Trajectory Data Mining: a Novel Distance Measure

Ayman Al-Serafi
Business Intelligence & Enterprise Data Warehousing
Teradata Corporation
E-mail: ayman.al-serafi@teradata.com

Ahmed Elragal
Department of Information Systems
German University in Cairo, GUC
E-mail: ahmed.elragal@guc.edu.eg

Abstract— **There is currently an increasing availability of large spatiotemporal datasets. Sequences of spatiotemporal data or paths, also known as trajectories, can be captured by modern technology and stored in moving-object databases (MOD) or a trajectory data warehouse. It is a common challenge for knowledge discovery within MODs to query proximities and distances, e.g. in clustering trajectories. Previously adopted distance measures focus on the complexity of geometric and/or mathematical models of trajectories, while ignoring several aspects common to all spatiotemporal trajectories, e.g. direction, distance covered, and duration. This research introduces a more comprehensive approach for trajectory distance measurement in spatiotemporal applications. The approach is simplified, yet novel, introducing a new set of dimensional variables, therefore called the Multi-Dimensional Trajectory Distance Measure (MTDM). The accuracy and relevance of MTDM is evaluated in experiments using multiple proximity metrics, for example MTDM based on Euclidean proximity calculation. A geospatial data analysis framework is utilized in the experiments. Efficiency evaluation of MTDM showed the feasibility of applying the measure to various trajectory datasets.**

Keywords-distance measurement; geospatial data analysis; moving-object databases; spatiotemporal data; trajectory data mining.

I. Introduction

There is currently an increasing availability of large spatiotemporal datasets. Data about movements and trajectories of objects are commonly captured using technology like Global Positioning System (GPS), Radio Frequency Identification (RFID) and Global System for Mobile communications (GSM) [8, 17]. The trend of increasing spatiotemporal data has been also supported by the advances in database management systems (DBMS) which support such kind of data. This can be seen with the increasing adoption of geospatial and temporal capabilities in DBMS which can support Mobile Objects Databases (MOD) [10, 14, 15].

Spatiotemporal data consists of a collection of points which have location and temporal references. Temporal references of spatiotemporal data are commonly denoted by a date and an instance on the world time system (like GMT). Meanwhile, location references from spatiotemporal points are commonly stored as spatial references (X and Y references on the Cartesian coordinate system), or as geospatial references consisting of a location on Earth referenced by a geodetic system like the World Geodetic System (standard format for GPS location references).

A trajectory is a path consisting of an ordered set of spatiotemporal points [8, 16]. This can be defined for any trajectory 'T' which can be seen as an ordered set of spatiotemporal points consisting of 3 dimensions: location in terms of x-coordinate 'x', y-coordinate 'y' and temporal dimension in terms of time 't'. This is formally defined as $T = \{(x_1, y_1, t_1), (x_2, y_2, t_2), \dots, (x_n, y_n, t_n)\}$ and can be seen in Figure 1.

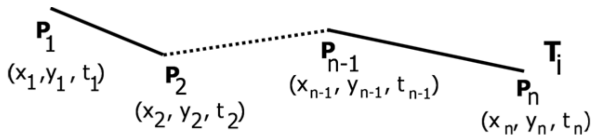


Figure 1. Formal representation of a trajectory consisting of 'n' points.

After storing trajectories in MODs, or a trajectory data warehouse (TDW), data mining is frequently used to analyse and to find knowledge within this data. Data mining involves multiple techniques like clustering, association rules and outlier detection. For techniques like clustering, proximity measurements are frequently used [8, 13, 16]. Proximity measurements can be also used to find outliers in datasets [4]. Examples of outliers include fraudulent transactions in e-commerce. Proximity measurements can also be used for classification using the nearest neighbour techniques [18] and trajectory similarity [14].

The aim of this paper is to develop a novel measurement of distance (proximity) between trajectories for spatiotemporal applications, and to test this distance measurement within a geospatial data analysis framework. The paper is organised as follows: in Section II a background about the research problem and proximity measures is provided, in Section III the Multi-Dimensional Trajectory Distance Measure (MTDM) is introduced, Section IV analyses the efficiency of our algorithms, Section V tests our approach in a series of experiments utilising the geospatial framework followed by a discussion in Section VI, and finally Section VII concludes the research study and advises on future research.

II. Problem Statement

Proximity (or similarity) can be defined as the degree of how close, or alike, are two instances [4]. Similarity coefficients measure the relationship between two individual items based on a number of variables. Distance, on the other hand, is seen as the degree of differences between instances based on specific dimensions or variables [16]. Different proximity metrics exist to calculate the distances between different instances under study. This includes Euclidean [12, p. 21], City Block [12, p. 21], Canberra [12, p. 21], and Chebyshev [11].

When calculating proximity between all instances the result is summarised in a proximity matrix [4]. The proximity matrix is a symmetric matrix with the diagonal equal to 0 when it is a distance measure. The number of unique elements for the matrix 'N' (which need to be calculated on one side from the diagonal; the other side is neglected as it is a mirror of the same values -similar to any symmetric matrix) is given by equation 1, where 'n' is the number of instances [4]. The proximity matrix compares all instances with each other. For distances the proximity matrix of trajectories is defined by the matrix in Figure 2.

$$N = \frac{n(n-1)}{2} \tag{1}$$

Previously researched measurements of trajectory distances include: the Euclidean distance, the Longest Common Sub-Sequences (LCSS) distance, Dynamic Time Warping (DTW) and edit distances like real sequences (EDR) and real penalty (EDP) [6, 8, 9, 13, 15].

$$\begin{bmatrix} & T_1 & T_2 & T_3 & \cdots & T_n \\ T_1 & 0 & & & & \\ T_2 & d_{(2,1)} & 0 & & & \\ T_3 & d_{(3,1)} & d_{(3,2)} & 0 & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ T_n & d_{(n,1)} & d_{(n,2)} & d_{(n,3)} & \dots & 0 \end{bmatrix}$$

Figure 2. Trajectory distance proximity matrix.

Previously researched measurements focus on the time-series approaches and geometric complexities similar to [5]. These measurements consider the trajectory as a time-series of spatiotemporal points which puts the burden of some limitations like time-shift complexities (related to trajectory distance measurement at different segments, or sub-trajectories, and for different temporal windows between the trajectories segments compared). Another limitation can be the duration of the trajectories to be compared. If they are not equal it also puts some complexities on the algorithms. This can be seen in [5] where the efficiency of such algorithms for non-equal time-windows can have high complexities.

Simple Euclidean distance is also a frequently utilised distance function. An example can be seen in the MINDIST Euclidean function used for trajectory queries in [10]. Distance measures like DTW and simple Euclidean distance do not allow for some points in a trajectory to be unmatched, however other approaches like LCSS allow for this [9]. On the other hand, algorithms based on Euclidean approaches similar to that of [19] require trajectories to have equal length.

Edit distance algorithms like EDP measure the cost of transforming one of the compared trajectories to the other. EDP utilises real spatial distances [13]. Both of EDR and LCSS focus on the matching or non-matching characteristic of comparison points in two trajectories. A match is considered so compared to a user-defined threshold for the distance measurement [13]. EDR returns the number of transformations required to match the trajectories being compared [13]. The LCSS algorithm also returns a non-metric distance measurement [17].

Other unique distance measures were also proposed. For example, spatial distance between trajectories can be a function of the area of the two-dimensional regions between trajectory intersections. This approach is called Locality In-between Polylines (LIP) [15]. This has an advantage that it calculates sub-sequence similarity between trajectories. However, the approach is limited by assumptions of directional similarities and intersections between trajectories requiring additional workarounds [15].

Other trajectory distance measures also include [8] which allow calculating distances for non-overlapping trajectories in terms of temporal aspects. Their approach is a modified Euclidean distance of the spatial dimensions, where the modifications consist of temporal aspects of the trajectories. However, their approach only considers spatiotemporal aspects related to privacy preservation. The work of [9] integrates temporal aspects of the trajectories compared into the distance measurement too. The shape of trajectories was also the focal interest of approaches like [19].

The majority of the state-of-the-art approaches consider (dis)similarities of geometric movements and proximity. The focus of those approaches is on the complexity of geometric and/or mathematical models of trajectories, which ignores several aspects common to all spatiotemporal applications like total distance covered by trajectories, duration of trajectories, minimum/maximum distances between trajectories, etc. In our research, we tackle the problem of finding trajectory proximities in geospatial space and taking the overall characteristics of trajectories into account. This approach was recommended for future research by multiple research studies like [5] which recommend using non-geometric and non-time-series approaches to find proximity between trajectories.

III. THE MULTI-DIMENSIONAL TRAJECTORY DISTANCE MEASURE (MTDM)

Most of the trajectory distance measurements analysed trajectory similarities based on matching sub-sequences of the trajectories in a manner similar to time-series mining approaches. Such approaches assume that trajectories are a series of fluctuations in space and over time. For example, [5] uses average Euclidean distance at specific time-windows to find proximity between trajectories. [7] can be referred to for a more detailed discussion about the time-series approaches. In addition, many of those approaches only consider sub-trajectories (segments of trajectories) instead of complete trajectories [5].

We argue that attention for the sub-sequence matching approaches from the time-series literature and which focus on the geometric (shapes) and mathematical (time-series theories) aspects neglect the specific properties of moving-object trajectories which are relevant to spatiotemporal applications. This includes properties like the duration of movement, the direction of movement, the locations of movements, etc. Therefore our approach will adopt the latter multidimensional approach examining moving-object trajectories. This was previously introduced by similar research studies which adopt the concept of applying trajectory properties to distance measures, for example [15] which introduced the speed and direction dimensions.

In our distance measurement we will apply this approach which uses distance metrics, like Euclidean and City Block, and which can handle the different data types of trajectory data. Multiple dimensions and factors of trajectories were recommended for computing proximities, such as spatial distance, trajectories' start and end points proximities, geometric movement patterns, and movement dynamics (like speed) [15, 16]. Such combination and special considerations was recommended by [16] and will therefore be considered in our approach.

The aim of our approach is to measure the distance between all trajectories in an MOD via an approximate, yet accurate, approach which focuses on the characteristics of trajectories. Approximation can be seen, for example, in that our approach does not consider the exact shape of the trajectories (only considers the overall occurrences of directionality, like northbound movement occurrences, relaxing the geometric assumptions) and the sequence of points does not matter too (relaxing the assumptions of time-series theories).

Approximation of trajectory distances is not a new concept and has been studied in previous research. For example, similar directional approximation based on the geographic orientations was also utilised in the study of [15]. Such approximation can improve the efficiency of the distance operators utilised in the algorithms without neglecting parts of the whole trajectory (similar to methods related to sub-trajectory distance measurements, like [5], which only seeks to find distance between trajectory segments and not the whole trajectory and which have some limitations concerning efficiency of the algorithms.) The sequence of points can be relaxed as the interest is not in the sequence patterns but rather on the complete trajectory proximity (with the shape dimension being captured by total directionality of the trajectory rather than the local directionality within trajectory segments or sequences).

Our approach will be called Multi-Dimensional Trajectory Distance Measure (MTDM) as it will involve a more comprehensive approach for trajectory distance measurement and which takes into account multiple characteristics of trajectories. MTDM can be defined for two trajectories T_i and T_j as a simplified function of the multiple dimensions as in equation 2. The MTDM distance function consists of multiple variables as described in Table 1.

$$MTDM_{T_i,T_j} = f\,(GeoDist,\quad Distance_{\Delta},\quad Direction_{\Delta},\quad Duration_{\Delta},\quad Min\_GeoDist,\quad Max\_GeoDist,\quad Min\_ExtDist,\quad Max\_ExtDist) \tag{2}$$

MTDM can be described as follows:-

1. Within each trajectory T, compare each point of the trajectory with the point succeeding it in order to calculate the direction of movement (see Figure 3), the duration of movement and the distance covered between both points. The output of this step is stored in a trajectory spatiotemporal table (see Trajectory_ST in the data model depicted in Figure 4.). In this table, each point in a trajectory is assigned an ID (ID_COL), an identifier of the object being traced (Object_id), identifier for the trajectory (trajectory_id), and the date, time and geospatial location (mov_Date, mov_time, geo_location respectively). The calculated distance of movement, duration of movement, and direction of movement are stored as geo_distance, duration and geo_direction respectively.

TABLE I. MTDM DIMENSIONS DESCRIPTION.

MTDM Dimension	Description	Detailed Step(s)
GeoDist	Average geographic distance measurement between two trajectories.	1. Find the smallest distance between each point of trajectory T_i with all points in T_j (and vice versa). 2. Average all the geographic distances calculated from step 1 and for each of the two trajectories. 3. *GeoDist* is finally assigned as the smallest average from the two calculated in step 2
Distance$_{\Delta}$	The difference between the total distances covered between the two trajectories.	• Find the absolute difference between total distances covered by the two trajectories
Direction$_{\Delta}$	The difference in directional movements between the two trajectories	• Find the absolute difference in directional counts for each of the 8 directions: N/ NE /E / SE /S/ SW/ W/ NW
Duration$_{\Delta}$	The difference between the total durations between the two trajectories.	• Find the absolute difference between durations of the two trajectories
Min_GeoDist	The minimum geographic distances between points in T_i and T_j	• Find the smallest distance between any point of trajectory T_i with any point in T_j (and vice versa).
Max_GeoDist	The maximum geographic distances between points in T_i and T_j	• Find the largest distance between any point of trajectory T_i with any point in T_j (and vice versa).
Min_ExtDist	The minimum geographic distances between the extreme (first and last) points of the trajectories.	1. Find the distance between the first and last point of trajectory T_i with the first and last point in T_j (and vice versa). 2. Assign Min_ExtDist as the smallest distance from step 1.
Max_ExtDist	The maximum geographic distances between the extreme (first and last) points of the trajectories	1. Find the distance between the first and last point of trajectory T_i with the first and last point in T_j (and vice versa). 2. Assign Max_ExtDist as the largest distance from step 1.

2. Calculate the sum of all distances and durations for each trajectory and store it into the Trajectory_Info table (see the data model in Figure 4) as distance and duration respectively. Count the number of occurrences for each direction and store it into the Trajectory_Info table (count the

directional orientation N/NE/SE/S/SW/W/NW for each trajectory.). The total number of points in the trajectory is also stored as “trajectory points number”.

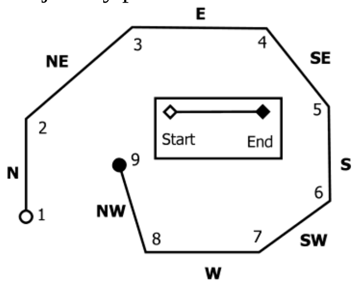
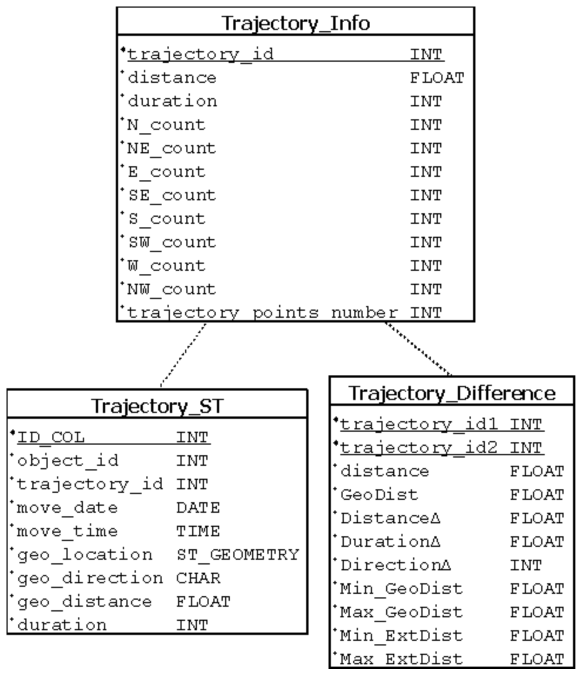


Figure 3. Calculating the direction of travel between each point and the point after it in a trajectory. For example, there is a positive change in both the x-axis and the y-axis between points 2 and 3 of the trajectory resulting in a direction of NE.



Trajectory_Info	
trajectory_id	INT
distance	FLOAT
duration	INT
N_count	INT
NE_count	INT
E_count	INT
SE_count	INT
S_count	INT
SW_count	INT
W_count	INT
NW_count	INT
trajectory points number	INT

Trajectory_ST	
ID_COL	INT
object_id	INT
trajectory_id	INT
move_date	DATE
move_time	TIME
geo_location	ST_GEOMETRY
geo_direction	CHAR
geo_distance	FLOAT
duration	INT

Trajectory_Difference	
trajectory_id1	INT
trajectory_id2	INT
distance	FLOAT
GeoDist	FLOAT
DistanceΔ	FLOAT
DurationΔ	FLOAT
DirectionΔ	INT
Min_GeoDist	FLOAT
Max_GeoDist	FLOAT
Min_ExtDist	FLOAT
Max_ExtDist	FLOAT

Figure 4. Trajectory MTDM data model.

3. Convert the direction counts of the trajectories (in table Trajectory_Info) into percentages by dividing the counts by the total number of points and multiplying by 100%. This normalises the directional counts relevant to the number of points in each trajectory so that trajectories with different number of points can be compared.

4. The differences between the trajectories’ characteristics (described in Table 1) are then calculated and stored in the Trajectory_Difference table (Figure 4). Figure 5 depicts the Min_ExtDist and Max_ExtDist dimensions. The geographical distances between points of trajectory Ti and trajectory Tj is calculated using the geospatial function ST_SphericalDistance (which calculates spherical geospatial distance, see [2] for a description of the function).

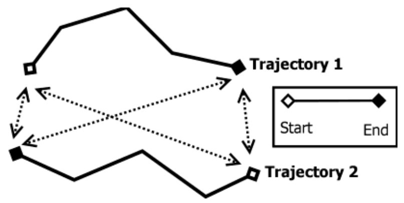


Figure 5. The first (trajectory start position) and last point (trajectory end position) of each trajectory is compared with the first and last point of the second trajectory. Four total comparisons (dotted-lines) are calculated.

5. Finally, standardise all the distance measurements ‘X_i’ in the Trajectory_Difference table by rescaling them between 1 and 100 using equation 3. Standardization to the distance measures is done by rescaling in order to make sure that there is no single (large) dimension dominating the distance measurement. This procedure is recommended so that the distance between two trajectories is invariant to spatial scaling and shifting.

$$Standard(X_i) = 1 + \frac{\left((X_i - \min(X)) * (100-1)\right)}{(\max(X) - \min(X))} \tag{3}$$

After completing the MTDM procedures and calculating specific characteristics and proximities between trajectories, the overall proximity can be calculated using multiple proximity metrics. The five proximity metrics we are going to use are modified Euclidean, City Block, Canberra, Chebychev, and an average proximity metric. The average proximity metric will also consider all of the dimensions as shown in equation 4. Average proximity metrics, like average Euclidean distance, was used in other trajectory similarity measures too [5].

$$MTDM_{average} = \frac{\begin{array}{c} GeoDist + Distance_{\Delta} \\ + Direction_{\Delta} + Duration_{\Delta} + Min\_GeoDist \\ + Max\_GeoDist + Min\_ExtDist + Max\_ExtDist \end{array}}{8} \tag{4}$$

IV. Efficiency Evaluation

MTDM can be decomposed into two main algorithms. *Algorithm 1* is used to calculate the distance, duration and direction between each point and the point after it for all points in each trajectory (step 1 of MTDM in Section III); and *Algorithm 2* is used to calculate the differences between all trajectories (step 4 of MTDM in Section III). Algorithm 1 runs in linear time O($t \cdot n$) where n is the number of points in each trajectory and t is the number of trajectories. Algorithm 2 runs in the polynomial time ($\frac{t(t-1)}{2} \cdot n^2$). The efficiency of algorithm 2 is based on comparing all trajectories together as expressed by equation 1, and within the comparison of every pair of trajectories there are quadratic steps executed on the points from both trajectories as described in step 4 of the MTDM approach (including finding the minimum, maximum and average distances between the trajectories- in Section III).

The overall big-O-notation is therefore O(n^2). The efficiency of MTDM is therefore comparable with other trajectory distance measures which run in O(n^2) polynomial time (with the exception of simple Euclidean distance which runs in O(n) linear time under the assumption that trajectories compared have equal number of points). The efficiencies of other algorithms from the time-series literature are: $O(n^2)$ for both EDP and DTW [13]. LCSS and EDR are non-metric and compare all points in the trajectories resulting into an efficiency of $O(n^2)$ too [13].

V. Experiments

In order to evaluate MTDM and its effectiveness with spatiotemporal applications a number of experiments are conducted. The first experiment tests the effectiveness of MTDM on a real dataset of spatiotemporal trajectories. In addition, experiment 1 compares MTDM with the simple Euclidean distance measurement as a benchmark. The simple Euclidean distance was calculated for two trajectories T_i and T_j as described in equation 5. The 'x' and 'y' are the coordinates of the i'th point of the trajectory (or j'th point of trajectory 2) retrieved using the standard ST_X and ST_Y methods in spatial databases and applied on the geo_location of each point from the Trajectory_ST table (Figure 4) –see [2]. The distance and duration are retrieved from the Trajectory_Info table and are used with the simple Euclidean distance in order to make it comparable with the MTDM approach (which also includes the duration and distance covered for trajectories. The simple Euclidean approach must also include those factors in-order to keep the evaluation non-biased towards the approach with the more representable variables).

$$\sqrt{\sum_{i,j=1}^{n} (x_i - x_j)^2 + (y_i - y_j)^2 + (distance_i - distance_j)^2 + (duration_i - duration_j)^2} \qquad (5)$$

We will not compare our MTDM approach with other time-series based approaches (like DTW or LCSS) because the shapes and geometric matching of trajectories is relaxed as described earlier in this paper. The main focus therefore is to assess the degree of relevance of the MTDM dimensions for assessing dissimilarity of trajectories in spatiotemporal applications.

Another experiment was conducted to test the accuracy of the MTDM distance measure as compared to simple Euclidean distance in experiment 2. Experiment 2 introduces a new technique for evaluating proximity measures which better differentiates the different measures and metrics evaluated as compared to experiment 1 (which uses classical evaluation techniques from previous literature).

For both experiments, we have used a subset of the trajectories dataset which consisted of 145 spatiotemporal trajectory traces from 2 school buses (moving-objects) transporting students within in the vicinity of the Athens metropolitan area, Greece (See [10] for more details about the dataset). The dataset was retrieved from [3].

The approach used in the experiments to execute the distance measurement on the sample datasets consisted of multiple steps and tools as summarized in the framework diagram (see Figure 6). First, the data is loaded into the Teradata TDW using Extract, Load and Transform (ELT) techniques. The original data was in the WGS 84 projection utilised in GPS. This format and multiple other projections are supported with Teradata Geospatial. To load spatiotemporal maps/GIS data from GIS/Map formats into the TDW we have used the TDGeoImport tool. The tool automatically converts the different spatiotemporal formats into the standard Teradata ST_Geometry format stored in the Teradata MOD / TDW. This allows for Teradata Geospatial functions to be used on the spatiotemporal data, which can also be combined with non-spatial data allowing for deeper and richer analytics. This data can also be visualized using data visualization tools by using the GeoServer geospatial data integration tool [1] or by direct SQL-querying over the ST_Geometry fields.

The distance measurement calculation then takes place using the Teradata Stored Procedures. The programming language used in implementing the algorithms and distance functions was in standard Teradata Database Stored Procedures (V13.0).

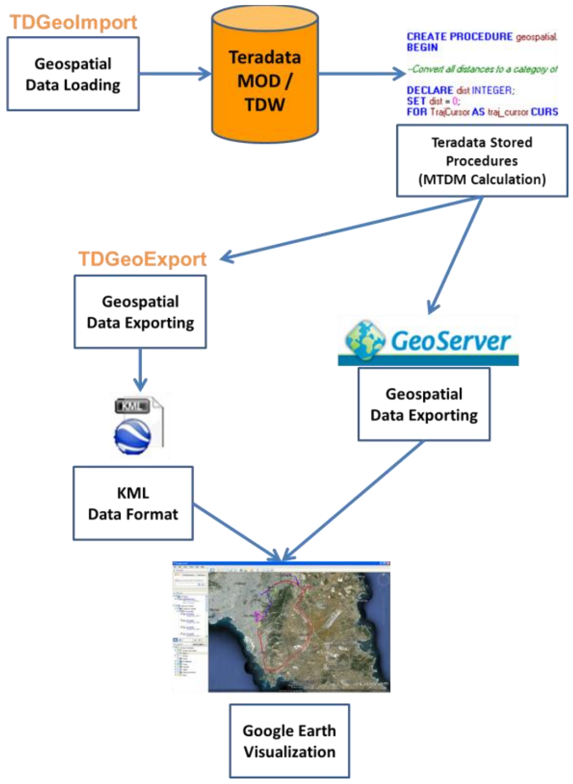
Following this, visualization takes place on Google Earth. There are two alternatives to achieve this goal (and both techniques were used). The first alternative uses GeoServer [1] middleware between the Teradata TDW and the Google Earth interface. GeoServer uses the Web Map Service (WMS) protocol. WMS is a standard protocol for serving geo-referenced map images over the Internet that are generated by a map server using data from a GIS database. GeoServer supports the Teradata Database (MOD). In this situation, GeoServer acts as the geospatial middleware connecting the client (Google Earth application) to the maps or geospatial data stored in the TDW. Alternatively, the data to be visualized can be exported by the TDGeoExport tool. This allows exporting the data to the Google Earth KML format, which can then be visualized.

A. *Experiment 1: Classification accuracy*

In the first experiment we test the alternative distance metrics applied within MTDM and compare it with the simple Euclidean distance. We use the "leave-one-out" approach which was used before in previous literature like [7] and [6]. In this approach the trajectories are already assigned to clusters and labelled. Using the distance measure to be evaluated, the nearest trajectory to each of the trajectories is found and the class label of the former trajectory is assigned to the latter trajectory. This is similar to the 1-nearest-neighbour classification approach which classifies an object to the same class-label as its nearest neighbour. If the assigned class matches the real (actual) class then it is a hit (accurate), otherwise it is a miss (inaccurate). The accuracy rate is then calculated as in equation 6.

For this experiment 78 trajectories were sampled from the original dataset and which consisted of 5850 points (75 points in each trajectory). The trajectories retrieved were selected so that they belong to one of 4 clusters (visualized in Figure 7): cluster 1 consisting of medium to long trajectories at areas remotely located from Athens city centre and shorter trajectories closer to the centre or NE area but with more jagged shapes (blue), cluster 2 consisting of trajectories of small to medium length trajectories with close proximity to

Athens city centre but with less jagged shapes (red), cluster 3 consisting of trajectories lying NE of Athens city centre and of small to medium length trajectories (green), and finally cluster 4 consisting of all longer trajectories located in remote regions around Athens (purple).


TDGeoImport
Geospatial Data Loading
Teradata MOD / TDW
Teradata Stored Procedures (MTDM Calculation)
TDGeoExport
Geospatial Data Exporting
KML
GeoServer
Geospatial Data Exporting
KML Data Format
Google Earth Visualization

```
CREATE PROCEDURE geospatial
BEGIN
--Convert all distances to a category of
DECLARE dist INTEGER;
SET dist = 0;
FOR TrajCursor AS traj_cursor CURS
```

Figure 6. The Geospatial Data Analysis Architecture.

This experiment resulted into absolute accuracy of 100% for all measures including the MTDM implementations (Euclidean, City Block, Canberra, Chebyshev, and average proximity metrics) and the simple Euclidean distance. In order to distinguish between the different measures another experiment was conducted which evaluates the accuracy of the distance measures on domain-expert labelled data (See experiment 2).

$$Accuracy\ Rate = \frac{No.of\ hits}{No.of\ hits + No.of\ misses} \tag{6}$$

B. *Experiment 2: Apriori-knowledge validation*

In the second experiment we compare the accuracy of the different proximity metrics utilised within MTDM and compare it to the simple Euclidean distance measurement. For this experiment the original dataset was sampled for 1386 points making up all 11 trajectories from two days (9th and 12th of February 2001) and resulting into 55 trajectory comparisons (as calculated for 'n' trajectories by equation 1). The duration of a trajectory was considered as a calendar day (trajectories were divided into sub-trajectories covering a single day). Those trajectories are depicted in Figure 8 (each individual trajectory visualized with a unique colour).

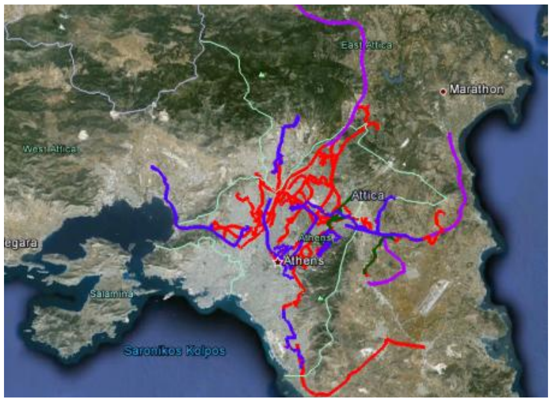


Figure 7. Visualization of the 4 trajectories' clusters used in experiment 1.

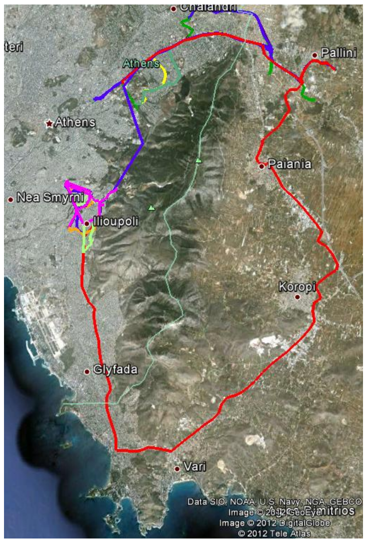


Figure 8. Visualization of the trajectories used in experiment 2.

We introduce a new approach for testing alternative distance measures based on expert apriori-knowledge evaluation. To test the accuracy of our approach, each of the distance measures was validated against a repository of pre-evaluated trajectory distance assigned by data mining experts. For our experiment we had 3 repositories of apriori-knowledge from 3 different data mining experts. The repositories identify the distance between two trajectories on a scale of four degrees of similarity {A, B, C, D}. Similarity of A would mean the two trajectories compared (from the expert's point-of-view) are most similar and similarity of D would mean least similarity. On the other hand, to convert the numerical MTDM / simple Euclidean distances 'd' to similar categories (as assigned in the knowledge repositories) we used the following conversions for the different ranges: ($1 \le d \le 25 \rightarrow$ A), ($26 \le d \le 50 \rightarrow$ B), ($51 \le d \le 75 \rightarrow$ C), ($76 \le d < \infty \rightarrow$ D).

The analysis and interpretation of human analysts is recommended to use in order to capture the embedded meanings within the data [16]. Human cognition is suitable for such classification problems because human knowledge and common-sense are utilised to capture what seems to be correct to human interpretation [16]. This apriori-knowledge testing-set will be used to evaluate the classification accuracy of MTDM using multiple alternative proximity metrics as compared to the simple Euclidean distance.

For each of the proximity measures we constructed a proximity matrix which compares all trajectories together similar to Figure 2. This totalled to 55 comparisons from the 11 trajectories as calculated by equation 1.The proximity measures were then converted to categories {A, B, C, D} similar to the approach used within the apriori-knowledge repositories. To validate the accuracy of each proximity measure with the benchmark of the human experts we constructed a confusion matrix which evaluates the degree of agreement in assigning (dis)similarity between two trajectories (see Figure 9). Full-agreement classifications are across the diagonal, namely AA, BB, CC, and DD. Those were assigned a coefficient of 100% accuracy. Partial agreement of a single degree difference (like classifying a distance between two trajectories as C instead of B) was assigned 66.67% classification accuracy. Two degree difference in classification agreement was assigned 33.33% accuracy while a three degree difference in classification agreement was assigned 0% accuracy.

MTDM utilising the alternative five proximity metrics was then evaluated along with the simple Euclidean distance. The resulting accuracy of the proximity measures (calculated for each proximity measure from its confusion matrix as in Figure 9) are summarised in Table 2 (in order of descending accuracy rates).

	Computed Classification
Actual Classification	$$\begin{array}{c} & A & B & C & D \\ A & 100\% * AA & 66.67\% * AB & 33.33\% * AC & 0\% * AD \\ B & 66.67\% * BA & 100\% * BB & 66.67\% * BC & 33.33\% * BD \\ C & 33.33\% * CA & 66.67\% * CB & 100\% * CC & 66.67\% * CD \\ D & 0\% * DA & 33.33\% * DB & 66.67\% * DC & 100\% * DD \end{array}$$

Figure 9. Confusion matrix for evaluating the trajectory distance measures in our experiments.

VI. Discussion

The In this paper, multiple proximity metrics and different combinations of dimensions from MTDM were tested in a spatiotemporal application. The results explain the capability of MTDM in accurately estimating the distance between trajectories. This was shown with its capability to correctly classify 100% of the instances in experiment 1 and by the overall accuracy rate ranging between 72.12% (with the MTDM Canberra metric) and 80.61% (with the MTDM Euclidean metric) in experiment 2. It must be noted that the top scoring proximity measures in experiment 2 were based on the MTDM approach which outperformed the simple Euclidean approach. The best MTDM distance metric was Euclidean as expected from the literature [7]. The average metric came second which can mean that all of the MTDM dimensions proposed had equal weights in the evaluation of trajectory distance and are all important as validated against the apriori-knowledge repositories. This was equally followed by the City Block MTDM implementation which had the same accuracy rate as the average MTDM implementation.

On the other hand, the Chebychev implementation of MTDM didn't perform as well as the Euclidean, City Block and average implementations because it takes the maximum distance making it highly sensitive to noise (or peaks in the variables) [7]. The Canberra implementation didn't perform as well as the other implementations of MTDM too. This is probably due to the high dimensionality of MTDM which can affect the Canberra distance. This is mainly because the Canberra distance assumes there is an origin (of expected values) and with values scattered around the origin. This was not the case for our trajectories experiment. In our experiments, the trajectories had random movements not scattered around a specific origin.

As for the simple Euclidean measurement, it came lagging behind all the MTDM implementations. This indicates the inadequacy of the simple Euclidean approach to estimate proximity of trajectories as human experts would expect. Alternatively, the MTDM implementations had considerable improvements in approximating the proximities of trajectories indicating the importance of the dimensions proposed in MTDM. Overall, the experiments indicate that the best proximity metric to use with MTDM is the Euclidean measure.

Concerning the correlation between the dimensions in MTDM, it was observed (using the dataset in experiment 1) that the absolute Pearson correlation coefficients between variables are negligible/small as they range between 0.0026 and 0.2192. The exception was the distance metrics relying on geospatial distance (GeoDist and ExtDist distances) which have high correlation ranging between 0.9536 and 0.9984 (as expected because they use the geospatial distance as a common base-metric). It was decided to keep all geospatial distance metrics for the experiments to maintain a comprehensive approach for MTDM, however, alternatively the variables relying on geospatial distance metrics can be used interchangeably according to application requirements.

There were some limitations with our approach and experiments. As for the approach, the approximation of the directional properties to the 8 compass directions could be considered as a limitation, however, we consider it an advantage which saves calculation costs of computing exact degree-based differences. Concerning the experiments, the accuracy rates were estimated but they could be largely affected by the reliability of the human approximations (for experiment 2).

VII. Conclusion

This research paper introduced a more comprehensive approach for trajectory distance measurement in spatiotemporal applications called MTDM. The approach is simplified, yet novel, introducing a new set of dimensional variables. MTDM can be used in trajectory data mining techniques requiring trajectory proximity measurement like trajectory clustering or similar trajectory querying.

TABLE II. THE ACCURACY RATES OF THE PROXIMITY MEASURES IN EXPERIMENT 2.

Proximity Measure	Accuracy Rate
MTDM (Euclidean based)	80.61%
MTDM (Average based)	75.15%
MTDM (City Block based)	75.15%
MTDM (Chebychev based)	72.73%
MTDM (Canberra based)	72.12%
Simple Euclidean	58.18%

MTDM is a specialised trajectory distance measure for spatiotemporal (geospatial) applications which does not hold presumptions about the relationship between compared trajectories. This gives the approach its flexibility towards comparing all sorts of spatiotemporal trajectories (mainly involving geospatial types). MTDM also has multiple advantages including: capability to adapt the distance measurement to the relative scale of the problem (using the standardisation steps of the approach) and the capability to approximate trajectory distance in a comprehensive approach (utilising the multiple MTDM dimensions). MTDM can also compare different length trajectories and is not limited by the requirement to have equal number of points within the trajectories compared. Efficiency evaluation of MTDM showed the feasibility of applying the measure to various trajectory datasets as was tested in this research paper.

To test MTDM, a geospatial data analysis framework was utilised. This led to the effective visualization of geospatial data (moving-object trajectories) for the data mining experts as was required for the apriori-knowledge validation experiments.

For future research, improvement to the efficiency of the algorithms with spatial indexes can be tested. In addition, the robustness of the algorithm to noise can be tested as it is expected for the MTDM approach to behave well under “noisy” conditions. Testing alternative weighting of the dimensions of MTDM can also contribute to the theoretical foundation as this paper assumed equal weights of the dimensions. Integrating semantics from different spatiotemporal applications into the multi-dimensional distance measure can also be analysed in case-studies which can demonstrate the capabilities of MTDM in handling multiple spatiotemporal domains.

ACKNOWLEDGMENT

This research study was funded by the research and development fund from Teradata Corporation, Egypt. Experiments were conducted within the laboratories of the German University in Cairo.

REFERENCES

[1] GeoServer Software. (2012). Retrieved March 31, 2012, from GeoServer: http://www.geoserver.org

[2] *SQL Geospatial Types.* (2010). Retrieved March 1, 2012, from Teradata Website: www.info.teradata.com/edownload.cfm?itemid=102320050

[3] Trajectory Datasets. (2012). Retrieved March 31, 2012, from R-Tree Portal: http://www.rtreeportal.org

[4] Badawy, S. A., Elragal, A., & Gabr, M. (2008). Multivariate Similarity-Based Conformity Measure (MSCM): an Outlier Detection Measure for Data Mining Applications. AIA '08 Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications, (pp. 314-320).

[5] Buchin, K., Buchin, M., van Kreveld, M., & Luo, J. (2009). Finding Long and Similar Parts of Trajectories. Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems GIS, (pp. 296-305).

[6] Chen, L., Özsu, M. T., & Oria, V. (2005). Robust and Fast Similarity Search for Moving Object Trajectories. Proceedings of the 2005 ACM SIGMOD international conference on Management of data, (pp. 491-502).

[7] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X., & Keogh, E. (2008). Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. Proceedings of the VLDB Endowment, 1(2), 1542-1552.

[8] Domingo-Ferrer, J., Sramka, M., & Trujillo-Rasúa, R. (2010). Privacy-preserving Publication of Trajectories Using Microaggregation. SPRINGL '10 Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS , (pp. 26-33).

[9] Frentzos, E., Gratsias, K., & Theodoridis, Y. (2007). Index-based Most Similar Trajectory Search. IEEE 23rd International Conference on Data Engineering, ICDE 2007, (pp. 816 - 825).

[10] Frentzos, E., Gratsias, K., Pelekis, N., & Theodoridis, Y. (2005). Nearest Neighbor Search on Moving Object Trajectories. Lecture Notes in Computer Science, 3633/2005, 328-345.

[11] Funes, A., Ferri, C., Hernández-Orallo, J., & Ramírez-Quintana, M. J. (2009). An Instantiation of Hierarchical Distance-based Conceptual Clustering for Propositional Learning. Lecture Notes in Computer Science, 5476/2009, 637-646.

[12] Gordon, A. D. (1981). Monographs On Applied Probability And Statistics Classification. Cambridge: The University Press.

[13] İnan, A., & Saygın, Y. (2006). Privacy Preserving Spatio-Temporal Clustering on Horizontally Partitioned Data. *Lecture Notes in Computer Science, 4081/2006*, 459-468.

[14] Panagiotakis, C., Pelekis, N., & Kopanakis, I. (2009). Trajectory Voting and Classification Based on Spatiotemporal Similarity in Moving Object Databases. *Lecture Notes in Computer Science, 5772/2009*, 131-142.

[15] Pelekis, N., Kopanakis, l., Ntoutsi, I., Marketos, G., & Theodoridis, Y. (2007). Mining Trajectory Databases via a Suite of Distance Operators. IEEE 23rd International Conference on Data Engineering Workshop, (pp. 575 - 584).

[16] Rinzivill, S., Pedreschi, D., Nanni, M., Giannotti, F., Andrienko, N., & Andrienko, G. (2008). Visually–driven analysis of movement data by progressive clustering. Information Visualization, 7(3-4), 225-239.

[17] Vlachos, M., Gunopulos, D., & Kollios, G. (2002). Robust Similarity Measures for Mobile Object Trajectories. Proceedings of the 13th International Workshop on Database and Expert Systems Applications, (pp. 721 - 726).

[18] Vlachos, M., Kollios, G., & Gunopulos, D. (2002). Discovering similar multidimensional trajectories. Proceedings of the 18th International Conference on Data Engineering , (pp. 673 - 684).

[19] Yanagisawa, Y., Akahani, J.-i., & Satoh, T. (2003). Shape-Based Similarity Query for Trajectory of Mobile Objects. Lecture Notes in Computer Science, 2574/2003, 63-77.