Compressing Trajectory for Trajectory Indexing

Kaiyu Feng
Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly,
Interdisciplinary Graduate School,
Nanyang Technological University
kfeng002@e.ntu.edu.sg

Zhiqi Shen
Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly,
Interdisciplinary Graduate School,
Nanyang Technological University
zqshen@ntu.edu.sg

ABSTRACT

Nowadays, as many devices like mobile phones and smart watch/band are equipped with GPS-devices, a large volume of trajectory data is generated every day. With the availability of such trajectory data, many mining tasks have been proposed and investigated in the past decade. Since the raw trajectory data is usually very large, it is a big challenge to analyse and mine the raw data directly. In order to address this issue, a branch of research has been done to compress the trajectory data. This paper surveys recent research about trajectory compression. An overview of existing techniques for trajectory compression is provided.

CCS CONCEPTS

• **Information systems** → *Data cleaning*; *Geographic information systems*;

KEYWORDS

Trajectory, Trajectory Compressing, Survey

1 INTRODUCTION

Nowadays, more and more devices (e.g., mobile phones, smart watches/bands, auto navigators, etc) are equipped with GPS sensors. As a result, a large volume of trajectory data is generated every day. The availability of such trajectory data enables us to perform various mining tasks to understand the patterns and properties of the moving objects. Applications in location-based social networks and urban computing benefit from the knowledge discovered from the trajectory data. Common trajectory mining tasks can be divided into the following categories: trajectory pattern mining, trajectory uncertainty, outlier detection and classification [28].

However, before we analyze and mine the trajectory data, there are some issues that we need to address. As the trajectory data is usually collected from GPS devices periodically, the raw trajectory data is usually extremely large. For example, assume that trajectory data of a taxi is collected from its GPS every 1 second. If there are 10k taxis in the city, the size of the collected trajectories for just one day is 20 GB. Apparently, it takes too much network bandwidth and space to transmit and store the raw trajectory data. Moreover, it takes too much time for existing query processing and mining algorithms to process the raw trajectory data. As a result, an important step before we use the trajectory data is *trajectory compression*.

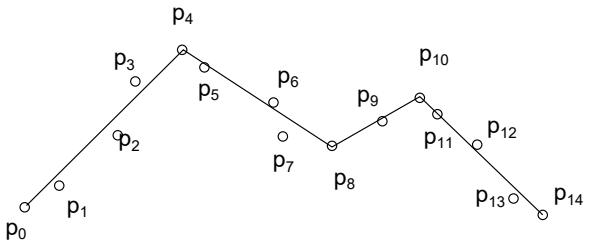

Figure 1: Trajectory Data.

Example. Consider the trajectory in Figure 1. The trajectory consists of fifteen points, i.e., $[p_0, \ldots, p_{15}]$. After compression, points $\{p_0, p_4, p_8, p_{10}, p_{14}\}$ are retained while the other points are discarded. The compressed trajectory contains four line segments, i.e., $\overline{p_0p_4}$, $\overline{p_4p_8}$, $\overline{p_8p_{10}}$, and $\overline{p_{10}p_{14}}$.

Trajectory compression aims to compress the size of a trajectory while maintaining the utility of the trajectory. The trajectory data is usually generated by periodically collected from GPS devices. To capture the trajectory of a moving object more accurately, we usually use a high sampling rate to collect its position. Thus, a large volume of trajectory data will be generated every day. Since the raw trajectory data is large and it is hard to completely load them into memory to perform any mining algorithms, a common idea is to compress the trajectory data to reduce the storage requirements.

An ideal compression method should achieve good compression ratio while has no loss of precision. However, the existing methods always make a trade-off between the compression ratio and the precision. Thus, existing trajectory compression methods can be divided into two categories: lossless compression and lossy compression.

In this paper, we review some recent works about trajectory compression. The remaining of the paper is organized as follows: In Section 2, we introduce the structure of trajectory data. Then we discuss lossless trajectory compression in Section 3. Next, research about lossy trajectory compression is reviewed in Section 4. We conclude our paper in Section 6.

2 TRAJECTORY DATA

Trajectory data corresponds to the mobility trace of moving objects like human, vehicle, animals. A trajectory is usually represented by a sequence of tuples $T = [t_1, \ldots, t_n]$. Each tuple t_i consists of a location $t_i.p$ and a time stamp $t_i.t$. In some applications, each tuple t_i is further associated with some additional information like keywords. A trajectory $T = [t_1, \ldots t_n]$ implies that the object is located at position $t_i.p$ at time $t_i.t$, for any $i \in [1, n]$. A common assumption is that the object moves along the line segment between $t_i.p$ and $t_{i+1}.p$ from time $t_i.t$ to time $t_{i+1}.t$.

An example of trajectory data is shown in Figure 1. The trajectory consists of fifteen poinst, $[p_0, \ldots, p_{15}]$.

3 LOSSLESS COMPRESSION

Lossless compression methods enable exact reconstruction of the original data from the compressed data without loss of precision. For example, delta compression scheme [6, 19] is a lossless compression method. The idea of delta compression is to store a delta for each timestamp in the trajectory. Specifically, the delta d_i for the i-th timestamp equals $p_i - p_{i-1}$ where p_i and p_{i-1} are the location at timestamp i and $i-1$, respectively. Since the deltas are usually small, they can be encoded to save space [6]. Then Nibali et al. [19] propose a system which has a better compression ratio. Specifically, compared with [6], the system in [19] uses a dynamic encoding scheme instead of a static leading zero encoding schemes. Moreover, the system in [19] can also be applied to perform lossy compression.

Though this compression method is simple and straightforward, the limitation of lossless compression lies in that its compression ratio is relatively poor.

4 LOSSY COMPRESSION

In order to achieve a better compression ratio, various lossy compression methods, which allow errors or derivations from the original trajectory, are proposed. Such methods typically select some important data points from the original trajectory. Then by joining the selected points, a series of line segments are generated to represent the original trajectory. Other unselected data points are discarded since they are redundant to the line segments. By ensuring that the disregarded points are within a range of the line segments, they can achieve a error threshold.

Many lossy compression methods have been developed since they have good compression ratio with bounded errors. The lossy compression methods can be classified into two categories: line generalization based methods and semantics based methods. We next review the two categories of compression methods, respectively.

4.1 Error Measures

The lossy compression methods aim to reduce the size without compromising much of the precision. A number of error measures have been defined. We next review

4.1.1 Perpendicular Euclidean Distance. We first illustrate the perpendicular Euclidean distance with the example in Figure 2. As shown in the figure, the original trajectory is $[p_1, p_2, p_3, p_4]$ where p_i is the location of the moving object at time t_i. The compressed trajectory is $[p_1, p_4]$, which is reduced from the original trajectory.

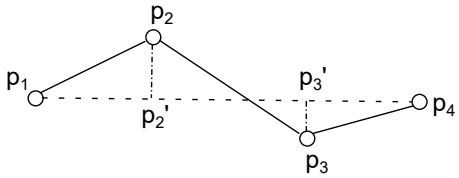

Figure 2: Perpendicular Euclidean distance.

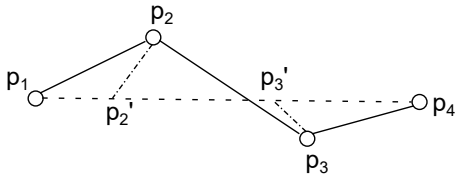

Figure 3: Time Synchronized Euclidean Distance.

The data points in the original trajectory are projected to the line segment $\overline{p_1p_4}$ in the compressed trajectory. Figure2 shows that p_2' and p_3' on segment $\overline{p_1p_4}$ are the projection of p_2 and p_3, respectively. The perpendicular Euclidean distance is the summation of the distances between each point from the original trajectory and its projection, i.e., $|\overline{p_2p_2'}| + |\overline{p_3p_3'}|$.

4.1.2 Time Synchronized Euclidean Distance. The perpendicular Euclidean distance were not suitable for trajectory compression because it does not consider the temporal information. Therefore, another error measure, named time synchronized Euclidean distance, measure the errors through distances between pairs of temporally synchronized positions. Specifically, as shown in Figure 3, it assumes the object moves with a constant speed between p_1 and p_4. Point p_2' and p_3' are the projection of p_2 and p_3 based on their timestamp.

4.1.3 Other Measures. Besides the two most widely used measures, other distance metrics are also considered. For example, speed information[18], direction information [15] are preserved in the error measure. In [22], position, speed and direction information are all considered in the error measure.

4.2 Line generalization based methods

Line generalization based methods are one of the most common trajectory compression methods. The line generalization problem [16] is first investigated in the cartography community. The cartographers want to extract features from detailed data and represent them using simple and readable maps. Given an ordered set of $n+1$ points, $v_0, \ldots, v_n$, let C be a series of line segment $\overline{v_0v_1}, \ldots, \overline{v_{n-1}v_n}$ joining the $n+1$ points. The line generalization problem aims to find a set C' containing less line segments which approximates C within an acceptable error margin. Based on how the trajectory data is processed, the literature about line generalization based methods can be further classified into offline compression and online compression. The offline compression takes as input all points in the trajectory data that are generated, while the online compression process a data point instantly once it is generated.

4.2.1 Offline Compression. Given a fully generated trajectory, offline compression generates an approximated trajectory by discarding some redundant points from the original trajectory. The offline compression can be either top-down or bottom up. The top-down offline compression divide the trajectory recursively. Specifically, it finds a position in a trajectory to split the whole trajectory into two sub-trajectories. This process is repeated until the stoping condition is met. One of the most well-known top-down line generalization based method is Douglas-Peucker (DP) algorithm [7]. The idea of the DP algorithm is to replace the original trajectory by an approximate line segment. If the replacement does not meet specified error tolerance, it select the point contributing the biggest error as the splitting point. This process continues until the error between the approximation and the original trajectory is below a specified error. Hershberger et al. [10] further improve the time complexity of the compression algorithm to $O(N \log N)$ from $O(N^2)$, where N is the number of points in a trajectory. Bellman [1] employs a dynamic programming technique with a complexity of $O(N^3)$ to guarantee that the approximated trajectory is optimal. These methods use *perpendicular euclidean distance* as the distance metrics to measure the error of a compression. As the *perpendicular euclidean distance* ignores the temporal information, these algorithms have a limitation on trajectory compression. In order to utilize the temporal information, Meratnia et al. [17] utilize *time synchronized euclidean distance* as the distance metric to measure the error of the compression. Long et al. [15] study the problem of direction preserving trajectory compression. The direction information is defined as the angle between a vector from v to v' and the x-axis when an object moves from v to v'.

On the contrary, the bottom-up offline compression recursively merge two adjacent line segments. Specifically, by discarding the point p bridging the two segments, it creates a line segment connecting the non-bridging end points of the two segments. The process is repeated until discarding any point violates the error tolerance. In order to consider the temporal information, a fast $O(N)$ polygonal approximation algorithm [3] is proposed. Its error measure is extended from *local integral square error* and *integral square error*, which can be evaluated efficiently in $O(1)$ time with precalculated information. Potamias et al. [22] propose two one-pass compression algorithms based on sampling which utilizes the spatial locality and temporal timeliness inherent in trajectory streams. Muckell et al. [18] also propose a one-pass trajectory compression method which further preserves speed information.

4.2.2 Online Compression. The offline compression methods require that the all trajectory points are fully generated. However, in many real life applications, the trajectory data should be compressed and transmitted immediately. Many online trajectory compression methods have been proposed to determine whether a newly collected point should be retained in a trajectory. Open window approach [17] is an incremental algorithm. It starts with a window that contains the first three data points. As long as all distances of intermediate data points are below the distance threshold, it attempts to move the end of the window to the next point. When the threshold is going to be exceeded, it uses a heuristic strategy to select a point inside the window, and makes it the start point of the next segment. If the threshold is not exceeded, it moves the end of the window to the next point in the trajectory. Another window-based algorithm is the SWAB algorithm [12]. It keeps a buffer window whose size is initially chosen to contain enough points to create 5–6 segments. In addition, it imposes upper and lower bounds on the window such that the window size cannot be too large or too small. In the algorithm, the bottom-up algorithm is applied to the window to merge the points to a set of segments. Then they take out the first segment in the window, and grow the window by incorporating a set of following points whose approximate ratio does not exceed a given threshold. This process is repeated until all the data points are processed. The SWAB algorithm combines the sliding window algorithm and the bottom-up algorithm together. It can do the segmentation in an online manner, and generate segments with a "semi-global" view to avoid excessive fragmentation. Dead-Reckoning [13] is also an online algorithm. It reads each position sequentially and determines whether this position is discarded or not according to a heuristic criterion. In [14], a one-pass error bounded algorithm is proposed. The algorithm scans each data point in a trajectory once and only once. They further extend the algorithm to a aggressive version, which allows interpolating new data points into a trajectory under certain conditions.

4.3 Semantic Meaning Preserved Compression

A branch of research aims to keep not only the structure and the shape of the whole trajectory but also the semantic meanings, e.g., the places where the user stayed, took photos or watching something attractive, etc. For instance, in [5], semantic meanings are considered when reducing redundant points from the trajectory. They first divide a trajectory into walking and non-walking segments. A point is weighted by its heading change degree and the distance to its neighbours. Richter et al. [24, 25] introduces the concept of semantic trajectory compression (STC). In STC, a semantic representation of the trajectory that consists of reference points localized in a transportation network replaces raw, highly redundant position information.

In the line generalization based methods, the data points in the trajectory can be located at any position. However, the trajectories of certain moving objects such as cars and trucks are constrained by road networks. These vehicles always travel along road networks instead of the line segment between two points. A series of research considers trajectory compression with the constraints of road networks [8, 11, 26]. These methods project data points in the trajectory onto roads. This enables us to reduce the redundant points on the same road segment, which helps achieve a better compression ratio. [11] first combines map matching and trajectory compression together and study the problem of the compression of a moving object's trajectory keeping it at the same time matched on the underlying road network. In [26], spatial representation of a trajectory is separated from its temporal representation. The spatial and temporal information of trajectories are compressed separately. The spatial compression combines frequent sequential pattern mining techniques with Huffman Coding to reduce the size of a trajectory to save storages. In [8], a path in road networks is encoded as a subsequence of vertices such that the path can be uniquely reconstructed from the vertices by computing for each pair of consecutive vertices a shortest path and concatenating these paths. In this branch of research, an important task in the trajectory compression with constraints of road

networks is map matching. Map matching is a process to convert a sequence of raw latitude/longitude coordinates to a sequence of road segments. In order to assign data points in a trajectory data to a road segment, a series of methods utilizes additional information. Some of them use geometric information[9] (like matching a point to the nearest road). Some of them utilize the connectivity of road network [4, 27]. The GPS noise is also considered to select the best path from multiple possible paths [20, 21, 23]. In addition, a group of methods match a point to a road segment based on the range of sampling points [2].

5 FUTURE DIRECTIONS

Trajectory compression is one of the most fundamental operations for trajectory mining. We proceed to discuss some factors to be considered in trajectory mining.

6 CONCLUSION

With the widely use of GPS equipped devices, a large scale of trajectory data is generated every day. The availability of the trajectory data enables us to discover knowledge by performing various mining tasks. Before we analyse and mine the data, an import step is trajectory compression, since the raw trajectory data is costly to transmit and store. This paper surveys recent research about trajectory compression. An overview of existing techniques for trajectory compression is provided.

ACKNOWLEDGMENT

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative.

REFERENCES

[1] Richard Bellman. 1961. On the approximation of curves by line segments using dynamic programming. *Commun. ACM* 4, 6 (1961), 284.

[2] Sudarshan S Chawathe. 2007. Segment-based map matching. In *Intelligent Vehicles Symposium, 2007 IEEE*. IEEE, 1190–1197.

[3] Minjie Chen, Mantao Xu, and Pasi Franti. 2012. A fast $o(n)$ multiresolution polygonal approximation algorithm for gps trajectory simplification. *IEEE Transactions on Image Processing* 21, 5 (2012), 2770–2785.

[4] W Chen, M Yu, ZL Li, and YQ Chen. 2003. Integrated vehicle navigation system for urban applications. (2003).

[5] Yukun Chen, Kai Jiang, Yu Zheng, Chunping Li, and Nenghai Yu. 2009. Trajectory simplification method for location-based social networking services. In *Proceedings of the 2009 International Workshop on Location Based Social Networks*. ACM, 33–40.

[6] Philippe Cudre-Mauroux, Eugene Wu, and Samuel Madden. 2010. Trajstore: An adaptive storage system for very large trajectory data sets. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*. IEEE, 109–120.

[7] David H Douglas and Thomas K Peucker. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization* 10, 2 (1973), 112–122.

[8] Ranit Gotsman and Yaron Kanza. 2015. A dilution-matching-encoding compaction of trajectories over road networks. *GeoInformatica* 19, 2 (2015), 331–364.

[9] Joshua S Greenfeld. 2002. Matching GPS observations to locations on a digital map. In *81th annual meeting of the transportation research board*, Vol. 1. 164–173.

[10] John Edward Hershberger and Jack Snoeyink. 1992. *Speeding up the Douglas-Peucker line-simplification algorithm*. University of British Columbia, Department of Computer Science.

[11] Georgios Kellaris, Nikos Pelekis, and Yannis Theodoridis. 2009. Trajectory compression under network constraints. *Advances in Spatial and Temporal Databases* (2009), 392–398.

[12] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. 2001. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, 289–296.

[13] Ralph Lange, Tobias Farrell, Frank Durr, and Kurt Rothermel. 2009. Remote real-time trajectory simplification. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*. IEEE, 1–10.

[14] Xuelian Lin, Shuai Ma, Han Zhang, Tianyu Wo, and Jinpeng Huai. 2017. One-pass error bounded trajectory simplification. *Proceedings of the VLDB Endowment* 10, 7 (2017), 841–852.

[15] Cheng Long, Raymond Chi-Wing Wong, and HV Jagadish. 2013. Direction-preserving trajectory simplification. *Proceedings of the VLDB Endowment* 6, 10 (2013), 949–960.

[16] Robert B McMaster. 1986. A statistical analysis of mathematical measures for linear simplification. *The American Cartographer* 13, 2 (1986), 103–116.

[17] Nirvana Meratnia and A Rolf. 2004. Spatiotemporal compression techniques for moving point objects. In *International Conference on Extending Database Technology*. Springer, 765–782.

[18] Jonathan Muckell, Jeong-Hyon Hwang, Vikram Patil, Catherine T Lawson, Fan Ping, and SS Ravi. 2011. SQUISH: an online approach for GPS trajectory compression. In *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*. ACM, 13.

[19] Aiden Nibali and Zhen He. 2015. Trajic: An effective compression system for trajectory data. *IEEE Transactions on Knowledge and Data Engineering* 27, 11 (2015), 3138–3151.

[20] Washington Y Ochieng, Mohammed Quddus, and Robert B Noland. 2003. Map-matching in complex urban road networks. *Revista Brasileira de Cartografia* 2, 55 (2003).

[21] Oliver Pink and Britta Hummel. 2008. A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*. IEEE, 862–867.

[22] Michalis Potamias, Kostas Patroumpas, and Timos Sellis. 2006. Sampling trajectory streams with spatiotemporal criteria. In *Scientific and Statistical Database Management, 2006. 18th International Conference on*. IEEE, 275–284.

[23] Mohammed A Quddus, Robert B Noland, and Washington Y Ochieng. 2006. A high accuracy fuzzy logic based map matching algorithm for road transport. *Journal of Intelligent Transportation Systems* 10, 3 (2006), 103–115.

[24] Kai-Florian Richter, Falko Schmid, and Patrick Laube. 2012. Semantic trajectory compression: Representing urban movement in a nutshell. *Journal of Spatial Information Science* 2012, 4 (2012), 3–30.

[25] Falko Schmid, Kai-Florian Richter, and Patrick Laube. 2009. Semantic trajectory compression. *Advances in Spatial and Temporal Databases* (2009), 411–416.

[26] Renchu Song, Weiwei Sun, Baihua Zheng, and Yu Zheng. 2014. PRESS: A novel framework of trajectory compression in road networks. *Proceedings of the VLDB Endowment* 7, 9 (2014), 661–672.

[27] Huabei Yin and Ouri Wolfson. 2004. A weight-based map matching method in moving objects databases. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*. IEEE, 437–438.

[28] Yu Zheng. 2015. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 3 (2015), 29.