A New Trajectory Similarity Measure for GPS Data

Anas Ismail
anas.ismail@kaust.edu.sa

Antoine Vigneron
antoine.vigneron@kaust.edu.sa

Visual Computing Center
King Abdullah University of Science and Technology (KAUST)
Thuwal 23955-6900, Saudi Arabia

ABSTRACT

We present a new algorithm for measuring the similarity between trajectories, and in particular between GPS traces. We call this new similarity measure the Merge Distance (MD). Our approach is robust against subsampling and supersampling. We perform experiments to compare this new similarity measure with the two main approaches that have been used so far: Dynamic Time Warping (DTW) and the Euclidean distance.

Categories and Subject Descriptors

I.5 [**PATTERN RECOGNITION**]: I.5.3 Clustering—*Algorithms, Similarity measures*

Keywords

Trajectory similarity measure, GPS trajectories, DTW

1. INTRODUCTION

Tracking devices are becoming more and more widely used, for instance in location based services, transportation system management, navigation planning, congestion control, vehicle fleet management, wildlife tracking... The amount of trajectory data is therefore increasing rapidly. In order to support these applications, databases management systems have been specifically designed for handling spatial trajectories [7, 18].

Many applications require to be able to perform classification or clustering on this type of data, for instance identifying traveling patterns or frequent behaviors, path planning, video surveillance [3, 6, 8, 9, 11]. Classification and clustering require a distance function to measure the similarity between two trajectories. Several such similarity measures are known, for instance Sum of Pairs Distance [1], Edit Distance on Real Sequences [2], Edit Distance with Real Penalty [4], Longest Common Subsequence [23], and Dynamic Time Warping [19]. Liu and Schneider [12] also considered measures that are not only geographic, but also semantic. Vlachos et al. presented another similarity measure that is robust against noise [16]. Surveys comparing these algorithms can be found in [13, 15, 17, 22], and an introduction is given in the book by Zheng and Zhou [24, Chapter 2.3.2].

In this paper, a trajectory p is a sequence of points $p_1, \ldots, p_n$. For instance, a GPS trace would be given as a sequence of triples (x_i, y_i, t_i), where x_i and y_i are the coordinates, and t_i is a timestamp. In this case, we have $p_i = (x_i, y_i)$. The *length* of this trajectory is $\ell(p) = \sum_i d(p_i, p_{i+1})$, where $d(p_i, p_{i+1})$ is the distance between p_i and p_{i+1}. We will not use the timestamps explicitly, but we will only assume that the points are given in chronological order, and hence $t_1 < \cdots < t_n$.

Our goal is to design a new distance function for measuring similarity between two trajectories that is suitable for GPS data. In particular, we would like it to be robust under subsampling and supersampling, as GPS devices only provide sampled points along the actual trajectory. Our new distance function, called the Merge Distance (MD), is based on the length of the shortest trajectory that is a supersequence of both trajectories. (See Section 2.) Intuitively, this length should be short when the two trajectories come from the same curve. The Merge Distance can be computed in quadratic time, like DTW. Then we perform an experimental comparison between MD, the Euclidean distance and DTW [19].

2. COMPUTING THE MERGE DISTANCE

In this section, we give a detailed description of our new similarity measure, that we call the *Merge Distance* (MD), and we give an efficient algorithm for computing it. We assume that two trajectories a and b are given as point sequences $a_1, \ldots, a_n$ and $b_1, \ldots, b_m$. We only need to assume that we can compute the distances between any two of these points in constant time. So for any $1 \leq i < i' \leq n$ and any $1 \leq j \leq j'$, we know $d(a_i, a_{i'})$, $d(b_j, b_{j'})$ and $d(a_i, b_j)$ where d is any metric—for instance, in our experiments, the trajectories a and b are sequences of points in $\mathbb{R}^2$, and we use the 2-dimensional Euclidean distance.

The *shortest supertrajectory* $s(a, b)$ through the sequence a, b is the trajectory with shortest length and such that a and b are subsequences of $s(a, b)$. (See Fig. 1, left.) its length is denoted by $\ell(a, b)$. We first compute $\ell(a, b)$ in quadratic time $O(mn)$ by the following dynamic programming approach.

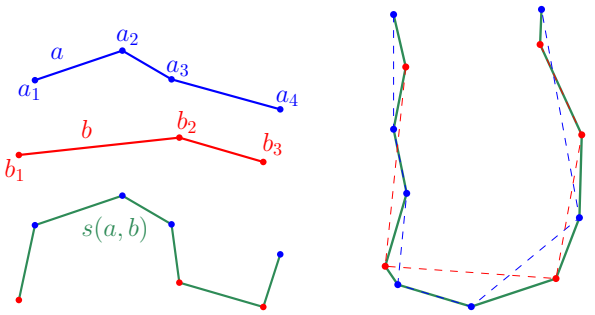

Figure 1: (Left) The green trajectory $s(a, b)$ is the shortest trajectory through a and b. (Right) The red and blue trajectories come from the same green path. The lengths of the red, green and blue path are approximately the same.

Let $a[1, i]$ and $b[1, j]$ denote the subtrajectories $(a_1, a_2, \ldots, a_i)$ and $(b_1, b_2, \ldots, b_j)$, respectively. We denote by A_i^j (resp. B_i^j) the length of the shortest trajectory that is a supersequence of $a[1, i]$ and $b[1, j]$, and such that its last point is a_i (resp. b_j). These quantities satisfy the following relations. For any $2 \leq i \leq m$ and $1 \leq j \leq n$, we have

$$A_i^j = \min\left(A_{i-1}^j + d(a_{i-1}, a_i), B_{i-1}^j + d(b_j, a_i)\right),$$

and for any $1 \leq i \leq m$ and $2 \leq j \leq n$, we have

$$B_i^j = \min\left(A_i^{j-1} + d(a_i, b_j), B_i^{j-1} + d(b_{j-1}, b_j)\right).$$

The boundary cases are given by the relations

$$A_1^j = \left(\sum_{k=1}^{j-1} d(b_k, b_{k+1})\right) + d(b_j, a_1), \text{ and}$$

$$B_i^1 = \left(\sum_{k=1}^{i-1} d(a_k, a_{k+1})\right) + d(a_i, b_1).$$

So we can compute all the values $A_{i,j}$ and $B_{i,j}$ by dynamic programming in $O(mn)$ time, and the length of the shortest supertrajectory is $\ell(a, b) = \min(A_m^n, B_m^n)$.

After computing the length $\ell(a, b)$ of the shortest supertrajectory, we obtain the merge distance $\mathrm{MD}(a, b)$ from the lengths $\ell(a)$ and $\ell(b)$ of the trajectories a and b by the following expression:

$$\mathrm{MD}(a, b) = \frac{2\ell(a, b)}{\ell(a) + \ell(b)} - 1.$$

We now briefly explain why we chose this expression. The merge distance $\mathrm{MD}(a, b)$ is at least as large as $\ell(a)$ and $\ell(b)$, so we normalize it by dividing it by the average of $\ell(a)$ and $\ell(b)$. After subtracting 1, this ensures that $\mathrm{MD}(a, b) \geq 0$ for any two trajectories a and b, and that $\mathrm{MD}(a, b) = 0$ when $a = b$. This expression also ensures that $\mathrm{MD}(a, b)$ is invariant under rigid motions. In addition, $\mathrm{MD}(a, b)$ is large when a and b are far apart, because in this case $\ell(a, b)$ is much larger than $\ell(a)$ and $\ell(b)$.

Intuitively, the merge distance $\mathrm{MD}(a, b)$ should be close to 0 when the trajectories a and b are sampled densely enough from the same curve, because in this case $\ell(a)$, $\ell(b)$, and $\ell(a, b)$ are roughly equal. (See Fig. 1, right.) It should therefore be appropriate for performing similarity search on GPS traces, as two GPS traces of the same actual trajectory are point sequences sampled from the same curve. This was the original motivation for this work.

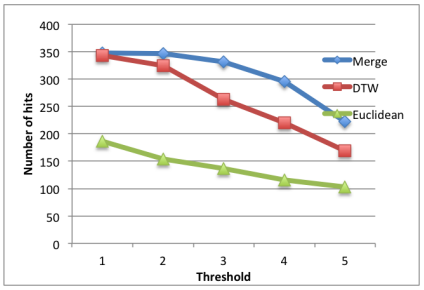

Figure 2: Subsampling experiment: Taxi data set consisting of 348 trajectories. A value v on the x-axis represents a threshold distance Δ of v percent of the total length of the trajectory.

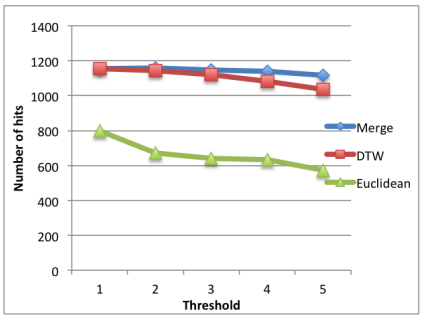

Figure 3: Subsampling experiment: Taxi data set consisting of 1157 trajectories.

3. EXPERIMENTS AND RESULTS

In order to test our algorithm, we conducted several experiments on two real data sets, following the recommendations of Keogh and Kasetty [10]. We tested our algorithm against DTW and Euclidean distance.

Similarity measures. The Euclidean distance between two trajectories a and b is $\sum_{i=1}^{n} d(a_i, b_i)$ where d denote the 2-dimensional Euclidean distance (we need to assume that the two trajectories have the same number of points $n = m$). DTW [14] is a commonly used similarity measure for time series which, similarly to our method, can be computed in quadratic time by dynamic programming, but the quantity that is minimized is the sum of the distances $d(a_i, b_j)$ over all pairings (a_i, b_j), and not the length of the supertrajectory.

Data sets. We used two data sets of real trajectories. The *truck data set* comes from a fleet of trucks[5]. The advantage of this data set is that it is clustered into two subsets, which we use it in our classifications experiments. The *taxi data set* is the T-Drive data set [20, 21]. It contains data for thousands of taxis and millions of data points. We divided the data into different sets of subtrajectories.

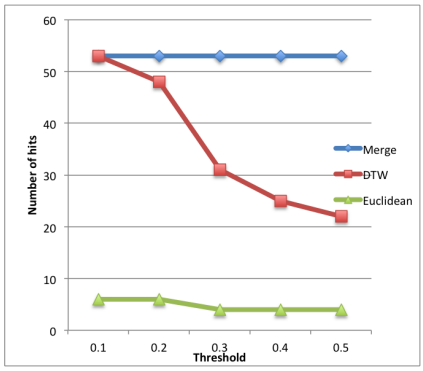

Figure 4: Subsampling experiment: First truck data set consisting of 53 trajectories.

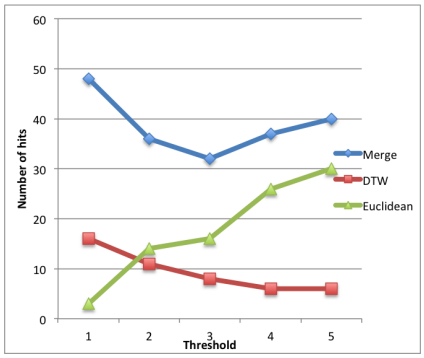

Figure 5: Subsampling experiment: Second truck data set consisting of 53 trajectories.

Subsampling experiments. In our first set of experiments, we compare the three similarity measures (Euclidean, DTW and MD) under subsampling, and using taxi and truck data sets. Our subsampling depends on a parameter Δ. The query q (see below) is subsampled as follows: We pick the first point q_1, then the first point q_i, $i > j$ that is at distance at least Δ from q_1, and we repeat the process until we reach the end of the query trajectory q. Instead of using this fixed threshold Δ, the trajectories in the data set are subsampled using a random threshold taken from a random Gaussian variable, with mean 0, standard deviation Δ, and ignoring negative values. This random variable is recomputed at each step, that is, for each point of the subsampled trajectory. (We use this random threshold so that the queries are not subtrajectories of the data, nor the converse.)

For each query trajectory q, we compute the subsampled version q' of q as described above. We then find, by brute force, the trajectory $n(q')$ in the whole subsampled data set that is closest to q'. If this point $n(q')$ is equal to q, we call it a hit, and otherwise a miss. We repeat this process for each possible query trajectory q in our data set, and count the number of hits.

We varied the threshold distance and counted the number of hits. We compared against DTW and the Euclidean Algorithms. The performance of MD was superior as shown in figures 2, 3, 4 and 5. The graphs show the number of hits for MD, DTW, and the Euclidean distance. MD outperforms

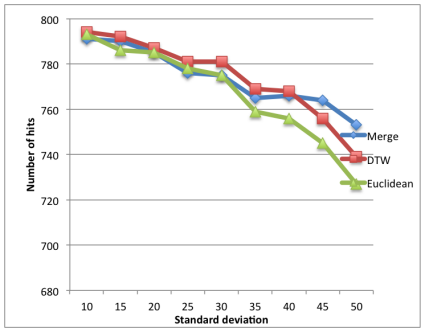

Figure 6: Adding Gaussian noise, taxi data set consisting of 796 trajectories. The value v on the x-axis means that we added random Gaussian noise with mean 0 and standard deviation $v\times$ average distance between two consecutive points in the trajectory.

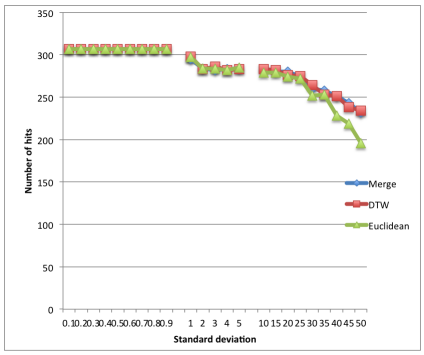

Figure 7: Adding Gaussian noise, taxi data set consisting of 307 trajectories.

the other two similarity measures, which is not surprising as we designed it specifically to be robust to subsampling.

Classification experiment. In this experiment, we only use the truck data set, because the taxi data set is not clustered. The trucks were given in two clusters. The experiment goes as follows. We first pick a query trajectory q in the data set. We compute the distance between q and all the other trajectories in D. Let $n(q) \in D \setminus \{q\}$ denote the closest such trajectory. If the label of $n(q)$ is the same as q, we call it a hit, and otherwise a miss. We repeat this process over all possible query trajectories $q \in D$, and we count the number of hits. Out of 53 trajectories, MD achieved 47 hits which is the same as DTW.

Adding Gaussian noise. In this experiment, instead of changing the resolution, we modify the query trajectory q by adding Gaussian noise with zero mean and fixed standard deviation to each point. The standard deviation is a multiple of the average distance between two consecutive points in the trajectory. We thus obtain a perturbed trajectory q', and we find the nearest point $n(q')$ in the dataset. If $n(q') = q$, we call it a hit. The results of the experiments can be found in figures 6 and 7. The results show that DTW and MD are comparable, and outperform the Euclidean distance in this experiment.

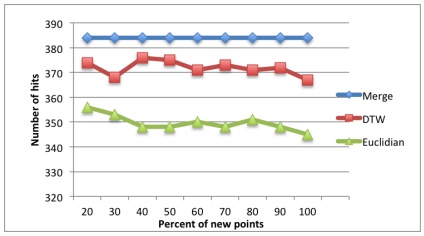

Figure 8: Supersampling experiment. Taxi data set consisting of 384 trajectories. The value on the x-axis is the percentage of new points in the trajectory.

Supersampling experiment. In this experiment, we insert new points along the query trajectory q. These points are added as follows: Pick two consecutive points at random in the trajectory, and insert the midpoint. Repeat this process until the desired number of points is reached; We denote by q' the supersampled trajectory obtained this way. Then we find the closest trajectory $n(q)$ to q' in the data set according to the Euclidean distance, DTW, and MD. If $n(q) = q$, we call it a hit.

The results of this experiment appear in Figure 8. MD gives a perfect answer, as the supersampled trajectory q' has the same length as the original trajectory q. DTW does not always give the right answer, and the Euclidean distance gives worse results.

4. CONCLUSION

In this paper we presented a new distance function, the merge distance (MD), for measuring trajectory similarity. We benchmarked it against DTW and the Euclidean distance through several experiments on real datasets. The results indicate that MD is robust under subsampling and supersampling, and that it has comparable results with DTW in several other settings. At this point, we do not know whether MD is a metric or not.

5. ACKNOWLEDGEMENTS

Anas Ismail was supported by KAUST base funding.

6. REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Proc. 4th International Conference on Foundations of Data Organization and Algorithms*, FODO '93, pages 69–84, 1993.

[2] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *Proc. ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 491–502, 2005.

[3] Z. Chen, H. T. Shen, and X. Zhou. Discovering popular routes from trajectories. In *Proc. 27th IEEE International Conference on Data Engineering*, ICDE '11, pages 900–911, 2011.

[4] Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie. Searching trajectories by locations: An efficiency study. In *Proc. ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 255–266, 2010.

[5] E. Frentzos. R-tree portal truck data. `http://www.rtreeportal.org`. Online; accessed December-2014.

[6] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag. Adaptive fastest path computation on a road network: A traffic mining approach. In *Proc. 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 794–805, 2007.

[7] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Trans. Database Syst.*, 25(1):1–42, 2000.

[8] W. Hu, D. Xie, Z. Fu, W. Zeng, and S. Maybank. Semantic-based surveillance video retrieval. *IEEE Trans. Image Processing*, pages 1168–1181, 2007.

[9] H. Jeung, Q. Liu, H. T. Shen, and X. Zhou. A hybrid prediction model for moving objects. In *Proc. 24th IEEE International Conference on Data Engineering*, ICDE '08, pages 70–79, 2008.

[10] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Min. Knowl. Discov.*, 7(4):349–371, Oct. 2003.

[11] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: A partition-and-group framework. In *Proc. ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 593–604, 2007.

[12] H. Liu and M. Schneider. Similarity measurement of moving object trajectories. In *Proc. 3rd ACM SIGSPATIAL International Workshop on GeoStreaming*, IWGS '12, pages 19–22, 2012.

[13] B. Morris and M. Trivedi. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '09, pages 312–319, 2009.

[14] C. Myers, L. Rabiner, and A. Rosenberg. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(6):623–635, 1980.

[15] K. Toohey and M. Duckham. Trajectory similarity measures. *SIGSPATIAL Special*, 7(1):43–50, 2015.

[16] M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering similar multidimensional trajectories. In *Proc. 18th International Conference on Data Engineering*, ICDE '02, pages 673–684, 2002.

[17] H. Wang, H. Su, K. Zheng, S. Sadiq, and X. Zhou. An effectiveness study on trajectory similarity measures. In *Proc.24th Australasian Database Conference*, ADC '13, pages 13–22, 2013.

[18] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: issues and solutions. In *Proc. 10th International Conference on Scientific and Statistical Database Management*, pages 111–122, 1998.

[19] B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proc. 14th International Conference on Data Engineering*, ICDE '98, pages 201–208, 1998.

[20] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proc. 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 316–324, 2011.

[21] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: Driving directions based on taxi trajectories. In *Proc. 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pages 99–108, 2010.

[22] Z. Zhang, K. Huang, and T. Tan. Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In *Proc. 18th International Conference on Pattern Recognition*, ICPR '06, pages 1135–1138, 2006.

[23] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proc. 18th International Conference on World Wide Web*, WWW '09, pages 791–800, 2009.

[24] Y. Zheng and X. Zhou. *Computing with Spatial Trajectories*. Springer, November 2011.