轨迹数据相似性综述

丁光伟

1、引言

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，其中就包括轨迹的相似性查询[1]，其定义如下。

定义 1（轨迹相似性查询）给定一个轨迹 T，轨迹数据库 S 和一个阈值η，返回 S 的一个子集S'，使得

$$\forall s' \epsilon S' \quad d(T,s') \leq \eta,$$

其中 d(·)是轨迹数据之间的相似性计算函数。

在移动对象的轨迹相似性查询中，相似性计算函数是核心。我们给出相似性计算函数[1]的形式化定义。

定义 2（轨迹相似性计算函数）给定一个定义在轨迹数据上的空间 T，以及任意两条 T 中的轨迹 x 与 y，则 T 上的相似性计算函数 d 定义为：

$$d: T \times T \rightarrow R$$

其中 R 是实数空间。

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。因此，在面对不同场景时我们需要根据具体情况采用不同的相似性查询方法。本文重点介绍当前主流的一些相似性计算函数。

本文组织结构如下：第 1 章是引言部分，第 2 章将依次介绍各种相似性计算

函数以及优缺点，第 3 章是全文总结。

2、相似性计算函数

2.1 欧氏距离（EU）

欧氏距离[1][3]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A = $\{a_1, a_2, \dots a_n\}$，B = $\{b_1, b_2, \dots b_n\}$，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。

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

$$l_p norms(A,B) = \sqrt[p]{\sum_{i=1}^{n}(a_i - b_i)^p} \tag{2}$$

欧氏距离有很多优点，比如计算简单，长度为 n 的两条轨迹，可以在$O(n)$时间内计算出它们的相似度，而且它满足三角不等式，如式（3）所示，运用三角不等式可以计算两条轨迹之间的距离下限，从而可以进行高效的轨迹查询。

$$\forall A, B, C \in T, EU(A,C) + EU(B,C) \geq EU(A,B) \tag{3}$$

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 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。

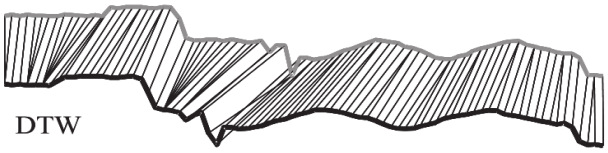

图 2.2 DTW 一对多的对应关系

DTW[4][5][6][7]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。

$$DTW(A,B)=\begin{cases}0, if\ \ n=0\ \ and\ \ m=0\\ \infty, if\ \ n=0\ \ or\ \ m=0\\ d\big(Head(A),Head(B)\big)+min\begin{cases}DTW\big(A,Rest(B)\big)\\ DTW\big(Rest(A),B\big)\\ DTW\big(Rest(A),Rest(B)\big)\end{cases}, \text{otherwise}\end{cases} \tag{4}$$

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

2.3 最长公共子序列距离（LCSS）

顾名思义，最长公共子序列距离[8][9]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。

$$\mathrm{LCSS(A,B)}=\begin{cases}0, if\ \ n=0\ \ or\ \ m=0\\ LCSS\big(Rest(A),Rest(B)\big)+1, if\ \ d\big(Head(A),Head(B)\big)\le\varepsilon, and\ \ |n-m|<\delta\\ \max\{LCSS(Rest(A),B),LCSS(A,Rest(B)\},otherwise\end{cases} \quad (5)$$

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

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

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

2.4 编辑距离（EDR）

EDR[10]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。

$$dist(r_i,s_i)=\begin{cases}0 & if\ \ r_i=s_i\\ 1 & if\ \ r_i\ \text{or}\ s_i\ \ is\ \ a\ \ gap\\ 1 & otherwise\end{cases} \quad (6)$$

然而时间序列中的元素是实数，有时候不会像字符那样完全相等，所以当两个实数之差小于阈值δ时，我们就认为这两个实数相等，因此时间序列中元素之间的距离如式（7）所示。EDR 是基于时间序列中元素的距离 $\mathrm{dist_{edr}}$ 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。

$$dist_{edr}(r_i, s_i) = \begin{cases} 0 & if\ \ |r_i - s_i| \le \delta \\ 1 & if\ \ r_i \text{ or } s_i\ \ is\ \ a\ \ gap \\ 1 & otherwise \end{cases} \tag{7}$$

$$\mathrm{EDR(R,S)} = \begin{cases} n & if\ \ m = 0 \\ m & if\ \ n = 0 \\ EDR\big(Rest(R), Rest(S)\big) & if\ \ dist_{edr}(r1, s1) = 0 \\ \min\begin{cases} EDR\big(Rest(R), Rest(S)\big) + dist_{edr}(r1, s1) \\ EDR(Rest(R), S) + dist_{edr}(r1, gap) \\ EDR\big(R, Rest(S)\big) + dist_{edr}(gap, s1) \end{cases} & otherwise \end{cases} \tag{8}$$

2.5 带真实惩罚的编辑距离（ERP）

ERP[11]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。

$$dist_{erp}(r_i, s_i) = \begin{cases} |r_i - s_i|\ \ ,if\ \ r_i\ \ ,s_i\ \ not\ \ gaps \\ |r_i - g|\ \ ,if\ \ s_i\ \ is\ \ a\ \ gap \\ |s_i - g|\ \ ,if\ \ r_i\ \ is\ \ a\ \ gap \end{cases} \tag{9}$$

$$\mathrm{ERP(R,S)} = \begin{cases} \sum_1^n |s_i - g|\ ,if\ \ m = 0 \\ \sum_1^m |r_i - g|\ ,if\ \ n = 0 \\ \min\begin{cases} ERP\big(Rest(R), Rest(S)\big) + dist_{erp}(r1, s1) \\ ERP(Rest(R), S) + dist_{erp}(r1, gap) \\ ERP\big(R, Rest(S)\big) + dist_{erp}(gap, s1) \end{cases} & ,otherwise \end{cases} \tag{10}$$

2.6 多维数据序列的相似性搜索

时间序列数据是一维数据，仅包含坐标和时间戳轨迹数据是二维数据，除此之外还有多维数据。多维数据序列的相似性搜索[12]是一种有效地查找与给定查询序列相似的多维数据序列的方案，以视频流作为多维数据序列的典型例子来讲解，可以解决以下两类问题：

1、查询子流：找出数据库中的视频，这些视频的某个子视频段和待查询视频相似，并且只播放这些子视频段。

2、长查询（待查询视频比数据库中视频长）：找出数据库中和待查询视频某

个子视频段相似的视频。

由于视频流不能直接进行相似度计算，需要将其转换为多维数据。所以在处理查询操作之前，需要进行预处理，即从视频流或图像等原始数据中提取特征向量，每个向量代表多维空间中的点，一系列向量构成一个多维序列。然后将数据库中每一个多维序列按照一定的规则分割成子序列，每个子序列都包含在最小边界矩形（MBR）中。由于矩形可以由一对对角端点确定，所以一个 n 维的 MBR 可以由两个端点 L 和 H 表示，即 MBR=(L,H)。MBR 之间的距离 D_{mbr} 计算如公式（11）所示，图 2.3 以二维空间的 MBR 为例，展示了 MBR 之间的距离。然后将 D_{mbr} 正态化处理得到 D_{norm}。

$$D_{mbr}(A,B) = (\sum_{1\le k\le n} x_k{}^2)^{1/2} \tag{11}$$

其中，$x_k = \begin{cases} |h_{A,k} - l_{B,k}| & if\ \ h_{A,k} < l_{B,k} \\ |l_{A,k} - h_{B,k}| & if\ \ h_{B,k} < l_{A,k} \\ 0 & otherwise \end{cases}$

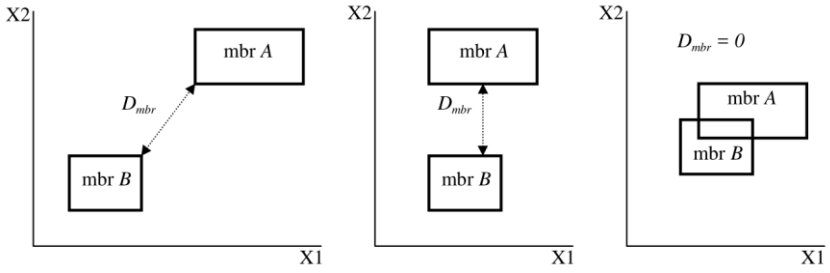

图 2.3 mbr 之间的距离

预处理结束之后，将进行相似性查询，相似性查询算法分为三步。

第一步，将查询序列按照相同的划分规则，划分为一个或多个子序列。如图 2.4 所示，其中 mbr_1 、mbr_2 、mbr_3 和 mbr_4 属于数据序列 S，mbr_q 属于待查询序列 Q。

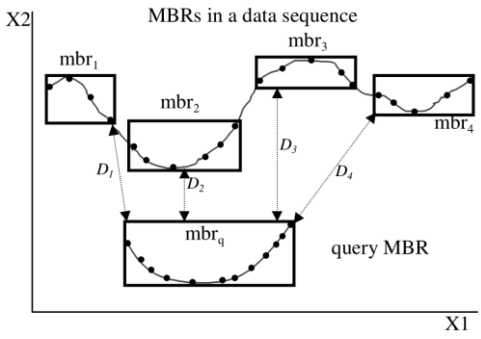

图 2.4 查询序列到数据序列的距离计算

第二步，第一次剪枝操作。筛选出数据库中满足公式（12）的数据序列 S_k。

其中 $mbr_i(Q)$是待查询序列 Q 的第 i 个 mbr，$mbr_j(S_k)$是数据序列 S_k 的第 j 个 mbr。

$$D_{mbr}\left(mbr_i(Q), mbr_i(S_k)\right) \le \varepsilon \tag{12}$$

第三步，第二次剪枝操作。从第二步得到的数据序列集合中筛选出满足公式（13）的数据序列 Sk 及 Sk 的满足条件的子数据段。

$$D_{norm}\left(mbr_i(Q), mbr_i(S_k)\right) \le \varepsilon \tag{13}$$

虽然本方法解决的是视频流的搜索问题，但是将视频流的各个属性看做多维度的时间序列数据来解决该问题，但是算法的主体部分仍然是对多维数据序列的相似性查询，我认为可以将其应用到轨迹数据的相似性查询中去。该算法的好处有以下几点：1、该算法基于 MBR，而不是序列中的每个点，因此和类似于滑动窗口的顺序扫描算法相比，速度快且存储开销小。2、该算法用来查找数据库中和待查询序列相似的序列或者待查询序列的子序列，对于查询序列和数据序列的长度没有限制。

2.7 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[13]。

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 T_a 和 T_b，轨迹格式为 $T=\{(l_1, v_1, t_1), (l_2, v_2, t_2),\ldots(l_m, v_m, t_m)\}$，其中 $l_i=(lg_i, la_i)$表示样本点坐标，vi 表示对象在 ti 时刻的移动速度。我们用 $d_a(lai,Tb)$表示从样本点 l_{ai} 到轨迹 T_b 的路网距离。l_{ai} 到 T_b 路网距离指，T_b 上距离 l_{ai} 最近的点到 l_{ai} 的路径距离。用 D_G 表示路网 G 的直径。在不同条件下，对图 G 中的轨迹进行相似性查询，我们有不同的距离函数。

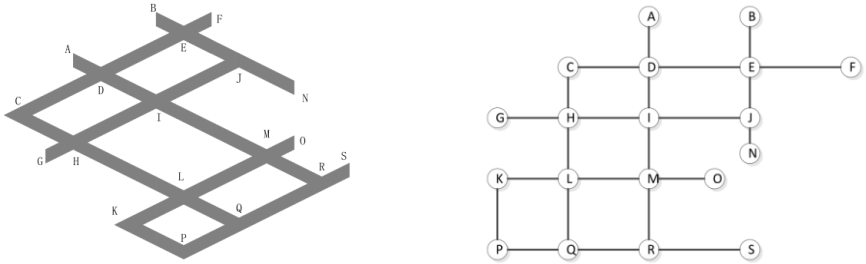

图 2.5 将道路转化为路网

2.7.1 仅考虑空间位置的轨迹距离

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。

$$d_N(T_a,T_b)=\frac{1}{m}\sum_{i=1}^{m}\frac{d_a(l_{ai},T_b)}{D_G} \tag{14}$$

2.7.2 带权值的轨迹距离

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。

$$d_{NW}(T_a,T_b)=\frac{1}{m}\sum_{i=1}^{m}\frac{w_{ai}d_a(l_{ai},T_b)}{D_G} \tag{15}$$

2.7.3 带速度信息的轨迹距离

当需要对路况信息或者交通拥堵信息进行分析时，轨迹的实时速度信息就十分重要了。令 S_G 代表当前道路最高限速，ds(lai,lbi)代表轨迹 A 上的第 i 个样本点 lai 与轨迹 B 上的第 i 个样本点 lbi 的速度之差，其中 lbi 是轨迹 B 上到 lai 最近的样本点。带速度信息的轨迹距离公式如式（16）所示。

$$d_{NWS}=\frac{1}{m}\sum_{i=1}^{m}\frac{w_{ai}d_a(l_{ai},T_b)}{D_G}\frac{d_s(l_{ai},l_{bi})}{S_G} \tag{16}$$

2.7.4 带时间信息的轨迹距离

分析路况信息时，时间信息也是一个很重要的因素。我们令$|d_t(l_{ai},l_{bi})|$代表轨迹 A 上的第 i 个样本点 lai 与轨迹 B 上的第 i 个样本点 lbi 的时间之差，其中 lbi 是轨迹 B 上到 lai 最近的样本点。la1 是轨迹 Ta 的第一个点，lam 是轨迹 Ta 的最后一个点，lb1 是轨迹 B 上距 la1 最近的点，lbm 是轨迹 B 上距 lam 最近的点。带时间信息的轨迹距离公式如式（17）所示。

$$d_r=\frac{1}{m}\sum_{i=1}^{m}\frac{|d_t(l_{ai},l_{bi})|}{max\{|d_t(l_{am},l_{b1})|,|d_t(l_{a1},l_{bm})|,|d_t(l_{am},l_{a1})|,|d_t(l_{bm},l_{b1})|\}} \tag{17}$$

2.7.5 各种因素结合的距离计算

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w_{NW}和w_r分别代表对应的子计算方法的权值参数，并且两个$w_{NW}+w_r=1$。如果将速度

信息加入公式，公式如式（19）所示。

$$d_{NWT} = d_{NW}(T_a, T_b) + w_r d_T \tag{18}$$

$$d_c = w_{NWS} d_{NWS}(T_a, T_b) + w_r d_r \tag{19}$$

2.7.6 小结

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。

2.8 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。

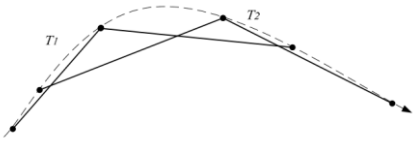

图 2.6 不同采样方法造成的样本点错位

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[14]。

2.8.1 点段距离

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$dist_{ps}$

为 p1 到轨迹 S 的点段距离。对应的距离公式如式（20）所示。

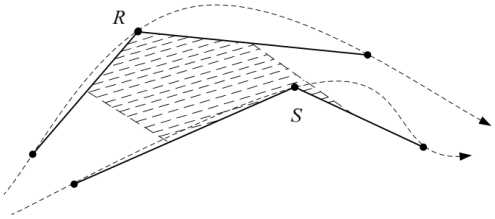

图 2.7 使用轨迹段面积代替点点距离

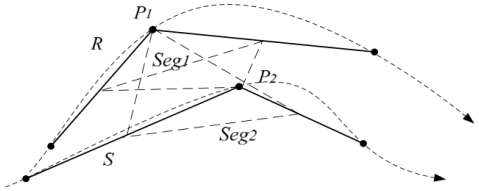

图 2.8 三角形面积正比于阴影面积

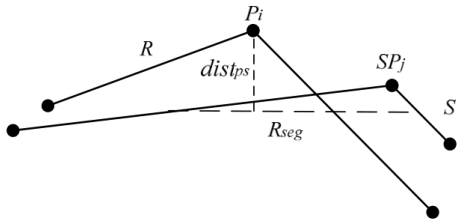

图 2.9 使用三角形的高代替三角形面积

$$dist_{ps}(P_1, R_{seg}) = \begin{cases} \sqrt{(x_i - x_{mid1})^2 + (y_i - y_{mid1})^2} & if \ \ r \le 0 \\ \sqrt{(x_i - x_{mid2})^2 + (y_i - y_{mid2})^2} & if \ \ r \ge L_{seg} \\ \sqrt{(x_i - dx)^2 + (y_{mid1} - dy)^2} & otherwise \end{cases} \tag{20}$$

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

2.8.2 预测距离

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

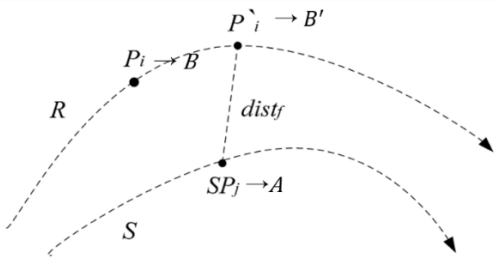

图 2.10 预测 B 在与 A 相同时刻的位置B′

然后利用轨迹 R 在 ti-1 时刻的位置和 ti-1 与 ti 间的平均速度，来预测 tj 时刻的位置，得到P_i'，$P_i' = (x_{B'}, y_{B'})$的预测距离计算公式如式（21）所示。

$$\begin{cases} x_{B'} = x_{i-1} + v_i^x (t_B + \Delta t - t_{i-1}) \\ y_{B'} = y_{i-1} + v_i^y (t_B + \Delta t - t_{i-1}) \end{cases} \tag{21}$$

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

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

$$dist_t(A,B) = dist(A,B') \tag{22}$$

2.8.3 段段距离

图2.11中，融合点段距离$dist_P(P_i, SP_j)$和预测距离$dist_t(P_i, SP_j)$，我们可以得到对应样本点P_i和SP_j之间的时空距离公式，如式（23）所示，其中t是时间距离的敏感参数，值越大表示时间距离越重要。

$$dist_{st}(P_i, SP_j) = dist_P(P_i, SP_j) + t \times dist_t(P_i, SP_j) \tag{23}$$

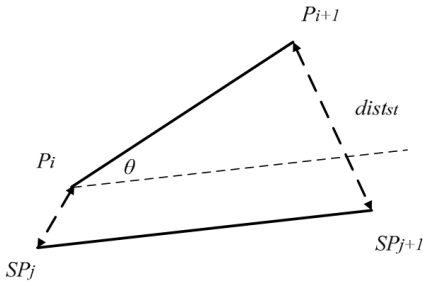

图2.11 利用夹角计算形状相似性

利用样本点之间的时空距离$dist_{st}(P_i, SP_j)$，可以得到该样本点和后一个样本点形成的轨迹段之间的距离$dist_{st}(S_i, SS_j)$，即段段距离，来计算轨迹段S_i和轨迹SS_j之间的形状相似程度，如式（24）所示，其中S_i是P_i与P_{i+1}之间的轨迹段，SS_j是SP_i与SP_{i+1}之间的轨迹段。

$$dist_{st}(S_i, SS_j) = dist_{st}(P_i, SP_j) + dist_{st}(P_{i+1}, SP_{j+1}) \tag{24}$$

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θ是两条轨迹段之间的夹角，θ和$f(\theta)$的计算公式如式（26）和式（27）所示

$$dist_s(S_i, SS_j) = f(\theta) dist_{st}(S_i, SS_j) \tag{25}$$

$$\theta = |arctan2(y_{i+1} - y_i, x_{i+1} - x_i) - arctan2(y_{j+1} - y_j, x_{j+1} - x_j)| \tag{26}$$

$$f(\theta) = \frac{dist_{smid}(S_i, SS_j)}{dist_{max}(R,S)} \times (\omega + \theta) \tag{27}$$

2.8.4 轨迹相似度函数 SDTW

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

$$SDTW(R,S)=\begin{cases}0, & if\ \ n=0\ \ and\ \ m=0\\ \infty, & if\ \ n=0\ \ or\ \ m=0\\ dist_s\big(Head(R),Head(S)\big)+\min\begin{cases}SDTW\big(T,Rest(S)\big)\\ SDTW\big(S,Rest(T)\big)\\ SDTW\big(Rest(T),Rest(S)\big)\end{cases}\end{cases} \tag{28}$$

2.8.5 小结

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，提高结果精度。

2.9 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[15]来找出带预测用户的相似用户组，以此支撑社会传染理论的运行，

根据用户在某地活动消耗的的时间比仅仅路过该地的时间长，并减少相似性计算的复杂度，我们将完整的轨迹分解成小轨迹。分解条件就是用户u_i的时间跨度$t_i > T_{cut}$，T_{cut}是一个时间阈值。分解后得到用户u_i的小轨迹集$Tr_i = \{trace_1, trace_2, \dots trace_n | trace_i \in \mathrm{L}, 1 \le \mathrm{i} \le \mathrm{n}\}$。我们可以得到所有用户的小轨迹集$Tr_{total} = Tr_1 \cup Tr_2 \cup \dots \cup Tr_n$。

然后两个小轨迹集之间的重合程度来计算他们的相似程度。令$V_i = \{v_1, v_2, \dots v_m\}$代表用户$u_i$的布尔型向量，$v_i$的值如式（29）所示。

$$v_i = \begin{cases} 1, & ith\ \ trace \in\ \ Tr_i \\ 1, & otherwise \end{cases} \tag{29}$$

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

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

2.10 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[16]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。

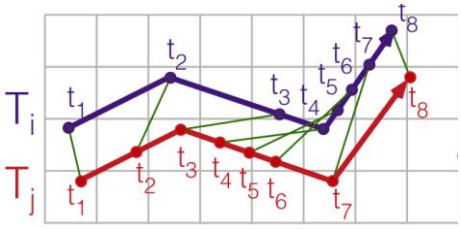

图 2.12 按时间匹配点对

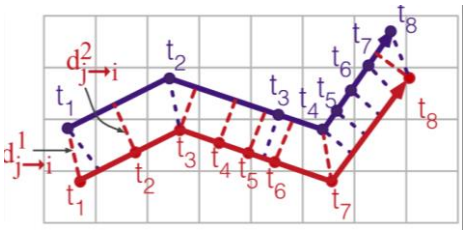

图 2.13 按点到轨迹段最小距离匹配

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$Dist_{PT}(p_i^k, T_j)$，其计算公式如式（30）所示。

$$Dist_{PT}(p_i^k, T_j) = min_{l \in T_j} Dist_{PL}(p_i^k, l) \tag{30}$$

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。

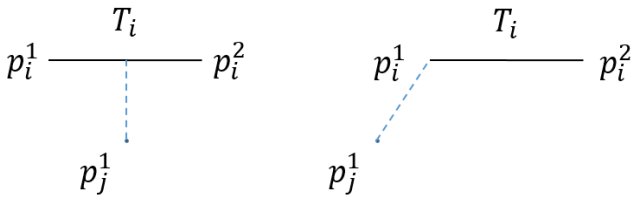

图 2.14 点到轨迹段的距离

BDS 的计算公式如式（31）所示，其中$d_{i\to j}^k$是一个归一化的距离，最终结果$SIM(T_i,T_j)$是一个介于 0 到 1 之间的值，表示轨迹T_i和轨迹T_j之间的相似程度，值越大，表示两条轨迹越相似。

$$SIM(T_i,T_j)=1-\frac{\sum_{k=1}^{|T_i|}d_{i\to j}^k+\sum_{k=1}^{|T_i|}d_{j\to i}^k}{|T_i|+|T_j|} \tag{31}$$

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

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

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

3、全文总结

本文中，我们介绍了移动对象轨迹相似性度量相关的概念、成熟的技术和一些新颖的研究。

针对相似性模型的研究主要产生了以下几种相似性函数，包括 Lp-norms、DTW、LCSS、EDR 和 ERP 等，其中 Lp-norms 计算两条时间序列每个点对之间的 p 范数，然后累积求和，但是它不能处理局部时间偏移。LCSS 和 EDR 可以处理局部时间偏移，但是没有考虑计算时间序列之间的真实距离。ERP 结合了 Lp-

norms 和 EDR 这两种相似度函数，计算结果可以表示出时间序列间的距离，同时也可以处理局部时间偏移。此外，一些学者还提出了这些方法的改进方法以及一些创新方法。

随着大数据时代的到来，海量数据给相关方面的应用带来了前所未有的机遇，但同时在数据管理、计算和挖掘方面的难度也大大提升了。而轨迹数据的应用前景非常广泛，对轨迹数据相似性的研究在数据查询方面占有重要地位，扩展轨迹数据的应用场景意义十分重大。

参考文献

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

[2]. Agrawal R, Faloutsos C, Swami A. Efficient similarity search in sequence databases[C]// International Conference on Foundations of Data Organization and Algorithms. Springer-Verlag, 1993:69-84.

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

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

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

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

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

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

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

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

[11]. Chen L, Ng R. On The Marriage of Lp-norms and Edit Distance[C]// Thirtieth International Conference on Very Large Data Bases. VLDB Endowment, 2004:792-803.

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

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

[14]. Mao Y, Zhong H, Xiao X, et al. A Segment-Based Trajectory Similarity Measure in the

Urban Transportation Systems [J]. Sensors 2017. 524-540.
[15]. Liu Z, Hu L, Wu C, et al. A novel trajectory Similarity-Based approach for location prediction[J]. International Journal of Distributed Sensor Networks, 2016, 12(11).
[16]. Na T, Li G, Xie Y, et al. Signature-Based Trajectory Similarity Join[J]. IEEE Transactions on Knowledge & Data Engineering, 2017, 29(4):870-883.