ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.9, September 2009, pp.2426−2435
doi: 10.3724/SP.J.1001.2009.03580

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于 R-Tree 的高效异常轨迹检测算法*

刘良旭[1,3], 乔少杰[2,4+], 刘 宾[5], 乐嘉锦[3], 唐常杰[4]

[1](宁波工程学院 电子与信息工程学院,浙江 宁波 315016)

[2](西南交通大学 信息科学与技术学院,四川 成都 610031)

[3](东华大学 计算机科学与技术学院,上海 200051)

[4](四川大学 计算机学院,四川 成都 610065)

[5](Department of Computer Science, School of Computing, National University of Singapore, 117590, Singapore)

Efficient Trajectory Outlier Detection Algorithm Based on R-Tree

LIU Liang-Xu[1,3], QIAO Shao-Jie[2,4+], LIU Bin[5], LE Jia-Jin[3], TANG Chang-Jie[4]

[1](School of Electronics and Information Engineering, Ningbo University of Technology, Ningbo 315016,China)

[2](School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China)

[3](School of Computer Science and Technology, Donghua University, Shanghai 200051, China)

[4](School of Computer Science, Sichuan University, Chengdu 610065, China)

[5](Department of Computer Science, School of Computing, National University of Singapore, Singapore)

+ Corresponding author: E-mail: qiaoshaojie@gmail.com

Liu LX, Qiao SJ, Liu B, Le JJ, Tang CJ. Efficient trajectory outlier detection algorithm based on R-tree. *Journal of Software*, 2009,20(9):2426−2435. http://www.jos.org.cn/1000-9825/3580.htm

Abstract: Recent progress on location aware services, GPS and wireless technologies has made it possible to real-timely track moving object and collect a large quarlity of trajectories data. As a result, how to effectively discover the knowledge from these trajectory data becomes an attractive and interesting research topic. The new trajectory outlier detection, proposed in this paper, can be used to determine whether two trajectories are globally matched by calculating the local matching degree between every base comparing unit pairs. Firstly, this paper proposes a new distance measure approach, which treats k consecutive points as a local comparing unit to depict the local features in terms of trajectories, via calculating the matching degree between trajectory segments. In addition, the critical concepts as local match, global match and trajectory outlier are presented. Secondly, based on this distance measure method, a new trajectory outlier detection algorithm based on R-tree is proposed to improve the efficiency of outlier detection. The main idea behind this algorithm is to eliminate unnecessary distance computation by R-tree and distance characteristic matrix between every trajectory pair. Extensive experiments demonstrate the efficiency and effectiveness of the proposed algorithm for trajectory outlier detection.

Key words: trajectory outlier detection; R-tree; minimum Hausdorff distance under translation; global match;

* Supported by the National Natural Science Foundation of China under Grant Nos.60773169, 60473071 (国家自然科学基金); the 11th Five-Years Key Programs for Sci. &Tech. Development of China under Grant No.2006BAI05A01 ("十一五"国家科技支撑计划); the Youth Software Innovation Project of Sichuan Province of China under Grant No.2007AA0032 (四川省青年软件创新工程)

Received 2008-08-13; Accepted 2009-01-15

local match

摘 要: 提出了异常轨迹检测算法,通过检测轨迹的局部异常程度来判断两条轨迹是否全局匹配,进而检测异常轨迹.算法要点如下:(1) 为了有效地表示轨迹的局部特征,以 k 个连续轨迹点作为基本比较单元,提出一种计算两个基本比较单元间不匹配程度的距离函数,并在此基础上定义了局部匹配、全局匹配和异常轨迹的概念;(2) 针对异常轨迹检测算法普遍存在计算代价高的不足,提出了一种基于 R-Tree 的异常轨迹检测算法,其优势在于利用 R-Tree 和轨迹间的距离特征矩阵找出所有可能匹配的基本比较单元对,然后再通过计算距离确定其是否局部匹配,从而消除大量不必要的距离计算.实验结果表明,该算法不仅具有很好的效率,而且检测出来的异常轨迹也具有实际意义.

关键词: 异常轨迹检测;R 树;基于平移的最小 Hausdorff 距离;全局匹配;局部匹配

中图法分类号: TP311 **文献标识码**: A

随着 GPS 定位、传感器网络、无线通信等技术的日臻成熟,越来越多的轨迹数据被收集和存储在数据库.因此,如何从轨迹数据中检测异常的轨迹已开始引起一些研究人员的兴趣.然而,轨迹通常是用若干相互关联的静态点表示,这使得传统异常点检测算法[1−5]中的对象距离度量方法无法直接用来检测轨迹之间的距离.传统异常点检测算法中对象之间的不匹配程度(距离)通常由一些相互独立的属性(用来描述对象特性)之间的差值加权和表示.Knorr 等人[1]通过将轨迹表示为几个相互独立的全局属性,如轨迹所处位置(由轨迹的起点和终点组成)、轨迹运动方向(由所有轨迹点切线方向的最大、最小和平均速度值组成)、轨迹运动速度(由轨迹中最大、最小和平均速度组成)、轨迹长度(轨迹点数目),然后再用基于距离的异常点检测算法检测数据集中的异常轨迹.这种方法的不足在于:由于忽略了轨迹之间的局部差异,它只适用于那些路径较短或较简单的轨迹.为了比较复杂轨迹,Lee[6]等人参照计算机图像和模式识别领域的线段 Hausdorff 距离[7]的思想,先将每条轨迹划分成表示轨迹的局部特征的 t-partition(即轨迹片段两个端点连成的线段)集合,然后按照每个 t-partition 与其他轨迹的 t-partition 相比较确定其是否异常,最后根据异常 t-partition 在整个轨迹中所占的比例来确定轨迹是否异常.该方法很好地解决了复杂轨迹之间难以比较的问题,但是它也存在以下不足:

(1) 轨迹之间的距离仅依赖于轨迹形状.轨迹尽管也用静态点集合表示,但是它与图像数据有本质的不同.首先,除了形状之外,轨迹中蕴含的对象运动规律(对象运动速度和方向)也是轨迹的一个重要特征,例如,强飓风形成的轨迹会因为其运动速度快而被认为异常轨迹.其次,轨迹数据还具有一些图像数据所没有的特征.例 1 描述了这些性质.

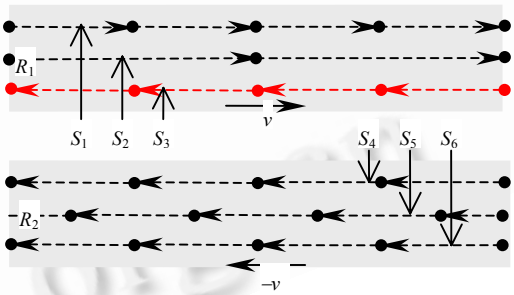


Fig.1 Features of the trajectory

图 1 轨迹的特征

例 1:如图 1 所示,R_1 与 R_2 是两个具有不同运动规律的路径,在路径 R_1,大多数移动对象都按照速度 v 从左到右运动,而在路径 R_2,大多数移动对象按照速度 v 从右到左运动.其中,S_1, S_2, S_3 分别表示 3 条经过路径 R_1 的轨迹,S_4, S_5, S_6 为 3 条经过路径 R_2 的轨迹.显然,轨迹 S_2 是一条异常轨迹,因为它所蕴含的对象移动速度明显大于其他轨迹;S_3 也是一条异常轨迹,因为它表示对象的移动方向与周围轨迹相反.而文献[6]提出的 TROAD 算法显然无法检测出这些异常轨迹.此外,S_4,S_5,S_6 都表示经过路径 R_2 的轨迹,但是由于采样时间不同(如 S_4 和 S_5),或是允许的误差范围,构成 3 条轨迹的点集之间存在一定的公共偏差.因此,一种好的复杂轨迹不匹配程度的度量方法除了能从形状和轨迹自身蕴含的运动规律角度来计算轨迹之间的不匹配程度之外,还能尽可能地缩小相似轨迹(如 S_4,S_5 和 S_6)之间的公共偏差.

(2) t-partition 是由不相互重叠的轨迹片段的两个端点的连线,因此,t-partition 集合所表示的局部特征并没有包含所有可能的局部特征.例 2 显示了这种方法存在的问题.

例 2:如图 2 所示,两条相似轨迹 $S_1=\{p_{11},\ldots,p_{19}\}$(白色空心点表示)和 $S_2=\{p_{21},\ldots,p_{27}\}$(黑色实心点表示).假设

构成 t-partition 的轨迹片段长度为 3,那么点划线就是 S_1 的 t-partition,而点线就是 S_2 的 t-partition.显然,这两个 t-partition 集合具有很大的差异性.

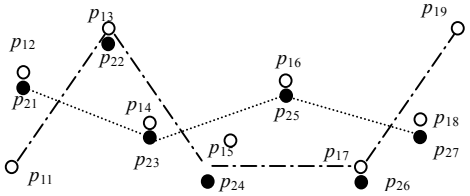


Fig.2 The problem of t-partition
图 2 t-partition 的问题

针对轨迹数据的上述特征,本文先将从轨迹中提取所有长度为 k 的轨迹片段(称为基本比较单元,长度 k 由应用领域确定),然后根据两条轨迹间基本比较单元之间的匹配情况来确定两条轨迹是否匹配.两个基本比较单元之间的距离度量方法是参照基于平移的最小 Hausdorff 距离的思想,并结合轨迹数据的特征,不仅比较两个基本比较单元的形状及其蕴含的运动规律,而且还能消除一定范围内两个基本比较单元的公共偏差.此外,利用算法中基本比较单元仅和其所在邻域内的基本比较单元相比较,本文提出了一种基于 R-Tree 的有效地异常轨迹检测算法,该算法能利用 R-Tree 和两条轨迹组成的点距离矩阵快速找出所有可能局部匹配的基本比较单元对,从而极大地提高算法效率.最后,本文通过详细实验分析了各个参数对算法的影响.实验结果表明,该算法不仅具有很高的效率,而且其挖掘出的异常轨迹也具有实际意义.

1 异常轨迹的定义

文献[6]采用的异常轨迹检测算法尽管可以计算复杂轨迹之间的不匹配程度,但是它存在的不足(如上节所述)也是非常明显的.为了克服上述问题,本文提出了一种新的复杂轨迹之间不匹配程度的度量算法,该算法首先抽取轨迹中所有长度为 k 的轨迹片段构成轨迹的局部特征单元,即基本比较单元,并在此基础上定义一种度量基本比较单元之间不匹配程度的度量方法,该方法不仅比较轨迹的形状以及其蕴含的运动规律,而且还能消除基本比较单元之间的公共偏差.最后,本文给出了轨迹局部匹配、全局匹配轨迹和异常轨迹的定义.本文假设轨迹中每两个轨迹点之间的时间间隔是相同的.

1.1 Hausdorff距离的相关概念

Hausdorff 距离和基于平移的最小 Hausdorff 距离是计算机图像和模式识别领域常用于计算点集之间不匹配程度的经典算法.Hausdorff 距离[8]在图像识别中用于计算两个静止点集之间的不匹配程度(距离).假设两个点集 $A=\{a_1,\ldots,a_n\}$ 和 $B=\{b_1,\ldots,b_m\}$,$dist(a,b)$ 为集合 A,B 中点之间的距离函数(如欧几里德距离),那么,A 和 B 之间的 Hausdorff 距离可以定义为

$$H(A,B)=\max(h(A,B),h(B,A)) \tag{1}$$

其中, $h(A,B)=\max_{a\in A}(\min_{b\in B}(dist(a,b)))$ 称为 A 和 B 之间的直接 Hausdorff 距离,即集合 A 中每个点到最近的集合 B 中的点的最大距离.而基于平移的最小 Hausdorff 距离[9]是用来计算两个图像进行平移之后的最小 Hausdorff 距离(即动态点集之间的不匹配程度),它消除了两个图像之间的公共偏差.基于平移的最小 Hausdorff 距离表示为

$$M_T(A,B)=\min_t H(A,B\oplus t) \tag{2}$$

其中,H 表示 Hausdorff 距离,⊕是一个标准的 Minkowski 符号(例如 $B\oplus t=\{b+t\mid b\in B\}$).

1.2 基本比较单元之间的距离定义

基本比较单元之间的距离定义(不匹配程度)是定义异常轨迹的关键.基于平移的最小 Hausdorff 距离就是两个点集经过平移之后的最小 Hausdorff 距离.因此,将平移量当作公共偏差,基于平移的最小 Hausdorff 距离可以被看作是两个点集在消除公共偏差之后的距离值.但是用它来计算轨迹之间的距离时,存在两个问题:(1) 其距离值与点在点集中的位置无关;(2) 它的公共偏差是无限制的,即图 1 中的 S_3 也会被认为是 S_4,S_5 和 S_6 的相似轨迹,这显然是不符合现实认知的.下面对该距离度量方法作一些改变.假设轨迹 $A=\{a_1,\ldots,a_n\}$ 和 $B=\{b_1,\ldots,b_m\}$,A 的基本比较单元表示为 $A_i=\{a_{i+1},\ldots,a_{i+k}\mid 0\le i\le n-k,k\le n\}$,B 的基本比较单元表示为 $B_j=\{b_{j+1},\ldots,b_{j+k}\mid 0\le j\le m-k,$

$k \le m\}$.由于轨迹中每两点时间间隔相同,那么只要将基于平移的最小 Hausdorff 距离中的每个点到另一个集合的最近点距离用该点到其在另一个点集中的对应点的距离替代,那么该距离度量方法就包含了轨迹片段在一定时间间隔内变化趋势之间的差异,即速度因素.而且,由于每个点都是与对应点进行比较,那么等式(2)中代表公共偏差的平移变量 t 可以用每对点之间距离的平均值 $\bar{t}$ 来代替.因此,两个基本比较单元之间的距离 $dist_k(A_i, B_j \oplus \bar{t})$ 可以定义为

$$dist_k(A_i, B_j \oplus \bar{t}) = \max_{p=1,\ldots,k}((a_{i+p} - b_{j+p}) \oplus \bar{t}) \tag{3}$$

其中,$\bar{t} = \frac{1}{k}\sum_{p=0}^{k-1}(a_{i+p} - b_{j+p})$.

等式(3)其实是基于平移的最小 Hausdorff 距离的简化,它的另一种解释是两个基本比较单元中对应点之间的最大距离减去所有对应点之间距离的平均值.它不仅代表着两个基本比较单元在形状上的差异,而且还蕴含着运动规律的差异.显然,这个距离还存在一个问题.如例 1 中 S_3 是异常轨迹,但是在路径 R_2 中轨迹的运动规律与其是相似的,如果按照等式(3)来计算基本比较单元之间的距离,那么 S_3 就会被认为是正常轨迹.因此,公共偏差必须被限定在一定范围内,即基本比较单元仅和其周围一定范围内的其他轨迹的基本比较单元进行比较.按照这个思想,下面先给出 ω-邻域点集 $N_\omega(a_i)$和两个基本比较单元之间距离为无穷的定义,然后再给出两个基本比较单元之间距离公式.

定义 1(ω-邻域点集). 给定一个邻域阈值 ω,一个轨迹点 a_{i+p},对于任一其他轨迹上的点 b_{j+p},如果 $dist(a_{i+p} - b_{j+p}) \le \omega$,那么 b_{j+p} 属于 a_{i+p} 的 ω-邻域点集,表示为 $b_{j+p} \in N_\omega(a_{i+p})$.

定义 2(距离无穷). 给定两个长度为 k 的基本比较单元 A_i,B_j,如果 A_i,B_j 之间有一对点满足 $dist(a_{i+p},b_{j+p})>\omega$,则 A_i,B_j 之间的距离为无穷大.

那么,两个基本比较单元 A_i, B_j 之间的距离 $dist_{k,\omega}(A_i, B_j \oplus \bar{t})$ 定义如下:

$$dist_{k,\omega}(A_i, B_j \oplus \bar{t}) = \max_{p=0,\ldots,k-1}(dist(a_{i+p} - (b_{j+p} \oplus \bar{t})) \tag{4}$$

其中, $dist(a_{i+p} - (b_{j+p} \oplus \bar{t})) = \begin{cases} dist(a_{i+p} - b_{j+p} - \bar{t}), & (a_{i+p} - b_{i+p}) \le \omega \\ \infty, & (a_{i+p} - b_{i+p}) > \omega \end{cases}$, $\bar{t} = \frac{1}{k}\sum_{p=0}^{k-1}(a_{i+p} - b_{j+p})$.

1.3 局部匹配、全局匹配和异常轨迹

根据公式(4)定义的基本比较单元之间的距离公式,下面给出轨迹局部匹配、全局匹配和异常的概念.

定义 3(局部匹配). 给定一个距离阈值 θ,如果两个基本比较单元 A_i,B_j 之间的距离 $dist_{k,\omega}(A_i, B_j \oplus \bar{t}) \le \theta$,那么这两个基本比较单元是相互匹配(简称局部匹配)的,符号表示为 $LM_{k,\omega,\theta}(A_i, B_j)$.

定义 3 是两条轨迹之间的局部匹配.因此,结合传统 $DB(p,D)$异常点的思想[3−5],下面给出全局匹配轨迹和异常轨迹的定义:

定义 4(全局匹配). 设 A^+表示目标轨迹 A 中与轨迹 B 的任一基本比较单元匹配 B_j 的所有基本比较单元 A_i 集合,联合 $Union(A \mid B) = \{a_q \mid a_q \in A_i, A_i \in A^+\}$.给定一个比例阈值 ξ,如果等式(5)成立,则称 B 为 A 的全局匹配轨迹.

$$\left|Union(A \mid B)\right| \ge \xi \times n \quad (0<\xi \le 1) \tag{5}$$

其中,$Union(A|B)$表示联合 $Union(A|B)$所包含的轨迹点数目,n 表示目标轨迹 A 的轨迹点数目.

定义 5(异常轨迹). 给定异常阈值 δ,如果与目标轨迹 A 全局匹配的轨迹数目小于 δ,则称 A 为异常轨迹.

显然,局部匹配是对称的,而全局匹配则是不对称的,即轨迹 A 是轨迹 B 的全局匹配轨迹并不表示轨迹 B 是轨迹 A 的全局匹配轨迹.其原因是基本比较单元的长度是相同的,而轨迹的长度是不同的.这也符合实际应用的要求(例如,两条长度相差很大的轨迹,长轨迹可能是短轨迹的全局匹配轨迹,而短轨迹一定不是长轨迹的全局匹配轨迹).

2 基于 R-Tree 异常轨迹检测算法

按照本文异常轨迹的定义,最直接的检测方法是将每一条轨迹与其他轨迹逐一比较,通过检测两者之间基本比较单元的匹配情况确定两条轨迹是否全局匹配,最后,根据与目标轨迹的全局匹配轨迹的数目确定该轨迹是否异常.这种方法实现简单,但其计算代价比较昂贵.与本文研究最为相关的文献[6]采用二阶段划分方法优化算法性能,即先将轨迹划分为较大的轨迹片段组成粗粒度 *t*-partition,并检测出异常的粗粒度 *t*-partition,然后再对异常部分进行划分,寻找其中异常的细粒度 *t*-partition.这种方法在一般情况可以提高算法性能,但是它的问题是可能导致异常的细粒度 *t*-partition 丢失.这是因为 *t*-partition 仅由轨迹片段的两个端点组成,是一个近视的描述,所以正常的粗粒度 *t*-partition 并不意味着其中没有异常的细粒度 *t*-partition(尽管文献[6]使用最小表述长度 MDL(minimum description length)原则来获得最佳的粗粒度划分尺寸,但这并不能从根本上解决异常的细粒度 *t*-partition 丢失的可能性).另外,按照本文的基本轨迹单元之间距离的度量方法,轨迹片段之间存在相互交叉重叠的情况.因此,本文提出了基于 R-Tree 索引的方法来加速算法效率,该方法利用基本比较单元比较区域性的特点,借助于 R-Tree 的搜索功能和轨迹之间距离特征矩阵消除两条轨迹之间的满足定义 2 的基本比较单元对,从而提高算法效率.

2.1 实现原理

由于基本比较单元仅和其邻域内的基本比较单元进行比较,那么,如果能消除那些满足定义 2 的基本比较单元对,算法效率将会大大提高.针对这一设想,本文提出了一种基于 R-Tree 的异常轨迹检测算法.该算法的主要目标就是找出所有来自不同轨迹且不满足定义 2(即距离不等于无穷)的基本轨迹单元对.这个过程分为两个步骤实现:

(1) 通过 R-Tree 搜索找出每个轨迹点的定义 1 中的ω-邻域点集,与该轨迹点构成点对集合.按照本文对基本比较单元之间距离的定义,每个轨迹点仅和其ω-邻域点集中的点进行比较.因此,如果事先采用 R-Tree 索引机制来索引轨迹数据中的所有轨迹点,那么借助于 R-Tree 搜索很容易找到每个轨迹点的整个ω-邻域点集.R-Tree 索引的创建可以在轨迹数据录入时进行.

(2) 借助于两个轨迹构成的距离特征矩阵的特性找出不满足定义 2 的基本比较单元对.这个过程主要实现从距离小于指定邻域阀值ω的点对集合中找出不满足定义 2 的基本比较单元对.

首先,给出两条轨迹的距离特征矩阵的定义以及其特征:

定义 6(距离特征矩阵). 设轨迹 $T_i=\{t_{i1},\ldots,t_{ip}\}$和 $T_j=\{t_{j1},\ldots,t_{jq}\}$,那么其距离特征矩阵可以表示为

$$M=\begin{Bmatrix}(t_{i1},t_{j1}) & \ldots & (t_{ip},t_{j1})\\ \ldots & \ldots & \ldots\\ (t_{i1},t_{jq}) & \ldots & (t_{ip},t_{jq})\end{Bmatrix}.$$

定义 7(正对角序号). 矩阵中元素(t_{ie},t_{if})的正对角序号是其行号和列号之差,表示为 $pssn(t_{ie},t_{if})=e-f$.

定义 8(k-正对角相邻). 如果矩阵的 k 个元素 $(t_{ie_1},t_{jf_1}),\ldots,(t_{ie_k},t_{jf_k})$ $(e_1<e_2<\ldots<e_k)$满足下面两个条件:

1) $pssn(t_{ie_1},t_{jf_1})=\ldots=pssn(t_{ie_k},t_{jf_k})$;

2) $e_2-e_1=\ldots=e_k-e_{k-1}=1$,

则称 $(t_{ie_1},t_{jf_1}),\ldots,(t_{ie_k},t_{jf_k})$ 是 k-正对角相邻.

显然,k-正对角相邻实际上是由两个基本比较单元组成.如果将矩阵元素中距离大于ω的点对置成空值∅,那么一个 k-正对角相邻的非空值元素就对应一个距离不满足定义 2 的基本比较单元对.因此只要找出所有 k-正对角相邻的非空元素组合,就是找出了所有可能局部匹配的基本比较单元对.

2.2 算法实现

按照上面的分析,基于 R-Tree 的异常轨迹检测算法的主要思想是先通过 R-Tree 找出每个轨迹点 t_{ip} 的 ω-邻域点集 $\{t_{jq}|t_{jq}\in N_\omega(t_{ip})\}$,并组成 ω-邻域点对集合 $\{(t_{ip},t_{jq})\}$(t_{ip} 称为目标轨迹点,t_{jq} 被称为比较轨迹点);然后将这些点对集合插入到一个按照比较轨迹点的轨迹编号、点对的正对角序号、目标轨迹点在轨迹中的序号 i 排序的有序堆栈中,那么只要逐个从堆栈中取出点对元素,并判断出栈元素是否和前 k–1 个出栈元素是否是 k-正对角相邻且目标轨迹是同一轨迹.如果为真,则表示这 k 个元素就构成一个可能局部匹配的基本轨迹单元对,那么只要检查其对应的两个基本比较单元之间的距离是否大于距离阈值,如果成立,则表示两个基本比较单元是匹配的.图 3 显示了检测轨迹 T_i 是否是异常轨迹的算法实现源代码,其中 T 表示轨迹数据集,*params* 表示算法所涉及的参数 $k,\omega,\theta,\xi,\delta$;数组 *CandArray* 存储当前正对角相邻的点对元素组合,类似于一个长度为 k 的移动窗口;数组 *MatchedArray* 存储目标轨迹和某条轨迹间局部匹配的基本比较单元对的目标轨迹点(去除重复轨迹点);*numMatchedTrajectory* 表示与当前轨迹全局匹配的轨迹数;算法首先通过 R-Tree 找出所有 ω-邻域点对集合 $\{(t_{ip},t_{jq})\}$ 并插入有序堆栈 *stackNeighbour*(第 1 行);接着,逐一从堆栈中取栈顶元素 *curEn*,并检查栈顶元素和上一次元素 *oldEn* 是否正对角相邻.如果为真,那么执行如下操作:(1) 检查对应比较轨迹点是否属于同一轨迹,如果为真,则清空数组 *CandArray*,同时,检查 *MatchedArray* 中轨迹点数目与目标轨迹中轨迹点总数的比例是否达到 ξ,如果是,则表示该比较轨迹为目标轨迹的全局匹配轨迹,那么 *numMatchedTrajectory* 加 1(第 6~9 行);(2) 清空数组 *CandArray*,并将栈顶元素插入该数组(第 10~11 行实现).如果栈顶元素和上一次栈顶元素属于正对角相邻,那么执行如下操作:(1) 将栈顶元素插入到数组 *CandArray*,并检查该数组的元素个数是否等于 k,如果等于 k(表示找到一个 k-正对角相邻的非空元素组合),则根据定义 3 检查对应的两个基本比较单元是否局部匹配,如果局部匹配,则将数组 *CandArray* 中目标轨迹点插入到数组 *MatchedArray*,检查两个基本比较单元是否局部匹配由函数 *KMatch* 实现,此外,该函数还删除了数组 *CandArray* 的第 1 个元素,并将其他元素往前移一个(第 12~16 行).上述处理之后,将栈顶元素赋给变量 *oldEn*.当堆栈为空时,将返回与目标轨迹全局匹配轨迹数目是否大于 δ.整个异常轨迹检测算法就是对每条轨迹执行 *EstimateTrajectory* 函数以确认其是否为异常轨迹.

```
EstimateTrajectory(T,i,params)
01: stackNeighbour=QueryRTree (Ti,ω);
02: numMatchedTrajectory=0;
03: while (stackNeighbour isn't empty)
04:     curEn=stackNeighbour. RemoveTop ();
05:     if (olden, curEn is not diagonally adjacent)
06:         if curEn.tid !=oldEn.tid
07:             MatchedArray. Clear();
08:             if MatchedArray.Size()>ξ×Ti.Size();
09:                 numMatchedTrajectory++;
10:         CandArray. Clear();
11:         CandArray. addEntry (curEntry);
12:     else
13:         CandtArray. addEntry (curEntry);
14:         if CandArray. Size()==k
15:             if KMatch(CandArray,θ,k)
16:                 MatchedArray. addEntry (CandArray);
17:     oldEn=curEn;
18: return (numMatchedTrajectory<δ);
```

Fig.3 Pseudo code of outlier detection

图 3 异常轨迹检测的伪代码

3 性能分析

异常轨迹检测算法的主要计算代价是点与点之间的距离计算,因此本节将以两个点之间的距离计算为基准代价分析算法的性能.下面将分析基本算法和基于 R-Tree 的异常轨迹检测算法的性能,并作相应的比较.

假设轨迹数据空间的整个区域面积为 L^2,并假设轨迹点在整个数据空间均匀分布.相关参数说明见表 1.

Table 1 Parameter description

表 1 参数说明

Parameter	Introduction
N_{bcu}	The area of the external rectangle w.r.t. ω-neighborhood circle of points in S_i
C_{base}	The cost of the benchmark algorithm
N_i	The number of distance computations of each point w.r.t. trajectory in a R-tree
N_{tp}	The number of points w.r.t. trajectories
C_q	The cost of the phase of search in a R-Tree
C_c	The cost of determining whether the basic comparison unit is globally matched
p	The comparison between the benchmark algorithm and the R-Tree based algorithm

最基本的算法就是找出所有的基本比较单元对,逐一比较它们之间的距离以确定其是否局部匹配,并最后得到异常轨迹.这种方法的计算复杂度就是所有基本比较单元对的个数 N_{bcu} 乘以计算基本比较单元对距离的平均代价 C_{bcu}.显然 $1<C_{bcu}<k$,因此我们可以得到其计算复杂度 C_{base} 是

$$C_{base}=N_{bcu}\times C_{bcu}=\sum_{i=1}^{n}m_i\times\sum_{j\neq i,j=1,\ldots,n}m_j\times C_{bcu} \tag{6}$$

基于 R-Tree 的距离计算分布两个函数 QueryRTree 和 KMatch,即 R-Tree 搜索和计算两个可能局部匹配的基本比较单元之间距离这两个阶段.首先,由于 ω-邻域是圆形区域,R-Tree 是矩形区域搜索方式,因此对于每个点,算法必须检查其 ω-圆形邻域的内接矩形和外接矩形之间的点是否处在 ω-邻域之内.那么,整个 R-Tree 搜索的计算复杂度为

$$C_q=N_{tp}\times N_i=\sum_{i=1}^{n}m_i\times\frac{4\omega^2-2\omega^2}{L^2}\sum_{j\neq i,j=1,\ldots,n}m_j=\sum_{i=1}^{n}\sum_{j\neq i,i=1}^{n}\frac{2\omega^2}{L^2}m_jm_i \tag{7}$$

后一个阶段,由于一个不属于 S_i 轨迹点处在 S_i 某个轨迹点的 ω-邻域之内的概率为 $\pi\omega^2/L^2$,因此,任一其他轨迹的基本比较单元和 S_i 的某个基本比较单元不满足定义 2 的概率为 $(\pi\omega^2/L^2)^k$.所以其计算复杂度为

$$C_c=\sum_{i=1}^{n}\sum_{j\neq i,j=1}^{n}(m_i-k+1)\times(m_j-k+1)\times k\times(\sqrt{\pi}\omega/L)^{2k} \tag{8}$$

综上所知,基于 R-Tree 的异常轨迹检测算法的总计算复杂度为

$$C=C_q+C_c=\sum_{i=1}^{n}\sum_{j\neq i,i=1}^{n}\frac{2\omega^2}{L^2}m_jm_i+\sum_{i=1}^{n}\sum_{j\neq i,j=1}^{n}(m_i-k+1)\times(m_j-k+1)\times k\times(\sqrt{\pi}\omega/L)^{2k} \tag{9}$$

为了简化起见,假设每条轨迹的长度相同,即 $m_i=m$,且 $\omega<<L$,且 $k\geq 2$,那么基于 R-Tree 的异常轨迹检测算法和基本算法的性能之比如下:

$$p=\frac{2n(n-1)m^2\dfrac{\omega^2}{L^2}\times C_{bcu}}{n(n-1)(m-k+1)^2\times k}=\frac{2m^2C_{bcu}}{(m-k+1)^2\times k}\frac{\omega^2}{L^2}\approx\frac{2\omega^2C_{bcu}}{kL^2}<\frac{2\omega^2}{L^2} \tag{10}$$

TROAD 算法中采用的二阶段划分方法[6]能够提高算法性能的比例是 $n_t^2/(n_c^2+n_f^2)$,其中,n_t 表示细粒度 t-partition 的数目,n_c 表示粗粒度 t-partition 的数目,n_f 表示异常粗粒度 t-partition 中细粒度 t-partition 的数目.由于 n_c 越小就意味着算法遗漏异常细粒度 t-partition 的情况就越严重,因此,n_c 的值不能太小.因此,即使不考虑其丢失异常细粒度 t-partition 的不足,二阶段划分方法对算法性能的提高也是有限的.在本文提出的算法中,由于 $\omega<<L$,因此,算法可以明显地提高算法效率,后面的实验结果也证明了这一结论.

4 实 验

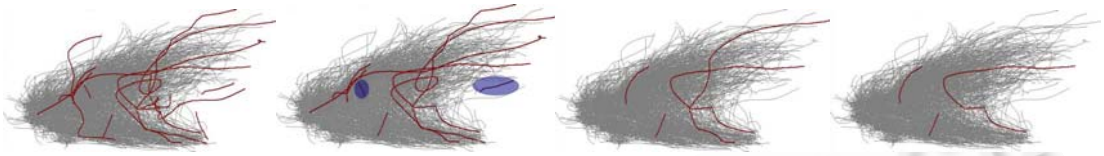
基于本文提出的轨迹检测算法,我们开发了异常轨迹检测系统 TrajDetector.该系统由 Visual C++ 6.0 开发,操作系统为 windows XP.实验硬件环境为:CPU 为 Centrino2 2.1G,内存为 1G,在 Thinkpad T61 上运行.R-Tree 采用的是 Beckmann[10]提出的 R*-tree(http://www.rtreeportal.org/),并对它进行适当的改变.实验数据集采用取自飓风数据集(http://weather.unisys.com/hurricane/index.html)中的经纬度数据.实验中被使用的数据集包括:(1) 大西洋飓风数据集 2.该数据集记录了从 1850 年~2006 年在大西洋上飓风移动的数据,包括 20 368 个点组成的 1 277 条轨迹;(2) 大西洋飓风数据子集 1.该数据集记录了 1950 年~2006 年大西洋上的飓风轨迹数据,包括 18 951 点构成的 608 条轨迹;(3) 大西洋飓风数据子集 2.该数据集记录了 1990 年~2006 年大西洋上飓风轨迹数据,包括 7 270 个点构成的 221 条轨迹.

4.1 参数分析

本文提出的算法主要涉及 5 个需要用户预先设定的参数:$\omega,k,\theta,\xi,\delta$.下面以数据集 II——大西洋飓风数据集为实验数据集对不同参数的影响进行详细分析.

邻域阈值 ω 的大小直接决定算法的计算代价,由于其含义比较直观,对于应用领域专家很容易确定一个比

较合适的阈值.表 2 显示了不同邻域阈值下(从 1~10)的计算代价和算法检测出的异常轨迹数目.从表 2 可以发现,随着邻域阈值的增大,算法的计算代价急速增加,ω=10 时的计算代价是ω=1 时的 600 多倍.由此可见,选择一个合适的ω值可以大幅减少算法的计算代价,进一步说明了本文所提算法具有很高的效率.另一方面,随着ω值增大,异常轨迹的数目也逐渐减少,主要原因是:ω值越大,意味着更大范围内寻找与目标轨迹全局匹配的轨迹,而本文提出的距离函数具有消除距离偏移的能力,因此,ω的增大就意味着在更大范围内寻找与目标轨迹形状相匹配的轨迹.图 4 显示了ω分别为 5,6,7,8 时算法检测出的异常轨迹.从图中可以发现:随着ω的增大,一些明显异常的轨迹(椭圆区域标识的轨迹)被归类为正常.其原因在于,该轨迹尽管与其相邻区域的轨迹不具有相似的运动规律,但却和较远区域的轨迹具有类似的运动规律.因此,给ω设定一个适当值,不仅有助于提高算法的计算效率,而且能找出一些比较隐蔽的异常轨迹.



ω=5 ω=6 ω=7 ω=8

Fig.4 Trajectory outlier under different ω

图 4 不同ω值的异常轨迹

基本比较单元的轨迹点数目 k 和局部匹配的距离阈值θ是与基本比较单元的局部匹配程度相关的参数.k可以看成是局部匹配的长度,θ则是局部匹配的精度.k越大,算法检测到的异常轨迹数就越多;θ越大,算法检测到的异常轨迹数就越少(其他参数固定不变).表 2 显示了随着k和θ的改变而导致算法的计算代价和异常轨迹数目的变化.随着 k 值的增大,算法计算代价会先增后降,但变化幅度不是很大,主要原因是随着 k 会导致每个基本比较单元的距离计算代价增加,但它同时又会使得局部匹配的基本比较单元对数目减少,而变化不大是因为在算法代价中,R-Tree 搜索代价占了整个算法代价的很大比重.因此,算法搜索到的异常轨迹数据与 k 成反比,k 的增大将导致要求匹配的基本比较单元更长,在其他条件相同的情况下,k 的增大会导致异常轨迹数的增加;而θ值的增大仅意味着两个点匹配的距离阈值增大,因而它仅会导致异常轨迹数减少,对算法性能影响几乎没有.

Table 2 Number of trajectory outlier and CPU time under different parameter values

表 2 不同参数值的 CPU 时间和异常轨迹数目

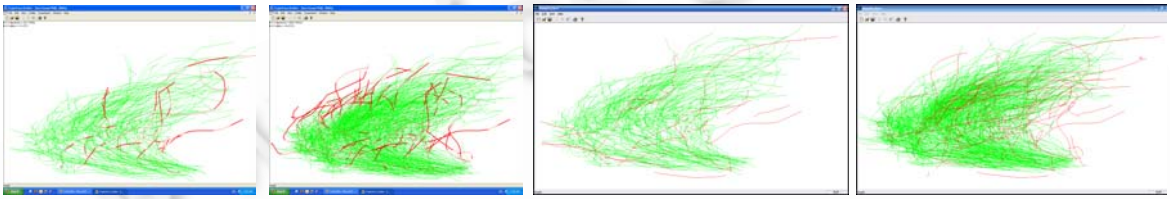
k=3, θ=0.3, ξ=0.5, δ=3										
ω	1	2	3	4	5	6	7	8	9	10
CPU time(s)	12	47	135	340	687	1352	2198	3605	5363	7886
Number of trajectory outliers	339	94	40	24	13	9	4	3	1	1
ω=5, θ=0.5, ξ=0.5, δ=2										
k		2	3	4	5	6	7	8	9	10
CPU time(s)		676	684	696	704	724	726	776	750	728
Number of trajectory outliers		6	6	14	35	72	130	260	412	587
ω=5, k=5, ξ=0.5, δ=2										
θ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
CPU time (s)	728	741	730	734	704	724	728	711	722	707
Number of trajectory outliers	1122	308	97	55	35	27	19	14	11	9
ω=5, k=3 ,θ=0.3, δ=5										
ξ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
CPU time (s)	696	705	708	704	701	729	704	709	698	713
Number of trajectory outliers	18	18	18	19	20	25	34	41	56	96
ω=5, k=3, θ=0.3, ξ=0.5										
δ	1	2	3	4	5	6	7	8	9	10
CPU time (s)	701	710	707	709	712	736	716	719	725	704
Number of trajectory outliers	10	10	13	17	20	23	26	28	29	32

全局匹配的比例阈值ξ和异常阈值 δ 是与全局匹配相关的参数.与 k 和θ类似,如果将ξ当作全局匹配的精度阈值,那么δ可以看成是全局匹配的数量阈值,即小于这个数量阈值的话,就是异常轨迹.表 2 显示了ξ和δ的改变

对算法的计算代价和检测到的异常轨迹数的影响.这两个参数对算法的计算代价的影响并不是很明显,其原因是这两个参数的变化并不影响候选局部匹配点对集的数量;但这两个参数可以明显影响到算法搜索异常轨迹集合,这是因为随着ξ值增大,就意味着两条轨迹间要有更多的轨迹点属于局部匹配的基本比较片段,这将导致全局匹配的轨迹数下降;而随着δ值的增加,就意味着每条轨迹需要有更多与之全局匹配的轨迹才能被判断不是异常轨迹.表 2 异常轨迹的数目也显示了这个结果.

4.2 实验结果

由于本文算法的轨迹片段距离定义与 TROAD 算法不同,因此,两种方法性能之间不具有直接的可比性.此外,由于 TROAD 算法中使用的二阶段划分的优化算法不仅自身存在问题,而且由于本文表示局部特征的轨迹片段是相互交叉的,算法中无法使用二阶段划分方法来优化算法性能,因此,在实验中仅与 TROAD 算法在检测到异常轨迹效果上进行比较.图 5 显示了 TROAD 算法在大西洋飓风轨迹数据子集 1 和大西洋飓风轨迹数据子集 2 中检测到的异常轨迹(http://netfiles.uiuc.edu/jaegil/www/icde08)[6].从图中可以发现:除了一些位于轨迹稀疏区域的轨迹被定为异常轨迹之外,那些与周围轨迹相比,拥有异常移动路径(形状)的轨迹也被标识为异常轨迹.图 6 显示了基于 R-Tree 的异常轨迹检测算法在同样两个数据子集中检测出的异常轨迹情况(参数ω=5,k=4,θ=5,ξ=0.5,δ=5),从图中可以发现,除了那些具有异常形状的轨迹之外,大多数强飓风由于其拥有更快的移动速度而被标为异常轨迹;而在 TROAD 算法中,却会由于其轨迹路径类似于其他一般飓风而被认为是正常轨迹.因此,本文提出的异常轨迹检测算法要比 TROAD 算法更具有现实意义.



(a) 1990~2006 (b) 1950~2006 (a) 1990~2006 (b) 1950~2006

Trajectory outliers mined by TROAD Trajectory outliers mined by R-tree based algorithm

Fig.5 Comparison of experimental results in outlying trajectories (v.s. TROAD)

图 5 异常轨迹实验结果的对比(与 TROAD 比较)

5 结束语

随着无线技术和定位服务的发展,挖掘轨迹数据中隐藏的有效信息已经成为数据挖掘的一个研究热点.本文针对现有轨迹数据的自身特性,首先从轨迹中抽取所有可能的基本轨迹单元构成轨迹的局部特征元素集合.然后,提出了一种适合计算基本比较单元之间距离的度量方法;并在此基础上提出局部匹配、全局匹配和异常轨迹的定义.接着,本文还利用基本比较单元局部性的特点,提出了一种基于 R-Tree 的异常轨迹检测算法.实验结果和性能分析都表明本文所提出的算法不仅具有很高的计算效率,而且其检测到的异常轨迹具有实际意义,是一种有效的异常轨迹检测算法.

致谢 在此,我们特别向对本文的工作给予支持的西南交通大学交通运输工程博士后流动站的老师和同学提出的宝贵建议和帮助表示感谢.

References:

[1] Knorr EM, Ng RT, Tucakov V. Distance-Based outliers: Algorithms and applications. VLDB Journal, 2000,8(3):237−253.

[2] Ramaswamy S, Rastogi R, Shim K. Efficient algorithms for mining outliers from large data sets. In: Chen WD, Jeffrey FN, Philip AB, eds. Proc. of the SIGMOD 2000. New York: ACM, 2000. 427−438.

[3] Breunig MM, Kriegel HP, Ng RT, Sander J. LOF: Identifying density-based local outliers. In: Chen WD, Jeffrey FN, Philip AB, eds. Proc. of the SIGMOD 2000. New York: ACM, 2000. 93−104.

[4] Papadimitriou S, Kitagawa H, Gibbons PB, Faloutsos C. LOCI: Fast outlier detection using the local correlation integral. In: Dayal U, Ramamritham K, Vijayaraman TM, eds. Proc. of the ICDE 2003. New York: IEEE Computer Society, 2003. 315−326.

[5] Aggarwal CC, Yu PS. Outlier detection for high dimensional data. In: Aref WG, ed. Proc. of the SIGMOD 2001. New York: ACM, 2001. 37−46.

[6] Lee J, Han J, Li X. Trajectory outlier detection: A partition-and-detect framework. In: Proc. of the ICDE 2008. New York: IEEE Computer Society, 2008. 140−149.

[7] Chen J, Maylor K. Leung, Gao Y. Noisy logo recognition using line segment hausdorff distance. Pattern Recognition, 2003,36(4): 943−955.

[8] Huttenlocher DP, Klanderman GA, Rucklidge WA. Comparing images using the hausdorff distance. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1993,15(9):850−863.

[9] Huttenlocher DP, Kedem K, Sharir M. The upper envelope of voronoi surfaces and its applications. Discrete and Computational Geometry, 1993,9(1):267−291.

[10] Beckmann N, Kriegel HP, Schneider R, Seeger B. The R*-tree: An efficient and robust access method for points and rectangles. In: Hector GM, ed. Proc. of the SIGMOD'90. New York: ACM, 1990. 322−331.

刘良旭(1974－),男,浙江奉化人,博士,讲师,主要研究领域为移动对象数据库,数据挖掘.

乐嘉锦(1951－),男,教授,博士生导师,CCF 高级会员,主要研究领域为数据仓库,软件工程,数据挖掘.

乔少杰(1981－),男,博士,主要研究领域为移动对象数据库,数据挖掘.

唐常杰(1946－),男,教授,博士生导师,CCF 高级会员,主要研究领域为数据库,数据挖掘.

刘宾(1983－),男,博士,主要研究领域为数据库查询.