了轨迹数据T从到的所有样本点。

**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，希望查询到数据库中与Q相似的轨迹或子轨迹。**

本文称该问题为轨迹局部相似性查询问题。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。

### 【问题】加上符号定义

## 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

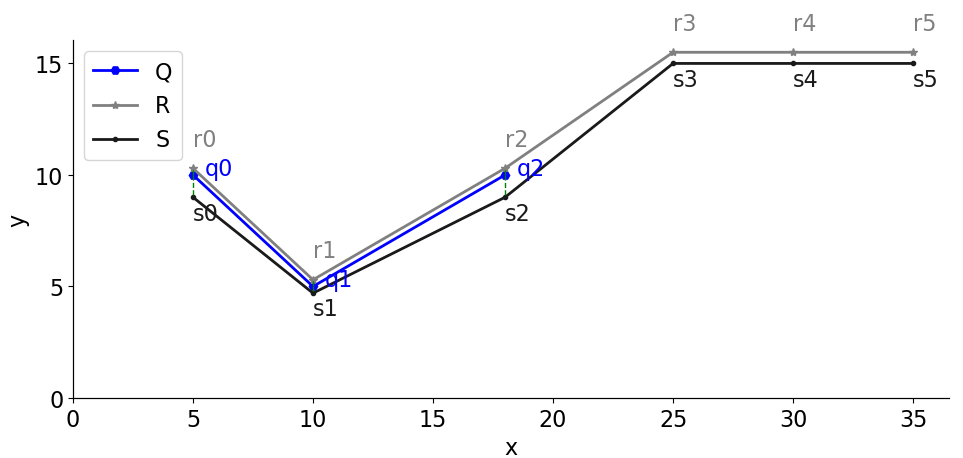
之前有很多研究人员研究过包含时间和空间两个维度的轨迹相似性查询技术，大多数采取的做法是将时间和空间分割开，分别计算相似程度或者距离，然后使用权值将二者进行结合。最典型的方法就是下面的PTM算法（Personalized trajectory matching）。

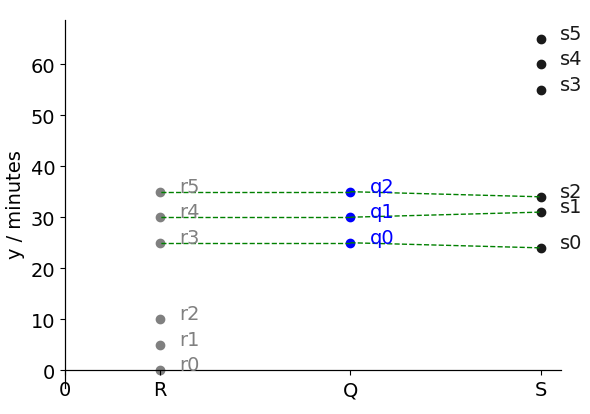
PTM算法主要利用轨迹间的时间和空间因素，计算轨迹间的相似程度。首先考虑空间因素，查询轨迹Q上的点和数据轨迹R上的点的空间距离为，这里的空间距离计算的是路网上的距离，基于此空间距离，获得和的空间距离影响因子，如公式（2）所示。距离越大，影响因子越小，当距离大到一定程度时，将该因子置0。然后借用LCSS的思想，基于轨迹上所有样本点的位置关系以及样本点的权重得到空间上最大相似程度，如公式（4）所示，其中权重是基于用户对某一个点的重视程度来确定，也是文章的创新点之一。然后考虑时间因素，查询轨迹Q上的点和数据轨迹R上的点的时间差为，时间影响因子为。然后基于所有点的时间关系得到时间相似因子，其计算方法与空间相似因子的计算方法类似。最后使用参数将空间相似度和时间相似度结合起来，得到轨迹相似度，如公式（6）所示。

下面使用一个例子来讨论该方法在时间和空间的结合上出现的一些问题。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。

假设我们有数据轨迹R和S，还有一条稍短的查询轨迹Q。在空间关系上面可以参考图一，x轴和y轴是空间的两个维度，图中可以看出轨迹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，但是很明显在这一段轨迹R和Q的距离更近一点。关于时间维度可以参考图二，这里需要解释一下，图二的纵轴代表从8点开始的分钟数，比如的y值为0，代表的时间戳为8点整，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。

解释完三条轨迹的时间和空间数据后，我们来使用PTM算法分别计算查询轨迹Q与R和S的相似性，关于参数设置如下，时间和空间阈值都取无穷大，每个样本点权重相等，时间和空间的权重一样，。先计算Q与R的PTM相似性，空间相似性为2.22，时间相似性为3，时空结合相似性为2.61。然后计算Q和S的PTM相似性，空间相似性为1.10，时间相似性为1.10，时空结合相似性为1.10。我们由PTM算法可以得出轨迹R和轨迹Q的相似度更大，将R作为查询结果返回。





现在我们分析一下这样计算存在的问题。我们可以回到原始查询请求以及轨迹数据的分析部分。根据查询轨迹，查询的目的应该是想找到一条轨迹，这条轨迹在8点25分左右出现在点，在8点30分左右出现在点，在8点35分出现在点。从描述上看，应该是轨迹S较为符合我们的查询需求。再看一看我们的查询结果R，对应时间下的空间位置与查询目的相去甚远，很显然不符合查询要求。

出现上述问题的原因是在PTM中，将时间和空间作为两个方面去考虑问题，空间上找距离Q最接近的样本点，计算空间相似度，然后在时间上也找距离Q最接近的样本点，计算时间相似度，这两步没有一点关联，导致空间上的样本点对应关系可能和时间上样本点的对应关系完全不同，最终导致了较差的查询结果。

因此，在考虑时间因素和空间因素的情况下分析轨迹数据相似性，应该将二者统一考虑，计算轨迹中的样本点的对应关系，这样查询得到的结果才符合轨迹相似性查询的要求。

### 3.1.2时空归一化方法

基于前面对PTM算法的讨论以及思考，得出了下面这个结论，即如果想同时考虑时间与空间因素对轨迹相似性造成的影响，那么必须在寻找轨迹样本点之间的对应关系的时候，同时考虑时间和空间在同一对样本点上的差异。

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，需要使用一个方法将时间向空间进行转换，因此我们需要找出空间和时间上的相关的地方。

如果应用场景是问题定义中的寻找到一辆车能拍摄小偷的车的话，在空间上分析，如果是在城市道路中，可能间隔约200米，小偷的车就会被其他车完全挡住拍不到，但是间隔再近一点，便能拍摄到。从时间上分析，两辆车一前一后，前车速度快，后车速度慢，在后车恰好看不到前车的时候，前车突然刹车，后车需要20秒才能赶上前车。对时间和空间上进行分析之后，我们发现空间上落后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。考虑到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，将时间维度向空间距离上做转化。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。如果是想找相似轨迹搭顺风车，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等于0.22米每秒。

然后就可以使用该转化因子将时间t转化为空间中的z。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。

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。

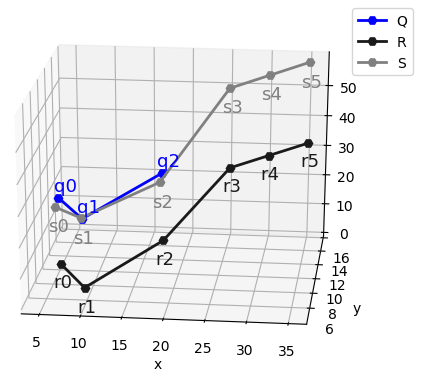
|  |  |
| --- | --- |
| **算法** 时空归一化 normalization(trajectorie, ) | |
| **输入**：原始轨迹数据trajectorie，时空转化因子 | |
| **输出**：三维时空中的轨迹数据trajectorie\_3d，时空转化因子 | |
| 1. | trajectory\_3d=[] |
| 2. | for trajectory |
| 3. |  |
| 4. | trajectory\_3d.add([]) |
| 5. | end for |
| 6. | return trajectorie\_3d |

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，点和的时空距离如公式所示。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

时空转化因子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，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所示，在引入了由时间转化后的第三维度之后，就可以很明显地看出，在至对应的时间段上，轨迹S的至子轨迹段与查询轨迹Q最接近，再分析轨迹数据的信息，发现至子轨迹段的描述确实是符合查询要求的。因此，这里提出的三维时空是有效的。



## 3.3对应点和对应轨迹段

本节主要介绍对应点以及基于对应点产生的对应轨迹段。一个样本点的对应点指的是该样本点到另一条轨迹中距离最短的点。有些算法中限制一个点的对应点必须为样本点，比如DTW，而一些算法则没有这个限制，如BDS（Bi-directional mapping similarity）算法。下面分别介绍这两种算法在寻找对应点时的优势与不足。

### 3.3.1BDS算法找对应点的优势与不足

之前的一篇论文中提到了双向映射相似性算法BDS。大致思想是，在二维空间中，有一条查询轨迹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。这里一个点的对应点指的是该点对齐到另一条轨迹上最接近的位置，这个位置就是该点的对应点，并且这个位置不一定是另一条轨迹的某一个样本点，可能只是两个相邻样本点之间连线上的点。

有时候由于采样策略的原因，导致样本点的分布很不均匀，如图一所示。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，会导致样本点对应的时候错位很严重，不仅前后不同时间的样本点对应错误，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，如图二所示。

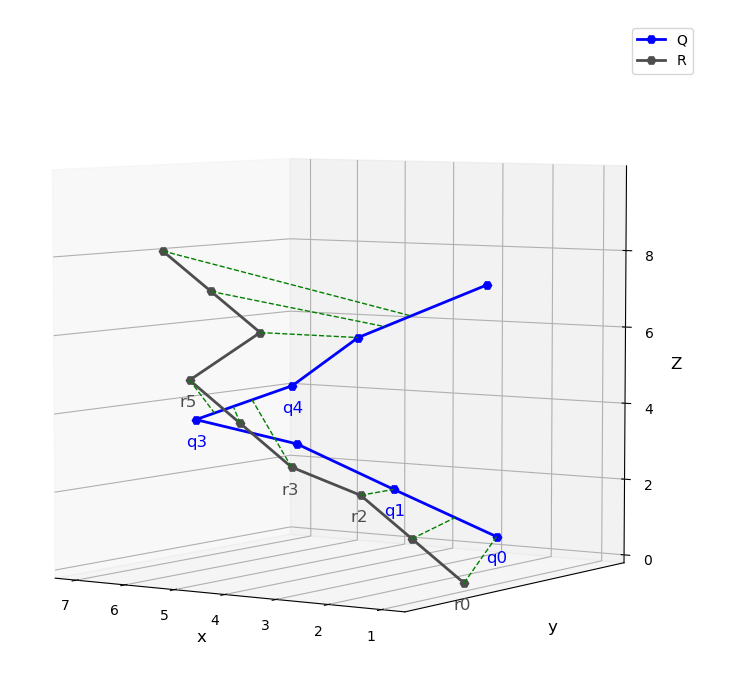
BDS的优点在于不考虑一个点的对应点必须要是样本点，减少了由于样本点位置和序列带来的限制，会更好地描述样本点之间的对应情况。





BDS算法使用了一种很好的样本点对应方法，但是BDS算法没有考虑时间因素以及轨迹方向因素，而本文研究的问题是在一个三维时空下，需要考虑时间的先后顺序以及轨迹的方向。如果将BDS算法直接使用在本场景中，有可能出现这样的情况，即轨迹R上的某个点在对应到轨迹Q上的一个点之后，时间戳在后面的点有可能对应到时间戳在前面的点。

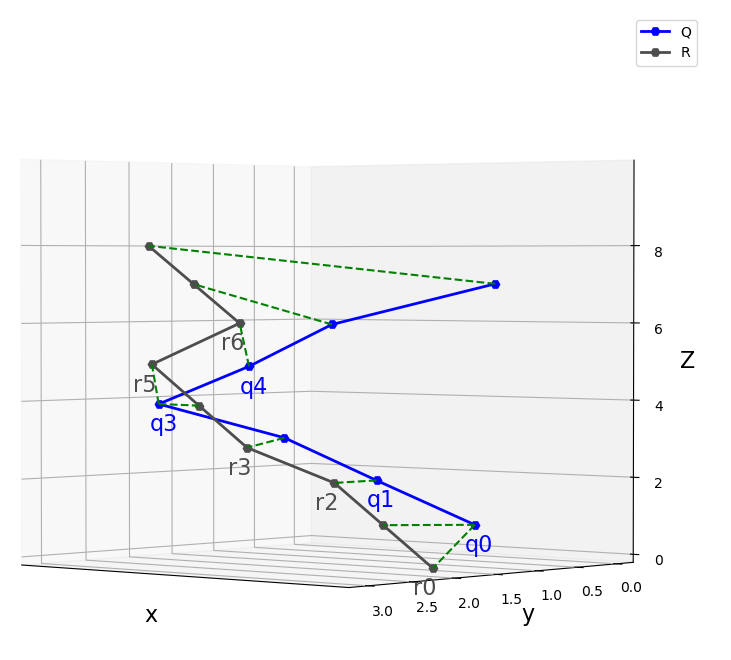
举个例子说明一下。如图一所示，先展示二维空间下的轨迹Q和R，其中R的至部分轨迹段和Q很相似，至部分轨迹段与Q形状相似，但方向相反。图二展示了加上时间维度的三维时空下的Q和R，由于空间表示不方便，所以隐藏了部分点的表示，图中可以看出对应点的关系，如果使用BDS中的方法，一个样本点到另一条轨迹距离最短的位置为该样本点的对应点的话，会导致轨迹R的至都与轨迹Q很相似，但是中间发生了行驶方向的翻转，对应样本点时序的错位，即至对应至，至对应至，因此与轨迹R进行比较的轨迹序列为<>，而不是我们输入的查询轨迹Q，虽然样本点都相同，但是一条时间序列还需要关注样本点时间上的先后顺序。由于BDS中寻找对应样本点的方法中没有考虑到同一条轨迹上样本点的时间先后，不能直接用于本文讨论的问题。

### 3.3.2DTW算法找对应点的优势与不足

在DTW算法中使用的样本点对齐方法很容易可以解决刚才BDS中产生的时序错位问题。针对样本点对齐问题，DTW是这么做的。首先计算轨迹Q和轨迹S的第一个节点之间的距离，然后加上下面三个距离的最小值，第一个是Q的第二个节点开始到S的DTW距离，第二个是Q到S的第二个节点开始的DTW距离，第三个是Q的第二个节点开始到S的第二个节点的开始的DTW距离。这种做法严格按照轨迹数据中样本点的时间顺序来往下计算，最后获得的轨迹间距离的值是所有对应点匹配方案中的最优方案贡献出来的，并且可以获得最优的对应点匹配方案。DTW算法肯定比欧氏距离算法中的使用样本点一一对应的方法要好很多。

如果使用DTW解决刚才的问题，那么其对应关系如图所示。样本点对应关系为：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上对应点时序的翻转，不仅解决了BDS中时序错位的问题，还得到了一个让对应点之间总体距离最小的匹配方案。



因此DTW算法的优点是在对应点匹配过程中不会出现时序的错乱。但是由于DTW限制了只能是样本点之间的对应，这其中必然存在有些对应关系不是最优的，如果考虑使用BDS算法中的思想，对应策略还有优化的空间。

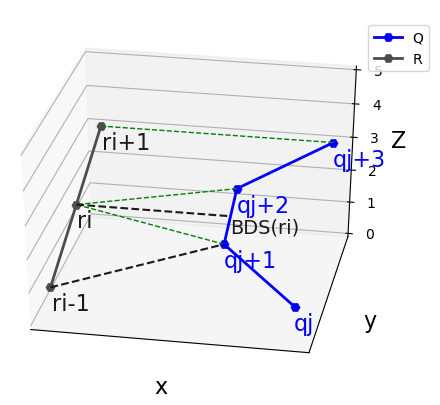
### 3.3.3DTW-BDS对应点匹配算法

基于上述对DTW和BDS算法的讨论，发现二者在寻找对应样本点可以优劣势互补。因此下面我们考虑在DTW算法中使用BDS对应点匹配方法进行改进。

假设轨迹R的样本点个数为n，轨迹Q的样本点个数为m，并且我们已经通过DTW获得了所有DTW对应点对集合，我们记的DTW对应点集合为，并且由于允许一对多，所以可能包含轨迹Q的多个样本点，我们记其中时间序列最早的为，时间序列最晚的为。我们现在要使用BDS算法中找最短对应点的思想，需要调整数据轨迹点的DTW对应点，称为的DTW-BDS对应点，简记为。我们记在轨迹段至间的BDS对应点为，且若和为同一个点，那么为。

如果为轨迹R的第一个点，即的BDS对应点在至的DTW对应点中时间序列最早的点之间，即。如果恰好的第一个DTW对应点就是，那么。

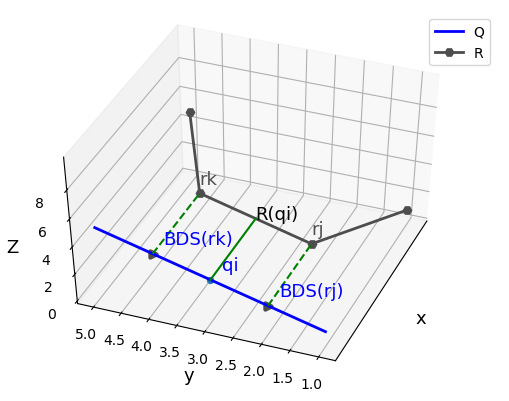
如果不是轨迹R的第一个点，也不是最后一个点，且已经找到了其BDS对应点，则的对应点在的对应点与的对应点之间，因此。如图所示，为，为，那么在与之间。



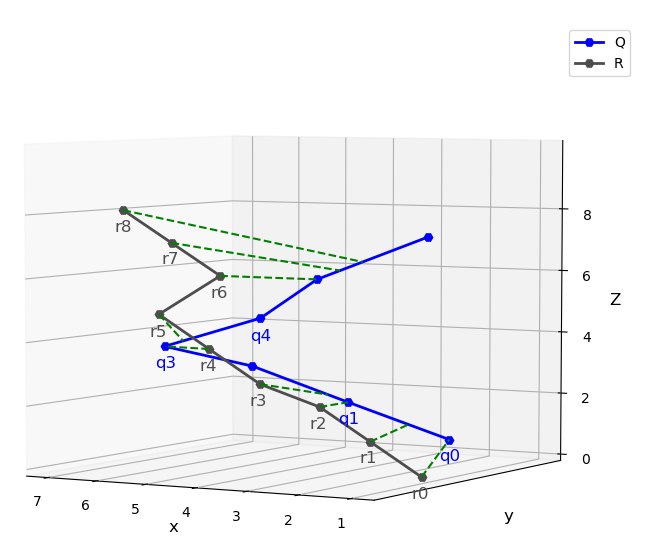
如果是轨迹R的最后一个点，那么。其中为轨迹Q的最后一个点。下面给出求DTW-BDS对应点匹配算法的伪代码。

|  |  |
| --- | --- |
| **算法** DTW-BDS对应点匹配算法 DTW\_BDS\_pair\_match(Q,R) | |
| **输入**：查询轨迹Q，数据轨迹R | |
| **输出**：R与Q的DTW-BDS对应点对pair | |
| 1. | 调用DTW(Q, R)，获得DTW最优对应矩阵array |
| 2. | 从array中，在Q上找到的DTW对应点 |
| 3. | for |
| 4. | if 是R的第一个点 |
| 5. | pair.add([]) |
| 6. | else if 不是R最后一个点 |
| 7. | pair.add([,]) |
| 8. | else |
| 9. | pair.add([,]) |
| 10. | end for |
| 11. | return pair |

在找到轨迹R上的所有样本点在Q上的对应点之后，我们再找轨迹Q在轨迹R上的对应点就简单了。如果想找在R上的对应点，通过时序关系，先找到的上一个BDS对应点和下一个BDS对应点，如图所示，那么的对应点在和之间，即。



在刚才的例子中，如果按照这种方法，先使用DTW算法获得轨迹R中每个样本点的DTW对应点，再使用BDS算法思想做局部优化，可以得到如下如所示的优化结果。我们发现轨迹R中的、、、和都分别找到了最佳的BDS对应点，而不再对应到轨迹Q中的样本点上，而、和的BDS对应点与DTW对应点相同，均为轨迹Q上的样本点。这样一来进一步减小了对应点之间的距离，因此该方法是行之有效的。



本文将这种先使用DTW算法寻找DTW对应点，再使用BDS算法中的思想进行局部优化的方法叫做DTW-BDS对应点匹配算法。因为该算法是建立在DTW算法找到的对应点的基础上，因此该方法找对应点的效果不会比DTW算法更差。第二步使用BDS算法对局部进行调整，不会打破DTW算法原本的建立起来的时序对应关系，并且可以尽可能地获得在三维时空中更加相近的对应点，减小了对应点间的距离，为对应关系做进一步优化，同时也避免了全局使用BDS算法可能导致的时序错位问题。由上面的例子可以看出，该方法是一个有效的方法。

由于本文目的是找出数据轨迹R中与查询轨迹Q最相似的部分，后面的算法中，只用到了数据轨迹段对应的查询轨迹段，因此这里只需要考虑数据轨迹R上的所有样本点在查询轨迹Q上的对应点即可，而不需要找出Q上的样本点在R上的对应点。因此这里不需要双向映射，而是单向映射，将数据轨迹R中的点向查询轨迹Q中映射。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，我们记样本点在轨迹Q上的DTW-BDS对应点为。

### 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。

对应轨迹段可能会出现多种情况，我们对每种情况的处理方式不尽相同。第一种情况是和之间的轨迹段是一条直线段。其中和有可能是轨迹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。这个情况在讲对应点已经讨论过了，如图所示，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，但是和中间可能隔着多个样本点，如图所示，和的对应点和中间夹着轨迹Q的样本点和。



## 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。在第三节中指出了DTW算法和BDS算法在寻找对应样本点时的优点与缺点，因此本文提出DTW-BDS对应点匹配算法，使用二者优点来弥补对方缺点，并给出了在三维时空下轨迹对应点匹配算法，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，为后文求轨迹局部相似性做出铺垫。

# 第四章 局部轨迹相似性算法

在第三章中的主要工作是提出了结合时间维度与空间维度的三维空间，并通过算法得到三维空间下对应点以及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，最后获得轨迹段距离。然后基于轨迹段距离得到局部轨迹相似性算法。

## 4.1对应轨迹段的时空距离

由于后面计算轨迹相似性的需要，本文提出三维时空下的对应轨迹段相似性的概念。在三维时空中，数据轨迹段和对应的查询轨迹段之间的相似性由两部分组成，第一部分是其形状上的相似性，第二部分是轨迹段之间时空距离。下面依次对这两部分进行讲解。首先介绍一下三维时空下，轨迹段间的空间距离的计算方法。

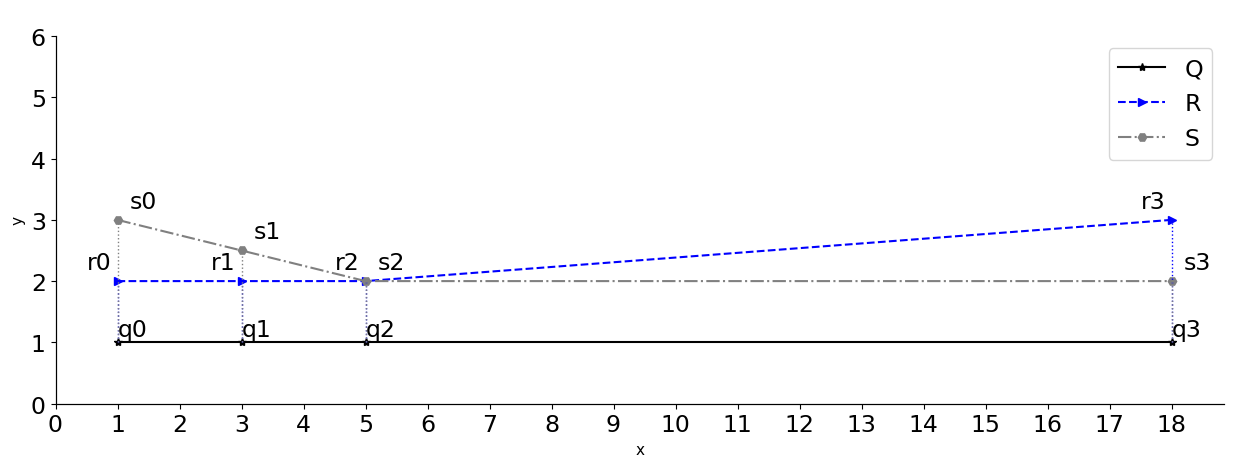
### 4.1.1DTW算法计算二维空间距离及其存在的问题

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，比如DTW算法中是使用动态规划思想先找到最优的对应样本点分布，然后计算所有对应点的距离之和，作为轨迹之间的距离。该方法的好处是利用了动态规划思想的优点，改进了使用轨迹间的欧氏距离算法中的样本点必须一一对应的缺点，使得样本点之间允许“一对多”，解决了时间偏移的问题。这是一个最典型的求轨迹空间距离的做法，后面又有很多基于DTW算法的研究工作，将DTW中使用欧式距离去计算对应样本点的方法做出改进，其中包括引入时间和速度等其它维度的特征，构造更为复杂的距离函数，从而可以多方位地描述了样本点信息。但是这些改进算法并没有改变DTW算法的核心部分，依然是先寻找最优的对应样本点分布，然后计算所有对应样本点的距离之和作为轨迹间的距离。

虽然DTW算法应用广泛，但是也有一定的不足之处，DTW算法只考虑不同轨迹样本点之间的最优对应关系，对样本点产生了严重的依赖，并忽视了轨迹段的概念。如果一条轨迹使用无穷多个点去记录，当然DTW算法有着比较不错的表现，不但可以找到轨迹之间的最优对应点，并且最后得到的轨迹间的距离值可以准确地描述轨迹之间的距离以及相似程度。使用DTW算法计算原始轨迹的相似性的缺点是算法时间复杂度大，耗时严重，如果不考虑耗时的影响，DTW看起来是一个还不错的计算方法。但实际情况是，首先我们采样设备的采样频率是有一定限制的，不可能使用无穷多个点去描述一条轨迹。还有就是在轨迹相似性查询中，经常使用兴趣点和转折点提取算法，将一条有成百上千个样本点的轨迹数据转化为仅包含十几个或几十个点的轨迹数据，这样做的好处是在相似性查询时很大程度地保留了原始轨迹的形状，并且更关注轨迹中的兴趣点，同时减少数据量，降低了相似性计算的时间。下面我们来探讨一下因为采样策略或者提取兴趣点和转折点算法不同的原因，导致同样的轨迹，使用不同的样本点进行表示，会对DTW算法产生多大的影响。

因为DTW只考虑了空间维度，不考虑时间维度，而本文的三维时空中包含的是二维空间和一维时间，因此，为了尽量减少二者场景的不同，我们给出一个使用DTW算法计算二维空间中两条轨迹空间距离的例子。

假设我们的采样设备以一定的采样策略采集到了三个物体的运动轨迹，包括轨迹Q和轨迹R和S。然后使用兴趣点和转折点提取算法提到到了如图所示的轨迹数据，其中轨迹Q包含了起点、终点以及中间的兴趣点和，轨迹R包含了起点、兴趣点和转折点和终点，轨迹S也是包含了起点、终点、转折点和兴趣点。这是三条比较简单的轨迹，该采样设备通过很少的样本点，将三个物体的移动路径准确地表达出来。



我们令轨迹Q为查询轨迹，我们想获取Q和R的相似性以及Q和S的相似性。下面通过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，得到的距离矩阵如表所示。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，Q和R的DTW距离小于Q和S的DTW距离。并且可以得到对应点的关系，分别对应和,分别对应和，分别对应和，分别对应和。我们会根据相似性计算结果中Q和R的距离与Q和S的距离，得到Q和R距离更小，轨迹相似性更大。

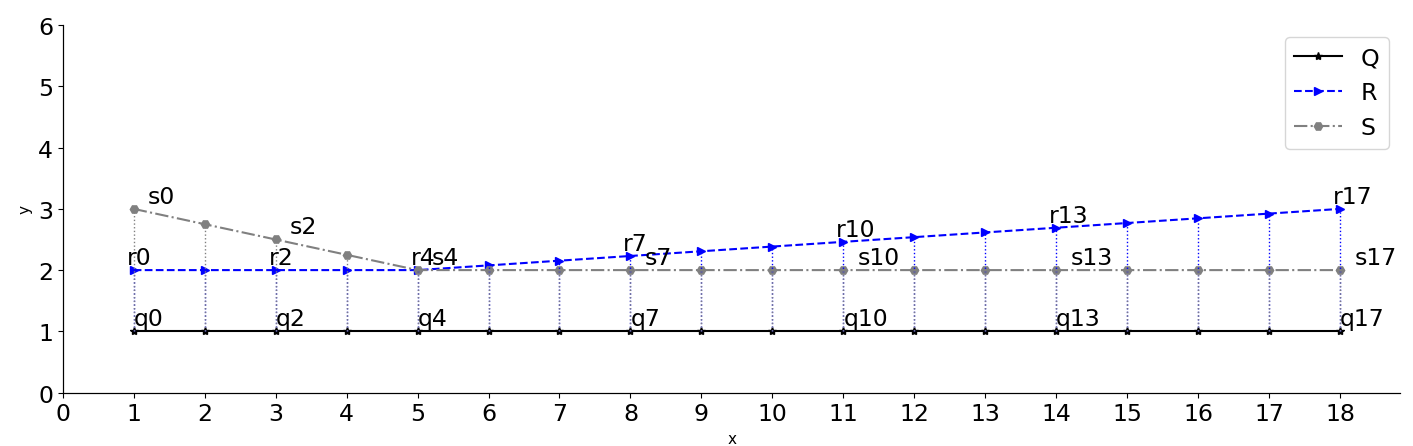
dtw(Q,R)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 |  |  |  |  |
| 0 | 0 |  |  |  |  |
|  |  | 1 | 3.24 | 7.36 | 24.48 |
|  |  | 3.24 | 2 | 4.24 | 19.37 |
|  |  | 7.36 | 4.24 | 3 | 16.15 |
|  |  | 24.39 | 19.27 | 16.04 | 5 |

dtw(Q,S)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 |  |  |  |  |
| 0 | 0 |  |  |  |  |
|  |  | 2 | 4.5 | 8.62 | 25.65 |
|  |  | 4.83 | 3.5 | 5.74 | 20.77 |
|  |  | 9.30 | 6 | 4.5 | 17.54 |
|  |  | 26.42 | 21.07 | 17.54 | 5.5 |

假设刚才产生Q、R、S轨迹的移动对象上还放着另外一套采样设备，这套采样设备的采样策略和刚才的设备稍有不同，最终采样得到的样本点稍微多一点，然后兴趣点和转折点提取算法也和刚才算法也略有差别，最终得到了与刚才不同的轨迹数据，如图所示。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。



现在我们再使用一次DTW算法，分别计算出Q和R的距离，以及Q和S的距离。由于矩阵长度过大，展示不太方便，我们直接给出最终计算结果。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。根据计算结果，我们可以得出查询轨迹Q和数据轨迹S的距离更小，相似程度更高。

对比上述实验，使用DTW算法，对不同采样设备获取相同移动对象的轨迹数据进行相似性计算，第一份数据的结论是Q和R的距离更小，相似程度更高，第二份数据的结论是Q和S的距离更小，相似程度更高。得到了两个截然相反的结论。出现这个问题的原因表面上是由于采样设备的采样策略造成不同的计算结果，根本原因还是由于DTW算法对样本点以及对应样本点之间的距离过于依赖，完全使用样本点而忽视采样间隔，从而导致对采样策略十分敏感，不同的采样策略产生不同的相似性结论。

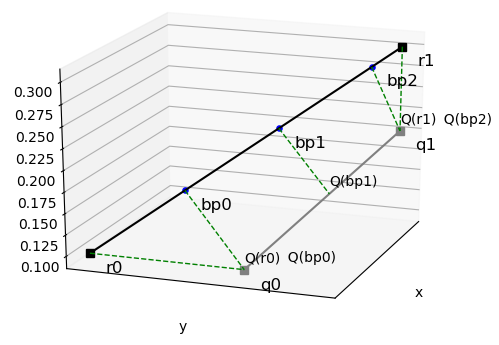
因此，仅根据轨迹数据中的少量样本点、兴趣点或转折点来计算轨迹间距离的做法是不合适的。因为单个的点只能表达了它自己的所具有的信息，没有描述出移动对象在两个点之间的移动的状态，比如移动距离。如果像第二个实验那样，样本点更加密集一点，使用更多的点去描述移动对象在和之间的移动，则会对这段直线段所包含的信息有更好地表达。

### 4.1.2断点

基于以上讨论，提出一个断点（break point）的概念。断点是轨迹段上的点，断点与断点之间距离相同，一条轨迹段中断点的个数取决于轨迹段长度以及规定的断点之间的距离。本文后面将使用断点将一条轨迹段分割为更短的轨迹段。

我们将断点与断点之间的距离阈值设置为，在轨迹段上，从样本点开始，每隔距离设置一个断点，使用断点将相邻样本点组成的一条轨迹段切分为条轨迹段，最终得到轨迹段上所有的断点集合。

下面给出一个断点的例子。如图所示，是查询轨迹R中的一条轨迹段，我们使用三个断点，和将轨迹段分成四段，其中前三段的长度、和的长度均为，而末尾的子段的长度小于。



### 4.1.3轨迹段三维时空距离

在得到轨迹段中的断点之后，我们使用寻找对应样本点的算法，在轨迹段上找到与断点集合中所有断点的对应点。因为我们事先已经使用DTW-BDS对应点匹配算法找到了和，在求断点的对应点时，只需要使用BDS算法，找出到轨迹段中距离最短的点，即的对应点。由于对应轨迹段存在三种空间分布情况，我们首先以第一种最基本的情况来进行讨论，即轨迹段的对应轨迹段是一条直线段。

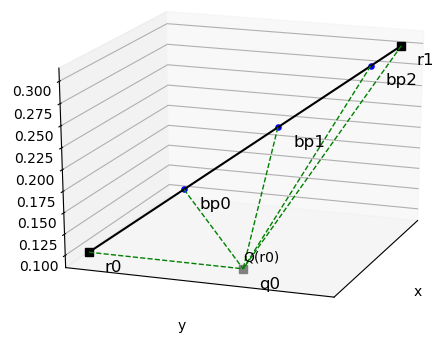
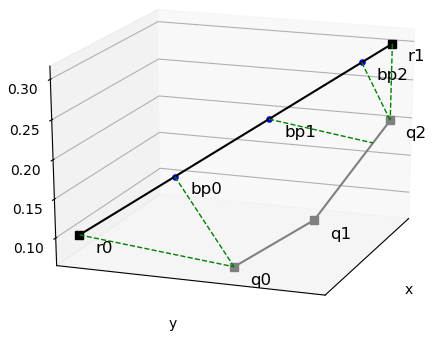
在得到断点的对应点之后，就可以由轨迹段样本点和这些断点集合计算对应点的距离。如果按照图中的例子，我们可以得到每一个断点与其对应点之间的距离、、、、，这是一条轨迹段中的点到对应轨迹段的空间距离。根据每一个端点和断点的位置为得到的距离赋予权值，样本点的权值为，中间的断点的权值为，最后一个断点的权值为，样本点的权值为，然后使用距离乘以权值，再求和，发现结果中很多权值中都包含，可以把提取出来。因为相似性查询使用的是同一个值，所以可以将提取出来的省去，更一般性的公式如公式所示。这里的的可以取大一点的值，这样会使轨迹中的断点少一点，找对应点的计算会少一点，也可以取一个小一点的值，使断点更多，描述轨迹段间隔更准确，但是计算量会变大。最好的情况就是确定一个可以将大多数轨迹段进行切分的值，这样使用更多更均匀的样本点去找对应点，可以更好地描述出轨迹段之间的距离。

当n>=1时，

当n=0时，

其中，为断点的个数。

上面讨论了的对应轨迹段为直线段的情况。当出现第二种空间分布情况，即与为同一个点，如图二所示，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，省了一步求对应点的步骤。当出现对应轨迹段的第三种空间分布情况，即与中间间隔着样本点，如图三所示，与第一种情况的区别就是每个断点需要在与中间间隔的折线段中求出对应点。当求出对应点之后，三种空间分布的对应轨迹段间距离的计算方法都相同，都是求对应样本点的距离乘上对应权值，最后再求和便得到了轨迹段到轨迹段的空间距离。

二 三

由此可以看出，无论对应轨迹段的空间分布是属于哪种情况，如果轨迹段的长度，就需要将轨迹段使用断点分割，然后获取每个断点到其对应点的距离，最后乘上对应权值。在一个确定的值下，最后得到的三维时空下的轨迹段距离，除了和对应轨迹段之间的位置有关，还和查询轨迹段自身的长度有关，查询轨迹段的长度越长，断点个数就越多，最后对应点距离累加值就会大。下面给出对应轨迹段时空距离计算的伪代码。

|  |  |
| --- | --- |
| **算法** 对应轨迹段时空距离 space\_distance\_calculate(Q, R, pair,) | |
| **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d\_space\_list | |
| 1. | for |
| 2. | if |
| 3. | 使用参数切分轨迹段，获得所有断点 |
| 4. | 从pair中获取和 |
| 5. | 计算断点对应点 |
| 6. | d=，， |
| 7. | w=[1/2, 1, 1…1, , ] |
| 8. | else if |
| 9. | d=， |
| 10. |  |
| 11. | d\_space 🡨 d乘以对应权值w再求和 |
| 12. | d\_space\_list.add(d\_space) |
| 13. | end for |
| 14. | return d\_space\_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对应轨迹段形状相似性

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。下面将从夹角和轨迹段长度两方面讨论轨迹段形状上的相似性。

### 4.2.1余弦距离

在我们建立的三维时空中，两条轨迹段可能存在异面的情况，如图所示。我们在这一小节采用向量的方式去讨论轨迹段的夹角，那么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。三维空间中向量和向量的余弦距离如公式所示。在轨迹段余弦距离中，的取值范围为[0, ]，的取值范围为[-1,1]。

如果把上面的公式做一下变换，两边同时乘以，如公式所示，得到的是将向量往向量方向上映射，映射后的向量的长度，为的线性映射。

，其中。



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。

### 4.2.2轨迹段形状影响因子

上一节介绍的余弦距离可以用来描述对应轨迹段之间的夹角的相似程度，但是两条轨迹段的形状相似除了与夹角有关，还和轨迹段长度相关。

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。因此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。

基于以上讨论，我们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。

其中，

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。

当出现第二种情况，即和是轨迹Q中的同一个样本点时，由于此时为0，所以这种情况对应的轨迹段形状相似性的最大值不超过0。计算具体的值时，由于和是同一个点，不能直接使用该点去计算余弦距离，首先需要找到轨迹中与相邻的两个样本点，分别记为和，由于在这两个样本点的中间，所以这一点处的方向应当看做至以及至两条轨迹段方向的平均值，如公式所示，虽然只是一个点，但是可以代表处的向量，该向量的方向为前后轨迹段方向的均值。因此当和处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，轨迹段形状相似性为负数。

当出现第三种情况，即和轨迹段中间经过一个或多个样本点，此时不是一条直线段，不能使用上面的公式直接进行计算。如图所示，和中间间隔着和，此时需要利用寻找对齐样本点的算法，在直线轨迹段上找到和的DTW-BDS对应点，记为和，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，然后计算每一对对应轨迹段的轨迹形状相似性。如果与某一个端点重合，比如就是，那么根据的公式，因为为0，所以此处轨迹段形状相似性为0。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，如公式所示，其中，至是与之间间隔的样本点，m<n。



轨迹段形状相似性计算的伪代码如算法所示。sim\_shape\_calculate的作用是计算对应轨迹段之间的形状相似性，而不是整体轨迹的形状相似性。

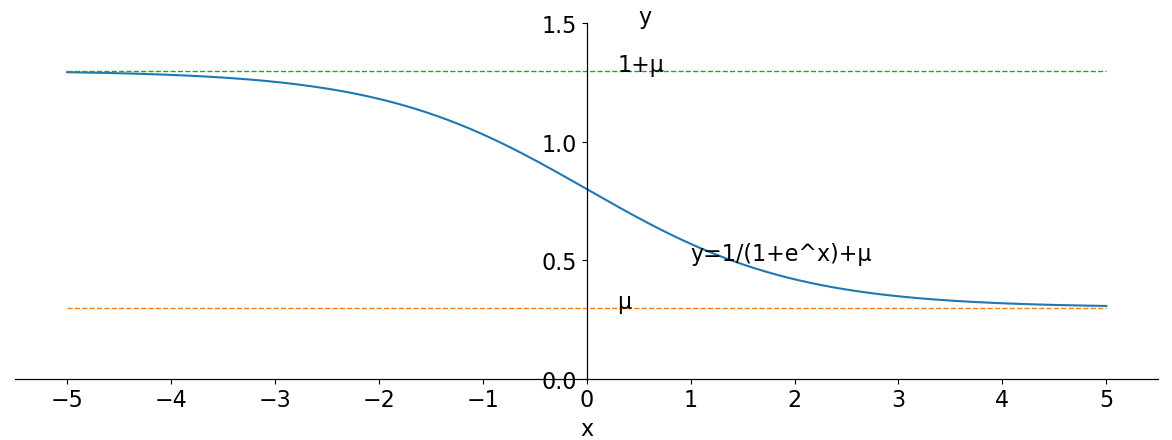
|  |  |
| --- | --- |
| **算法** 轨迹段形状性计算 sim\_shape\_calculate(,, , ) | |
| **输入**：样本点，及其对应点, | |
| **输出**：轨迹段到对应轨迹段, 的形状相似性sim\_shape | |
| 1. | if or |
| 2. | 计算 |
| 3. |  |
| 4. | return |
| 5. | else |
| 6. | for segment in |
| 7. | = segment[0],segment[1] |
| 8. | 获得的对应点和 |
| 9. | sim\_shape += sim\_shape\_calculate(,, , ) |
| 10. | end for |
| 11. | return sim\_shape |

为了将形状相似性转化为影响轨迹段距离的形状影响因子，需要引入sigmoid函数，如公式所示，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，函数形状如图所示。



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的计算公式，如公式所示。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。

其中，。



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，提供给下一步进行局部相似性计算。

|  |  |
| --- | --- |
| **算法** 轨迹段形状影响因子计算 I\_shape\_calculate(Q,R,pair,) | |
| **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 | |
| **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 in R |
| 2. | sim\_shape= sim\_shape\_calculate(,, , ) |
| 3. | I\_shape= |
| 4. | I\_shape \_list.add(I\_shape) |
| 5. | end for |
| 6. | return I\_shape \_list |

### 4.2.3形状相似性与SSD计算方法的比较

之前也有很多论文研究过轨迹段形状上的相似性给轨迹相似性带来的增益，比如前人的一篇论文中，提出SSD（Segment-Segment Distance）方法，使用一个与轨迹段之间形状相似程度相关的函数作为轨迹段距离的参数，如公式所示，描述的是对应轨迹段之间形状上的不相似程度，轨迹段越不相似，的值越大，就会让时空距离乘上更大的系数，使最后描述轨迹之间距离的变大，代表轨迹相似性越低。其中和是对应的轨迹段，是轨迹段，是其对应轨迹段。公式中使用到的角度和本论文中使用的角度含义相同，均为两条有向轨迹段的夹角，范围在[0,]，只不过该论文描述的是二维空间的情况，而本篇论文是在三维时空中进行讨论，研究问题的背景上没有其他区别。

其中，



该论文中使用的Segment-Segment Distance与本论文中提出的轨迹段形状相似性有两个不同点。

第一个不同点在于夹角的使用，由于该论文中形状相似函数是作为距离的参数，那么形状越相似，的值应该越小，使得轨迹段之间距离越小，因此与是正相关的关系，关于的单调递增。而本论文中由于需要获得与形状相似程度成正相关的值，采用了余弦距离的形式，使余弦距离是关于的单调递减函数，余弦距离的值越小表示形状越相似。

第二个不同点在于Segment-Segment Distance中没有考虑对应轨迹段长度这方面，仅考虑了所夹角度。本文的形状相似性距离考虑到了对应轨迹段的长度给形状相似上带来的影响。因为针对两条轨迹，需要找到其中最相似的部分轨迹段，找出的相似部分越长越好。那么本文中的形状相似就不完全由轨迹段所夹的角度决定，对应轨迹段所夹角度相同的情况下，两条轨迹段长度越长，对最后的结果越有利，考虑到这种情况，本文使用了数据轨迹段通过改进余弦距离映射到其对应轨迹段中的长度来计算其对形状相似产生的影响。

## 4.3三维时空下的轨迹局部相似性查询

### 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，对于轨迹R上的轨迹段及其在轨迹Q上的对应轨迹段，我们得到了评价对应轨迹段形状相似的形状影响因子以及描述对应轨迹段三维时空相似程度的空间距离。下面我们给出结合形状影响因子和空间距离的对应轨迹段距离，如公式所示。对应轨迹段的距离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。

### 4.3.2轨迹局部相似性计算

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距离做局部相似性查询。

轨迹相似性查询的最好的情况是返回一条和查询轨迹在时间、空间、方向上完全相同的轨迹。如果回到问题定义中的场景，最希望找到的是一辆车，从A点到C点一直跟着小偷开那辆车，因为这样才最有可能拍下小偷中途丢弃物品的过程。但是由于现实路况较为复杂，获得这么完美的轨迹的可能性几乎没有，很多汽车可能只是在某一段路程能拍摄到小偷的车辆，为了使查询结果更加精确，我们通过局部相似性查询获得的子轨迹不能太长，因为小偷只在从A到C的那段路程以及那段行驶时间中丢弃了物品，其他时间或空间不可能拍摄到该画面。但是获得的子轨迹也不能太短，太短的话，拍摄到小偷丢弃物品的概率就会大大降低。因此我们需要的子轨迹段的长度需要与查询轨迹从长度很接近。

基于上述讨论，我们给出对查询结果中子轨迹段长度激励函数，如公式所示，当子轨迹段的长度偏离查询轨迹Q的长度过长或者过短，就会很小，如图所示。因此我们要设置子轨迹长度限制参数，在查询中保证中不能小于，达到使查询结果的子轨迹长度接近查询轨迹的长度的目的。

在得到这些轨迹R所有轨迹段的与其对应轨迹段的距离后，我们对轨迹段的所有子轨迹进行遍历，如果子轨迹段的长度的值符合小于限制参数就舍弃该子轨迹段，否则就计算每条子轨迹到Q的距离，即计算上所有轨迹段到Q上的对应轨迹段距离之和，将该值作为子轨迹与轨迹Q的距离。轨迹局部相似性计算方法如算法所示。

|  |  |
| --- | --- |
| **算法** 局部相似性计算 sub\_similarity\_calculate(Q, R, , , , ) | |
| **输入**：查询轨迹Q，数据轨迹R，激励参数，断点切分阈值，形状权重，子轨迹长度限制参数 | |
| **输出**：Q中与R最相似的子轨迹best\_segment，轨迹距离min\_distance | |
| 1. | pair= DTW\_BDS\_pair\_match(Q,R) |
| 2. | d\_space\_list= space\_distance\_calculate(Q, R, pair,) |
| 3. | I\_shape \_list= I\_shape\_calculate(Q, R, pair, ) |
| 4. | d\_segment\_list = d\_space\_list I\_shape \_list |
| 5. | min\_distance = inf |
| 6. | for R |
| 7. | l = d() |
| 8. | if and <min\_distance |
| 9. | best\_segment = |
| 10. | min\_d= |
| 11. | return best\_segment, min\_distance |

下面介绍一下整体的查询过程，首先给定时空转换因子，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进行局部轨迹相似性计算。在局部相似性计算中，首先通过DTW\_BDS\_pair\_match获取轨迹对应点，接着分别计算所有数据轨迹段到查询轨迹段的空间距离以及形状影响因子，然后在计算每条轨迹段到对应轨迹段的距离，最后获取数据轨迹上在限制下的与轨迹Q距离最短的子轨迹以及该子轨迹到查询轨迹Q的距离。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，那么该保留子轨迹，加入到查询结果中去，否则丢弃。

### 【问题】缺少一个贯穿始终的例子

## 4.4本章小结

为了获得与查询轨迹最相似的数据子轨迹，本章首先提出了对应轨迹段之间的时空距离以及对应轨迹段形状影响因子，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。最后通过轨迹局部相似性计算，可以得到每一条数据轨迹中与查询轨迹最相似的子轨迹。

# 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 |

[1] 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  
[2] 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.  
[3] 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.

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，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轨迹数据集NorthAmericaRoadNetwork，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。

由于微软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，如公式所示，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。

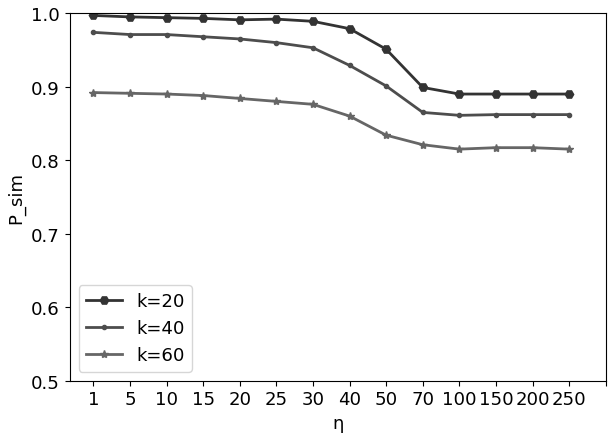
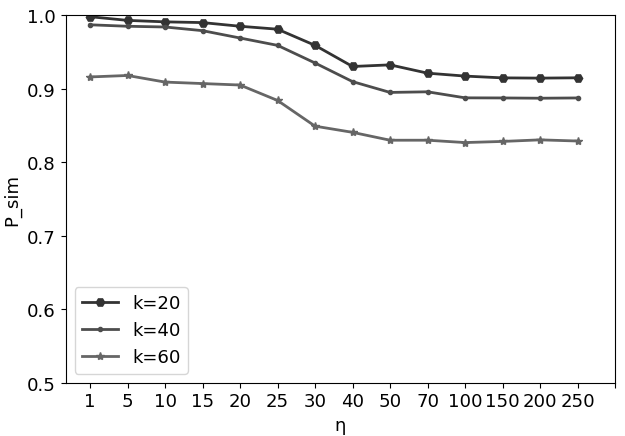
查全率R（Recall）如公式所示，其中FN表示正例被归为反例，即原本相似的轨迹，却没有出现在查询结果中。查全率R与查准率P不同的地方在于查全率R的分母是所有与查询轨迹相似的轨迹数目，来量化查询结果有没有覆盖所有真实结果。和P\_mul的定义类似，我们给出多次查询的查全率R\_mul的定义，如公式所示。

## 5.2算法中参数对结果的影响

本文提出的STS算法中包含如下参数，划分断点的距离阈值，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。

### 5.2.1距离阈值对查询结果的影响

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1米至500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。

a GL数据集 b NA数据集

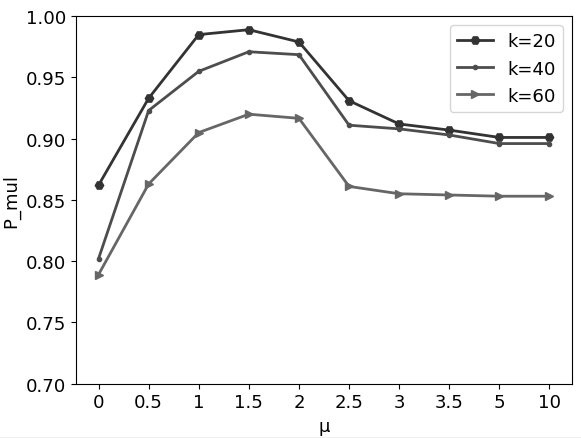
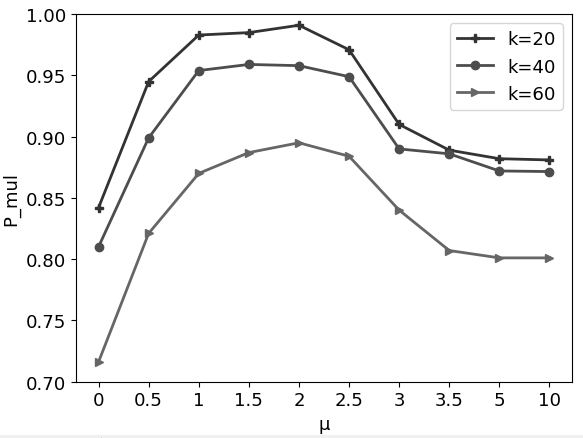
通过对GL和NA两个数据集的实验，得出了不同数据集上的查准率，如图所示。从图中可以看出，不管是那个数据集，相似性查询结果的查准率都随着的变大而减小。

在GL数据集中，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，时查准率下降很严重，其他区域趋于平缓，当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P\_mul的影响程度，如下图所示。

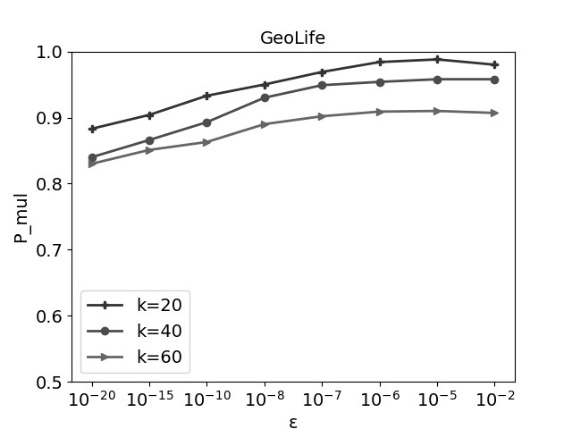
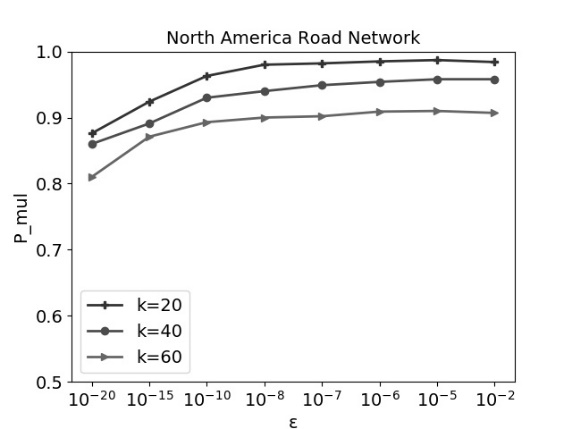
a GL数据集 b NA数据集

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，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。

下面分析一下为什么会出现这个情况。当=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长度限制参数对查询结果的影响

长度限制参数的作用是保证查询结果中子轨迹的长度必须要与查询轨迹长度接近，以便于在数据轨迹中找到与查询轨迹相似的子轨迹，由于其余部分轨迹段距离查询轨迹较远，有了的限制，会过滤掉太远的轨迹段，减小其对轨迹距离的影响。的取值范围为0到1，但是随着越接近1，对数据轨迹的长度要求越严格。当，允许查询轨迹与查询结果子轨迹的长度差10米左右，可以作为的上限，如果太大，过于限制子轨迹长度，可能会因为单条轨迹段长度的原因导致数据轨迹中子轨迹长度没有能符合要求。

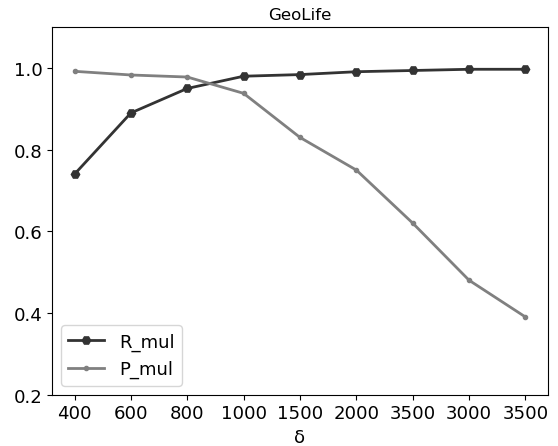
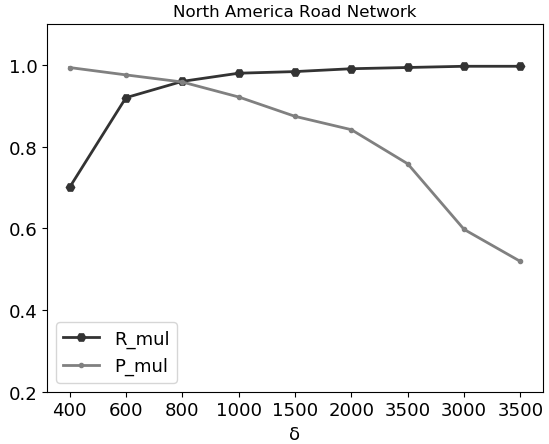
 

由实验可以看出，在区间内，相似性查询结果的查准率P\_mul随着的增大而增大。在 =时，由于对子轨迹长度的限制过小，可能会导致一些与查询轨迹总长度相差较大的查询结果混入进来，虽然由于参数设置原因被算法纳入查询结果中，但是由于其相似部分长度过短，不能被当做与查询轨迹相似的轨迹，因此在GL数据集和NARN数据集中均表现较差，尤其是计算结果的top-60中，掺杂的这种轨迹较多，在NARN数据集中的查准率低至80%。但是随着增大至，对子轨迹长度要求更加严格，很大程度地避免了这种情况的出现。

因此参数的作用不仅在于能再轨迹中找出最相似的子轨迹，还可以保留下来一些局部接近的轨迹，过滤掉整体和局部均相距较远的轨迹。在一定范围内，随着的增大，要求子轨迹的长度接近查询轨迹长度，可以更加准确地找到局部相似的轨迹，提升查准率。

### 5.2.4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P\_mul和查全率R\_mul来共同监控查询结果。

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R\_mul和查准率P\_mul呈现相反的增长趋势。随着不断增大，查询结果中，查全率R\_mul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P\_mul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，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。

## 5.3对比实验

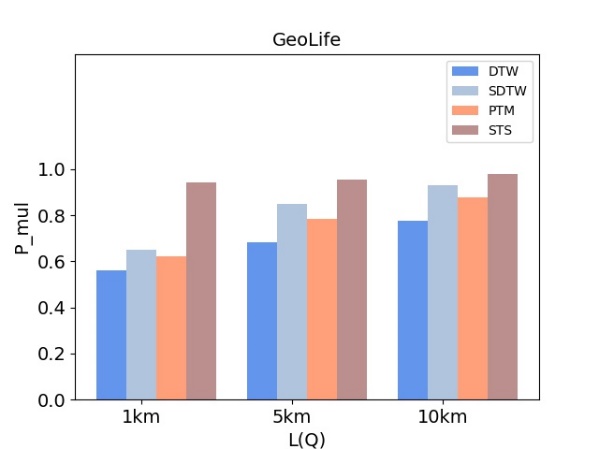
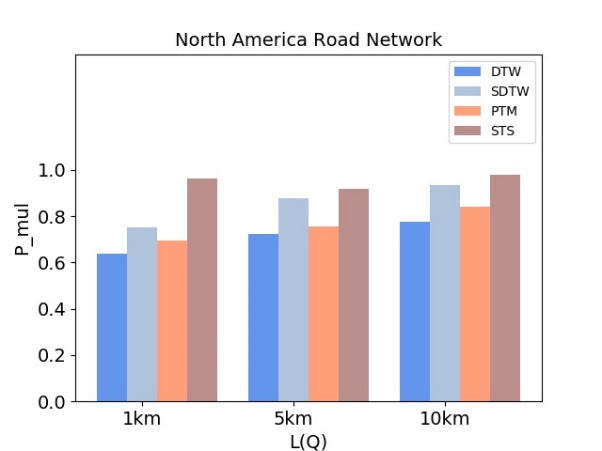
### 5.3.1不同查询轨迹长度给相似性算法的影响

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，可以对SDTW算法中对轨迹形状相似性的权重进行调节，取在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，断点距离阈值取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。

### 5.3.2研究三维时空的有效性

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，分别使用原始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，其中原始DTW算法指的是采用的二维欧式空间距离的DTW算法，这里为了区分，我们将原始DTW算法记为DTW，引入时空距离的DTW算法记为DTW-3d。为了减小查询轨迹长度对实验的影响，这里的查询轨迹只使用长度大于7千米的轨迹，并且为了详细地探究算法的表现，这里将原始数据集划分为不同规模进行实验，分别

### 5.3.3加噪音的算法查准率对比

# 第六章 总结与展望