Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

A comparative study of similarity evaluation methods among trajectories of moving objects

Nehal Magdy *, Tamer Abdelkader, Khaled El-Bahnasy

Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt

ARTICLE INFO

Article history:
Received 3 March 2017
Revised 19 February 2018
Accepted 19 March 2018
Available online 26 March 2018

Keywords:
Trajectory similarity
Regression
Interpolation
Barcoding
TWEDistance

ABSTRACT

Evaluating similarity between moving objects' trajectories has gained much attention in many application domains. There exist similarity measures in the literature that propose evaluating similarity between trajectories in the form of time stamped values. Their main drawback is that the similarity evaluation is affected by the different sampling rates as it is defined over sequences of time stamped values. One of these measures is the Time Warp Edit distance measure that was our base of measuring similarity in our TWEDistance operator. Therefore, in this paper, a comparative study is made between four different approaches: TWEDistance operator, regression, interpolation and curve barcoding. Similarity evaluation is made over trajectories of different sampling rates. Results show that interpolation achieves the highest accuracy compared to the other approaches with an average accuracy up to 90%. Experimental evaluation was made over synthetic and real datasets.

© 2018 Production and hosting by Elsevier B.V. on behalf of Faculty of Computers and Information, Cairo University. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

There has been a tremendous growth in movement data due to availability of devices that could be used to track movement of objects. Moving objects are objects that change their location/value over time such as cars, stock shares, temperature and animals [1]. Tracking their movement leads to a sequence of time stamped values/locations known as trajectories. Trajectories hold the historical movement of an object and it can be expressed as a sequence of positions $[l_1, l_2, \cdots, l_m]$ observed at discrete time instances $[t_1, t_2, \cdots, t_m]$ so a trajectory T can be expressed as follows $T = [(l_1, t_1), (l_2, t_2), \cdots, (l_m, t_m)]$ where m is the length – number of observations – of T [2,3].

Evaluating similarity between trajectories is important for many application domains. For example, analyzing trajectories of customers in a supermarket to find similar movement patterns for a better management of products. Another example is finding frequent migration patterns of a migrating group of birds. In surveillance systems, it is possible to find suspicious object movements and rare trajectory patterns. Other domains such as stock and data analysis, recommender systems of travelling routes, recommender systems for friends based on their interests, visited locations, likes, etc. and future prediction of phenomena such as storms, hurricanes and earthquakes [3,4].

Evaluating similarity between trajectories in their discretized form as a set of (x,y) pairs are affected by the sampling rate differences. Applying discrete similarity measures on a trajectory and a resampled version – same trajectory with different sampling rate – of it may result a distance that is bigger than the distance between two truly different trajectories. While what is expected, is that a trajectory with its all resampled forms shall be the head nearest ones with the minimum distance (i.e. the same trajectory with different sampling rate (each 2, 4 and 6 sec for example). Therefore, this paper handles the sampling rates issue based on the continuous form of a trajectory's data. Three different approaches were used, which are regression, interpolation and curve barcoding.

Regression is used to predict an equation that best fits the trajectories' points with the least error. For evaluating similarity between two trajectories R and S, R's movement is approximated by a function $f(n)$ with minimum residual error RSS_R. Then, how much the $f(n)$ approximates S's trajectory is evaluated with a minimum error RSS_S. Regression similarity is then equal to the absolute difference between RSS_S and RSS_R. Three types of regression are

* Corresponding author.
E-mail addresses: nehalmagdy@cis.asu.edu.eg (N. Magdy), tammabde@cis.asu.edu.eg (T. Abdelkader), khaled.bahnasy@cis.asu.edu.eg (K. El-Bahnasy).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.eij.2018.03.001
1110-8665/© 2018 Production and hosting by Elsevier B.V. on behalf of Faculty of Computers and Information, Cairo University.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

considered: Linear, Logarithmic and polynomial regression of degree up to 10.

Using interpolation, the objective is to estimate the value of the function that the data points represent for an intermediate value. For evaluating similarity between two trajectories R and S, R's movement is divided into intervals and for each interval a function is generated. Then, the points of trajectory S are interpolated in R's functions. Interpolation similarity is then equal to the Euclidean distance between the interpolated values and the original values of S. Two types of interpolation are used, Linear and Cubic Interpolation.

In curve barcoding, a trajectory is represented as an image and this image is converted into a binary barcode sequence and the Hamming distance – the number of positions at which the corresponding elements are different – is used to evaluate similarity between two trajectories' barcodes.

The Experiments are based on the previously mentioned approaches in addition to a discrete similarity operator [5] – TWEDistance – that is based on Time Warp Edit Distance (TWED) [6]. Similarity evaluation was based on different similarity meanings which are spatial proximity, speed and direction. The experiments were made on synthetic and real datasets. The results show that interpolation gives the highest accuracy results over the other approaches. So our contribution is twofold:

- Proposing generic continuous trajectory similarity measures based on regression, barcoding and Interpolation.
- A Comparative study between these continuous based approaches and our TWEDistance operator that is proposed in [5], by evaluating the measures through a set of experiments over different datasets to show the accuracy of the approaches with the existence of different sampling rates.

The rest of the paper is organized as follows: In Section 2, the background and the related work are presented. In Section 3, a detailed discussion of the TWEDistance operator is given. In Section 4, three approaches which are regression, interpolation and curve barcoding and how they are utilized to evaluate the similarity are discussed. Then, the experiments, and the output results are shown in Section 6. Finally, Section 7 concludes the paper, and sets ideas for future research work.

2. Related work

One of the important topics related to moving objects is how to evaluate the similarity between their trajectories [3,7]. The work in [8] is based on spatial raw representation where trajectories' fixes are aligned at the same positions, trajectories must have same number of fixes, local time shifting[1] is not taken into consideration and its efficiency decreases with the existence of noise. The works in [9–12], are based on the geometric shape of trajectories. They consider local time shifting, and compare trajectories of possibly different number of fixes. In [9,10], they consider local time shifting. Other works, such as in [13,14], conclude that similarity measures that are based on raw representation of trajectories are sensitive to rotation, shifting and scaling. Therefore, they proposed a measure based on movement direction. The work in [13] builds on the work in [14], and takes into account local time shifting, robustness to noise and the possibility of comparing trajectories of different lengths. Other measures that are based on time series representation of a trajectory are usually associated with a distance function that can either be metric or non-metric [7]. A distance function is said to be metric if it satisfies non-negativity, uniqueness, symmetry and triangle inequality [6,7]. Based on the used distance function, these measures can be divided into two classes. The first class uses L_1-norm and L_2-norm as a distance measure such as dynamic time warping [15], edit distance with real penalty [16], and time warp edit distance [6]. The second class scores similarity based on a matching threshold such as longest common subsequence [17], and edit distance on real sequences [18]. They handle local time shifting, and trajectories can have different lengths. Another work is proposed in [19], where similarity is based on both the speed and the path of moving objects. It follows a warping approach based on dynamic time warping. This approach works on trajectories of different lengths and handles local time shifting, but it is not robust to noise.

Most of the proposed measures are not reliable with the existence of different sampling rates between trajectories. For example, in Fig. 1, Trajectory R and S are the same but with different sampling rates. Using TWED measure to evaluate similarity, the result is that the two trajectories are not the closest. Therefore, in this paper, a comparative study is made between a discrete based similarity operator – TWEDistance – and three continuous based approaches: regression, interpolation and curve barcoding.

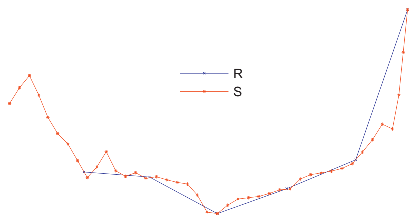
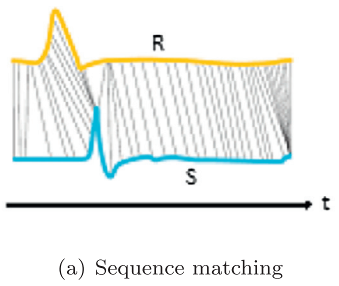


Fig. 1. Same trajectories with different sampling rate (R: every 200 sec, S: every 30 sec).



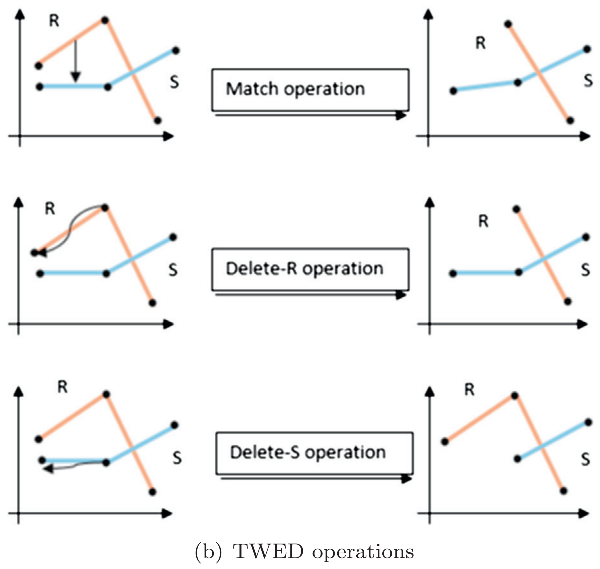


Fig. 2. TWED similarity measure.

[1] Local time shifting occurs when one element of one sequence is shifted along time axis to match an element of another sequence even if the two matched elements appear in different positions.

3. TWEDistance discrete similarity based operator

TWEDistance operator [5] is a generic discrete similarity based operator that is built on top of one of the moving objects database management systems which is called SECONDO [20,21]. This operator is proposed in our previous paper [5] and is based on the time warp edit distance measure (TWED) [6]. TWED is favorable over other discrete measures as it deals with local time shifting, a metric measure and an elastic metric as detailed in [5].

Given two trajectories R and S, as in Fig. 2(a), TWEDistance edits the two trajectories using three operations: *delete*$_R$, *delete*$_S$, and *match*. Each of the three operations has a penalty, and the distance between the two trajectories is the sum of penalties of the minimal-cost sequence of editing operations needed to transform one trajectory into the other.

A graphical illustration of the TWEDistance operations is given in Fig. 2(b). The *Delete* operation inside R or S involves dragging and dropping the sample to be deleted to its previous one. The cost associated with this delete operations is the length of the vector from the deleted sample to its previous one, using the d_{lp} which is a user defined function. An extra associated constant penalty called Lambda λ is added to this delete cost. So, the cost of deleting element m from the series R would be $d_{lp}(r_m, r_{m-1}) + \lambda$. The *Match* operation involves dragging and dropping the segment between two samples in the first trajectory to the matching segment in the second trajectory. The cost associated with the match operation is the sum of lengths of two vectors connecting the start and end of both segments, which is $d_{lp}(r_m, s_n) + d_{lp}(r_{m-1}, s_{n-1})$.

To provide the controlled temporal elasticity, TWED includes the time stamp difference in all its cost functions. These time differences are multiplied by a stiffness parameter γ to control the elasticity. TWED, hence, accept four parameters $R_1^m, S_1^n, \lambda, \gamma$ and returns the minimum of:

$$\begin{cases} \text{TWED}_{\lambda,\gamma}(R_1^{m-1}, S_1^n) + \gamma \cdot (t_{r_m} - t_{r_{m-1}}) \\ \quad + d_{lp}(r_m, r_{m-1}) + \lambda \\ \qquad\qquad\qquad\qquad \textit{delete} - R \\ \text{TWED}_{\lambda,\gamma}(R_1^m, S_1^{n-1}) + \gamma \cdot (t_{s_n} - t_{s_{n-1}}) \\ \quad + d_{lp}(s_n, s_{n-1}) + \lambda \\ \qquad\qquad\qquad\qquad \textit{delete} - S \\ \text{TWED}_{\lambda,\gamma}(R_1^{m-1}, S_1^{n-1}) + d_{lp}(r_m, s_n) \\ \quad + \gamma \cdot (t_{s_n} - t_{r_m}) + d_{lp}(r_{m-1}, s_{n-1}) \\ \quad + \gamma \cdot (t_{s_{n-1}} - t_{r_{m-1}}) \\ \qquad\qquad\qquad\qquad \textit{Match} \end{cases} \tag{1}$$

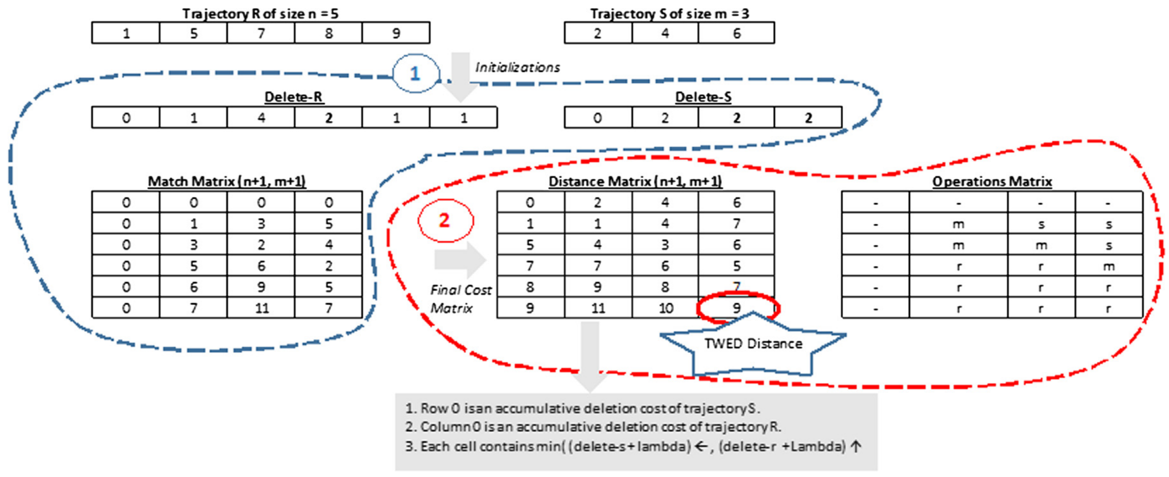
Lambda and stiffness values are data dependent and can be used for normalizing the value difference and the temporal difference scales [5]. The operator accepts trajectories of any meaning (e.g. the speed profile of a car, the spatial path, etc.) and the distance function used can be any user defined function (i.e. L$_n$-norm where n = 1, 2,…). TWEDistance operator, hence, supports genericness of both the meaning of similarity and the used distance function.

Algorithm 1 illustrates the evaluation of TWEDistance. It takes as an input two moving objects' trajectories divided into two sequences the spatial values and the time-stamp values, TWED's Stiffness and Lambda, and a user defined distance function. It returns as an output the TWEDistance that represents the sum of penalties of the minimal-cost sequence of editing operations needed to transform one trajectory into the other.

Algorithm 1. TWEDistance Algorithm

```
Input:
  $R$          → The sequence of values of the first object
  $S$          → The sequence of values of the second object
  $t_r$        → The sequence of time stamps for the first object
  $t_s$        → The sequence of time stamps for the second object
  $d_{lp}$     → The user defined distance function
  $\lambda$    → Lambda
  $\gamma$     → Stiffness
Output:
  $TWED[R_{Size}, S_{Size}]$   → The cost/distance between two moving objects
1: $R_{Size} \leftarrow Length(R)$
2: $S_{Size} \leftarrow Length(S)$
3: Allocate $TWED[0..R_{Size}, 0..S_{Size}]$
4: for $i \leftarrow 1, S_{Size}$ do
5:     $TWED[0, i] \leftarrow \infty$
6: end for
7: for $j \leftarrow 1, R_{Size}$ do

8:     $TWED[j, 0] \leftarrow \infty$
9: end for
10: $TWED[0, 0] \leftarrow 0$
11: for $i \leftarrow 1, R_{Size}$ do
12:     for $j \leftarrow 1, S_{Size}$ do
13:         $Delete - R \leftarrow TWED[i-1, j] + d_{lp}(R[i-1], R[i]) + \gamma * (t_r[i] - t_r[i-1]) + \lambda$
14:         $Delete - S \leftarrow TWED[i, j-1] + d_{lp}(S[j-1], S[j]) + \gamma * (t_s[j] - t_s[j-1]) + \lambda$
15:         $Match \leftarrow TWED[i-1, j-1] + d_{lp}(R[i], S[j]) + \gamma * (|t_r[i] - t_s[j]|))$
            $+ d_{lp}(R[i-1], S[j-1]) + \gamma * (|t_r[i-1] - t_s[j-1]|)$
16:         $TWED[i, j] \leftarrow Minimum(Delete - R, Delete - S, Match)$
17:     end for
18: end for
19: return $TWED[R_{Size}, S_{Size}]$
```



Trajectory R of size n = 5

1	5	7	8	9

Trajectory S of size m = 3

2	4	6

Delete-R

0	1	4	2	1	1

Delete-S

0	2	2	2

Match Matrix (n+1, m+1)

0	0	0	0
0	1	3	5
0	3	2	4
0	5	6	2
0	6	9	5
0	7	11	7

Distance Matrix (n+1, m+1)

0	2	4	6
1	1	4	7
5	4	3	6
7	7	6	5
8	9	8	7
9	11	10	9

Operations Matrix

-	-	-	-
-	m	s	s
-	m	m	s
-	r	r	m
-	r	r	r
-	r	r	r

1. Row 0 is an accumulative deletion cost of trajectory S.
2. Column 0 is an accumulative deletion cost of trajectory R.
3. Each cell contains min((delete-s + lambda) ←, (delete-r +Lambda) ↑

Fig. 3. TWED example using L_1-norm, $\lambda = 1$ and $\gamma = 0.0$.

A TWEDistance example is shown in Fig. 3 that shows how TWEDistance implementation works between two 1-D trajectories R and S, where four matrices are defined: Delete-R, Delete-S, Match and Final Distance Matrix [5].

Where Delete-S and Delete-R at position 'index' is measured as follows:

$$delete[index] = \begin{cases} 0, & if\ index = 0 \\ d_{lp}(sample_{i-1}, 0), & if\ index = 1 \\ d_{lp}(sample_{i-1}, sample_{i-2}), & if\ index > 1 \end{cases}$$

Match cost between the two trajectories R and S is measured as follows:

$$Match[i,j] = \begin{cases} d_{lp}(R_{i-1}, S_{j-1}) + d_{lp}(R_{i-2}, S_{j-2}), \\ \qquad\qquad if\ i \& j > 1 \\ d_{lp}(R_{i-1}, S_{i-1}), \\ \qquad\qquad otherwise \end{cases}$$

where d_{lp} is the used defined function. After building deletion and match matrices, the final distance matrix holds the accumulative cost required (added to $\gamma *$ the time stamp differences) to superimpose both trajectories where $distanceMatrix[n+1, m+1]$ is the TWED distance [5].

4. Continuous based similarity measures

In this paper, similarity evaluation is proposed to be based on the trajectory's data as a continuous function. The proposed continuous similarity evaluation was based on different approaches which are regression, interpolation and curve barcoding. In this section, these approaches and how they are utilized in evaluating the similarity are discussed.

4.1. Regression

Regression is one of the statistical curve fitting approaches in which an equation is predicted to describe discrete set of points. One of its types is the linear regression that depends on one independent variable to predict the value of the dependent variable as described in Eq. (2).

$$Y = a + bX \tag{2}$$

where

Y is the dependent variable that needs to be predicted.
X is the independent variable.
a is the intercept (The value of y when x equals zero).
b is the slope.

Given a set of $n\,(x, y)$ pairs, the objective is to find the best fitting function $f(X_i)$ with the minimum residual error that deviates from the observed data the least using least square method. Residual error for a point at index i is evaluated as shown in Eq. (3), and for the entire set is as in Eq. (4). To achieve the objective, partial derivative of the function $r(X)$ is taken with respect to its coefficients (e.g. a and b in Eq. (2)). After equalizing the partial derivatives of the coefficients to zero, the result is a set of equations – where their number is equal to number of coefficients – to be solved using linear algebra and matrices.

$$r_i(x_i) = y_i - f(x_i) \tag{3}$$

$$r(X) = \sum_{0}^{n-1} (y_i - f(x_i))^2 \tag{4}$$

where y_i is the original value of y in the data and $f(x_i)$ is the predicted value.

A linear regression model then can be solved using the following matrix:

$$\begin{vmatrix} n & \sum_0^n x_i \\ \sum_0^n x_i & \sum_0^n x_i^2 \end{vmatrix} \begin{vmatrix} a \\ b \end{vmatrix} = \begin{vmatrix} \sum_0^n y_i \\ \sum_0^n x_i y_i \end{vmatrix}$$

The previous matrix could be generalized for polynomial regression [22,23] with function $Y = a_0 + a_1 X + a_1 X^2 + a_2 X^3 + \cdots\cdots + a_p X^p$ of degree p as follows:

$$\begin{vmatrix} n & \sum_0^n x_i & \sum_0^n x_i^2 & \sum_0^n x_i^3 & \cdots & \sum_0^n x_i^p \\ \sum_0^n x_i & \sum_0^n x_i^2 & \sum_0^n x_i^3 & \sum_0^n x_i^4 & \cdots & \sum_0^n x_i^{p+1} \\ \sum_0^n x_i^2 & \sum_0^n x_i^3 & \sum_0^n x_i^4 & \sum_0^n x_i^5 & \cdots & \sum_0^n x_i^{p+2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum_0^n x_i^p & \sum_0^n x_i^{p+1} & \sum_0^n x_i^{p+2} & \sum_0^n x_i^{p+3} & \cdots & \sum_0^n x_i^{2p} \end{vmatrix} \begin{vmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_p \end{vmatrix} = \begin{vmatrix} \sum_0^n y_i \\ \sum_0^n x_i y_i \\ \sum_0^n x_i^2 y_i \\ \vdots \\ \sum_0^n x_i^p y_i \end{vmatrix}$$

Using Gauss-Jordan Elimination technique [24], the linear or polynomial system can be solved to get coefficient values by building the augmented matrix as shown in Eq. (5) and converting it to its reduced echelon form using row operations (i.e. swapping rows, multiplication/division of rows by a non zero constant and adding/subtracting one row to a multiple of another row).

$$\left[\begin{array}{cc|c} n & \sum_0^n x_i & \sum_0^n y_i \\ \sum_0^n x_i & \sum_0^n x_i^2 & \sum_0^n x_i y_i \end{array}\right] \tag{5}$$

Logarithmic functions could also be used for modeling a set of discrete points to a continuous function using linear transformations [25,26] as shown in Table 1. In this paper, similarity is evaluated between trajectories using Linear, Logarithmic and Polynomial Regression.

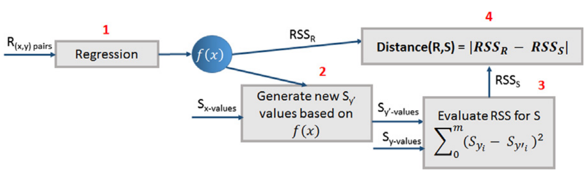
4.1.1. Regression similarity

In regression, the movement of the trajectory is approximated in the form of a function $f(x)$ with the minimum residual error *RSS*. For evaluating the similarity between two trajectories R and S, the following steps are followed and they are shown in Fig. 4.

1. For a trajectory R represented as (x, y) pairs, a function $f(x)$ is generated that approximates the movement with a residual error RSS_R.
2. The x value of the generated function $f(x)$ is set with the x-values of the second trajectory S ($S_{x\text{-}values}$) to generate the new $S_{y'\text{-}values}$ values.
3. Then, the residual error is evaluated for the second trajectory RSS_S using sum of square differences between original y values $S_{y\text{-}values}$ and predicted y′ values $S_{y'\text{-}values}$.
4. Finally the similarity between two trajectories R and S is the absolute difference between their RSS: $|RSS_R - RSS_S|$.

Table 1
Linear regression transformation [27].

Regression type	Function representation	Description
Logarithmic	$y = a + b\,ln(x)$	Relation between $ln(x)$ and y is linear. a and b are evaluated using least square method on y and $ln(x)$ instead of x



1
2
3
4
R(x,y) pairs
Regression
$f(x)$
RSS_R
Distance(R,S) = $|RSS_R - RSS_S|$
RSS_S
S_x-values
Generate new S_y' values based on $f(x)$
S_y'-values
S_y-values
Evaluate RSS for S
$\sum_0^m (S_{y_i} - S_{y'_i})^2$

Fig. 4. Regression approach steps.

Fig. 5, shows a numeric example for linear regression.

Where linear regression is used to approximate the trajectory R points getting up with a linear function that equals to $1.323199212x + 1197.900894$ with a residual error equal to 258442.184. After substituting with the x values of S trajectory in the linear equation, new y values are generated with a residual error equal to 2934795l4.9. After evaluating both error the similarity is the absolute difference between then that is equal to 293221072.716.

4.2. Interpolation

Interpolation is used for curve fitting and for constructing new data points within the range of a discrete set of known data points. Having a discrete set of points represented as (x, y) pairs with the condition $x_0 < x_1 < \ldots x_N$, it is required to estimate/interpolate the value of the function that the data points represent for an intermediate x value where $x_k \leqslant x \leqslant x_{k+1}$ (k is an index inside the set of pairs).

In this paper, two types of interpolation methods are considered:

- Linear Interpolation.
- Piecewise Cubic Interpolation.

4.2.1. Linear interpolation

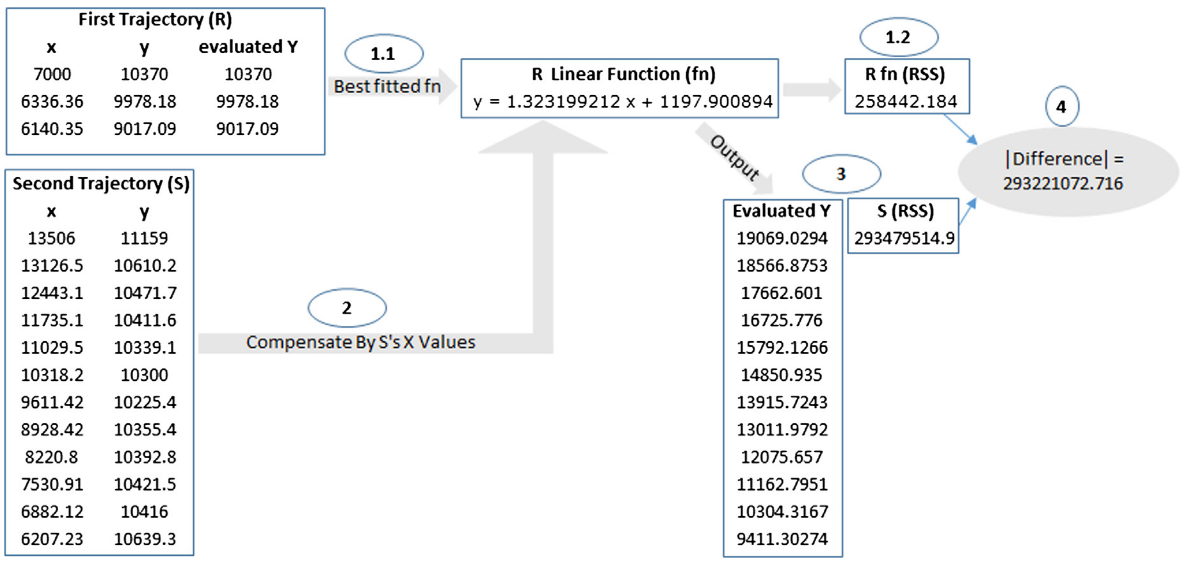
In Linear interpolation, If the value of the independent variable x falls between two data points (x_0, y_0) and (x_1, y_1) in the set of discrete points, the value of the dependent variable y can be estimated using Eq. (6). For example, in Fig. 6, the value of y at $x = 9000$ is calculated to be $y = 1.0342$.

$$y = y_0 + (y_1 - y_0)\frac{x - x_0}{x_1 - x_0} \tag{6}$$

Linear interpolation on a discrete set of data points $\{(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)\}$ of length n is the concatenation of linear interpolants between each pair of data points. This results in a continuous curve.

4.2.2. Piecewise cubic interpolation

In this section, types of interpolation based on piecewise cubic Hermite interpolation [28] such as Shape-Preserving [29,30] and



First Trajectory (R)

x	y	evaluated Y
7000	10370	10370
6336.36	9978.18	9978.18
6140.35	9017.09	9017.09

Second Trajectory (S)

x	y
13506	11159
13126.5	10610.2
12443.1	10471.7
11735.1	10411.6
11029.5	10339.1
10318.2	10300
9611.42	10225.4
8928.42	10355.4
8220.8	10392.8
7530.91	10421.5
6882.12	10416
6207.23	10639.3

Evaluated Y
19069.0294
18566.8753
17662.601
16725.776
15792.1266
14850.935
13915.7243
13011.9792
12075.657
11162.7951
10304.3167
9411.30274

Fig. 5. Linear regression example.

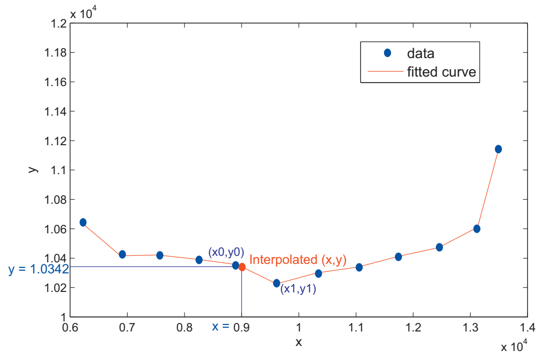


Fig. 6. Interpolation of $x = 9000$ that falls between $[x_0, x_1]$ using linear interpolation.

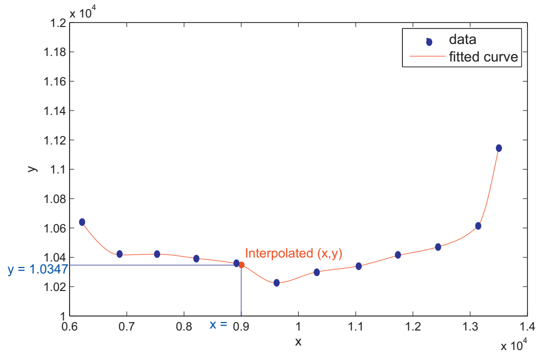


Fig. 7. Shape preserving cubic interpolation example where the $y = 1.0347$ at $x = 9000$.

cubic spline [31,30,32] are discussed. Both types generate a polynomial of degree 3 for an interval $[x_k, x_{k+1}]$.

The polynomial is defined as follows

$$P(x) = \frac{3hs^2 - 2s^3}{h^3} y_{k+1} + \frac{h^3 - 3hs^2 + 2s^3}{h^3} y_k + \frac{s^2(s-h)}{h^2} d_{k+1} + \frac{s(s-h)^2}{h^2} d_k \tag{7}$$

where $s = x - x_k, h = x_{k+1} - x_k$(interval length)$, y_k = P(x_k), y_{k+1} = P(x_{k+1}), d_k = P'(x_k), d_{k+1} = P'(x_{k+1})$ and d_k and d_{k+1} are the slopes at x_k and x_{k+1} respectively. Both types differ in the way they compute the slopes, d_k. Once the slopes have been computed, the interpolant can be evaluated using the power form with the local variable s: $P(x) = y_k + s\,d_k + s^2\,c_k + s^3\,b_k$, where the coefficients of the quadratic and cubic terms are

$$c_k = \frac{3\,\delta_k - 2\,d_k - d_{k+1}}{h}$$

$$b_k = \frac{d_k - 2\,\delta_k + d_{k+1}}{h^2}$$

where δ is measured as $\delta_k = \frac{y_{k+1} - y_k}{h_k}$. In the following subsections, how the slopes are measured for each type are presented.

Shape-Preserving Piecewise Cubic Interpolation

In Shape-Preserving piecewise cubic interpolation, to evaluate slopes there are two main cases:

- Slopes for the interior points
  1. If $\delta_k * \delta_{k+1} \leqslant 0$, then $d_k = 0$.
  2. If $\delta_k * \delta_{k+1} > 0$ and the two intervals have the same length then $d_k = \frac{2\,\delta_{k-1}\,\delta_k}{\delta_{k-1} + \delta_k}$.
  3. If $\delta_k * \delta_{k+1} > 0$ and the two intervals have different lengths then $d_k = \frac{\delta_k\,\delta_{k-1}\,(\omega_1 + \omega_2)}{\omega_1\,\delta_k + \omega_2\,\delta_{k-1}}$ where $\omega_1 = 2h_k + h_{k-1}$ and $\omega_2 = h_k + 2h_{k-1}$.
- Slopes for the data end points
  1. At Point 0, $d_1 = \frac{((2\,h_1 + h_2)\delta_1) - (h_1\,\delta_2)}{h_1 + h_2}$ with the following cases

     $$d_1 = \begin{cases} 0, & d_1 * \delta_1 \leqslant 0 \\ 3\,\delta_1, & \delta_1 * \delta_2 \leqslant 0 \,\&\&\, |\,d_1\,| > 3\,|\,\delta_1\,| \end{cases}$$

  2. At Point n, $d_n = \frac{((2\,h_{n-1} + h_{n-2})\delta_{n-1}) - (h_{n-1}\,\delta_{n-2})}{h_{n-1} + h_{n-2}}$ with the following cases

     $$d_n = \begin{cases} 0, & d_n * \delta_{n-1} \leqslant 0 \\ 3\,\delta_{n-1}, & \delta_{n-1} * \delta_{n-2} \leqslant 0 \,\&\&\, |\,d_n\,| > 3\,|\delta_{n-1}| \end{cases}$$

Fig. 7 shows an example of interpolating the value of y at $x = 9000$ on the approximated curve which is found to be $y = 1.0347$.

Cubic Spline Interpolation

In cubic spline, the following system of equations are solved to get the slopes d:

$$Ad = r$$

where A is a tridiagonal matrix represented as follows:

$$\begin{vmatrix} h_2 & h_1 + h_2 & 0 & 0 & \cdots & 0 \\ h_2 & 2(h_2 + h_1) & h_1 & 0 & \cdots & 0 \\ 0 & h_3 & 2(h_3 + h_2) & h_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & h_{n-1} & 2(h_{n-2} + h_{n-1}) & h_{n-2} \\ 0 & 0 & 0 & 0 & h_{n-1} + h_{n-2}) & h_{n-2} \end{vmatrix}$$

And r is

$$3\begin{vmatrix} r_1 \\ h_1\delta_2 + h_2\delta_1 \\ h_2\delta_3 + h_3\delta_2 \\ . \\ . \\ h_{n-2}\delta_{n-1} + h_{n-1}\delta_{n-2} \\ r_n \end{vmatrix}$$

where r_1 and r_n is evaluated using the following equations

$$r_1 = \frac{(h_1 + 2(h_2 + h_1)) * h_2 * \frac{y_2 - y_1}{x_2 - x_1} + h_1^2 * \frac{y_3 - y_2}{x_3 - x_2}}{h_2 + h_1}$$

$$r_n = \frac{h_{n-1}^2 * \frac{y_{n-1} - y_{n-2}}{x_{n-1} - x_{n-2}} + (2(h_{n-1} + h_{n-2}) + h_{n-1}) * h_{n-2} * \frac{y_n - y_{n-1}}{x_n - x_{n-1}}}{h_{n-1} + h_{n-2}}$$

Fig. 8 shows an example of interpolating the value of y at $x = 9000$ that results in $y = 1.0343$.

4.2.3. Interpolation similarity

In interpolation, a trajectory is represented as ((x, y), t) pairs or (x, t) pairs where they are divided into intervals based on a sorted independent variable. Therefore, the time dimension is chosen as our independent variable for predicting (x, y) pair or x. For evaluating similarity between two trajectories R and S, the following steps are followed and they are shown in Fig. 9:

1. For a trajectory R, the functions $f(t)$ for each data interval are generated ((n – 1) functions).

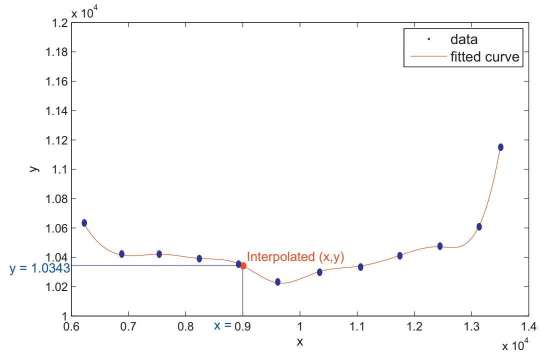
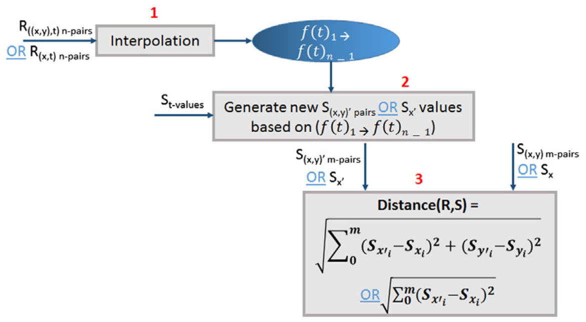


Fig. 8. Cubic spline interpolation example where $y = 1.0343$ at $x = 9000$.



1
$R_{((x,y),t)\ n\text{-pairs}}$
OR $R_{(x,t)\ n\text{-pairs}}$
Interpolation
$f(t)_1 \rightarrow f(t)_{n-1}$
2
$S_{t\text{-values}}$
Generate new $S_{(x,y)'\ pairs}$ OR $S_{x'}$ values based on $(f(t)_1 \rightarrow f(t)_{n-1})$
$S_{(x,y)'\ m\text{-pairs}}$ OR $S_{x'}$
$S_{(x,y)\ m\text{-pairs}}$ OR S_x
3
Distance(R,S) =
$\sqrt{\sum_0^m (S_{x'_i}-S_{x_i})^2 + (S_{y'_i}-S_{y_i})^2}$
OR $\sqrt{\sum_0^m (S_{x'_i}-S_{x_i})^2}$

Fig. 9. Interpolation based similarity approach.

2. Then, $S_{t\text{-}values}$ are interpolated in the generated functions to generate corresponding new (x, y) pairs ($S_{(x,y)'}$) or new X ($S_{x'}$) for S.
3. Finally, the distance between R and S is evaluated using Euclidean distance between the new predicted values ($S_{(x,y)'}$ or $S_{x'}$) and the original ones ($S_{(x,y)}$ or S_x).

4.3. Radon barcoding

Barcoding was proposed in medical image retrieval where it is used with feature based retrieval for better accurate results. One of the approaches that is used for image barcoding is the Radon Transform. Applying the Radon transform on an image f (x, y) for a given set of angles can be thought of as computing the projection of the image along the given angles. The resulting projection is the sum of the intensities of the pixels in each direction (i.e. a line integral). The result is a new image $R(\rho,\theta)$. The new image can be expressed using its projections which is defined as follows $\rho = x\cos\theta + y\sin\theta$. Then, Radon Transform of the image can be expressed as defined in Eq. (8).

$$R(\rho,\theta) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)\,\delta(\rho - x\cos\theta - y\sin\theta)\,dx\,dy \tag{8}$$

where $\delta(\cdot)$ is the Dirac delta function. In [33], the authors proposed to binarize the resulted radon transform. Therefore, after generating radon transform of an image, the projections are thresholded by calculating a typical value using the median operator. This median operator is applied to all non-zero values of each projection. After predicting median value it is used for building the binarized radon barcode where values $\geqslant$ median is replaced with 1, otherwise it is replaced with 0. To receive barcodes with the same length, images are normalized and resized into $R_N \times C_N$ images (i.e. $R_N = C_N = 2^n; n \in \mathbb{N}^+$).

Algorithm 2 describes how the binary radon barcodes are generated.

Algorithm 2. Radon Barcode Algorithm

```
Input:
  IMG            → The input image
  numOfProj      → Number of projections
  $R_N$          → Number of rows for image normalization
  $C_N$          → Number of columns for image normalization
Output:
  BRBC           → Binary Radon Barcode
1: $BRBC \leftarrow \varnothing$
2: Initialize $\theta \leftarrow 0$
3: $\overline{IMG} \leftarrow$ Normalize$(IMG, R_N, C_N)$
4: while $\theta < 180$ do
5:     Get projection $\rho$ for the angle $\theta$
6:     Find typical value:
       $T_{typical} \leftarrow Median_{\rho_i \neq 0}(\rho)$
7:     Binarize projection:
       $b \leftarrow \rho \geqslant T_typical$
8:     Append the new binary projection barcode at $\theta$ to BRBC:
       Append$(BRBC, b)$
9:     $\theta \leftarrow \theta + \frac{180}{numOfProj}$
10: end while
11: return BRBC
```

4.3.1. Barcode similarity

Similarity evaluation using curve barcoding deals with a trajectory as an image instead of a sequence of points. To evaluate similarity between two trajectories R and S, the following steps are followed:

1. After converting both trajectories (R and S) from (x, y) pairs into an image, barcode sequences are generated for them ($R_{barcode}$ and $S_{barcode}$). Fig. 10 shows an example of a trajectory image and its corresponding barcode.
2. Then, similarity between trajectories' barcodes is evaluated using Hamming distance that can be defined as

$$Distance(R_{barcode}, S_{barcode}) = \sum_0^n R_{barcode_i} \neq S_{barcode_i}$$

where n is the length of the barcodes. Fig. 11 shows an example for Hamming distance evaluation between two trajectories' barcodes.

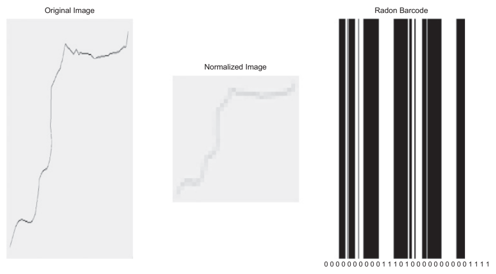


Fig. 10. Example of a 32×32 normalized trajectory image using 4 projections barcoding.

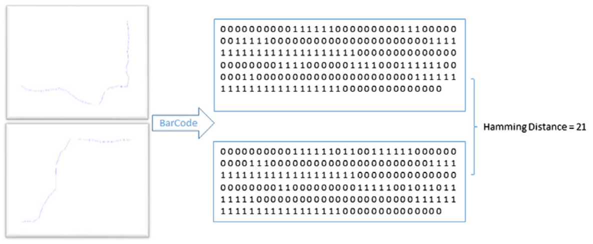


Fig. 11. Hamming distance between two trajectories' barcodes example.

5. k-nearest neighbor classification

The k-nearest neighbors (k-NN) [34] is a method that is used for classification in our experiments. The input consists of set of trajectories and a query trajectory that needs to be classified. Classification is done by evaluating distance between the query trajectory and the other trajectories. The predicted class is evaluated based on the K specified value. If k = 1 then the predicted class is the class of the first nearest neighbor based on the evaluated distance. If k > 1 then the predicted class is the most frequent class in the k nearest neighbors. The distance can be evaluated using any distance metric such as Euclidean distance (ED).

Algorithm 3 describes how K-NN Works.

Algorithm 3. K-NN Classification

```
Input:
  $[x, class]$        → the input trajectories with their class label
  $q$                 → the query trajectory
  $k$                 → the k value for nearest neighbors
Output:
  $c$                 → The predicted class label
1: $XSize \leftarrow Length(X)$
2: Distances[0..$XSize$]      → Contains distance with the class label
3: for $i \leftarrow 1$ to $XSize$ do
4:    Compute distance Distances.add ($(d_i([x, class]_i, q)$,class)
5: end for
6: sort(Distances) Ascending
7: if $k == 1$ then c ← the trajectory that has $d_0$ with q
8: else if $x > 1$ then
9:    c ← majority class of Distances from index 0 to k
10: end if
11: return c
```

6. Experiments and results

In the experiments, the similarity between trajectories is evaluated using the four approaches. Three datasets are used, SECONDO's berlintest synthetic dataset [35], the Trucks real dataset [36] and the Geo-Life real dataset [37–39].

Berlintest dataset contains the *Trains* relation. This relation was created by simulating the underground trains in Berlin. Simulation was based on the real train schedule and the real underground network of Berlin. The period of simulation is about 4 h in one day. It contains 9 lines identified by an ID (Line attribute) and group of trains travel per each line and identified also by an ID. Each train's trajectory is stored in a *Trip* attribute. This relation contains 562 trajectories, consisting of a total of 54,595 observations. Trucks dataset consists of 1100 trajectories of 50 trucks delivering concrete to several construction places around Athens with a total of 188,197 observations. Geo-Life is a GPS trajectory dataset that was collected in the Geolife project (Microsoft Research Asia) by 182 users in china in a period of over three years (from April 2007 to August 2012). It contains 17,621 trajectory with different sampling rates. Each trajectory is represented by a sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude. The dataset were filtered to year 2008 that contains 7334 trajectory with 5,474,814 observations.

6.1. Spatial based similarity

In this experiment, the accuracy of the four approaches is evaluated based on the spatial similarity and how it is affected by different sampling rates. Trains dataset is filtered by picking up 9 trains one from each line then resampling these trains to ten different samples coming up with 90 trajectories. Same for Trucks dataset, they are filtered into 205 trajectory and each trajectory was resampled to 6 samples yielding 1230 trajectories. These final trajectories contain from 4 to 287 samples per each. Table 2 concludes the datasets description that was used in this experiment.

In regression similarity, for all trajectories, linear, logarithmic and polynomial regression – degree 10 or less if trajectory has less than 9 samples – are used to predict the function that best fits trajectories' points with the minimum residual error. For each trajectory's function $f(n)$, the x variable are set with the x values of the other trajectories' points to evaluate the Y values to evaluate how much the function $f(n)$ fits the points. After that the Residual sum of squares RSS as defined in Eq. (4) is measured. Then the difference between $f(n)$ RSS and other trajectories' RSS are computed and the results are sorted according to the evaluated difference.

In TWED similarity, the generic TWEDistance operator that is implemented in SECONDO [5] is used. The Similarity evaluation is based on spatial proximity with stiffness = zero, Lambda = 1 and Euclidean Distance as our distance function [5]. The Query used is as follows

```
let SpatialDist=
  Trucks feed
  extend[Dist: TWEDistance (
    .Trip_a,
    .Trip_b,
    0.0,
    1.0,
    fun(x: point, y: point)
    sqrt( pow( getx(x) - getx(y), 2 ) +
    pow( gety(x) - gety(y), 2 )))]
    sortby[ID_a asc, Dist asc]
  consume;
```

where $Trip_a$ is the first trajectory and $Trip_b$ is the second one. After evaluating TWED distance between all pairs of trajectories, results were sorted with respect to the computed distance.

In radon barcode similarity, trajectories are represented as a curve image. Radon transform approach is then applied to the

Table 2
Description of trucks and trains datasets.

Parameters	Dataset	
	Trucks	Trains
Number of trajectories	1100	562
Number of Observations/points	188,197	54,595
Number of Filtered Trajectories	205	9
Number of versions per trajectory (V_N)	6	10
Total number of trajectories including versions (T_N)	1230	90
Sampling rates	Every (100, 50, 900, 80, 200) sec	Every (100, 50, 80, 200, 60, 30, 70, 20 and 15) sec
Number of samples per trajectory	4–287 sample	
Attributes defines a trajectory	ID, Version_ID and Trip(Trajectory)	

image after normalization (32 × 32) and the result is thresholded to generate the binarized barcode. For both datasets, radon transform is applied using different values of projections (i.e. 4, 8, 16 and 32) for all trajectories' images then binary barcode is generated and hamming distance is used to measure how much the barcodes are similar. The hamming distance results are then sorted.

In interpolation similarity, the two types of previously discussed interpolation are used. First, the approximated curve of each trajectory are generated then the x values of the other trajectories' points are interpolated to generate the y values (i.e. The x values in this experiment are the time and y values are the (x, y) coordinates). Then, the distance is measured using Euclidean Distance between the interpolated points and the original one. Finally, the results are sorted according to the evaluated distance.

6.1.1. Accuracy evaluation

Accuracy of the spatial similarity experiment is evaluated based on Head Nearest Neighbor approach. After evaluating distance between all trajectories, for each trajectory, the results are sorted from the least distance to the maximum. After that, the head k trajectories are evaluated if they contain trajectories from the same class under the assumption that trajectory and its versions form a class. In the trains dataset, each trajectory has a total number of versions (V_N) equals to 10 with a total number of trajectories (T_N) equals to 90. In the trucks dataset, each trajectory has a total number of versions (V_N) equals to 6 with a total number of trajectories (T_N) equals to 1230. First, accuracy per each trajectory is evaluated as follows:

$$Trajectory\_Accuracy = \frac{N_{sameclass}}{V_N}$$

where $N_{sameclass}$ is the number of trajectories of the same class that appears at the Head K. For each trajectory, k is determined based on the distance value. K is increased by 1 while reaching the following condition: if the next trajectory to compare with is not of the same class and previous trajectory's distance is different from the current one (i.e. stops when a different id appears). Fig. 12 shows two illustrative examples for evaluating accuracy of a trajectory with V_N equals to 7. In the first example, trajectory with id equal to 1 has head four trajectories with zero distance. The ones having same id are two. Therefore, k here is equal to 2 as there are other trajectories with different ids appears in zero distance. In second example, k is equal to 4 as in the zero distance there are three trajectories with the same id (initially k is equal to three), no different ids appears in that distance so go to the next distance (1717.17). In the next distance there is one trajectory with the same id so k now is 4, stop here as there is different id appears in that distance.

After evaluating accuracy per each trajectory, the total accuracy is computed using the following formula:

$$Total\_Accuracy = \frac{\sum_0^{T_N} \text{Trajectory\_Accuracy}}{T_N}$$

Fig. 13 shows the results of the four approaches using the head nearest neighbor approach where interpolation gives higher accuracy on both datasets.

6.2. Direction based similarity

In this experiment, accuracy of the approaches is based on the direction similarity. The *Trains* relation of the *berlintest* dataset was used as each trajectory is labeled with its direction. It contains trajectories of trains travelling at different lines. The subset of trains that travel on Line 1 that has a total of 58 trajectories are selected, from which 29 are heading on one direction, and the rest are heading on the opposite direction. One train is picked from each group[2] and then re-sampled into five versions. The sampling rates were every 42, 50, 80, 100, 200 sec.

In interpolation, regression and binary barcode, the time was considered as the x-axis and direction as the y-axis. The results should be as follows

- Each train should have the minimum distance with all other trains travelling at the same direction (i.e. Head distance results should be the other trains (train's versions) that travel at the same direction then comes the ones travelling at the opposite direction).

In TWED, the following query is used with lambda = 1.0 and stiffness = 0.0:

```
let DirectionDist=
  TrainsLine1 feed
  extend[Dist: TWEDistance (
    direction(.Trip_a),
    direction(.Trip_b),
    0.0,
    1.0,
    fun(x: real, y: real)
    abs( x - y))]
    sortby[ID_a asc, Dist asc]
  consume;
```

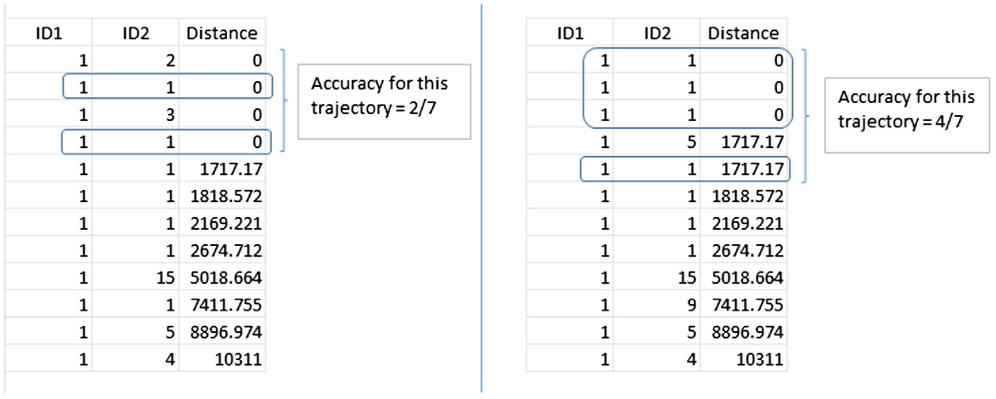
where TrainsLine1 is the relation that hold trajectories of all trains travelling at Line 1. $Trip_a$ and $Trip_b$ are the trajectories of two trains in each tuple where distance is measured between their direction using absolute difference as our defined function. After that, the results are sorted ascending according to the computed distance.

Same approach followed in the spatial experiment is used in this experiment for evaluating the accuracy of the approaches using V_N equal to 5 and T_N equal to 10 (i.e. the stopping condition is when a different direction appears). Fig. 14 shows the accuracy results for this experiment where cubic interpolation, linear interpolation and TWEDistance gives higher accuracy.

6.3. Speed based similarity

In this experiment, the Geo-life dataset is used as each trajectory is labeled by a transportation mode. These modes include car, bus, train, bike, walk, etc. This experiment is applied on 46

[2] Only one train is picked from each group because all trains in a group will have the same direction values as they travel at the same line and direction.



ID1	ID2	Distance
1	2	0
1	1	0
1	3	0
1	1	0
1	1	1717.17
1	1	1818.572
1	1	2169.221
1	1	2674.712
1	15	5018.664
1	1	7411.755
1	5	8896.974
1	4	10311

Accuracy for this trajectory = 2/7

ID1	ID2	Distance
1	1	0
1	1	0
1	1	0
1	5	1717.17
1	1	1717.17
1	1	1818.572
1	1	2169.221
1	1	2674.712
1	15	5018.664
1	9	7411.755
1	5	8896.974
1	4	10311

Accuracy for this trajectory = 4/7

Fig. 12. Trajectory accuracy evaluation examples with a total number of versions V_N equal to 7.

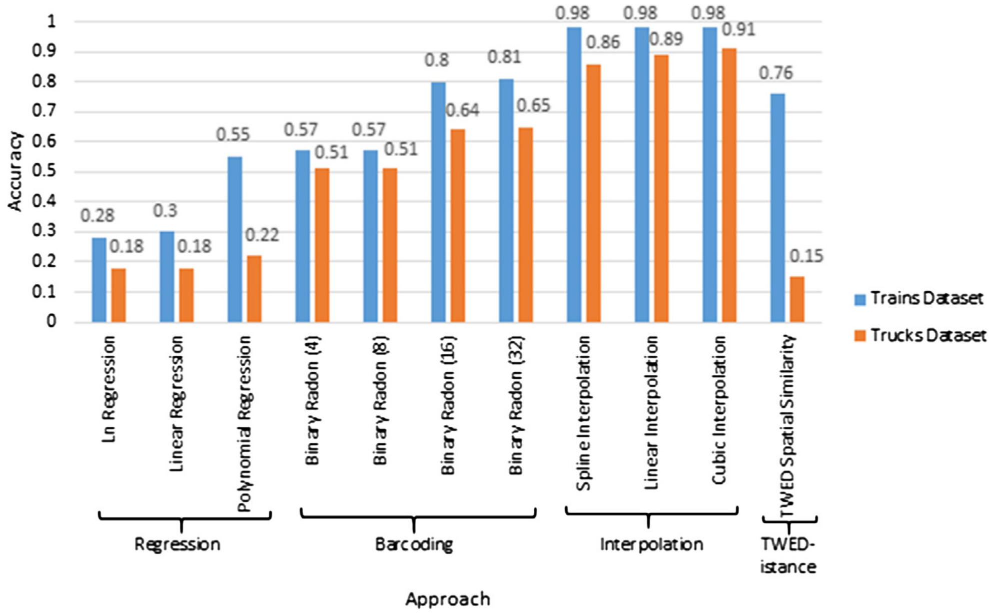


Fig. 13. Head nearest accuracy results where cubic interpolation in the experiments is the shape preserving type.

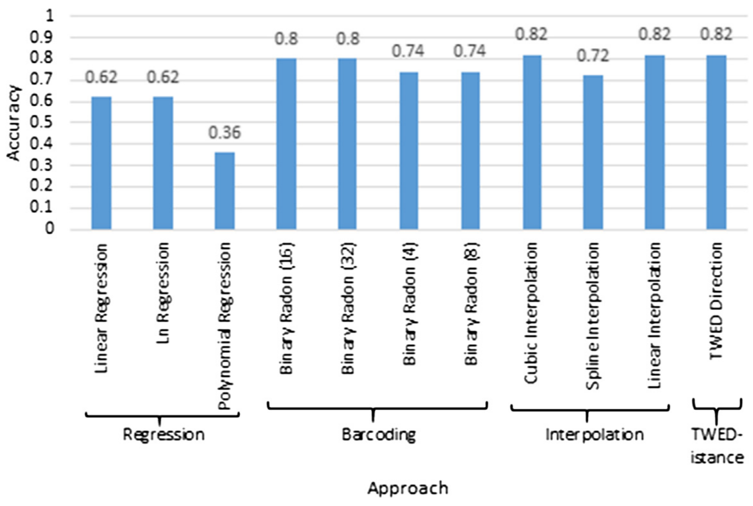


Fig. 14. Direction based similarity accuracy.

trajectories including original trajectories and sampled versions from them. The sampling rates differs based on the transportation mode as some sampling rates causes number of observations to be less that what is needed (i.e. in cubic interpolation at least four observations are needed).

In interpolation, regression and binary barcode, the time was considered as the x-axis and speed as the y-axis. The results should be as follows

- Each train should have the minimum distance with all other trains having the same transportation mode.

In TWED experiment, the following is the used query with lambda = 50.0 and stiffness = 0.0:

```
let SpeedDist=
  Trucks feed
  extend[Dist: TWEDistance (
    speed(.Trip_a),
    speed(.Trip_b),
    0.0,
    50.0,
    fun(x: real, y: real)
    abs( x - y))]
    sortby[ID_a asc, Dist asc]
  consume;
```

where Trucks is the relation that hold trajectories of all trains travelling at Line 1. $Trip_a$ and $Trip_b$ are the trajectories of two trains in each tuple where distance is measured between their speed using absolute difference as our defined function. After that, the results are sorted ascending according to the computed distance.

Same approach used for evaluating the accuracy in the spatial and direction based similarity is used here with T_N equal to 46 and different V_N for each transportation mode (i.e. the stopping condition is when a different mode appears). Table 3 shows each mode with its V_N.

Fig. 15 shows the accuracy results for all approaches where regression and interpolation gives higher accuracy.

Table 3
Transportation mode and V_N.

Mode	V_N
Train	11
Taxi	4
Bus	7
Walk	4
Bike	4
Subway	6
Car	10

6.4. Results

In this section, the overall accuracy for all approaches is evaluated based on the average accuracy of all experiments. For evaluating the average accuracy, the following formula is used:

$$Avg\_Accuracy = \frac{\sum C_{Exper_{sameclass}}}{\sum T_{N_{Exper}}}$$

where $C_{Exper_{sameclass}}$ is the number of trajectories of the same class that appears at the Head K per each experiment and $T_{N_{Exper}}$ is the total number of trajectories per each experiment as shown in Table 4.

After evaluating the average accuracy, the results are as shown in Fig. 16.

In the results of average accuracy, interpolation shows the highest accuracy results with the existence of sampling rate differences with an average accuracy up to 90% as it generates an equation for each data interval.

Table 4
Total number of trajectories per each experiment.

Experiment	T_N
Spatial similarity	1320 (i.e. 1230 + 90)
Direction similarity	10
Speed similarity	46

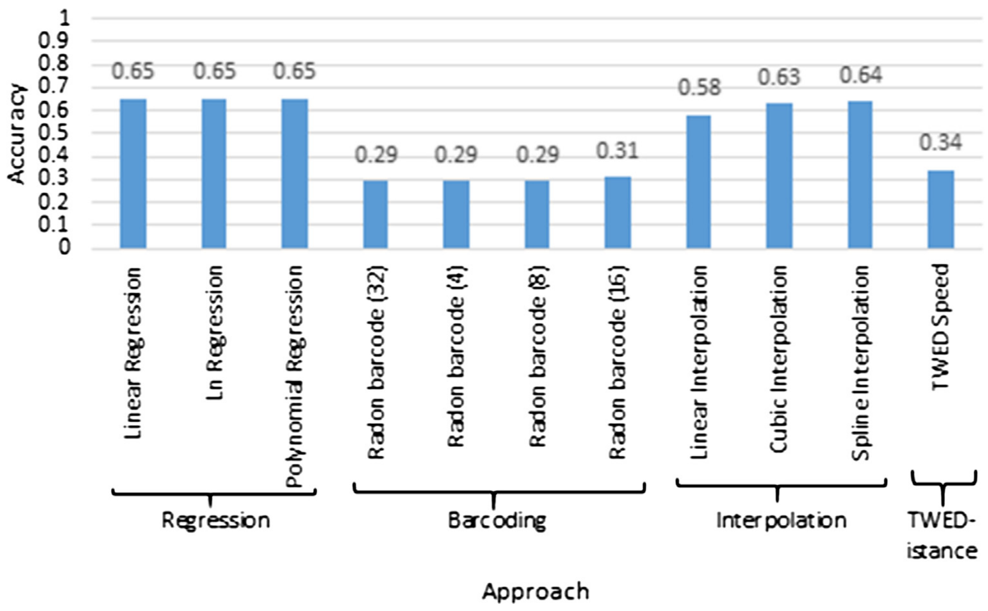


Fig. 15. Accuracy of speed similarity.

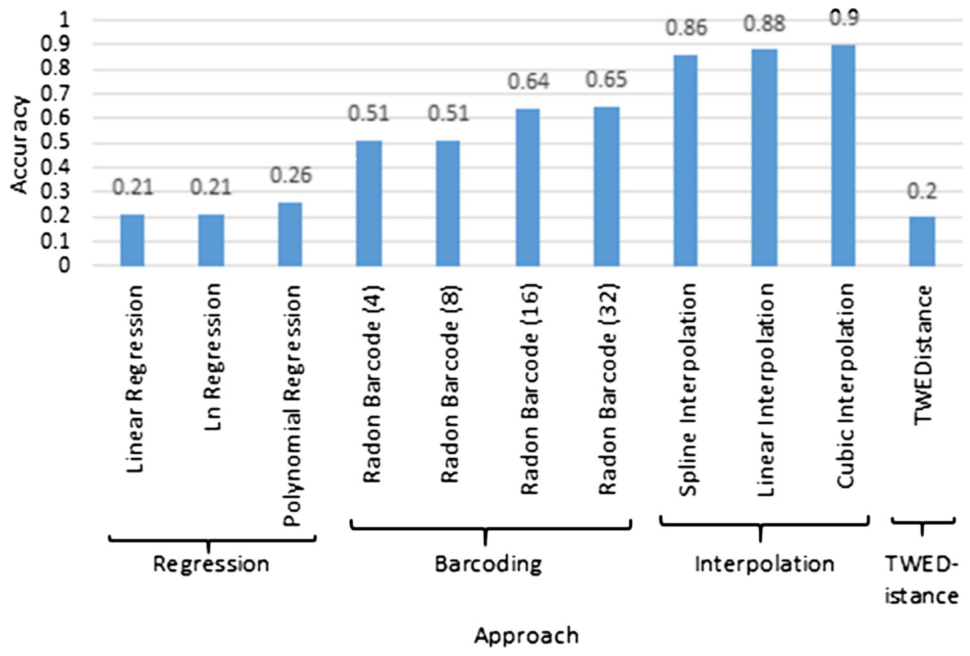


Fig. 16. Average accuracy.

Table 5
Complexity in big O notation.

Approach	Complexity	Description
TWED	$O(m * n)$	m and n are the length of the two trajectories
Barcoding	$O(l)$	l is the length of the trajectories' barcodes
Regression	$O(t^3)$	t is the size of the matrix to be solved
Interpolation	$O(n * m)$	m and n are the length of the two trajectories

6.5. Approaches complexity

In this section, complexities of the four approaches are presented in a tabular format in a Big O notation as shown in Table 5.

The complexity of TWED is O(n*m) as based on its previously mentioned algorithm there is a nested loop over the two trajectories with different lengths (m and n) to compute the distance. The barcoding compare barcodes of the same length (l), so there is only one loop over the two trajectories' barcodes and compare points at the same index to compute the hamming distance. There for the barcode complexity is O(l). Interpolation evaluates the distance over different trajectories length (n and m), it interpolates one trajectory's points into the approximated functions of the other trajectory to evaluate the distance as explained in the interpolation section. Therefore, interpolation complexity is $O(n * m)$. Regression complexity is the complexity of solving the matrix which is $O(t^3)$.

7. Conclusion and future work

Similarity evaluation between moving objects' trajectories is a very important aspect for many application domains. There are variety of similarity measures that evaluate similarity based on the discrete form of a trajectory. These measures are affected by the sampling rate differences. We used different approaches that build up the continuous form of a trajectory's data (Interpolation, Regression and Curve barcoding) and compared them with a discrete similarity based operator which is TWEDistance. We evaluated the four approaches on the trains synthetic dataset, the trucks and Geo-life real datasets. We measured their accuracy over the datasets based on spatial, direction and speed similarity. After evaluating the accuracy per each experiment, we evaluated the average accuracy and the results show that interpolation yields the highest accuracy results against the other approaches with an average accuracy up to 90%. For a future work, there is a need for an auto feature prediction generic continuous similarity operator that predicts what features to be used as a similarity criteria based on the input data.

References

[1] Sakr MA, Güting RH. Group spatiotemporal pattern queries. GeoInformatica 2014;18(4):699–746.
[2] Vlachos M, Gunopulos D, Das G. Rotation invariant distance measures for trajectories. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining. ACM; 2004. p. 707–12.
[3] Magdy N, Sakr MA, Mostafa T, El-Bahnasy K. Review on trajectory similarity measures. In: 2015 IEEE seventh international conference on intelligent computing and information systems (ICICIS). IEEE; 2015. p. 613–9.
[4] Wang H, Su H, Zheng K, Sadiq S, Zhou X. An effectiveness study on trajectory similarity measures. Proceedings of the twenty-fourth Australasian database conference, vol. 137. Australian Computer Society, Inc.; 2013. p. 13–22.
[5] Magdy N, Sakr MA, El-Bahnasy K. A generic trajectory similarity operator in moving object databases. Egypt Inform J 2017;18(1):29–37.
[6] Marteau P-F. Time warp edit distance with stiffness adjustment for time series matching. IEEE Trans Pattern Anal Mach Intell 2009;31(2):306–18.
[7] Ranacher P, Tzavella K. How to compare movement? A review of physical movement similarity measures in geographic information science and beyond. Cartogr Geogr Inform Sci 2014;41(3):286–307.
[8] Faloutsos C, Ranganathan M, Manolopoulos Y. Fast subsequence matching in time-series databases, vol. 23(2). ACM; 1994.
[9] Chen Y, Nascimento MA, Ooi BC, Tung AK. Spade: on shape-based pattern detection in streaming time series. In: 2007 IEEE 23rd international conference on data engineering. IEEE; 2007. p. 786–95.
[10] Nakamura T, Taki K, Nomiya H, Seki K, Uehara K. A shape-based similarity measure for time series data with ensemble learning. Pattern Anal Appl 2013;16(4):535–48.
[11] Alt H. The computational geometry of comparing shapes. In: Albers S, Alt H, Nher S, editors. Efficient algorithms. Springer; 2009. p. 235–48.
[12] Alt H, Behrends B, Blömer J. Approximate matching of polygonal shapes. Ann Math Artif Intell 1995;13(3–4):251–65.
[13] Chen L, Özsu MT, Oria V. Symbolic representation and retrieval of moving object trajectories. In: Proceedings of the 6th ACM SIGMM international workshop on multimedia information retrieval. ACM; 2004. p. 227–34.
[14] Li JZ, Ozsu MT, Szafron D. Modeling of moving objects in a video database. In: Proceedings, IEEE international conference on multimedia computing and systems' 97. IEEE; 1997. p. 336–43.

[15] Keogh E, Ratanamahatana CA. Exact indexing of dynamic time warping. Knowl Inform Syst 2005;7(3):358–86.
[16] Chen L, Ng R. On the marriage of lp-norms and edit distance. Proceedings of the thirtieth international conference on very large data bases, vol. 30. VLDB Endowment; 2004. p. 792–803.
[17] Vlachos M, Kollios G, Gunopulos D. Discovering similar multidimensional trajectories. In: Proceedings. 18th International conference on data engineering, 2002. IEEE; 2002. p. 673–84.
[18] Chen L, Özsu MT, Oria V. Robust and fast similarity search for moving object trajectories. In: Proceedings of the 2005 ACM SIGMOD international conference on management of data. ACM; 2005. p. 491–502.
[19] Little JJ, Gu Z. Video retrieval by spatial and temporal structure of trajectories. In: Photonics west 2001-electronic imaging. International Society for Optics and Photonics; 2001. p. 545–52.
[20] Güting RH, Böhlen MH, Erwig M, Jensen CS, Lorentzos NA, Schneider M, et al. A foundation for representing and querying moving objects. ACM Trans Database Syst (TODS) 2000;25(1):1–42.
[21] Forlizzi L, Güting RH, Nardelli E, Schneider M. A data model and data structures for moving objects databases, vol. 29(2). ACM; 2000.
[22] Que A. Polynomial regression; 2014. <http://polynomialregression.drque.net/math.html> [accessed: 2017-01-14].
[23] Lutus P. Polynomial regression; 2009. <http://arachnoid.com/sage/polynomial.html> [accessed: 2017-01-14].
[24] Gaussian elimination; 2016. <http://pdfonpoint.com/PDF/LEVEL%20200/CSC%20202%20Matlab/NUMERIC%20METHODS/GAUSS-JORDAN/Gauss_Jordan.pdf> [accessed: 2017-01-14].
[25] Que A. Non-linear functions that can be linearized for least-square regression; 2014. <http://www.drque.net/index.php?ArticleNumber=2872#_2872> [accessed: 2017-01-14].
[26] Trek S. Transformations to achieve linearity; 2016. <http://stattrek.com/regression/linear-transformation.aspx?Tutorial=AP> [accessed: 2017-01-14].
[27] Roberts D. Linear regression models; 2016. <http://mathbits.com/MathBits/TISection/Statistics2/IntroStat2.htm> [accessed: 2017-01-14].
[28] Matlab. Interpolation; 2016. <https://www.mathworks.com/moler/interp.pdf> [accessed: 2017-01-14].
[29] Fritsch FN, Carlson RE. Monotone piecewise cubic interpolation. SIAM J Numer Anal 1980;17(2):238–46.
[30] Iwashita Y. Opengamma quantitative research: piecewise polynomial interpolations; 2013. <https://developers.opengamma.com/quantitative-research/Piecewise-Polynomial-Interpolation-OpenGamma.pdf> [accessed: 2017-01-14].
[31] De Boor C, De Boor C, Mathématicien E-U, De Boor C, De Boor C. A practical guide to splines, vol. 27. New York: Springer Verlag; 1978.
[32] Jcooper. Splines; 2016. <http://www.math.umd.edu/jcooper/amsc460/spline.pdf> [accessed: 2017-01-14].
[33] Tizhoosh HR. Barcode annotations for medical image retrieval: a preliminary investigation. In: 2015 IEEE international conference on image processing (ICIP). IEEE; 2015. p. 818–22.
[34] Han J, Pei J, Kamber M. Data mining: concepts and techniques. Elsevier; 2011.
[35] Güting RH, Behr T, Düntgen C. Secondo: a platform for moving objects database research and for publishing and integrating research implementations. Fernuniv., Fak. für Mathematik u. Informatik; 2010.
[36] Pelekis N. Trucks revised; 2012. Available at <http://chorochronos.datastories.org/?q=node/10> [accessed: 2017-01-14].
[37] Zheng Y, Zhang L, Xie X, Ma W-Y. Mining interesting locations and travel sequences from gps trajectories. In: Proceedings of the 18th international conference on world wide web. ACM; 2009. p. 791–800.
[38] Zheng Y, Li Q, Chen Y, Xie X, Ma W-Y. Understanding mobility based on gps data. In: Proceedings of the 10th international conference on ubiquitous computing. ACM; 2008. p. 312–21.
[39] Zheng Y, Xie X, Ma W-Y. Geolife: a collaborative social networking service among user, location and trajectory. IEEE Data Eng Bull 2010;33(2):32–9.