硕士学位论文

面向轨迹数据的函数连接及相似性查询算法研究

TRAJECTORY DATA ORIENTED FUNCTION JOIN AND SIMILARITY SEARCH ALGORITHM

张冠男

哈尔滨工业大学

2014 年 7 月

万方数据

国内图书分类号：TP315　　学校代码：10213
国际图书分类号：004.4　　密级：公开

工学硕士学位论文

面向轨迹数据的函数连接及相似性查询算法研究

硕士研究生：　张冠男
导　　师：　张炜
申 请 学 位：　工学硕士
学　　科：　计算机科学与技术
所 在 单 位：　计算机科学与技术学院
答 辩 日 期：　2014 年 7 月
授予学位单位：　哈尔滨工业大学

万方数据

Classified Index: TP315
U.D.C: 004.4

Dissertation for the Master Degree in Engineering

TRAJECTORY DATA ORIENTED FUNCTION JOIN AND SIMILARITY SEARCH ALGORITHM

Candidate: Zhang Guannan
Supervisor: Zhang Wei
Academic Degree Applied for: Master of Engineering
Speciality: Computer Science and Technology
Affiliation: School of Computer Science and Technology
Date of Defence: July, 2014
Degree-Conferring-Institution: Harbin Institute of Technology

万方数据

摘　要

当今时代，信息化特征明显，人们观察物理世界所获得的数据均用计算机信息表示。为了更好的观察和分析人们生活的物理世界，人们采用了多种多样的信息数据收集方法。而如何能够将这些数据融合，如何检索到相似的数据已成为研究的热点。本文主要分成两大部分，第一部分是讲述如何采用函数连接算法解决传统方法解决不了的目标轨迹数据与属性值数据的关联问题；第二部分是讲述如何在给定的目标轨迹下，查询相似的轨迹数据。

在面向轨迹数据的函数连接算法中，本文先给出了目标轨迹数据和属性值数据的数据模型的定义，这些定义反映了关系数据库中的数据的存在方式。继而本文给出了循环函数连接算法，这种算法使用循环的方法，对任意一条轨迹和属性值数据进行判断，看是否满足连接条件。在这里判断的条件采用的是函数的方法，本文利用轨迹数据和属性值数据作为函数的输入条件，函数通过数据是否存在相近的时间空间关系和属性值数据是否满足属性值限制条件来判断是否满足条件。当满足条件时，连接成功。之后本文提出了一种基于哈希的算法来改进原有的循环连接算法。基于哈希的函数连接算法能够有效的解决循环算法的缺点，即不能在大规模的数据下使用。在这两种算法基础上，我们又给出了一种优化的策略，这种优化是将目标轨迹的数据进行压缩，从而减少连接操作的运算量。对于这种优化策略，我们给出了相应的理论分析，并给出了误差率。

在面向轨迹数据的相似性查询算法中，本文先给出了新颖的目标轨迹的距离函数度量的方法。这种方法有效的解决了目标轨迹之间相似度的计算，并为后文提出的算法奠定了基础。接下来本文给出了相似性的定义和判断，在本文当中，我们采用欧式空间的距离函数来判断是否两个目标轨迹数据相似。之后本文给出了一种判断相似性的算法，即利用给定的轨迹数据，截取轨迹的子串并和数据库中的轨迹数据做相似性判断的算法。通过这种算法我们能够查询相似的轨迹数据。最后我们给出了面向轨迹数据的相似性查询算法的改进策略，给出了改进算法的召回率，并同时给出了对比实验。

关键词：函数连接；目标轨迹；相似性查询；密度查询

万方数据

Abstract

Nowadays, the features of information are more explicitly. All the Data got by people is stored in database using computer binary information. In order to observe and analyze people's lives in the physical world better, we use a variety of information and data collection methods. How to do fusion operation to these data and how to retrieve similar data has become a hot research field. This paper is divided into two parts. The first part is about how to use the function join to improve the traditional method which cannot solve the problem that target trajectory data associated with the attribute value data. The second part is about how to do the similar query with a given target trajectory.

In function join algorithm based on target trajectory, the paper first gives the definition of the data model of the target data and attribute value data. These definition reflect the existence of data in a relational database. In this paper, we give the loop function join algorithm. The algorithm which uses the loop method judges whether the join conditions are met by one trajectory and one tips of attribute data. Here the join condition to determine uses function join. The paper use the trajectory data and the attribute data as the function of the input. The function gives whether conditions are met through judging whether the data satisfies a similar relationship between time and spatial and whether the attribute data satisfies the attribute value constraint. If the condition is satisfied, we say the data joins successfully. Then the paper gives hash based function join algorithm. This method can improve the loop function join algorithm. Hash based method can solve the problem effectively that loop method cannot use in big data. To improve the two basic algorithm, we propose an optimized strategy. This optimization is to compress the trajectory data, thereby reducing the amount of calculation of join operation. For the optimization strategy, we give the corresponding theoretical analysis and error rate.

In similarity search algorithm based on trajectory data, the paper first gives a novel target trajectory distance function metric approach. This approach is an effective solution to calculate the similarity between the target trajectories, and lays the foundation for the proposed algorithm later. In this paper, the following

万方数据

definitions and similarity judgment are proposed. Then we use the distance function of Euclidean space to determine whether the two data are similar. Then we give a method to determine similarity between trajectories which use a given trajectory data interception trajectory substring and trajectory data in the database to make similar judgments algorithms. By this algorithm we are able to query a similar trajectory data. Finally, we present an optimization strategy of similarity search algorithms based on the target trajectory to improve performance with the recall rate and comparative experiment.

Keywords: function join; target trajectory; similarity query; density query

万方数据

目　录

摘　要 I
ABSTRACT II
目　录 IV
第 1 章 绪论 1
1.1 课题背景及研究的目的和意义 1
1.2 本文的主要研究内容 3
1.3 本文的组织结构 4
第 2 章 面向轨迹数据的函数连接及查询简介 6
2.1 目标轨迹及属性值 6
2.2 函数连接算法 8
2.3 查询算法 9
2.4 难点及未来发展 11
2.5 本章小结 11
第 3 章 面向轨迹数据的函数连接算法研究 13
3.1 问题的定义 13
3.2 面向轨迹数据的函数连接处理过程 16
3.2.1 匹配属性值元组和轨迹段元组集合 16
3.2.2 循环连接算法 18
3.2.3 基于哈希的连接算法 19
3.3 面向轨迹数据的函数连接算法的改进 20
3.4 误差边界 22
3.5 面向轨迹数据的函数连接算法实验分析 22
3.5.1 连接算法的效率 22
3.5.2 效率的比较 24
3.6 本章小结 27
第 4 章 基于目标轨迹相似性查询算法研究 29
4.1 目标轨迹相似性查询 29
4.2 问题的定义 31
4.2.1 轨迹的定义 31
4.2.2 轨迹的相似性 33

万方数据

4.3 面向轨迹数据的查询算法 35
4.4 优化策略 37
4.5 查询算法实验分析 38
4.6 本章小结 39
结　论 40
参考文献 41
攻读硕士学位期间发表的论文 46
哈尔滨工业大学学位论文原创性声明和使用权限 47
致　谢 48

万方数据

第1章 绪论

1.1 课题背景及研究的目的和意义

当今时代，为了观察分析物理世界，人们采用了多种多样的信息数据收集方法。为了能够获得更好理解的信息，人们要重新梳理通过不同方法获得的信息数据，尽量多的得到感兴趣物体的各类属性信息并加以融合。例如，

动物保护组织和生物学家往往使用不同种工具来观测野生动物[5]。像位置跟踪设备、全球定位导航和无线电遥感器等设备通常被用来跟踪动物的运动轨迹。配有红外触发的相机[1]也被使用来探测野生物种的存在，并以此来估计种群的密度等参数[18]。同样，种群的规模依靠飞行器从空中的观测，物种栖息地的采光，温度，湿度等数据依靠传感器网络采集[6]。有了这些收集数据的方法，我们能够得到同一种群的大量的数据，比如图片、种群规模、运动轨迹、气候温度湿度和栖息地生存条件。如果能够将获得的这些数据通过某种方法融合起来，这将对人们研究野生动物的行为习惯产生非常大的帮助。

在军事侦查领域中，许多设备和复杂的系统被使用来完成相关的任务。因此雷达设备的需求越来越旺盛，多种不同的雷达系统被人们建造起来。这些雷达系统能够检测目标的方位，速度，经度和维度等等，这些目标包括汽车，船只，飞机等[3]。雷达警示接收器[4]或者被动雷达能够收集目标所发出的各种雷达信号[11]。频率信息，脉宽、脉冲能量波等很多雷达信号特征能够被这两种雷达捕获。如果能够将主动侦测到的目标信息与被动捕获到的信息结合在一起，这将对人们分析目标的特征带来非常大的帮助。

然而，由于这些信息通过不同种观测方法来收集到的，所以并不存在一种直接独特的方法来自动的关联这些信息数据。为了解决这个问题，我们给出了一种新颖的面向轨迹数据的函数连接算法。在算法中，目标的轨迹信息数据作为索引，剩余其他的信息数据通过建立的索引来和轨迹信息关联。在面向轨迹数据的函数连接算法中，基于时间和空间关系的条件用于决定获得的轨迹信息是否关联属性信息。这其中，时空关系条件可以理解为在同一时间段内，多组数据在空间上是否相近。同时，与轨迹信息关联的其他信息数据可以被看作被观察物体的属性信息数据。除了时间空间关系，面向轨迹数据的函数连接算法

万方数据

还需要满足人为给定的限制条件。例如，红外触发的相机能够探测一个扇形区域，但人们仅仅需要分析角度在 27 度和 42 度之间的区域信息[9]，这时，角度成为了限制条件，我们的目标就变成了在给定范围做函数连接算法。

对于所有和 GPS 获得的数据所关联的图像信息，限制条件是野生动物的毛发颜色和图像中种群的大小。通过关联图像信息和 GPS 轨迹信息，生物学家能够得到更加精确的来自不同野生动物栖息地的种群密度估计。同样，通过被动雷达接收到的雷达信号能够和飞行器的轨迹想关联，条件是在满足时空条件的前提下，信号方向恰好位于轨迹的点集的内部。在满足认为给定的限制条件下，大量的信号数据信息能够被关联到轨迹信息当中。

同样，我们可以利用以上提及的技术完成获取用户轨迹或者移动的目标的轨迹的任务。那么，轨迹数据就会被大量存储在提示位置信息的系统，交通导航系统和其他基于位置信息的信息系统。这些应用已经存储了非常多的轨迹信息，而其中的一些轨迹则包含有很有用处的用户个人信息。例如，通过分析处于大厦的办公室的用户的行为轨迹，我们能够找到许多座位、房间、楼梯和一些其他的被频繁使用的基础设施。这些分析的结果通常被用来管理和维护工作大厦。在交通导航系统的实际应用中，一个司机能够通过查找其他司机的更早的开车路线来检查道路，判断道路的拥挤程度或者可通过程度。在其他应用方面，我们可以在运动物理学中应用，通过在运动员身上安装位移传感器，分析运动员的各个肢体运动的轨迹来改善运动员的一些运动的细节，进而提高运动成绩和效果。因此为了找到隐藏在轨迹当中的模式信息，我们提出了面向轨迹数据的相似性查询。

基于距离的查询有范围查询、NN 查询等等。这些查询使用的距离是目标轨迹与点之间的距离，而这些距离能够利用数据库很好的管理。然而，这些查询在找出目标轨迹的运动模式方面缺差强人意，不能很好的完成任务。正如上文所提到的，因为我们对解析目标轨迹的运动模式很感兴趣，所以很有必要给出一些新颖的方法来完成这种分析。有关于相似性序列的研究已经进行了许多年了[20,21,23,32,34,44]，但是传统的针对数据序列的技术，包括距离函数，索引方法不能够被完全用到目前的面向轨迹数据的相似性分析上面。因此在第四章中我们将给出一种新的数据模型来定义目标轨迹数据，同时也将给出一种新的相似性查询方法，这种方法基于时间序列数据库中目标轨迹的之间的新的相似度的距离度量。

万方数据

基于密度的查询是另一种对目标轨迹的重要的查询方法。这种查询的目的是找出指定目标轨迹当中密度比较高的区域。密度查询可以在城市交通管理系统当中找出已经拥堵的道路或者具有潜在可能拥堵的道路。例如，利用交通的监控系统定期的检测密度区域来识别是否有潜在的交通拥堵状况。

目前的关于密度查询的研究成果[33,35]假定目标是自由移动的状态，并且是在欧式空间定义这种密度查询的。在这些前提下，很难有效率的给出一般性的密度查询结果。因此密度查询的焦点转移到了简化的查询[33]或者说特定条件下的密度查询[35]。这些方法利用网格将数据划分成不想交的块，然后根据既定的参数给出固定大小的区域的密度分析。然而，真实的目标的密度区域往往并不是正方形，而且区域可能更大或更小。在实际应用中，简化的密度查询方法不能反应出真实的密度区域和形状区域。我们关注于道路网络中的密度查询。道路上存在着非常多的车辆，这些车辆可能以任意的方向和形状行驶在道路上，即这些车辆可能会形成不同种密度。在第四章中，我们对于密度查询仅仅做了扩展的探讨，给出了当前的一些密度查询方法和策略。

本课题来源于国家 973 项目"海量信息可用性基础理论与关键技术研究"(编号：2012CB316200)，国家自然科学基金项目“数据质量管理中实体识别关键技术的研究”(编号：61003046)和教育部博士点基金项目“复杂数据上实体识别关键技术的研究”(编号：20102302120054)。

1.2 本文的主要研究内容

目前，大部分对于目标轨迹的连接算法的阐述不够完全，传统的连接算法主要是使用相同的属性值来做连接，采用朴素循环和哈希的算法完成对数据库中的数据连接。本文则主要给出了数据库中的关系表中不含有相同的属性时的连接算法，即面向轨迹数据的函数连接算法。这种方法突破了只能利用相同属性连接的局限性，将传统的连接条件扩展到函数连接，从而能够使这种方法更广泛的应用到更多的连接操作中。

而对于传统的面向轨迹数据的相似性查询算法，本文指出了这些算法所存在的问题，即相似性算法运行的条件过于苛刻，与实际应用脱节。本文给出了一种新的基于目标轨迹数据模型的定义，相似性判断的方法（相似性的距离度量），相似性查询算法，最后提出了面向轨迹数据的密度查询的一些方法的扩展概述。

本文的主要工作和主要的贡献有以下几点：

万方数据

(1)目标轨迹数据和属性值的数据模型的定义。本文给出了目标轨迹数据和属性值数据的形式化定义，指出了前人的参考文献当中目标轨迹数据和属性值数据的定义的缺点和不足并完善了一些概念性的定义。

(2)朴素的函数连接算法。本文给出了一种利用循环的函数连接算法。首先我们先定义了函数连接的概念，即利用不同的关系表的不同属性集来建立函数关系。本文采用的函数是以时间和空间为参数，通过一些运算来确定函数连接的条件,即如果满足时空关系吻合,则函数返回真值的方法来确定条件的真假。对于连接操作我们还定义了属性值的限制条件。存在这种条件则是由于不同的实际应用背景是满足不同的特定环境的。当关系表满足函数条件和限制条件时，连接成功。我们利用了最朴素的循环的方法判断任意目标轨迹和属性值元组是否满足连接条件。

(3)基于哈希的函数连接算法。在提出了朴素的基于循环的连接算法后，本文给出了另外一种基于哈希的函数连接算法。这种算法能够处理数据量特别大的情况，即将目标轨迹和属性值元组全部哈希之后，在每组哈希桶内进行连接操作，当满足函数连接条件和限制条件后，得到连接结果。这种算法有效的解决了大数据下朴素的循环算法效率低下的问题。

(4)函数连接算法的分析与实验。在这一部分中本文首先给出了目标轨迹的压缩的算法，即优化的策略。这种策略能够有效的增加函数连接算法的运行效率。之后我们给出了这种优化策略的误差率分析。最后我们给出了函数连接算法的大量实验，并同时给出了实验的结果与分析。

(5)目标轨迹的距离函数度量。本文给出了新颖的目标轨迹的距离函数度量方法。首先形式化的定义了轨迹的数据模型，包括连续的目标轨迹，离散的目标轨迹等。这些方法为后文相似性判断奠定了基础。

(6)目标轨迹的相似性的定义与判断。这部分本文给出了目标轨迹之间相似性的度量，即利用基于欧式空间的距离函数来确定。之后本文给出了目标轨迹的相似性查询算法描述，即如何在给定轨迹下查找相似的目标轨迹。最后本文给出了相似性查询算法的复杂性分析，算法改进策略和对比实验。

1.3 本文的组织结构

本文分为五个章节，主要内容和贡献安排如下：

第一章：绪论。这一章主要给出了全文的概述，包括课题研究的背景意义和目的，全文的主要研究内容以及全文的章节安排。

第二章：面向轨迹数据的函数连接及查询简介。这一章主要介绍了相关的

万方数据

概念，算法。主要包括目标轨迹和属性值的概念，函数连接算法的概念，相似性查询、密度查询等概念以及未来这个方向上难点。

第三章：面向轨迹数据的函数连接算法研究。这一章主要介绍了目标轨迹的函数连接算法。包括函数连接问题的定义，面向轨迹数据的函数连接处理的过程(匹配属性元组与轨迹元组集合、循环的连接算法、基于哈希连接的算法)，面向轨迹数据的函数连接算法的改进，误差边界及分析，连接算法的实验和分析（连接效率、效率的比较）。这章从多个角度给出了面向轨迹数据的函数连接的算法的主要研究内容。

第四章：面向轨迹数据的相似度查询算法研究。这章主要讲述了面向轨迹数据的相似度查询算法，包括目标轨迹的相似性查询简介，问题的定义（轨迹的定义，轨迹的相似性定义），面向轨迹数据的相似性查询算法，算法的复杂性分析。最后本章给出了面向轨迹数据的相似性查询算法的优化策略，并给出了对比实验。

第五章：总结全文。

万方数据

第2章 面向轨迹数据的函数连接及查询简介

2.1 目标轨迹及属性值

目标轨迹通常被定义成一系列位置信息的集合，这些位置信息一般是处于多维的欧式空间当中（二维或者三维的居多）。在许多应用中，轨迹数据是由不断侦测给定目标的位置信息并聚合而得到的，因此这些轨迹数据还含有时间属性信息。例如野生动物的观测，移动手机应用，地图应用，移动制导等等。

更进一步讲，信息技术发展快速，移动计算、传感器、GPS[2]等技术的大规模使用使高效的收集大量时间空间数据成为可能。在已经收集到的这些数据上，数据分析和应用便成为了一个人们感兴趣的方向。例如，在地图应用当中，人们想办法利用获取到的数据标示出人们想去的路线。在环境信息系统中，跟踪野生动物与天气条件已经非常常见，并且大量的时间空间信息数据被存数。人们需要在这些轨迹数据上进行分析，找出这些目标轨迹信息的共同的模式[28,29]，因此合适并有效率的定义轨迹数据的模型将对轨迹数据的分析产生重要影响。

通常在相似性轨迹查询过程中，我们往往利用时空数据库，即给定一个数据库 D 和一个查询 Q，我们要找到一个或一组轨迹 T，满足 Q 与 T 的距离最小。那么在这里我们需要定义的就是轨迹之间的距离和数据库的索引，在本文当中，我们主要关注轨迹之间的距离。

轨迹之间的相似度，或者说轨迹之间的距离函数一般需要考虑以下问题：

(1)不同的采样率和速度。由于轨迹数据本身是连续的数据，因此我们需要对其进行离散化，使用轨迹点集的形式来表示轨迹，因此对于线的采样需要根据具体的应用来判断。虽然在数据的收集过程中我们得到的是离散的时间与二维点信息，但我们真正使用的时候并不一定使用全部的点，而是采用不同的策略采样来完成相应算法。

(2)不同空间领域的相似的模式。这里的模式指的是方向与形状。目标可以以一种相似的规律运动，这时我们采集到的轨迹便是相似的轨迹。轨迹的模式往往难以挖掘和捕捉。

万方数据

(3)不同的长度。一般的距离函数能够处理的不同轨迹是相同长度的。当遇到不同的长度这种情况时，我们不得不决定是否要截断较长的轨迹序列，或者用零或其他值增加较短的轨迹序列。一般来说这种处理较为复杂。

(4)效率的问题。当计算相似性的时候，效率往往是需要人们重点考虑的，因为过低的效率会导致难以承受的计算量和等待的时间。

二维平面的点集，或者是三维的时间序列的数据集。在三维数据中，第三维表示的是时间戳。在相似性查询各个方向当中，我们更关心的是轨迹形状的相似程度。轨迹点的采样对测量两个轨迹的相似程度是有重要影响的，而对于时间维度，我们仅仅需要判断重合程度即可，并不会花费太大的代价。在基于一维时间序列的相似性分析已经被充分考虑和研究过了，而且已经应用在诸如股票预测，经济价格预测，销售量，天气数据和生物学动物迁徙等实际用途中。然而，这种基于距离函数和索引的方法不能够直接用在判断目标轨迹相似性上面，主要原因是因为他们的方法过于独特。

为了在数据库中更好的管理这些轨迹数据，我们定义了轨迹的数据模型，就像二维空间上的直线一样。轨迹之间的相似性定义成了离散的线段之间的欧氏距离。我们给出的相似性查询方法能够被用来发现更有用的嵌入在轨迹信息的模式。例如在某个地方，野生动物的密度突然增加，可能原因是前方的道路阻塞或其他原因。

属性值数据是指人们在观察目标时所收集到的含有目标各种属性的数据。人们通过一些可以侦测目标轨迹的手段来收集目标轨迹的信息，同样也用一些传感器等手段获取除目标轨迹之外的其他的属性信息，这些属性信息包括目标各种特征数据，时间数据，以及相应的空间参考数据。

由于属性值数据是描述目标的各项特征的，因此在实际应用当中，属性值往往都含有属性值域的约束。例如，在观测某个野生动物栖息地时，人们收集到的温度与湿度的数据是有值域范围的，温度过高或者过低并不属于临时的研究范围。这里的温度太高或者太低并非是数据有错误，只是人们在做特定的研究是需要的温度范围可能很小，此时就需要我们剔除一部分数据，这便形成了属性值的约束。

为了能够将属性值数据和目标轨迹数据想关联，以获得更有参考价值的信息，我们需要提取属性值数据与目标轨迹数据相关联的部分并加以融合。属性值数据中含有空间参考信息，而目标轨迹数据中含有轨迹的时间空间信息。因此，利用两类数据的时间空间信息就可以做出关联操作。通过关联操作我们能

万方数据

够得到新的数据集合，该集合的元组来自之前的两类数据，这些新的数据集合能够更好的帮助我们完成相应的分析任务。

2.2 函数连接算法

对于数据库中的两个或者多个关系表，传统的连接算法有等值连接、θ-连接、直连接、半值连接等等。然而这些连接算法的共同特点是要有相同的特征属性或属性集合。通过比对相同的特征属性的值是否相等或满足一定条件，来判断两个元组是否能够连接到一起。总之，传统方法能够连接的关系表必须含有相同的属性或属性集合，但本文当中的属性值数据与目标轨迹数据并不完全满足传统连接条件。因此我们重新定义了一种方法来处理这种没有相同属性或属性集合的关系表的连接。

在关系数据库领域中，[19]给出了一种利用新的数据库索引技术来改进关系表之间的连接操作，[17]提出了一种空间索引的方法来提升空间连接操作的效果。这种算法需要利用空间的网格来将数据划分，进而在网格内进行连接操作的运算。网格文件[18]和 KD 树[8]一般也用来被解决这类多维属性的连接操作问题[14]。对于一些高维空间的连接操作，[7]亦给出了划分区域的方法。在处理分块的目标轨迹的方法中，[10,12,16]给出了一些判断特定区块之间的相隔程度的方法，然而这些算法并不是很有效率，主要的瓶颈在于磁盘操作次数。为了更好的解决这种关系表的连接，我们先来分析这两类数据的特征。

对于属性值数据和目标轨迹数据，我们先来分析一下两类数据的特征。目标轨迹数据含有目标物体的运行轨迹点，以及每个点的采集时间。这两部分数据组成了目标轨迹数据的时间空间信息。属性值数据中含有空间参考信息和数据采集的时间，这两组数据构成了属性值数据的时间空间信息。这里，空间参考信息主要有两种：方位角度信息，距离信息。本文利用方位角度信息来完成两类数据的关联操作。

传统的等值连接操作是利用相同属性的相同值完成连接。这两类数据含有相同的时间特征属性，但相同的时间并不能作为连接的依据，因为空间上两组数据可能并不相交，或者说并不处于同一个空间的位置。因此，传统的等值连接算法并不适用本文给出的两类数据。对此我们提出了基于函数的连接算法。

由于传统方法不完全适用，于是我们采用了函数的方法。我们给出了函数的定义，并确定了函数的参数和返回值。属性值数据和目标轨迹数据的关联采用的是函数的方式，利用两类数据的时间空间值作为函数的参数，之后通过计算函数值，得到函数的真值或假值来判定两组数据是否满足关联条件。

万方数据

函数的设计至关重要。函数要完成对属性值数据和轨迹数据的关联性的判断。首先，函数要保证两类数据的时间上的重合性，即满足两组观测数据在时间上是吻合的。其次函数要保证两类数据在空间上的重合性，即满足两组观测数据所观测的物体近似在一个空间位置上。当函数判断同时满足这两个条件之后，函数会返回真值，我们就确定了当前的两组数据是满足函数连接条件的，可以进行连接操作，进而生成新的元组数据。

利用函数，我们能够关联属性值数据和目标轨迹数据，但这些关联后的结果并非是我们都想要的。根据具体应用的背景，我们能够得到属性值数据的相应的约束，因此在函数连接过程中，我们必须保证参与连接的属性值数据满足背景约束条件，即满足属性值限制条件。

这种基于函数的连接算法能够有效的处理没有完全相同的属性或属性集合的情况。后文将给出具体的问题的定义、算法的流程以及实验的结果。

2.3 查询算法

本文讲述的查询算法包括面向轨迹数据的相似性查询和密度查询。在相似度查询方面，最简单的方法就是将需要比较的两条轨迹映射到空间向量之中，然后利用 P 度距离来定义两个向量之间的距离。P 度距离定义为：$\mathrm{L}_P(\bar{x},\bar{y}) = (\sum_{i=1}^{n}(x_i - y_i)^P)^{\frac{1}{P}}$。当 P 为 2 时，P 度距离就变成了欧式距离。P 为 1 时为曼哈顿距离。很多不同的距离度量被使用和扩展延伸[21,37,33,29,39,43,42]。

另一种方法是基于最长公共子序列（LCSS）[26,29]。这种方法展示了良好的距离度量，只要我们允许序列增、删、改，那么就能得到良好的距离效果。由于目标轨迹的值都是真实的值，我们需要对值进行近似而不是完全的匹配，例如浮点数在计算机中不应完全比较相等。在[]中引入了概率性的算法来计算最长公共子序列。

其他的用来定义时间序列的相似性主要是基于一些特定的属性[32,36,37]。这些属性来自时间序列数据，并通过他们来定义序列的相似性。一种很有趣的表示好似时间序列的方法是利用在规定的时间区间的方向来表示[44]。一个最近的工作给出了一种利用 EM 算法（期望最大化）来解决较小的轨迹集合的聚类问题。然而这种方法存在扩展性的问题，而且它的前提假设也比较严格。

一般来讲，查询算法是给定一个已知的轨迹，然后通过算法在给定集合中寻找与给定轨迹相似的轨迹并返回。但有些时候我们需要直接寻找一个轨迹的

万方数据

集合，其中这些轨迹彼此都相似或满足一定的相似条件。因此给出相似集合的查询算法对某些应用是很重要的。本节给出移动目标的集合查询算法。

对于某些应用，找出一组一起运动的目标是很有意义的。给定一个目标轨迹集合，密度常数 m 和 e，时间周期 k，一个集合查询返回所有的满足特定条件的目标轨迹，这些目标轨迹的每一条都至少有 m 个轨迹与其相似，在这里，用于判断相似的距离 e 至少是使用连续的 k 个时间段周期。在一个大型的数据库中寻找这种目标的集合需要包含很多基于时间空间关系连接的合并操作。然而，连接操作会导致更多的复杂的计算。这一部分给出了三种有效的算法来处理这种集合查询。一种方法是将寻找移动目标的集合转化为寻找移动的簇，通过寻找簇来给出移动目标的集合。另两种方法是通过减小轨迹点的总数，并利用线性近似的方法来简化轨迹，进而搜寻简化的轨迹的相似集合。

为了解决发现并识别簇的问题，我们可以利用基于密度的方法来搜寻空间上相近的目标点[21]。在减少目标轨迹点的数目的方面，我们可以采用 Douglas-Peucker 算法来简化轨迹[30]，进而简化运算的复杂性。而在轨迹的简化方面，还有一些其他的近似方法，例如采用压缩的方法[21]，我们也可以采用空间轨迹点的压缩的方法来简化轨迹，减少运算。

基于密度的查询目标轨迹的问题最早是在[33]提出的。这篇文章的目的是为了找到空间中时间和空间的密度高于一个阈值的区域。在[33]中，作者找到了基于密度的查询的一般性的困难，并给出了一个很有效率的方法来简化这种查询。具体的讲，他们将数据划分成了很多不连接的块，并简化密度查询返回块，而不是直接检查某个空间中的区域是否满足密度条件。但这种方法有可能产生结果的遗漏或错误。为了解决这个问题，[35]定义了一种更加有效的方法来却表并没有结果遗漏或者错误。这两篇文章都假设目标在自由状态下移动，并在欧式空间下定义了密度查询。值得注意的是，对很多应用来讲，高效的在给定的空间中查询相似的目标轨迹是很关键的。考虑一些真实的场景，在道路网络中，人们希望查询道路网络中的车辆分布，这样就能够分析出车辆的密度，就知道了道路中哪里堵车，哪里不堵车。[31]则给出了在空间网络中密度查询处理的方法。对于连续侦测到的目标轨迹数据，且目标区域的密度会随着环境的变化而变化的密度查询处理，[35]给出了一种比较高效的算法来解决连续的目标轨迹密度查询问题。

万方数据

很多与密度查询计算相似的基于时空关系的查询都对密度查询做了比较深入的讨论[34]。基于密度的查询需要知道确定范围内的物体的总数，而且基于密度查询的范围必须要满足给定的密度阈值。

已经存在的一些聚类的方法能够给出大多数密度集中的地方的中心点，[31]给出了比较好的计算中心点的方法，但需要花费比较高的代价。这些技术对于密度查询的要求并没有太多的依赖，即聚类并不需要查询的条件。他们不能够识别那些没有给定阈值的区域的密度大小。而且他们在跟踪连续的目标轨迹的点的时候效率也不是非常高。

在本文中密度查询并不作为重点讨论，而是在文中给出一些前人在面向轨迹数据的密度查询方面的工作。

2.4 难点及未来发展

在目标轨迹数据与属性值数据的关联方面，函数连接方法仍然有许多可以研究的问题，例如如何在时间空间关系并不明确的地方使用函数连接，如何能够更好的利用属性值限制条件，如何能够进一步简化运算，给出更好的复杂性理论分析方面，使用效果的使用函数连接等等。

在面向轨迹数据的相似性查询方面，主要是使用距离函数作为轨迹相似性的度量。在使用距离函数作为度量方法时一般需要考虑一些难点问题，例如对于目标轨迹的采样速率和速度、目标轨迹的数据在不同的空间领域的相似模式（方向、形状等）、目标轨迹的长度的使用、以及计算效率等问题。在未来的相关的研究中，在使用除距离函数之外的其他的度量相似性的方法可能成为研究的热点。

还有一些方法使用维度缩减的技术来完成相似性查询。在维度缩减之后，一般会用多为索引结构在空间中重新索引数据。在这里使用的维度缩减的技术包括快速傅里叶变换，主成分分析，矩阵奇异值分解等等。对于一个给定的索引结构，有效率的索引依靠维度缩小后的空间所提供的简化运算。然而，在选择维度缩减方法的时候，并不是简单的选择一个简化效果最好的技术，而是要考虑如何在简化的过程中达到的数据损失最小。

2.5 本章小结

本章是对面向轨迹数据的函数连接和查询算法的简要概述。首先，本章介绍了目标轨迹数据的相关概念。这些概念包括目标轨迹的定义、应用以及目标轨迹的相似性判断等等。目标轨迹的表示有很多种，主要包括二维表示、三维

万方数据

表示的时间序列数据。本章对每种表示给出了优缺点和相应的应用环境。对于目标轨迹之间的相似度，本章给出了轨迹之间的距离的度量的一般准则，即不同的采样率和速度；不同空间领域的相似模式；不同的长度和算法效率。

其次，本章给出了属性值数据的相关概念。其中包括属性值数据的来源，表示，特征，应用背景等等。属性值数据是收集起来的观测目标的特征信息，而这些信息需要满足特定的背景，因此便引入了属性值限制条件。利用属性值限制条件，人们就可以获得更加精确的经过连接操作的数据。

再次，本章给出了函数连接算法和相似性查询的算法。传统的函数连接算法包括等值连接、θ-连接、直连接、半值连接等，我们给出了这些传统方法的优缺点，并针对必须含有相同属性的这一缺点给出了改进的方案，即采用函数连接的方法。接下来本章介绍了函数连接的具体做法和在实际应用中的实践。函数连接算法中的函数的设计对整个连接算法是至关重要的。

对于相似性查询算法，本章给出了一些前人的工作成果的总结。其中包括面向轨迹数据的相似性查询和密度查询的一些文献。例如，本章给出了度量相似性的距离函数的一些相关的工作，相似性查询中的最长公共子序列的相关算法。利用时间序列数据的特性的一些相关工作。对于密度查询，本章也给出了一些相关的成果。

最后，本章给出了面向轨迹数据的相似性查询和密度查询的难点及未来的发展方向。

万方数据

第3章 面向轨迹数据的函数连接算法研究

在信息时代，信息以爆炸式的速度增长，信息的来源多种多样，人们可以通过多种方法来获取信息、存储信息、提取信息和分析信息。为了获得能让人们更好理解这些来自不同信息源的轨迹及其相关信息，让人们能够更好的分析感兴趣的事物的信息，我们给出了一种将不同基于轨迹的信息融合的方法。这种方法叫做面向轨迹数据的函数连接算法。我们提取出关系数据库的关系表中的元组，并根据时间和空间的限制条件将属于不同关系表的元组关联起来。本文给出了的连接算法和基于哈希的连接算法。针对这两种算法，我们给出了一个轨迹压缩的优化方法来提高算法执行的效率。这种轨迹压缩的方法存在的误差边界，误差边界公式将在后文给出。理论的分析和实验的结论将会展示出给出的算法的优点。

本章主要讲述了面向轨迹数据的函数连接算法的研究内容与结果。内容与结果的规划如下：

第一部分给出了问题的定义，包括轨迹点集、轨迹段的定义，属性值数据的定义，并给出了例子来详细描述各种定义，以便更好的说明问题。所有关于问题的描述均采用形式化的定义。

第二部分给出了面向轨迹数据的连接算法的处理过程，这些算法包括基础的匹配算法、循环连接算法和基于哈希的连接算法。基础的匹配算法描述了如何检查时间空间关系限制条件和属性值限制条件，另外两个算法则完成了轨迹段与属性值数据的连接操作。所有的算法都给出了算法的描述，算法的步骤和相关的复杂性分析。

第三部分和第四部分给出了基于以上两种连接算法的改进算法，这种改进算法能够提升一定的连接效率。这种改进算法属于优化算法的一种，同时我们给出了误差率和误差边界的分析。

第五部分给出了实验，并对实验结果进行了分析。最后一部分则给出了本章的总结，总结了面向轨迹数据的函数连接算法的研究结果。

3.1 问题的定义

定义 1. 一条轨迹段是一系列具有时间戳的连续的点，这些点有一个独特的标识，表示为$traj(id) = \{(p_i, t_i) | i = 1,2, \dots, m; p_1 = (x_i, y_i), t_i < t_{i+1})\}$，其中$(p_i, t_i)$表示物体在时间戳$t_i$的坐标是$p_i$。

万方数据

在这部分章节当中，我们假定所有通过不同种侦测手段获得到的数据都存储在关系数据库当中。在关系数据库中轨点集的关系模式为$Tr(ID,X,TS)$，其中ID是每个物体的特定标识，X是轨迹集合中点的坐标，TS是每个点的时间戳属性。通过观测手段获得的数据集合的关系模式是$Mon(A,S,TS)$，其中$A=\{a_i|i=1,2,\dots,k\}$表示收集到的信息的属性特征，S代表相对于观测点（例如红外摄像头或者被动雷达的坐标位置）的空间参考信息，TS代表获得到的属性信息的时间戳。这里有需要注意的一点，轨迹点信息的观测时间戳和属性信息的观测时间戳是独立的，即多种设备的时间信息并不同步。

定义 2. 给定轨迹点集$TR=Tr(ID,X,TS)$，数据信息集合$MD=\{Mon_j(A,S,TS)|j=1,2,\dots,k\}$，时间空间限制条件$F(X,S,Mon_j.A,Mon_j.TS)$，和属性值限制条件$C=\{V_j|V_j\subseteq Mon_j.A\}$，面向轨迹数据的函数连接就可以表示为，在限制条件$F$和$C$下，数据信息集合$MD$与轨迹点集$TR$做连接运算。运算表示为，

$$TRJ_{(F,C)}(TR,MD)=\prod\nolimits_{Attr}(\delta_{(F,C)}(TR\times Mon_1\times\dots\times Mon_k))$$

其中$Attr=\{Tr.ID,Tr.TS,Tr,X,Mon_1.TS,Mon_1.A,\dots,Mon_k.TS,Mon_k.A\}$，

(1) 时间空间关系限制条件F：对任意两个连续的坐标点$p_i,p_{i+1}\in Tr.X$，并且相应的时间戳$t_i,\mathrm{t}_{i+1}\in Tr.TS$，存在一系列关联结果$JR=\{r(id,t_i,ma_1,mt_1,\dots,ma_k,mt_k)|r.id\in Tr.ID,r.t_i\in Tr.TS,r.ma_j\in Mon_j.A,r.mt_j\in Mon_j.TS\}\subseteq TRJ_{(F,C)}(TR,MD)$满足：

a) 对$\forall mt_j\neq NULL,t_i\leq mt_j\leq t_{i+1}$；

b) 对$\forall r\in JR$，让$s_j\in Mon_j.S$成为相对于$r.ma_j$在$r.mt_j$的空间参考信息，如果$\mathrm{r}.ma_j\neq NULL$并且$\mathrm{r}.mt_j\neq NULL$，$\mathrm{F}(\{p_i,p_j\},\{t_i,t_j\},\{s_j\},\{mt_j\})$为真。

(2) 属性值限制条件C：对任意有独特标识c的轨迹点集traj(c) ∈ TR，令$\mathrm{T}=[\mathrm{t}_0,\mathrm{t}_1]$表示traj(c)的时间段，则存在一系列结果集$JR=\{r(id,t_i,ma_1,mt_1,\dots,ma_k,mt_k)|r.id\in c,r.t_i\in T,r.ma_j\in Mon_j.A,r.mt_j\in T\}\subseteq TRJ_{(F,C)}(TR,MD)$，满足对$\forall r\in JR$，条件为$\mathrm{r}.ma_j\neq NULL$，$\mathrm{r}.mt_j\neq NULL$并且$\mathrm{r}.ma_j\in V_j$。

万方数据

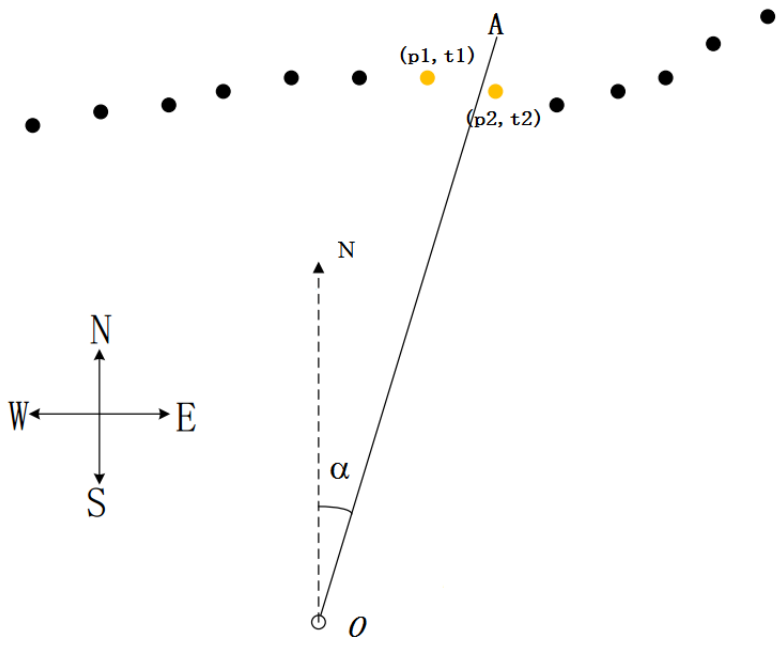


图 3-1 轨迹点集与属性值匹配过程

在面向轨迹数据的函数连接的定义中，我们用一般的形式表示出了时间空间限制条件和属性值限制条件。这些条件主要根据具体的实际应用来决定。例如，在图 3-1 当中，O 点表示摄像头的位置坐标，AO 表示所有捕获的图像相对于 O 点的空间参考信息。通过分析图像的空间参考信息，我们就可以得到动物相对于我们摄像头的方位。ON 表示正北方向，动物图像的角度 OA 是以 ON 为起点，顺时针旋转所得到的角度，可以表示为$\mathrm{Dir}(\mathrm{O},\mathrm{A})=\alpha$。同样，轨迹点也可以计算出每一个点相对于 O 点的角度。时间空间限制条件可以被定义为野生动物的方位恰好在某一对连续的轨迹信息点，同时，当前图像的获取时间也要在这两个轨迹点的时间戳之间。形式化表示为$\mathrm{Dir}(\mathrm{O},\mathrm{p}_1)\le\alpha\le\mathrm{Dir}(\mathrm{O},\mathrm{p}_2)$且$\mathrm{t}_1\le t\le\mathrm{t}_2$，其中$\mathrm{p}_1,\mathrm{p}_2$是两个连续的轨迹点，$\mathrm{t}_1,\mathrm{t}_2$是两个连续的时间戳。数据属性值的限制条件可以使一系列值的集合，也可以一段连续的值，即一段范围。该限制条件可以被设定在没一个属性值上，也可以设定在每一个轨迹点上。例如野生动物皮毛的颜色特征，以及每个图像中野生动物最少的数目。

下面我们将给出连接轨迹关系表和属性值关系表的连接算法。连接多个关系表可以通过线性扩展。当连接时属性数目超过一个时候，如果一些属性值不能被连接，则他们可以被设为空。

万方数据

3.2 面向轨迹数据的函数连接处理过程

这一部分首先描述了来自轨迹关系表和数据值关系表的元组的匹配过程，展示了如何实现时间空间关系限制条件和属性值限制条件。在这个过程的基础上，我们给出了循环的算法和基于哈希的算法。最后我们给出了一种近似计算的方法，并同时给出了该算法的理论分析过程。

3.2.1 匹配属性值元组和轨迹段元组集合

令$(\mathrm{p}_i, \mathrm{t}_i)$和$(\mathrm{p}_{i+1}, \mathrm{t}_{i+1})$作为两个连续的轨迹点和时间戳信息。给定关系表$Mon(A, S, TS)$中某一个元组$\mathrm{r}_1(a, s, t)$，首先检查属性值的限制条件。当$\mathrm{r}_1$满足这些条件时，检查时间空间关系限制条件$F$。$F$的定义需要根据应用的实际情况，可以为线性函数，二次函数或者其他类型的函数。

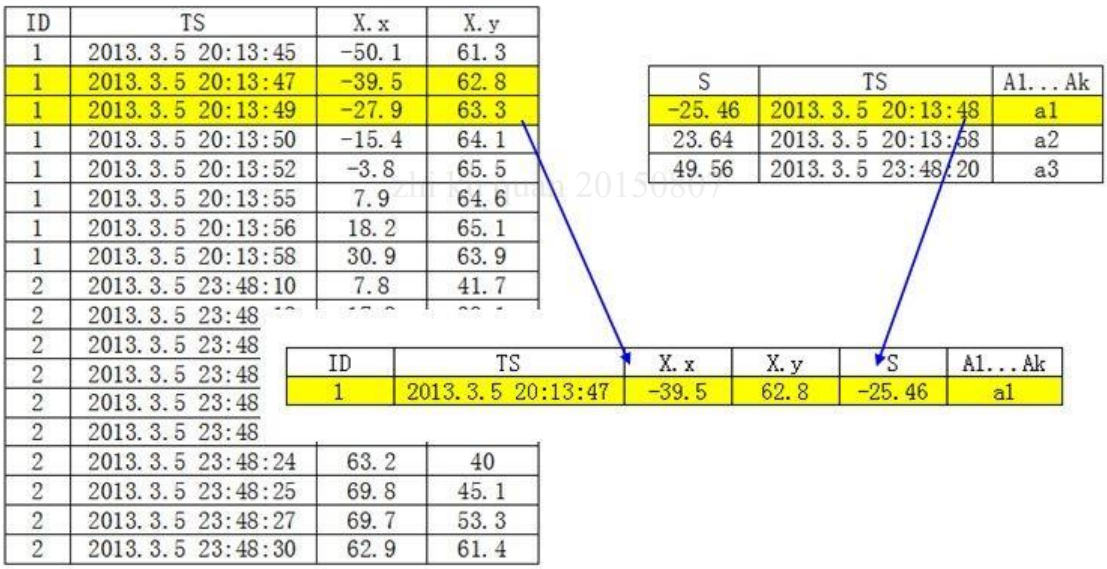


图 3-2 轨迹点集与属性值匹配具体实例

例如，在图 3-2 中，轨迹关系表中存储着所有轨迹点集。属性值关系表中存储着所有属性值数据。在本文中，我们采用线性函数满足时间空间限制条件以简化计算。在本例中，匹配过程计算了两个连续的点 20:13:47 和 20:13:49 相对于原点的角度。由于这一组数据满足了时间空间限制条件和属性值限制条件，算法返回了一个新的连接好的元组。如果一个元组能够和一个轨迹段满足连接条件，那么轨迹段的第一个时间点信息将被用于生成的新的元组。算法 3-1 如下。

万方数据

Algorithm 1 Matching Segment($\mathcal{F}, \mathcal{C}, seg, Mon_1(A,S,TS), \ldots, Mon_k(A,S,TS)$)

Input: spatial-temporal constraint $\mathcal{F}$, set value constraint $\mathcal{C}$, trajectory segment: $seg = \{(id, p_j, t_j), (id, p_{j+1}, t_{j+1})\}$, Data tables $Mon_1(A,S,TS)$, $\ldots$, $Mon_k(A,S,TS)$.
Output: Join result JR

```
initialize k records by next tuple of corresponding data tables Mon_i(A,S,TS) → r_i(a,s,t);
JR = ∅;
flag = true;
while flag do
    flag = false;
    new joined tuple jr = (id, t_j, NULL, NULL, ....., NULL);
    read a block of each data table Mon_k(A,S,TS);
    for i = 1 to k do
        if (t_j < r_i.t < t_{j+1} then
            flag = true;
            if r_i.a satisfies C then
                if r_i satisfies F({p_j, p_{j+1}}, {t_j, t_{j+1}}, r_i.s, r_i.t) then
                    replace NULL in jr at corresponding slots with r_i.a and r_i.t;
                end if
            end if
            read next tuple Mon_i(A,S,TS) → r_i(a,s,t);
        end if
    end for
    if not all r_i.s, r_i.t in jr are NULL then
        JR = JR ∪ {jr};
    end if
end while
return JR;
```

算法 3-1 匹配过程，包含时空关系与属性值的限制条件检查

如果有超过一个属性值关系表可以和当前轨迹点关系表连接，那么在那些关系表的每一条元组都按照顺与轨迹点进行比对。得到结果集的属性值的时间戳并不需要按照时间顺序，只要该属性元组满足了时间空间关系和属性值限制条件即可。例如，令$r_1(a,s,t) \in Mon_1(A,S,TS)$，$r_2(a,s,t) \in Mon_2(A,S,TS)$和$r_3(a,s,t) \in Mon_3(A,S,TS)$分别是三个属性值关系表的一个元组，$(p_1,t_1)$和$(p_2,t_2)$是$traj(1)$的两个连续的轨迹点。当这三个元组与轨迹段连接时，如果r_3不满足时间空间限制条件，那么连接的结果就可以表示为$jr = (1, t_1, p_1, r_1.t, r_1.a, r_2.t, r_2.a, NULL, NULL)$。如果下一个在$Mon_1(A,S,TS)$的元组不满足连接条件，第二个关系表的元组满足关系条件，则连接结果就可以表示为$jr' = (1, t_1', p_1', NULL, NULL, r_2'.t, r_2'.a, NULL, NULL)$。当三个属性值关系表的当前元组都和当前轨迹段做匹配连接之后，我们再读取下一段轨迹段数据，重新使用刚才元组进行匹配连接。在连接的过程中，每一个属性值数据仅仅被读取一次。

万方数据

算法 1 给出了匹配过程的细节。令k表示属性值关系表的总数，m表示最大的属性值关系表的元组的总数。当前匹配的算法的时间复杂度为$O(kmT)$。其中T表示检查时间空间关系限制条件和属性值限制条件的总时间。

3.2.2 循环连接算法

最直接的用于连接属性值数据和轨迹点集数据的连接算法就是循环连接算法。所有的轨迹点都是从轨迹点关系表中获取的。对每一个轨迹点关系表的轨迹段，循环算法不断的调用匹配算法来实现连接操作。算法可以通过排序的方法来进行优化，即通过对属性值数据的元组根据元组的时间戳进行排序来提升性能。由于数据是通过侦测设备来获取的，因此所有的数据都是按照时间顺序存储在数据库中。如果内存足够大，能把所有的元组全部放在里面时，循环连接算法将会非常有效率。否则，即使元组按照时间排序了，也会在算法运行时产生很多的磁盘 I/O。一般来讲，轨迹点关系表元组的总数要少于属性值关系表元组的总数，因此轨迹点可以被放在外层循环当中。算法 3-2 给出了面向轨迹数据的循环函数连接算法。假定在轨迹点关系表中有n个元组，最长的轨迹段包括c个段，那么循环函数连接算法需要调用匹配段过程复杂度为$O(cn)$。

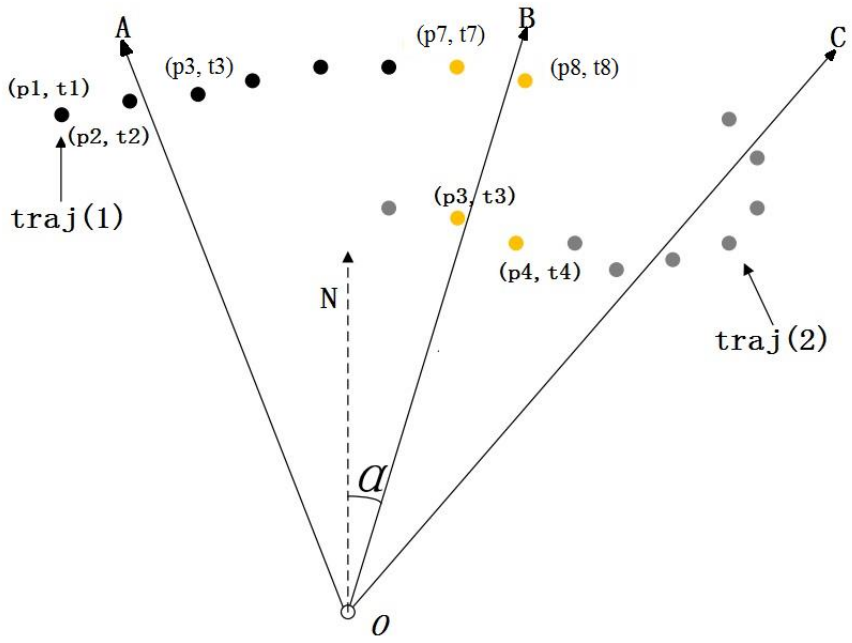


图 3-3 连接算法的具体情形

万方数据

例如，在图 3-3 中，循环算法首先获得了$traj(1)$和$traj(2)$。然后提取出每一个轨迹段，并和三个属性值元组进行匹配，OA，OB，OC 分别表示这三个属性值元组。特别的，OB 满足了时间空间关系限制条件。尽管获得的属性值元组可能匹配多个轨迹段数据，但我们可以通过属性值限制条件来增强判断条件。这种情况也暗示了我们不能通过标记一个元组是否已经匹配过来提高我们的算法的运行效率。在未来的工作中我们将给出方法来消除这种匹配的含糊的情况。

Algorithm 2 Nested Loop TRAIN($\mathcal{F}, \mathcal{C}, seg, Mon_1(A,S,TS), \ldots, Mon_k(A,S,TS)$)

Input: spatial-temporal constraint $\mathcal{F}$, set value constraint $\mathcal{C}$, trajectory table TR, Data tables $Mon_1(A,S,TS)$, $\ldots$, $Mon_k(A,S,TS)$.
Output: Join result JR

```
JR = ∅;
while exists unprocess trajectory data do
    read blocks of trajectory data in buffer as much as possible;
    for every trajectory traj(i) in buffer do
        for for every segment seg = {(id, p_j, t_j), (id, p_{j+1}, t_{j+1})} in traj(i) do
            JR = JR ∪ Matching Segment(F, C, seg, Mon_1(A,S,TS), ..., Mon_k(A,S,TS));
        end for
    end for
end while
return JR;
```

算法 3-2 循环连接算法

接下来分析 I/O 代价。令 B(T)表示轨迹点数据的数据块总数，B(D)表示属性值数据的数据块总数，M 表示内存缓冲区的大小。假设 B(T)<=B(D)，如果(M-k)>B(T)，则所有的轨迹段信息可以被放入内存当中，这时 I/O 代价为$B(T)+k\times B(D)$。如果 M<B(T)，轨迹点数据和属性值数据就不能全部装入内存当中。这时算法会试图尽可能多的装载轨迹点数据和属性值数据。在每一次迭代的过程中，需要载入(M-k)块数据块。总的 I/O 代价为$B(T)+B(T)\times k\times B(D)/(\mathrm{M}-\mathrm{k})$。

3.2.3 基于哈希的连接算法

在仅有几个观测点的情况下（例如一个或者两个基站来侦测飞行器），我们可以使用一些基于划分的方法来提高连接结果的效率。由于观测点比较少，那么在使用循环算法时会产生大量的中间数据，造成效率低下。通过采取极坐标框架的方法，我们能够改善一定的效率。极坐标的中心可以使用观测点的坐标，此时，所有轨迹点关系表的元组和属性值关系表的元组可以根据他们相对于空间参考点的方位计算哈希值，之后根据哈希值将各个元组划分到哈希桶中。

万方数据

在这里我们用角度来简化方位，角度范围为[0,360)。如果在使用角度来对数据进行划分的过程中，划分出来的桶非常不均匀，此时可采用角度和属性值限制来构造二维哈希函数，并通过计算哈希值，重新根据哈希值划分至各个桶中。在划分到桶中之后，在每个桶中利用循环的算法来实现轨迹点数据和属性值数据连接。算法 3-3 给出了基于哈希的连接算法的细节。

Algorithm 3 Hash based TRAIN($\mathcal{F}, \mathcal{C}, seg, Mon_1(A,S,TS), \ldots, Mon_k(A,S,TS)$)

Input: spatial-temporal constraint $\mathcal{F}$, set value constraint $\mathcal{C}$, trajectory table TR, Data tables $Mon_1(A,S,TS)$, $\ldots$, $Mon_k(A,S,TS)$.
Output: Join result JR

```
1: JR = ∅;
2: Partition trajectory segment and tuples in data tables into M − 1 buckets;
3: for each bucket of partitioned data tables H(Mon_i, n) and trajectory segments H(TR, n) do
4:     for for every segment seg = {(id, p_j, t_j), (id, p_{j+1}, t_{j+1})} in H(TR, n) do
5:         while exists unprocess trajectory data in H(TR, n) do
6:             read blocks of H(TR, n) in buffer as much as possible;
7:             JR = JR ∪ Matching Segment(F, C, seg, H(Mon_1, n), ..., H(Mon_k, n));
8:         end while
9:     end for
10: end for
11: return JR;
```

算法 3-3 基于哈希的连接算法

接下来分析 I/O 代价。基于哈希的连接算法通过划分数据的方法提高了算法的运行效率。这种方法在运行算法时减少了不必要的时间空间关系限制条件的检查，同时，这种方法在有限的内存下减少了磁盘读写的 I/O 操作。假设数据能够平均了被划分至每个桶中，所有划分的桶能够在一次循环算法中完成连接操作，那么磁盘的 I/O 代价可以表示为$3 \times (B(T) + k \times B(D))$，限制条件为$B(T) < (M-k)^2$。

3.3 面向轨迹数据的函数连接算法的改进

在这一部分中我们将给出一种优化的方法，即采用轨迹点集的近似压缩的方法来提高面向轨迹数据的函数连接算法的效率。之后我们给出了这种压缩的算法的误差边界。最后提出了一种在给定误差率条件下的连接算法。

在循环连接算法和基于哈希的连接算法中，对于每一个轨迹段，匹配算法过程调用了属性值关系表元组来进行连接。每一个属性值元组都需要与轨迹段中每一组连续的两个点进行匹配操作。如果时间空间关系满足条件，且属性值

万方数据

限制条件也满足，则可以生成新的结果元组。当时空关系函数设计的比较复杂的时候，在匹配过程中势必增加了程序的复杂性，程序的效率也会大大降低。尤其是当轨迹点集关系表元组数据和属性值关系表中元组数目特别大的时候，效率的降低将是灾难性的。因此有必要对这种匹配方法做出改进。

改进匹配方法的主要做法就是减少在一条轨迹段内精确匹配的次数，即减少时间空间函数的运行次数。给定一个物体 o，一个侦测设备 D，如果 o 围绕着 D 做匀速圆周运动时，我们可以根据 o 相对于 D 的方位信息的起点和终点来估计任意时刻 o 点所在的方位信息。

更一般的情况下，考虑轨迹点集中 k 个连续的点$\{(p_1,t_1),(p_2,t_2),\dots,(p_k,t_k)\}$，其中 k>2，令$(p_1,t_1)$和$(p_k,t_k)$来代表压缩后的轨迹段，用两个点来代替 k 个点，压缩比为 k/2。假设 0 是观测设备，$\mathrm{Dir(O,p_1)}$和$\mathrm{Dir(O,p_k)}$表示起始点和终点相对于远点的方位信息，令$\Delta\theta$表示平均角速度，则有

$$\Delta\theta=\begin{cases}\frac{|\mathrm{Dir(O,p_k)-Dir(O,p_1)}|}{k},\ if\ |\mathrm{Dir(O,p_k)-Dir(O,p_1)}|\le 180\\ \frac{(360-|\mathrm{Dir(O,p_k)-Dir(O,p_1)}|)}{k},\ if\ |\mathrm{Dir(O,p_k)-Dir(O,p_1)}|>180\end{cases}\tag{3-1}$$

令$\Delta t=(t_k-t_1)/k$作为每一步的平均时间，总的时间为$[t_1,t_k]$。给定元组r，相应的观测角度α和时间戳t。根据假设元组r可以与轨迹段连接，当且仅当α满足轨迹段第i个夹角，其中$i=|t-t_1|/\Delta t$，$\alpha\in[(Dir(O,p_1)+i\times\Delta\theta),\big((Dir(O,p_1)+(i+1)\times\Delta\theta)\big)]$。

例如，在图表 3 中，$traj(1)$的轨迹的点的角速度接近均匀。我们使用$Dir(O,p_1)$和$Dir(O,p_8)$作为轨迹段在时间t来对匹配算法进行估算。时间段$[t_1,t_8]$的每一步的平均时间为$\Delta t=(t_8-t_1)/k,\Delta\theta=(360-|\mathrm{Dir(O,p_k)-Dir(O,p_1)}|)/k$。这时，我们不需要检查时间限制条件，由于$Dir(O,\mathrm{A})\in[(Dir(O,p_1)+i\times\Delta\theta),\big((Dir(O,p_1)+(i+1)\times\Delta\theta)\big)]$，我们就可以认为当前轨迹段与属性值元组满足连接条件。如果不使用压缩的方法，按照原来的匹配算法，8 个轨迹点分成 7 段，共 3 个属性值元组，因此需要检查$3\times7=21$次。而在压缩之后，只有 2 个点和 3 个元组，因此只需要 3 次检查就能够判断是否满足条件。

上述压缩算法的正确性是基于物体角速度均匀的假设条件的。然而，物体实际的运动往往不完全满足于这个假设条件。接下来我们将讨论这个压缩方法的错误边界。

万方数据

给定一个轨迹点集，包含 k 个点$\{(p_1,t_1),(p_2,t_2),\dots,(p_k,t_k)\}$，令$(p_1,t_1)$和$(p_k,t_k)$来代表压缩后的轨迹段。假设$a_1,a_2,\dots,a_k$是 k 个均匀分布的角度。而$b_1,b_2,\dots,b_k$是真实的角度，则可以计算误差率如下，

$$\varepsilon=\frac{\sum_{i=2}^{k-1}|a_i-b_i|}{|a_1-a_k|} \tag{3-2}$$

其中$a_0=b_0,a_k=b_k$。

3.4 误差边界

误差率在上文已经给出。在这里我们讨论给出一个算法来满足算法误差率小于具体应用给定的误差率。事实上，误差边界依赖于具体的应用背景。在这里我们假定误差率是e，我们给出了一种计算 K 值的方法，满足得到的误差率小于给定的e。

首先，我们定义一个初始的点，令 K=2 然后我们不断的增加 K 值，同时根据上文提到的误差率公式计算此时的误差率，当误差率最大且不超过e的时候，停止 K 值的增加，这时得到的 K 值就可以作为匹配运算的值。进而在每次匹配的时候都用这个 K 值来完成。

然而，我们可以利用这个方法自动的让算法选择最大的 K 值，以减小多余的检查算法。显然，当轨迹点趋于角速度均匀时，K 值会选择的比较大。

3.5 面向轨迹数据的函数连接算法实验分析

我们给出了大量的实验来评估面向轨迹数据的函数连接算法。我们评估了四个版本的连接算法。我们给出了循环算法与优化版的循环算法的对比实验，哈希算法与优化版的哈希算法的对比实验。优化版的算法使用的是自适应的 K 值。同时，我们也给出了在相同数据规模下 K 值的选取对算法运行效率的影响。之后我们给出了图表来显示数据规模与效率的关系。实验环境：PC 机 3.10GHz CPU，4G 内存，Ubuntu 12.04 系统。

3.5.1 连接算法的效率

我们用了八组数据集来完成了本次实验。在这里，数据集合的大小是两个关系数据库表的笛卡尔积大小。我们假设数据采集时对同一物体的采样频率接近一致，因此这两个关系表的元组数目大致相等。所以我们仅仅用其中一个关

万方数据

系表的元组数目作为我们的输入规模即可。下面的表格给出了八组数据在四种算法下的运行效率，单位为秒。四种算法采用的是自适应的 K 值。

表 3-1 各种算法在不同数据规模下的运行时间

	Nested Loop	NL with opt	Hash-Based	HB with opt
1000	3.124	2.873	1.873	1.719
2000	13.394	12.329	2.795	2.412
3000	24.521	21.387	3.623	3.319
4000	50.573	30.487	4.447	4.084
5000	78.374	50.823	5.423	4.912
6000	117.722	71.381	6.389	5.528
7000	173.056	95.110	7.361	6.210
8000	241.382	128.182	8.344	6.617

万方数据

3.5.2 效率的比较

表 3-1 描述了每一个算法在不同的数据规模下的运行时间，接下来我们利用这些时间给出图表来进一步显示时间和对比效率。图表显示了不同算法的效率的曲线。

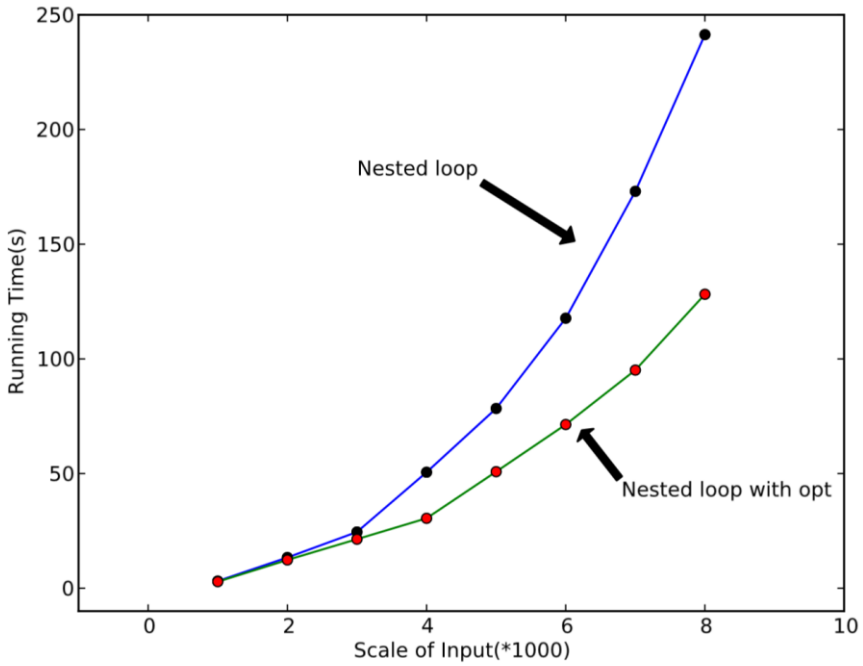


图 3-4 循环算法及优化

万方数据

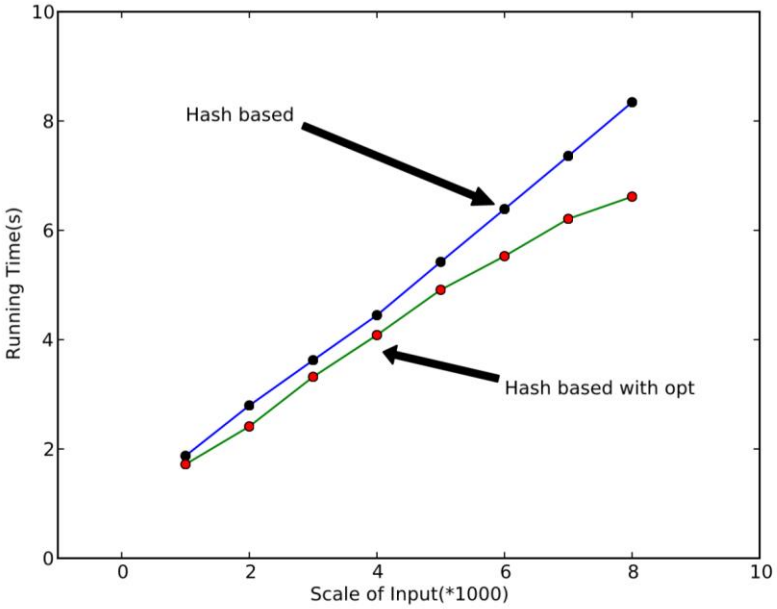


图 3-5 哈希算法及优化

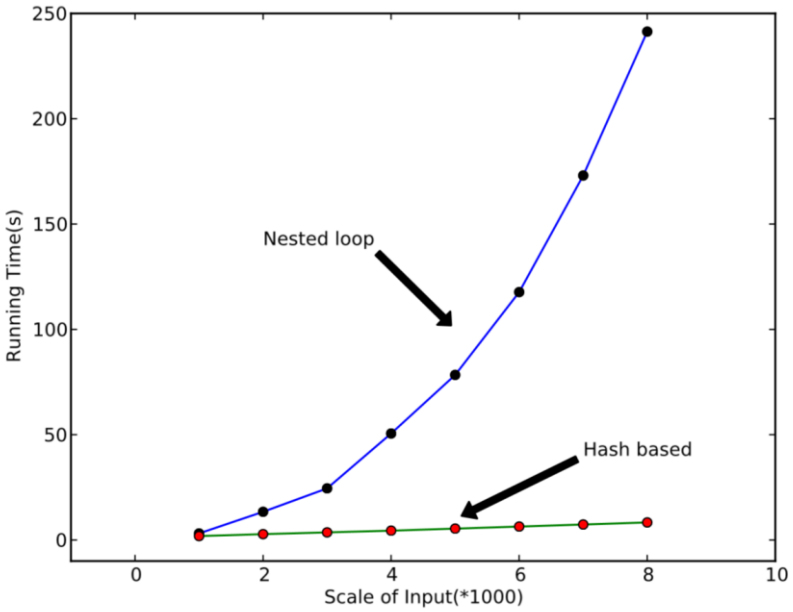


图 3-6 循环与哈希算法对比

万方数据

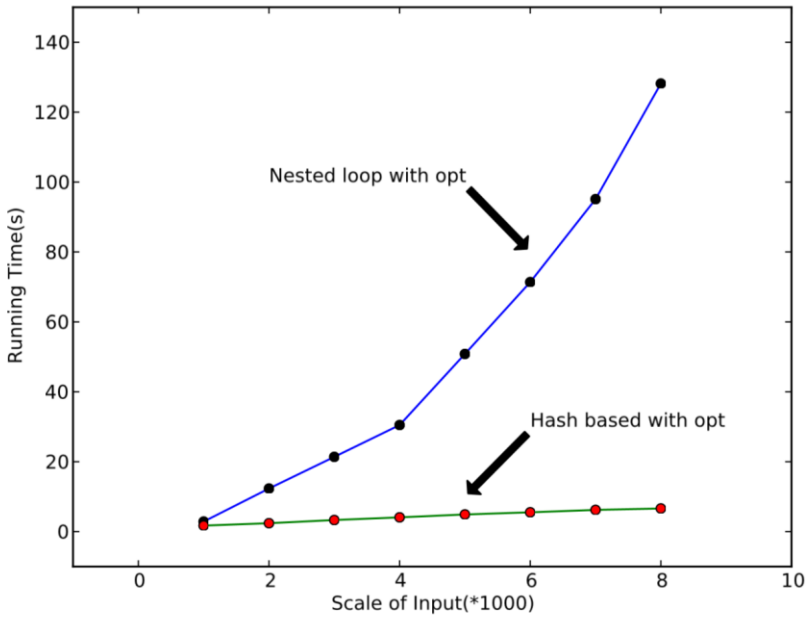


图 3-7 两种优化的算法的对比

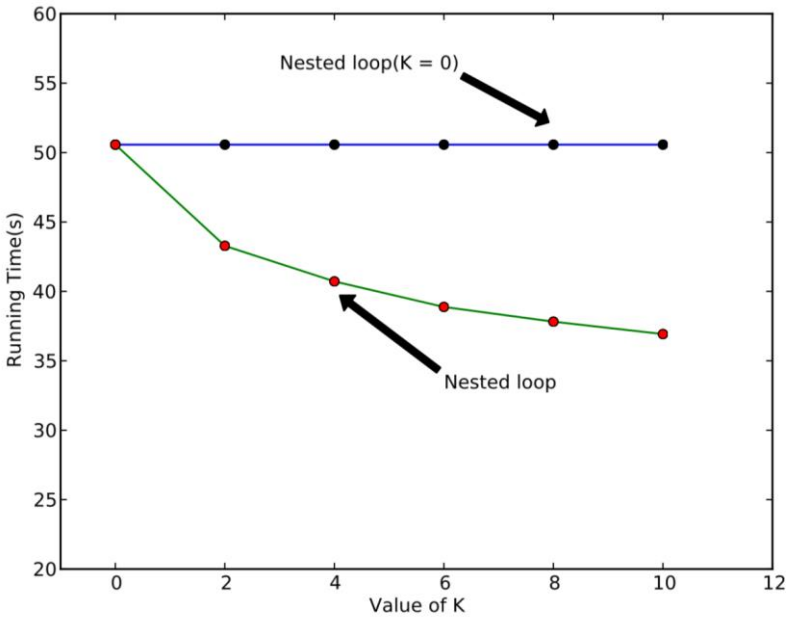


图 3-8 K 值对循环算法的影响

万方数据

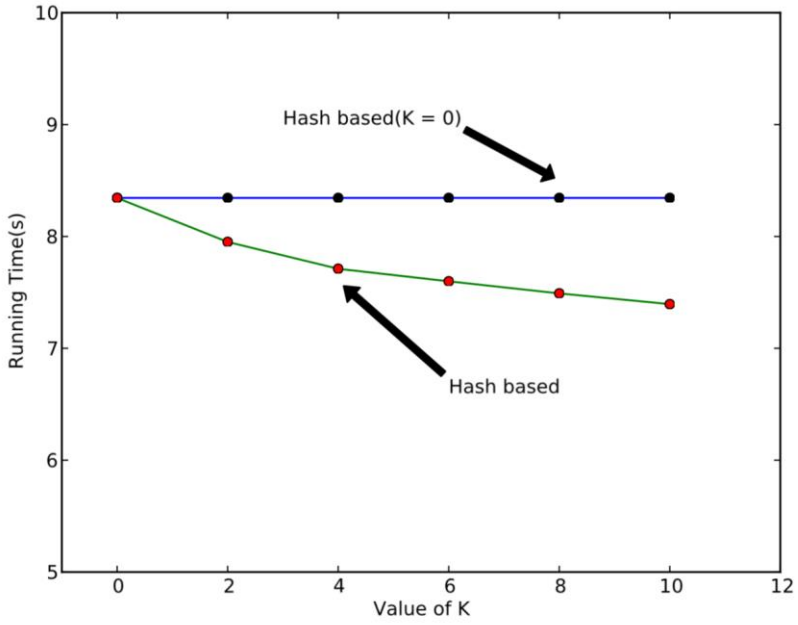


图 3-9 K 值对哈希算法的影响

从图表中我们可以看到每个算法的运行时间随着数据规模的扩大而增加。例如左侧图表和右侧图表对比，循环算法的运行时间增加的最快，但它有一个非常低的起点，即数据规模小的时候运行的很快。而基于哈希的算法运行时间增加的比循环的算法要慢，但它有一个非常高的起点，即当数据规模比较小的时候也运行了非常长的时间。当两种算法都用轨迹点集压缩算法优化后，算法的运行时间都会相应的减少。这表明了轨迹点集压缩算法能够起到一定的优化作用。

最后我们给出了 K 值的选取如何影响不同的算法的运行时间。我们给出了固定的 K 的最大值的限制，即每种算法在使用自适应的 K 值时，K 值不能超过的最大值。如图所示，当 K 值最大限值增加的时候，算法的运行时间减少。而当 K 值最大限值不断增加时候，算法的效率提高的并不明显，这主要是由于轨迹段的长度是有限的，在当前条件下，我们不能无限制的压缩轨迹段来达到优化算法的目的。

3.6 本章小结

本文阐述了面向轨迹数据的连接算法的研究结果。为了获得能让人们更好理解这些来自不同信息源的轨迹及其相关信息，让人们能够更好的分析感兴趣

万方数据

的事物的信息，我们给出了一种将不同基于轨迹的信息融合的方法。这种方法叫做面向轨迹数据的函数连接算法。我们提取出关系数据库的关系表中的元组，并根据时间和空间的限制条件将属于不同关系表的元组关联起来。本文给出了的连接算法和基于哈希的连接算法。针对这两种算法，我们给出了一个轨迹段压缩的优化方法来提高算法执行的效率。这种轨迹压缩的方法存在的误差边界，我们给出了误差的计算方法。最后我们的理论的分析和实验的结论展示出了算法的优点。

万方数据

第4章 基于目标轨迹相似性查询算法研究

到目前为止，我们给出了目标轨迹数据与属性值数据连接的算法，但当我们有多种观测轨迹的手段来收集轨迹信息时，针对同一种物体的轨迹信息就出现了冗余，如果能够合并这些冗余信息并提取出一个最全面的轨迹数据结果，那么这将对数据分析产生重要的影响。

在本章中，我们给出了一些高级的轨迹相似性查询算法，尤其是在目标轨迹的相似性查询和密度查询方面。目标轨迹的相似性查询的目的是为了寻找目标轨迹的共同的位移模式，并将具有极高相似性的轨迹加以合并操作，完成更好的对轨迹数据的预处理。而密度查询则是为了找出感兴趣的目标的轨迹当中密度最大的一段或若干段区域。很多应用场景对应这相似查询和密度查询。我们在本章会讨论相似性轨迹的查询算法与密度查询算法。

本章组织结构如下：

第一部分给出了目标轨迹相似性查询的综述，介绍了目标轨迹相似性查询的应用背景，查询算法的介绍以及相似性查询的一些研究成果和这些成果所存在的问题。引出本章讨论的内容。

第二部分给出了问题的相关定义，包括目标轨迹数据的定义、面向轨迹数据的相似性查询的定义。其中对相似性查询中欧式距离的定义我们给出了连续和离散的情形以应对不同的应用需求。

第三部分和第四部分给出了面向轨迹数据的相似性查询算法。给出了算法的描述，算法的步骤、算法的复杂性分析。这种相似性的算法能够应用在大多数应用场景的数据库中。

第五部分给出了面向轨迹数据的相似性查询算法的一个改进策略。我们使用了局部敏感哈希的算法来实验算法的改进，我们给出了算法的召回率，并在最后给出了对比实验。通过实验我们可以了解这种优化策略对算法的效率有着很大的提升

最后一部分是本章小结。

4.1 目标轨迹相似性查询

为了能够合并相同目标物体的轨迹信息，我们需要给出轨迹的相似性判断方法，在这里我们可以换一种思路，即查询相似的轨迹路线。通过这种方法能

万方数据

够得到相同目标的轨迹信息，并采取一定的方法将其合并。本章我们重新定义轨迹，轨迹的相似度以及目标轨迹相似性的查询。

移动的目标轨迹能够被定义成二维（X-Y）平面或者三维（X-Y-Z）空间的时间序列数据。在基于相似的查询中，我们更关心目标轨迹的形状。对两条轨迹而言，采样的向量对度量轨迹之间相似度是很重要的，但时间属性相对与空间属性而言并没有那么重要，因此在判断形状相似的时候可以忽略时间因素。这一点就和传统的基于时间空间数据库的相似性查询区分开来[42]。考虑到一维时间序列数据相似性判断的研究已经非常广泛，例如股票或商品价格预测，商品的销售额，天气数据和生物学应用等等。然而，距离函数和一些索引的方法并没有直接应用在一维时间序列数据上，但这主要是由于时间序列数据的独特性。

目标轨迹通常都是二维的或者三维的数据序列，并且一条轨迹通常有不同的长度。然而大多数研究主要关注了一维时间序列数据的相似性问题[20,25,37,48,46]。

有些目标轨迹有一些离群点。不像股票，天气或者其他商品价格数据等等，移动的目标的轨迹通常在一段时间内被记录下来。这种记录可能是用 GPS、相机、视频等方式。因此，由于一些设备上的缺陷，人们收集到的数据可能含有一定数量的离群点。一般人们用最长公共子序列这种方法来解决这种问题[46]。然而这种方法并没有考虑相似的子序列间断的不同情况，所以会导致一定的不准确率。这种间断可能使得判断子轨迹的相似性带来失败。

在不同区域的轨迹可能出现相同的移动的模式。对于目标轨迹的不同的采样频率可能造成对比目标轨迹时的错位。尽管有一些相似性度量的方法，例如DTW[27,36,48]或者编辑距离[26]能够用来度量轨迹之间的相似度，并且起到消除错位的作用，但这些方法对于目标轨迹的错位过于敏感。

为了在数据库中管理这些目标轨迹数据，我们定义了轨迹的数据模型，把目标轨迹数据当做空间中的近似的直线，然后不同的目标轨迹之间的相似性就可以用离散的线的欧氏距离来近似。我们给出的这种相似性查询方法能够被用来去寻找嵌入在目标轨迹中的隐藏的模式信息，例如图 4-1，在城市中道路上的汽车可能会含有可能的交通拥堵等状况。

万方数据

图 4-1 北京交通地图

4.2 问题的定义

轨迹点集可以被看做空间当中的线。在空间中，定义两条线的相似性是比较困难的。然而我们在研究时间序列的数据库时发现了一些很有用的线索[24,37,43]。时间序列数据库系统能够存储时间序列的数据，例如温度数据，经济预测数据，人口数据，无线电波信号等，其中还包括一些支持时间序列数据的查询操作。大多数支持时间序列的数据库在分析数据序列时，采取了欧氏距离来评定两组数据序列的差距[37]。由于轨迹数据就属于一种时间序列的数据，那么支持时间序列的数据库就能够很有效率的支持轨迹数据的处理。然而轨迹数据不仅包含时间属性，同样还包含空间的一些属性。例如，利用时间序列数据库就比较难完成地理方面的一些查询。

4.2.1 轨迹的定义

为了定义轨迹数据之间的相似性，我们首先要定义轨迹数据。下面我们定义了目标轨迹的数据模型。

万方数据

一个真实的轨迹是一个点集序列，即一条线，包含一个起点和一个终点。给定一个二维空间R^2和一个闭区间时间间隔$I_\lambda = [t, t']$，其中$t < t'$，一条轨迹λ的定义如下：

定义 1. 一条轨迹就是一个连续的映射中的映像：$\lambda : I_\lambda \to R^2$。

这是一个基于时间的对一条简单的线的定义[22]。下面我们给出空间R^2中目标轨迹的长度L_S的定义以及目标轨迹的时间区间L_T的定义。

定义 2. 给定时间区间$[t_0, t_1]$，目标轨迹λ的长度$L_S(\lambda, [t_0, t_1])$可以用如下方法计算：

$$L_S(\lambda, [t_0, t_1]) = \int_{t_0}^{t_1} \sqrt{(dx/dt)^2 + (dy/dt)^2}dt, \text{where } \lambda(t) = (x, y) \tag{4-1}$$

整条目标轨迹的长度可以表示为$L_S(\lambda, [t, t'])$。

定义 3. 给定空间R^2中的向量$\vec{x} = (x, y)$，那么位于轨迹λ中$\vec{x}_i$和$\vec{x}_j$之间的时间区间定义如下：

$$L_T\left(\lambda, \left[\vec{x}_i, \vec{x}_j\right]\right) = \left|t_j - t_i\right|, \text{where } \lambda(t_i) = \vec{x}_i, \lambda\left(t_j\right) = t_j \tag{4-2}$$

$$L_T(\lambda) = |t' - t| \tag{4-3}$$

以上的定义针对的是连续的点组成的轨迹，即一条曲线。然而，类似与 GPS 这样的定位设备不能连续的测量移动物体的坐标位置，这类设备仅仅能够获得整条轨迹的一部分样本。图 4-2 中描述了实际当中目标轨迹的表示和在计算机中目标轨迹的表示，就此我们可以看到在计算机中，离散的轨迹表示往往是比较主要的形式。因此，我们定义离散的轨迹$\dot{\lambda}$，每一个向量$\vec{x}_i$代表轨迹上在时间$T_{\dot{\lambda}} = \{t_0, t_1, \dots, t_m\}$的一个向量点。

定义 4. 一条离散的轨迹是一系列离散的映射中的映像：$\dot{\lambda} : T_{\dot{\lambda}} \to R^2$。

一个离散的轨迹同样也可以用一系列向量序列$\dot{\lambda} < \vec{x}_1, \dots, \vec{x}_m >$来表示。在此我们假定当$\dot{\lambda}(t_i) = \vec{x}_i$时，我们引入了若干符号注释：$T_{\dot{\lambda}}(i) = t_i$，$X_{\dot{\lambda}}(i) = x_i$，$|\dot{\lambda}|$表示向量的膜长，接下来我们给出两个向量$\vec{x}, \vec{x}'$之间在空间$R^2$中的距离的定义。

万方数据

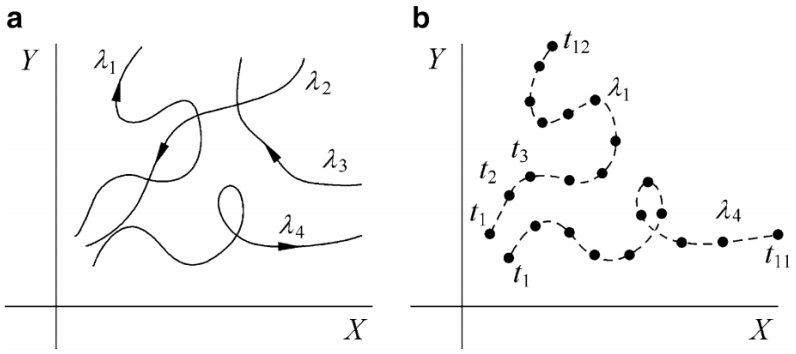


图 4-2 移动目标的轨迹(a)表示真实的轨迹(b)表示实际存储在数据库中的轨迹

定义 5. 两个向量$\vec{x},\vec{x}'$之间在空间R^2中的距离定义为：

$$D(\vec{x},\vec{x}') = \sqrt{(x-x')^2+(y-y')^2} \tag{4-4}$$

至此我们给出了轨迹的相关定义，其中包括连续的轨迹的定义，轨迹空间段、时间段的定义，轨迹长度的定义。再给出连续的轨迹的定义之后又给出了离散状态下的轨迹的定义，以及轨迹的时间、空间、长度的定义。这些定义将会是以后定义轨迹相似性的基础。接下来我们将给出轨迹的相似性的定义。

4.2.2 轨迹的相似性

在时间序列数据库中，两个时间序列数据集的相似程度通常是用欧氏距离来界定的[24,37]。这种利用欧氏距离的方法效率比较高。然而在空间中比较两条线的相似程度很少讨论，主要是因为之前的方法集中在讨论点和线之间的距离[28,29,38]。前面的的方法的目标是为了寻找在给定点附近经过另一个点的物体，例如一辆汽车穿过一条街。另一方面我们也关注轨迹形状。为了计算在不同轨迹之间基于形状的相似度，定义轨迹之间的相似度是很有必要的。

一般来讲，相似度查询普遍采用 kNN 查询[28,38]。有两种已经存在的查询的方法，一种是基于空间相似性的查询，另一种是基于时间序列的相似性查询。由于目标轨迹有时间和空间的特征属性，因此我们考虑以下三种目标轨迹的相似性查询：

- ✧ 时空相似性：基于时间和空间的特征属性，即$R^2 \times T$。
- ✧ 时间相似性：仅基于时间特征属性，无空间属性，即$R^1 \times T$。

万方数据

✧ 空间相似性：仅基于空间特征属性，无时间属性，即R^2。

正如上文提到的，目标轨迹含有时间序列的特征属性。我们在两条目标轨迹之间定义相似度与定义时间序列数据相似度查询是一致的[37]。对于基于时间序列的数据库，两组基于时间序列的数据的相似度是利用欧式距离来定义的。假设两组数据均含有 n 个值，那么相似度可以用两个向量在R^n的欧式距离来表示。在[24,37]中，给定两组时间序列数据$c =< w_1, \dots, w_n >$，$c' =< w'_1, \dots, w'_n >$，那么距离$D(c, c')$定义如下：

$$D(c, c') = \sqrt{(w_1 - w_n')^2 + (w_1 - w_n')^2} \tag{4-5}$$

这个定义可以被扩展成空间R^2上的向量的距离。当给定的两组时间序列数据是$\vec{X} =< \vec{x}_1, \dots, \vec{x}_n >$，$\vec{X}' =< \vec{x}'_1, \dots, \vec{x}'_n >$，我们就可以扩展原来的距离定义$D(c, c')$，那么新的**轨迹的相似性定义**可以定义为向量的欧式距离$D(\vec{X}, \vec{X}')$，可以表示为：

$$D(\vec{X}, \vec{X}') = \sqrt{D(\vec{x}_1 - \vec{x}_1')^2 + D(\vec{x}_n - \vec{x}_n')^2} \tag{4-6}$$

在这里，$D(\vec{X}, \vec{X}')$是有适用范围的，当每一个向量$\vec{X}$都在相同的时间区间侦测数据的时候，即侦测数据时$\Delta t = t_{i+1} - t_i (i = 1, \dots n-1)$全部相同的时候，$D(\vec{X}, \vec{X}')$才可以被使用，其中$\vec{x}_i$采集时对应的时间是$t_i$。然而，由于侦测设备容易丢失数据的原因，每个侦测到的向量的间隔时间Δt不可能完全相同。因此，为了利用我们的定义计算目标轨迹的相似度，我们定义了时间间隔规格化的离散的目标轨迹数据，定义如下：

定义 6. 给定一条轨迹λ，侦测时间区间$[t_S, t_E]$和一个自然数m，那么时间规范化离散轨迹$\dot{\lambda}_{\Delta t}$定义如下：

$$\dot{\lambda}_{\Delta t} =< \lambda(t_S), \lambda(t_S + \Delta t), \dots, \lambda(t_s + m\Delta t) >, \text{where } t_S + m\Delta t = t_E \tag{4-7}$$

事实上，离散轨迹$\dot{\lambda}_{\Delta t}$是对原来的轨迹λ的重新采样，即采样时保证相等的时间区间或者说时间间隔。换句话说，$\dot{\lambda}_{\Delta t}$是通过将λ按照相等的时间间隔等分生成的。对于离散轨迹$\dot{\lambda}_{\Delta t}$我们可以使用线性近似的方法来表示每一段轨迹，即利用近似的$\tilde{\lambda}$来代替λ。

定义 7. 给定两条轨迹λ和λ'，他们有着相同的时间长度（即$L_T(\lambda) = L_T(\lambda')$），给定自然数$m$，那么在$\lambda$和$\lambda'$之间的时空距离$D_{TS}(\lambda, \lambda')$定义如下：

万方数据

$$\mathrm{D_{TS}}(\lambda,\lambda')=\frac{1}{m+1}\sqrt{\sum_{i=0}^{m}D(\vec{X}_{\dot{\lambda}_{\Delta t}}(i),\vec{X}_{\dot{\lambda}'_{\Delta t}}(i))^2},where\ \Delta t=\frac{\mathrm{L}_T(\lambda)}{m}=\frac{L_T(\lambda')}{m} \quad (4\text{-}8)$$

这里需要注意的是$\mathrm{D_{TS}}(\dot{\lambda},\dot{\lambda}')$可以被定义为$\mathrm{D_{TS}}(\tilde{\lambda},\tilde{\lambda}')$。在这个定义当中，在两条目标轨迹之间的相似度是利用$m+1$维向量的欧式距离来表示的，其中每条轨迹的时间间隔都被规范化。利用这种定义，找到形状相似的目标轨迹就成为了可能，因此我们能够得到查询相似的目标轨迹的算法。

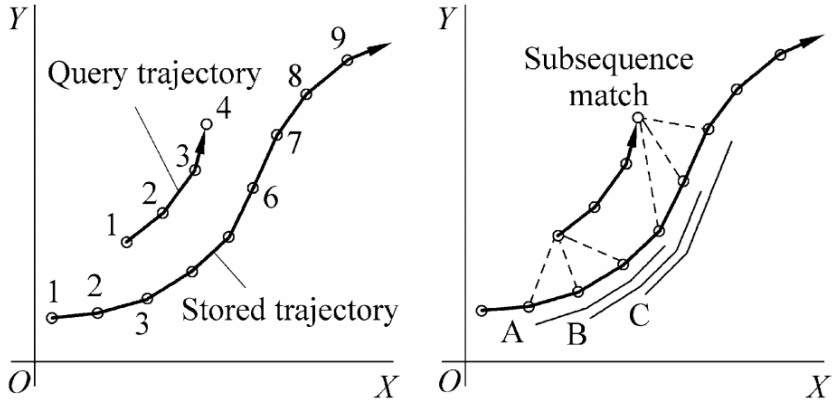


图 4-3 目标轨迹的相似性查询

4.3 面向轨迹数据的查询算法

基于以上章节的定义，我们考虑一种对目标轨迹的基于形状的相似度查询算法。这里，$\dot{\Lambda}$是存储在数据库中的离散轨迹点的集合，每一条轨迹$\dot{\lambda}_i(\dot{\lambda}_i\in\dot{\Lambda})$作为一条离散轨迹表示为$\dot{\lambda}_i=<\mathrm{x}_1,\mathrm{x}_2,\dots,\mathrm{x}_m>$。查询的轨迹$\dot{\lambda}_q$表示为$\dot{\lambda}_q=<\mathrm{x}_1,\mathrm{x}_2,\dots,\mathrm{x}_n>$。那么根据全面对两组时间序列数据的向量距离的定义，我们能够给出算法的描述，如图 4-3。同样我们能够利用$\dot{\Lambda}$和$\dot{\lambda}_q$来给出基于形状的查询的算法$Q_{range}(\theta:integer,\dot{\Lambda},\dot{\lambda}_q)$，算法 4-1 如下：

万方数据

Algorithm 13: $Q_{range}(\theta : integer, \dot{\Lambda}, \dot{\lambda}_q) : \dot{\Lambda}_a$

Input: $\dot{\Lambda}$,$\dot{\lambda}_q$,θ(θ is a natural number)
Output: $\dot{\Lambda}_a$,$\{\dot{\lambda}_{a1}, \ldots, \dot{\lambda}_{ak}\} \in \dot{\Lambda}_a$
begin
  $l = |\dot{\lambda}_q|$; $\dot{\Lambda}_a = \phi$;
  foreach $\dot{\lambda}_i$ *in* $\dot{\Lambda}$ **do**
    for $j = 1$ *to* $|\dot{\lambda}_i| - l + 1$ **do**
      $\dot{\lambda}_{ij} = subsequence(\dot{\lambda}_i, j, l)$;
      //This function will return a subsequence of the original sequence $\dot{\lambda}_i$, such as $< \mathbf{x}_j, \mathbf{x}_{j+1}, \cdots, \mathbf{x}_{j+l-1} >$, each $\mathbf{x} \in \dot{\lambda}_i$;
      if $D(\dot{\lambda}_q, \dot{\lambda}_{ij}) < \theta$ **then**
        Add $\dot{\lambda}_{ij}$ to $\dot{\Lambda}_a$;
      end
    end
  end
  return $\dot{\Lambda}_a$
end

算法 4-1 面向轨迹数据的相似性查询算法

算法给出了查询相似目标轨迹的处理过程。通过不断的枚举目标轨迹的范围段并计算与所查询的目标轨迹之间的向量距离，我们就能够得到与当前目标轨迹最相似的轨迹段。目标轨迹之间的最近邻查询也可以利用距离函数来定义。在我们的定义中，时间特征属性并没有在查询中显示的说明出来，我们认为时间属性与范围查询独立。

接下来我们给出算法的复杂性分析。面向轨迹数据的查询算法是利用给定的一条轨迹，通过遍历数据库中每一条轨迹，进而利用上文所提到的距离定义方法来计算两条轨迹的距离，最后判断距离是否满足既定的阈值，如果满足，则说明找到了相似的轨迹，并将其输出，如果没有找到，则返回空值。算法需要扫描整个数据库一次，因此所需要的磁盘 I/O 操作仅仅只有扫描数据库的操作。

令 B(T)表示轨迹数据的数据块总数，假设任意两条目标轨迹均能够存放到内存，那么运行一次算法所需要的 I/O 操作次数为$1 \times$ B(T)。算法在内存操作的时间复杂度主要取决于轨迹的长度，设目标轨迹的最大长度为 L，查询轨迹的长度L′，由于算法需要枚举所有的轨迹段来判断轨迹段是否与所给定的轨迹相似，因此在枚举两条轨迹的时候，算法的复杂度为O(L)，而计算距离来判断相

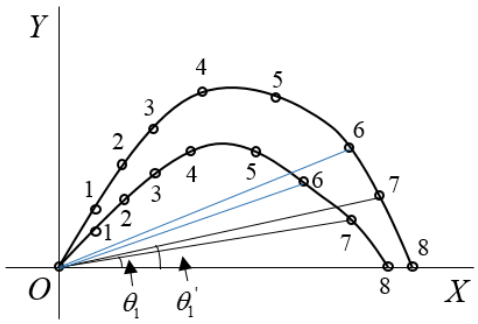
万方数据

似度的算法复杂度为$O(L')$，因此总的判断两条目标轨迹的相似度的复杂度为$O(LL')$。

4.4 优化策略

上文给出的算法是在原有的目标轨迹上直接计算欧几里得距离，进而得到查询的轨迹和数据库中的轨迹之间的相似度。这种算法存在一定的效率问题，由于计算欧式距离所需要的浮点计算量比较大，因此当数据量特别大的时候，算法的效率变成为了不可忽视的问题。本节利用局部敏感哈希的策略给出了上述算法的一种优化的算法，有效的解决了数据量特别大的时候算法的运行效率问题。

我们将两条目标轨迹分别做一些初始化的处理，首先将目标轨迹的起始点平移到坐标系的原点，继而通过将目标轨迹旋转，来将目标轨迹的终点移动到坐标横轴上。在这里，旋转可以通过坐标变换来实现，并且不改变目标轨迹的形状。最后，我们通过计算目标轨迹上的点来得到每个点相对于坐标原点的角度。例如图表 4-4，这样我们就得到了转化后的目标轨迹的角度集合，即目标轨迹对应的角度向量$\theta =< \theta_1, \theta_2, \ldots, \theta_m >$。



图表 4-4 转化后的目标轨迹

对于目标轨迹角度向量，我们可以定义相似度距离为角度向量的相关系数，但利用相关系数计算相似性和利用原来的欧式距离算法的复杂度近乎一致，因此我们在这里采用局部敏感哈希（LSH）的策略来实现相似度的度量。

局部敏感哈希（LSH）是利用随机矩阵将原有数据哈希，并简化计算的一种方法。我们采用的是$m \times 64$维的 01 矩阵来将 m 维向量哈希成 64 位的 01 数字串，通过对查询串和数据库中的轨迹串预处理成 64 位数字串，我们就可以通过比对两对 64 位数字串的汉明距离来给出这两对数字串的相似度，但会造成一定

万方数据

的误差率。这就是局部敏感哈希（LSH）的原理。因此，在利用局部敏感哈希（LSH）的策略来实现相似性查询时，我们需要首先预处理数据库中的目标轨迹串，得到的 64 位数字串，继而通过哈希的方法来将这些数字串哈希至若干桶中，之后在查询相似的轨迹串时，我们就可以根据查询串的 01 数字串来快速定位桶，从而加速查询的处理。

在我们处理数据的过程中，我们使用的是 64 位 01 串计算汉明距离来比较相似度。我们假定汉明距离小于等于 4 的时候得到近似的轨迹串，即当汉明距离大于 4 的时候，目标轨迹与查询轨迹不相似。在这里，我们将 64 位 01 串分成连续的 4 部分，每一部分 16 位，根据 LSH 的召回率概率分析，我们可以得到这种情况下处理的概率值为：

$$1-(1-s^r)^b, \tag{4-9}$$

其中$b=4$，$r=16$，$s=1-\frac{d}{64}$，d = 4。d 表示的是相似情况下最大的汉明距离，s 表示文档中最小哈希签名矩阵中某个具体行中两个哈希值相等的概率。根据计算当前实验的召回率约为 82.8%。

4.5 查询算法实验分析

我们给出了大量的实验来评估面向轨迹数据的函数连接算法。我们评估了欧式空间距离算法和利用 LSH 优化的算法。我们给出了两种算法的对比实验。之后我们给出了图表来展示这种效率上的优化。实验环境：PC 机 3.10GHz CPU，4G 内存，Ubuntu 12.04 系统。

我们用了六组数据集来完成了本次实验。这六组数据分别采用了两种不同的随机哈希矩阵。在这里，数据集合的大小是数据库中关系表的元组数目。下面我们给出了两种矩阵下的效率的对比。

表格 4-1 利用欧式距离的算法与 LSH 优化算法的对比

数据规模(万)	60	80	100	160
欧氏距离(s)	11.011	14.829	19.872	31.784
LSH 优化(s)	0.561	0.752	1.026	1.529

万方数据

表格中数据规模的单位是万，即数据库元组数目的规模是万。基于欧式距离的算法和 LSH 优化的算法的单位是秒，即算法的运行时间用秒来表示。通过对比两种算法的效率，我们可以确定 LSH 能够对原来的算法产生明显的优化效果。从表格中我们可以看到每个算法的运行时间随着数据规模的扩大而增加。利用 LSH 方法优化后的算法的效率约为原来的基于欧式距离的算法的 20 倍。在召回率为 82.8%的情况下效率提升 20 倍，这种改进可以应用在一些对精度要求不是很高的环境背景下。

4.6 本章小结

本章给出了面向轨迹数据的相似性查询算法，我们首先列举了一些目标轨迹相似性查询算法，尤其是在目标轨迹的相似性查询和密度查询方面。目标轨迹的相似性查询的目的是为了寻找目标轨迹的共同的位移模式，并将具有很高的相似性的轨迹加以合并操作，从而完成更好的对轨迹数据的预处理。而密度查询则是为了找出感兴趣的目标的轨迹当中密度最大的一段或若干段区域。很多应用场景对应这种相似查询和密度查询。

之后我们给出了问题的定义，将问题形式化表示出来。我们给出了轨迹的相关定义和相似性距离的定义，本文主要阐述的是欧氏距离。通过对轨迹的距离来判断目标轨迹之间的相似性。再次基础上我们给出了解决问题的算法，即面向轨迹数据的相似性查询算法。我们同样给出了算法的复杂性分析，同时给出了算法的一些特点。再次基础上我们有给出了优化的算法，即利用局部敏感哈希的方法实现轨迹查询，同时我们给出了召回率的计算。最后我们给出了实验结果，对比了两种算法的效率。

万方数据

结　论

当今时代，信息化特征明显，人们观察物理世界所获得的数据均用计算机信息表示。为了更好的观察和分析人们生活的物理世界，人们采用了多种多样的信息数据收集方法。而如何能够将这些数据融合，如何检索到相似的数据已成为研究的热点。本文取得的研究成果如下：

(1)本文阐述了面向轨迹数据的连接算法的研究结果。为了获得能让人们更好理解这些来自不同信息源的轨迹及其相关信息，让人们能够更好的分析感兴趣的事物的信息，我们给出了一种将不同基于轨迹的信息融合的方法。这种方法叫做面向轨迹数据的函数连接算法。我们提取出关系数据库的关系表中的元组，并根据时间和空间的限制条件将属于不同关系表的元组关联起来。本文给出了的连接算法和基于哈希的连接算法。针对这两种算法，我们给出了一个轨迹段压缩的优化方法来提高算法执行的效率。这种轨迹压缩的方法存在的误差边界，我们给出了误差的计算方法。最后我们的理论的分析和实验的结论展示出了算法的优点。

(2)本文给出了面向轨迹数据的相似性查询算法，我们首先列举了一些目标轨迹相似性查询算法，尤其是在目标轨迹的相似性查询和密度查询方面。目标轨迹的相似性查询的目的是为了寻找目标轨迹的共同的位移模式，并将具有很高的相似性的轨迹加以合并操作，从而完成更好的对轨迹数据的预处理。而密度查询则是为了找出感兴趣的目标的轨迹当中密度最大的一段或若干段区域。很多应用场景对应这种相似查询和密度查询。其次，我们给出了问题的定义，将问题形式化表示出来。我们给出了轨迹的相关定义和相似性距离的定义，本文主要阐述的是欧氏距离。通过对轨迹的距离来判断目标轨迹之间的相似性。再次基础上我们给出了解决问题的算法，即面向轨迹数据的相似性查询算法。我们同样给出了算法的复杂性分析，同时给出了算法的一些特点。最后我们给出了查询算法的优化策略，即利用局部敏感哈希的策略实现算法的效率提升。我们还给出了局部敏感哈希算法的召回率，并在最后给出了系统的实验结果。

总之，这篇论文提出了新颖的面向轨迹数据的函数连接算法，并给出了不同的算法的实现。之后提出了目标轨迹相似性查询的算法，通过这种算法能够实现查询相似的目标轨迹。未来我们将更加集中精力，努力的致力于函数连接算法和目标轨迹的相似性及密度查询算法的研究。

万方数据

参考文献

[1] Camera traps. http://worldwildlife.org/initiatives/camera-traps. Accessed: 2014-01-05.

[2] Gps wildlife tracking. http://en.wikipedia.org/wiki/GPS wildlife tracking. Accessed: 2014-01-05.

[3] Radar. http://en.wikipedia.org/wiki/Radar. Accessed: 2014-01-05.

[4] Radar warning receiver. http://en.wikipedia.org/wiki/Radar warning receiver. Accessed: 2014-01-05.

[5] Wildlife monitoring. http://www.nps.gov/pore/naturescience/wildlife monitoring.htm. Accessed: 2014-01-05.

[6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. Computer Networks, 38:393–422, 2002.

[7] L. Becker, K. Hinrichs, and U. Finke. A new algorithm for computing joins with grid files. In Data Engineering, 1993. Proceedings. Ninth International Conference on, pages 190–197. IEEE, 1993.

[8] J. L. Bentley. Multidimensional binary search trees used for associative searching. Com- munications of the ACM, 18(9):509–517, 1975.

[9] R. Bischof, H. Ali, M. Kabir, S. Hameed, and M. Nawaz. Being the underdog: an elusive small carnivore uses space with prey and time without enemies. Journal of Zoology, 2014.

[10] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf. Computational geometry. Springer, 2000.

[11] G. Fabrizio, F. Colone, P. Lombardo, and A. Farina. Adaptive beamforming for high- frequency over-the-horizon passive radar. IET radar, sonar & navigation, 3(4):384–405, 2009.

[12] R. H. Gu ¨ting and W. Schilling. A practical divide-and-conquer algorithm for the rectangle intersection problem. Information Sciences, 42(2):95–112, 1987.

[13] E. G. Hoel, H. Samet, et al. Benchmarking spatial join operations with spatial output. In VLDB, pages 606–618, 1995.

[14] M. Kitsuregawa, L. Harada, and M. Takagi. Join strategies on kd-tree indexed relations. In Data Engineering, 1989. Proceedings. Fifth International Conference

万方数据

on, pages 85–93. IEEE, 1989.

[15]J. Marcus Rowcliffe, C. Carbone, P. A. Jansen, R. Kays, and B. Kranstauber. Quantifying the sensitivity of camera traps: an adapted distance sampling approach. Methods in Ecology and Evolution, 2(5):464–476, 2011.

[16]C. Mead and L. Conway. Introduction to VLSI systems, volume 1080. Addison-Wesley Reading, MA, 1980.

[17]D. Rotem. Spatial join indices. In Data Engineering, 1991. Proceedings. Seventh Inter- national Conference on, pages 500–509. IEEE, 1991.

[18]J. M. Rowcliffe, J. Field, S. T. Turvey, and C. Carbone. Estimating animal density using camera traps without the need for individual recognition. Journal of Applied Ecology, 45(4):1228–1236, 2008.

[19]P. Valduriez. Join indices. ACM Transactions on Database Systems (TODS), 12(2):218– 246, 1987

[20]Agrawal R, Faloutsos C, Swami AN (1993) Efficient similarity search in sequence databases. In: Proceedings of the 4th international conference on foundations of data organization and algorithms (FODO 1993), Chicago, pp 69–84

[21]Agrawal R, Lin KI, Sawhney HS, Shim K (1995) Fast similarity search in the presence of noise,scaling,andtranslationintime-seriesdatabases. In:Proceedings ofthe21stinternational conference on very large data bases (VLDB 1995), Zurich, pp 490–501

[22]Andoni A, Deza M, Gupta A, Indyk P, Raskhodnikova S (2003) Lower bounds for embedding edit distance into normed spaces. In: Proceedings of the 14th annual ACM-SIAM symposium on discrete algorithms (SODA 2003), Baltimore, pp 523–526

[23]Cai Y, Ng R (2004) Indexing spatio-temporal trajectories with chebyshev polynomials. In: Proceedings of the 2004 ACM SIGMOD international conference on management of data (SIGMOD 2004), Paris, pp 599–610

[24]Chakrabarti K, Keogh E, Mehrotra S, Pazzani M (2002) Locally adaptive dimensionality reduction for indexing large time series databases. ACM Trans Database Syst 27(2):151–162

[25]Chan KP, Fu AW (1999) Efficient time series matching by wavelets. In: Proceedings of the 15th international conference on data engineering (ICDE

万方数据

1999), Sydney, p 126

[26]Chen L, Ng R (2004) On the marriage of edit distance and Lp norms. In: Proceedings of the 30th international conference on very large data bases (VLDB 2004), Toronto, pp 792–803

[27]Chen S, Kashyap RL (2001) A spatio-temporal semantic model for multimedia presentations and multimedia database systems. IEEE Trans Knowl Data Eng 13(4):607–622

[28]Chon H, Agrawal D, Abbadi AE (2002) Query processing for moving objects with space- time grid storage model. In: Proceedings of the 3rd international conference on mobile data management (MDM 2002), Singapore, pp 121–129

[29]Cormode G, Muthukrishnan S (2002) The string edit distance matching problem with moves. In: Proceedings of the 13th annual ACM-SIAM symposium on discrete algorithms (SODA 2002), San Francisco, pp 667–676

[30]Douias D, Peucker T (1973) Algorithm for the reduction of the number of points required to represent a line or its character. Am Cartogr 10(42):112–123

[31]EsterM,Kriegel HP,Sander J, XuX(1996) Adensity-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2th international conference on knowledge discovery and data mining (SIGKDD 1996), Portland, pp 226–231

[32]Faloutsos C, Ranganathan M, Manolopoulos Y (1994) Fast subsequence matching in time- series databases. In: Proceedings of the 1994 ACM SIGMOD international conference on management of data (SIGMOD 1994), Minneapolis, pp 419–429

[33]Hadjieleftheriou M, Kollios G, Gunopulos D, Tsotras VJ (2003) On-line discovery of dense areas in spatio-temporal databases. In: Proceedings of the 8th international symposium on advances in spatial and temporal databases (SSTD 2003), Santorini Island, pp 306–324

[34]Jagadish HV, Mendelzon AO, Milo T (1995) Similarity-based queries. In: Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems (PODS 1995), San Jose, pp 36–45

[35]Jensen CS, Lin D, Ooi BC, Zhang R (2006) Effective density queries on continuously moving objects.In:Proceedings ofthe22ndinternationalconference ondataengineering(ICDE2006), Atlanta, p 71

万方数据

[36]Keogh E (2002) Exact indexing of dynamic time warping. In: Proceedings of the 28th international conference on very large data bases (VLDB 2002), Hong Kong, pp 406–417

[37]Keogh E, Chakrabarti K, Pazzani M, Mehrotra S (2001) Dimensionality reduction for fast similarity search in large time series databases. J Knowl Inf Syst 3(3):263–286

[38]Kollios G, Tsotras VJ, Gunopulos D, Delis A, Hadjieleftheriou M (2001) Indexing animated objects using spatiotemporal access methods. IEEE Trans Knowl Data Eng 13(5):758–777

[39]Korn F, Jagadish H, Faloutsos C (1997) Efficiently supporting Ad hoc queries inlarge datasets of time sequences. In: Proceedings of the 1997 ACM SIGMOD international conference on management of data (SIGMOD 1997), Tucson, pp 289–300

[40]Martina N and By RA (2004) Spatiotemporal compression techniques for moving point objects. In: Proceedings of the 9th international conference on extending database technology (EDBT 2004), pp 765–782

[41]Papadias D, Zhang J, Mamoulis N, Tao Y (2003) Query processing in spatial network databases. In:Proceedings ofthe29thinternationalconference onverylargedatabases(VLDB 2003), Berlin, pp 802–813

[42]Pfoser D, Jensen CS, Theodoridis Y (2000) Novel approaches in query processing for moving objecttrajectories.In:Proceedings ofthe26thinternationalconference onverylargedatabases (VLDB 2000), Cairo, pp 395–406

[43]Priyantha N, Miu A, Balakrishnan H, Teller S (2001) The cricket compass for context-aware mobile applications. In: Proceedings of the 7th annual international conference on mobile computing and networking (MOBICOM 2001), Rome, pp 1–14

[44]Rafiei D (1999) On similarity-based queries for time series data. In: Proceedings of the 15th international conference on data engineering (ICDE 1999), Sydney, pp 410–417

[45]Saltenis S, Jensen CS, Leutenegger ST, and Lopez MA (2000) Indexing the positions of continuously moving objects. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data (SIGMOD 2000),

万方数据

Dallas, pp 331–342

[46]Vlachos V, Kollios G, Gunopulos D (2002) Discovering similar multidimensional trajectories. In: Proceedings of the 18th international conference on data engineering (ICDE 2002), San Jose, p 673

[47]Yi B, Faloutsos C (2000) Fast time sequence indexing for arbitrary Lp norms. In: Proceedings of the26thinternational conference on very large databases (VLDB 2000), Cairo, pp 385–394

[48]Yi B, Jagadish H, Faloutsos C (1998) Efficient retrieval of similar time sequences under time warping. In: Proceedings of the 14th international conference on data engineering (ICDE 1998), Orlando, pp 201–208

[49]Lu-An Tang, UIUC; Xiao Yu, University of Illinois at Urbana-Champaign; Quanquan Gu, CS, UIUC; Jiawei Han, UIUC; Alice Leung, BBN; Thomas La Porta, PSU: Mining Lines in the Sand: On Trajectory Discovery From Untrustworthy Data in Cyber-Physical System (KDD 2013)

万方数据

攻读硕士学位期间发表的论文

1. 张冠男，张炜. 面向轨迹数据的函数连接算法研究. 智能计算机与应用，2014年.（已收录，待发表）
2. Guannan Zhang, Wei Zhang. On Processing Trajectory Based Function Join. The Journal of Digital Information Management (JDIM), 2014.6.（论文在投）

万方数据

哈尔滨工业大学学位论文原创性声明和使用权限

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《面向轨迹数据的函数连接及相似性查询算法研究》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果，且学位论文中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本学位论文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签字：  日期： 2014 年 6 月 30 日

学位论文使用权限

学位论文是研究生在哈尔滨工业大学攻读学位期间完成的成果，知识产权归属哈尔滨工业大学。学位论文的使用权限如下：

（1)学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文，并向国家图书馆报送学位论文；（2）学校可以将学位论文部分或全部内容编入有关数据库进行检索和提供相应阅览服务；（3）研究生毕业后发表与此学位论文研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。

本人知悉学位论文的使用权限，并将遵守有关规定。

作者签名：  日期： 2014 年 6 月 30 日

导师签名：  日期： 2014 年 6 月 30 日

万方数据

致　谢

在这篇论文结束的时候，谨向在我硕士学习期间关心和帮助我的各位老师、同学、朋友们致以诚挚的谢意。

感谢我的导师张炜老师，张老师在我两年的研究生阶段中不论是生活还是学术上都给了我悉心的指导和帮助。在课题选择上，张老师分析了我现有研究工作，帮我最终确定课题研究方向。在撰写和发表论文期间，张老师不断督促我的研究进度，并帮助我修改检查论文内容以及论文格式。张老师认真的科研态度和对学生的亲和态度给我留下深刻的印象。

感谢海量数据计算研究中心的李建中教授、高宏教授、姜守旭教授、石胜飞老师、骆吉洲老师、张炜老师和邹兆年老师。他们无论是从学习上还是生活上都给予我热切的关怀，他们一丝不苟的科研精神和诲人不倦的教学态度都给我留下了深刻的印象，令我从学术上和生活上都受益匪浅。

感谢海量数据计算研究中心的兄弟姐妹，两年里，我们互相鼓励、互相支持、互相学习，没有他们的热心帮助和支持，我将很难顺利完成我的毕业设计，正是和他们融洽的相处使我始终保持一种乐观、轻松的态度面对困难和挫折。研究生阶段的这两年将是我永生难忘的记忆和宝贵财富。

感谢多年来一直关心和照顾我的家人，他们的理解与支持给予我前进的动力和克服困难的勇气，他们对我的期望与勉励使我奋发图强，他们的关心与支持使我安身立命。

最后，感谢在论文撰写期间所有关心和帮助过我的人！

万方数据