Algorithms for Compressing GPS Trajectory Data: An Empirical Evaluation

Jonathan Muckell
Dept. of Informatics
University at Albany–SUNY
Albany, NY 12222
jonmuckell@gmail.com

Jeong-Hyon Hwang
Dept. of Computer Science
University at Albany–SUNY
Albany, NY 12222
jhh@cs.albany.edu

Catherine T. Lawson
Dept. of Geography & Planning
University at Albany–SUNY
Albany, NY 12222
lawsonc@albany.edu

S. S. Ravi
Dept. of Computer Science
University at Albany–SUNY
Albany, NY 12222
ravi@cs.albany.edu

ABSTRACT

The massive volumes of trajectory data generated by inexpensive GPS devices have led to difficulties in processing, querying, transmitting and storing such data. To overcome these difficulties, a number of algorithms for compressing trajectory data have been proposed. These algorithms try to reduce the size of trajectory data, while preserving the quality of the information. We present results from a comprehensive empirical evaluation of many compression algorithms including Douglas-Peucker Algorithm, Bellman's Algorithm, STTrace Algorithm and Opening Window Algorithms. Our empirical study uses different types of real-world data such as pedestrian, vehicle and multimodal trajectories. The algorithms are compared using several criteria including execution times and the errors caused by compressing spatio-temporal information, across numerous real-world datasets and various error metrics.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Applications—*Spatial databases and GIS*

General Terms

Algorithms and Applications

Keywords

Trajectories, Compression, Error Metrics

1. INTRODUCTION

In recent years, the number of GPS-enabled devices sold has drastically increased, following an impressive exponential trend. Canalys, an information technology firm that studies market patterns, reported a 116% increase in the number of GPS units sold between 2006 and 2007 [4]. Location-based services and applications built from GPS-equipped mobile devices is a rapidly expanding consumer market. In 2009, there was an estimated 27 million GPS-equipped smart phones sold, bringing the world-wide GPS user-base to at least 68 million in the consumer market alone [5].

Data generated from GPS units are commonly used in a variety of business and public sector applications, such as supply-chain management and traffic modeling [7, 9, 12]. These efforts are being hampered by the sparse nature of data collection strategies, the sheer volume of the data, and technical issues associated with the use of the data. The enormous volume of data can easily overwhelm human analysis. If data is collected at 10 second intervals, a calculation due to Meratnia and de By [11] shows that without any data compression, 100 Mb of storage capacity is required to store just over 400 objects for a single day. This motivates the need for automated methods to compress and analyze the data. As a result, database support for storing and querying GPS traces is an area of active research [1, 15].

Transportation mode is defined as the method of traveling between locations (e.g, walking, bus, rail or airplane). The characteristics of a GPS trace (i.e., changes in direction, accuracy, speed and acceleration) differ substantially based on the transportation mode of the moving object. In this empirical study, we identify three data profiles based primarily on the transportation mode: buses, urban pedestrians and multimodal travel (involving modes such as walking, vehicle, subway or rail). Understanding how well each algorithm compresses traces from different transportation modes assists in matching the appropriate compression technique to business and organizational requirements.

Various metrics have been previously defined in the literature to measure the information loss associated with compression. However, to the best of our knowledge, no previous work has compared these metrics across a wide range of compression algorithms and across different transportation modes. In this research, seven different compression algorithms presented in the literature are compared on the basis of their actual execution times as well as several error metrics. These metrics allow the determination of the strengths and weakness of each algorithm on the different transportation modes.

Table 1: Summary of GPS Trajectory Algorithms

Algorithm	Running Time	Error Metrics
Uniform Sampling	$O(n)$	N/A
Douglas-Peucker	$O(n \log n)$	Spatial Distance
TD-TR	$O(n^2)$	Time Distance Ratio
OPW-TR	$O(n^2)$	Time Distance Ratio
OPW-SP	$O(n^2)$	Time Distance Ratio, Max Speed
Bellman's	$O(n^2)$	Spatial Distance
STTrace	$O(n^2)$	Synchronized Euclidean Distance, Heading, Speed

A more detailed version of this paper is available as a technical report [13].

2. TRAJECTORY-SPECIFIC ERROR METRICS

Algorithms for compressing GPS trajectories attempt to minimize one or more of the following error metrics: spatial distance, synchronized Euclidean distance (sed), time-distance ratio, speed and heading (which indicates the direction of the moving object). Table 1 lists the seven algorithms compared in this study, along with their worst-case running times and the metric(s) they attempt to minimize. The metrics considered by the algorithms serve as a logical mechanism for distinction and classification.

Synchronized Euclidean distance (sed) measures the distance between two points at identical time stamps [14]. To quantify the error introduced by the missing points, distance is measured at the identical time steps. The total error is measured as the sum of the distance between all points at the synchronized time instants, as shown below.

$$sed = \sum_{i=1}^{n} \sqrt{(x_{t_i} - x'_{t_i})^2 + (y_{t_i} - y'_{t_i})^2}$$

In the above expression, (x_{t_i}, y_{t_i}) and (x'_{t_i}, y'_{t_i}) represent the coordinates of a moving object at time t_i in the uncompressed and compressed traces respectively. Also, n represents the total number of points considered.

Another trajectory error metric is the time-distance ratio proposed by Meratnia and de By [11]. This metric uses both spatial and temporal information to determine whether to store or discard points. The temporal component is the ratio of two time intervals: the travel time across the original trajectory and the travel time across the approximation. The spatial component is the positions of discrete points from both trajectories. Various compression techniques such as the opening window algorithms OPW-TD and OPW-SP, and the modified version of Douglas-Peucker TD-TR, use the above time-distance ratio in order to select points for compression.

3. OVERVIEW OF COMPRESSION ALGORITHMS

The Douglas-Peucker Algorithm [6] is a popular line generalization heuristic, commonly used to fit a series of line segments to a curve, thereby reducing storage requirements. Often implemented in computer graphics applications, the Douglas-Peucker algorithm is applicable in a variety of geospatial applications. Hershberger et al. present an implementation of this algorithm with a running time of O($n \log n$) where n is the number of original points [8].

Despite the popularity of line generalization algorithms for a wide range of applications in cartography and computer graphics, Meratnia and de By indicate that such algorithms are not suitable for GPS trajectory data since both spatial and temporal data should be taken into account [11]. To fit the dataset more accurately, an algorithm called TD-TR [11] modifies the distance formula of the Douglas-Peucker algorithm to utilize the temporal component of the trajectory data stream.

Similar to the Douglas-Peucker and TD-TR algorithms, Opening Window Algorithms fit numerous line segments to the original GPS trajectory data. Two points are used to fit each line segment: the first point in the series, called the **anchor**, and the third point in the series called the **float**. If the distance between the original and the compressed sequence is greater than the defined error tolerance, then either the point causing the maximum error is added to the compressed series (Normal Opening Window Algorithm or NOWA) or the point just before the one that causes the maximum error is added (Before Opening Window or BOPW). If the threshold is not violated, the float slides forward to each subsequent point in the GPS trace until either a violation occurs or the end of the trace is reached. Algorithm OPW-TR [11] is a modified opening window algorithm that incorporates both the spatial distance and the temporal distance (time-distance error ratio) to determine when the error threshold is violated. Another algorithm called OPW-SP [11] is similar to OPW-TR except that an additional condition is included when deciding whether or not to add a point. This condition allows a new point to be added when the speed error introduced is greater then a user-defined tolerance.

Bellman's algorithm [2, 3], based on dynamic programming, also fits a sequence of line segments to a curve. The solution produced by the algorithm is provably optimal; the algorithm minimizes the root mean square (RMS) error under specific conditions. Therefore, Bellman's algorithm can be thought of as providing a very accurate compression of the GPS data, preserving the most important information. A straightforward implementation of the algorithm has a worst-case running time of $O(n^3)$, where n is the number of points in the trajectory. This is a serious drawback when large traces must be compressed. Using additional storage, the running time of the algorithm can be reduced to $O(n^2)$ [10].

The STTrace algorithm [14] is designed to preserve spatio-temporal, heading and speed information in a trace. A hybrid between an online and batch approach, STTrace defines a safe area by first using the previous two points in the series. A vector defining the speed and direction between the two locations is used to predict the location of the next point. Two input parameters are used to make this prediction. One of these parameters is the speed tolerance which defines how much the speed can vary while still remaining in the predicted range. The other input parameter is the heading tolerance that defines how much the heading can vary while still remaining in the predicted range.

4. DATA COLLECTION METHODS

For this research, two distinct datasets were collected; one dataset was obtained from a fleet of buses in Albany, New York (Public-Transit dataset); another was obtained from 24 volunteers at the

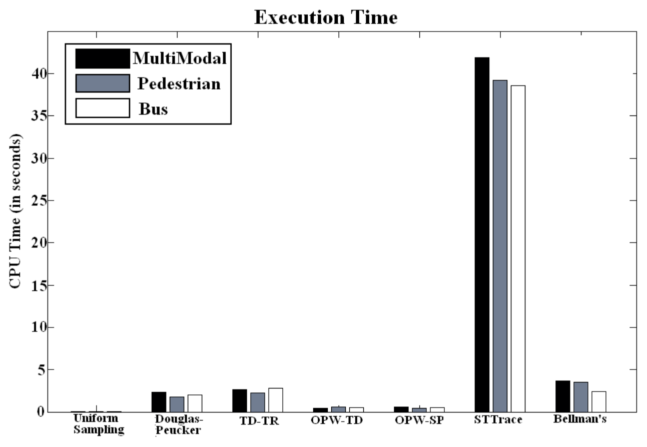

Figure 1: Algorithm Execution Time across different compression algorithms and travel modes. Input GPS trajectories consist of 30,000 points.

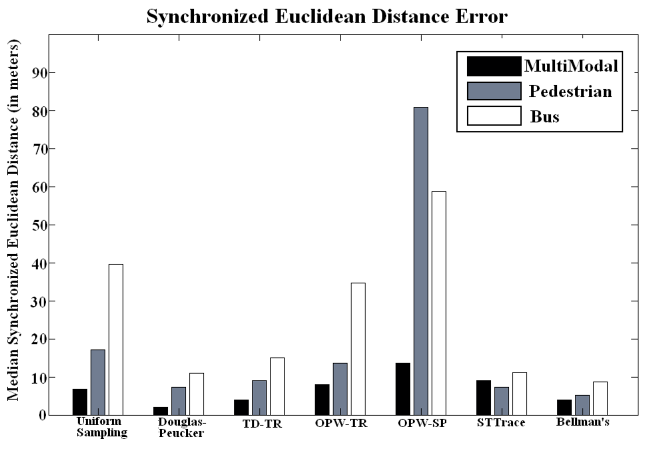

Figure 2: Difference in spatiotemporal error across different compression algorithms and travel modes. Error is measured using median synchronized Euclidean distance.

New York Metropolitan Transportation Council (NYMTC dataset).

The Public-Transit dataset was obtained from GPS units on-board buses traveling in Albany, New York, over a period of 12 weeks during from October to December, 2009. Forty one buses were tracked during this time period, operating on four different routes. The data for the buses was separated for each day the bus was tracked, resulting in over 3144 trajectories. GPS devices were originally installed in these vehicles in order to measure the on-time performance and to determine an optimal routing of buses for the public transportation system in Albany. Bus routes were fairly constant over the time period in which the buses were monitored, but traffic, weather and other variables caused significant variation for each bus across the different traces. The on-time performance of the buses was between 60 and 70 percent, indicating significant deviation from the bus schedule.

In the second dataset, twenty four volunteers at the New York Metropolitan Transportation Council (NYMTC) were recruited and asked to carry GPS units for one weekday. Each GPS unit was configured to automatically log the person's position every 5 seconds along with date, time, speed, etc. Each respondent was asked to turn on the GPS unit at the beginning of each day and carry the unit with them at all times. The GPS unit was only turned off at the end of the day when the person came home and didn't plan to go out again.

5. PERFORMANCE COMPARISON

5.1 Comparison Based on Execution Times

The actual execution time of each algorithm across the three travel modes are shown in Figure 1. Since higher compression ratios typically result in slower run-times, a common compression ratio of 7 was chosen for each trace; therefore, the final size of the compressed trace is $(1/7)$th the original size. There was no significant difference in the run-time performance for any algorithm among the different travel modes when the common compression ratio of 7 was used.

Substantial differences in the run-time performance among the different algorithms was observed. STTrace was by far the slowest algorithm, with an execution time of about 40 seconds; not surprisingly, uniform sampling was the fastest, with an execution time of about 0.002 seconds. STTrace incurs significant computational overhead due to the calculation of the points that define the safe area polygons and the test to determine whether additional points reside within the convex hull of the polygons. In contrast, uniform sampling is very fast since it simply chooses every ith point from the original trace. In our experiments, i was set to 7 to achieve a compression ratio of 7.

5.2 Comparison Based on Error Metrics

A comparison of the algorithms with respect to the median synchronized Euclidean distance (sed) error metric (shown in Figure 2) demonstrates significant differences among the various algorithms, as well as the three travel modes. All algorithms were compared at a common compression ratio of seven, and on the same input data size of $5,000$ points. On average, the bus dataset had the highest degree of error, followed by the pedestrian travel mode; the multimodal dataset had the least amount of error. The bus dataset has two properties that make compression difficult. First, GPS units inside the bus have an obstructed view of the sky, leading to horizontal dilution of precision (HDOP). This causes the location data to be less accurate. The second reason is that it is common for buses to stop either due to traffic or due to designated stops. High error, combined with frequent stops, causes random fluctuations and noise to be introduced into the GPS trajectory, resulting in less redundancy and higher measured error between the original and the compressed traces.

The pedestrian dataset had a higher median *sed* error rate than the multimodal dataset. The reason for this is similar to the bus dataset, in that GPS units in a complex urban environment, such as New York City, often have low accuracy due to the urban canyon effect that occurs from the reflection of GPS signals off tall buildings. Furthermore, pedestrians can often change directions and walk inside buildings, which causes difficulties when compressing the trajectories. In contrast, the multimodal dataset has individuals that are sometimes outside of urban areas, traveling in cars or trains. Travel on major roadways and trains leads to less fluctuation in movement, speed and heading.

For the seven algorithms, the results of the median difference in

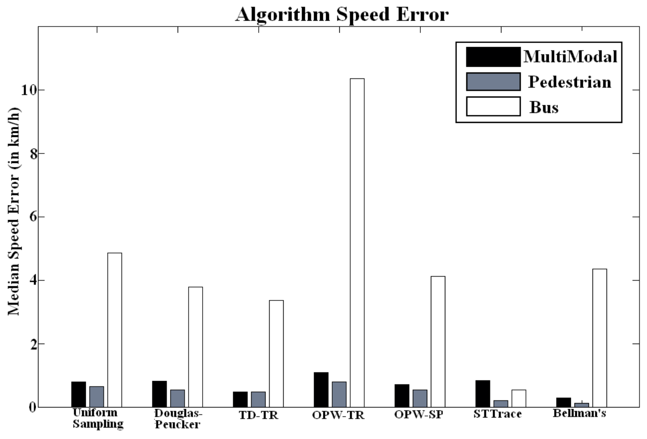

Figure 3: Difference in median speed error across different compression algorithms and travel modes.

speed between the original trajectory and the compressed trajectory are shown in Figure 3. Similar to the *sed* error metric results, the bus dataset had the highest error rate compared to pedestrian and multimodal datasets. This is most likely due to the near constant fluctuations in speed that occur due to bus stops and traffic conditions. Furthermore, unlike the pedestrian dataset, buses are capable of a high rate of speed, allowing the amount of speed change to be far greater than an individual traveling by foot. The multimodal data contained traces with more or less uniform velocities, thereby reducing the amount of error compared to the bus dataset.

6. DISCUSSION AND CONCLUSION

We believe that this empirical study fills a gap in the literature, by comparing numerous algorithms on the same dataset using different error metrics. By comparing these different algorithms side-by-side, trade-offs between accuracy and computation time can be evaluated on an equal footing. By understanding how well these compression algorithms work on traces corresponding to different transportation modes, application-specific compression can be used to determine the best algorithm for a specific context.

A key finding of this study is the substantial difference in the compression performance based on the travel mode. This suggests that there is no one-size-fits-all approach in selecting a compression algorithm. Application-specific algorithms are needed to match the best algorithm to the type of information utilized by the application or business process.

In this work, the effectiveness of the compression techniques was measured using metrics that indicate how well spatiotemporal information is preserved and the actual run-times of the techniques. Since compressed representations are also used to process queries, it is of interest to study the effectiveness of the compression techniques using application-specific metrics that quantify the differences in the responses to queries on uncompressed and compressed representations.

Acknowledgments

We thank the reviewers of ACM SIGSPATIAL GIS 2010 for their comments. Special thanks to Professor Siwei Lyu (Computer Science Department, University at Albany) for his assistance in various stages of this work. This work is supported in part by the Research and Innovative Technology Administration of the U.S. Department of Transportation through the Region 2 - University Transportation Research Centers (UTRC) Program and the University at Albany through the Faculty Research Awards Program (FRAP) - Category A.

7. REFERENCES

[1] P. K. Agarwal, L. J. Guibas, H. Edelsbrunner, J. Erickson, M. Isard, S. HarPeled, J. Hershberger, C. Jensen, and L. Kavraki. Algorithmic Issues in Modeling Motion. *ACM Computing Surveys*, 34:550–572, 2002.

[2] R. Bellman and S. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, NJ, 1962.

[3] R. E. Bellman. On the Approximation of Curves by Line Segments Using Dynamic Programming. *CACM*, 4(6):284, 1961.

[4] Canalys. Worldwide Mobile Navigation Device Market More Than Doubles. Technical report, Canalys Research Release, 2007.

[5] Canalys. North America Overtakes EMEA as Largest Satellite Navigation Market. Technical report, Canalys Research Release, 2009.

[6] D. Douglas and T. Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Line or its Caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[7] S. P. Greaves and M. A. Figliozzi. Commercial Vehicle Tour Data Collection Using Passive GPS Technology: Issues and Potential Applications. *Transportation Research Record*, 2049:158–166, 2008.

[8] J. Hershberger and J. Snoeyink. Speeding Up the Douglas-Peucker Line Simplification Algorithm, 1992.

[9] C. Jones and J. Sedor. Improving the Reliability of Freight Travel. *Public Roads*, 70(1), 2006.

[10] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley, Reading, MA, 2005.

[11] N. Meratnia and R. A. de By. Spatiotemportal Compression Techniques for Moving Point Objects. In *Advances in Database Technology*, volume 2992, pages 551–562. Springer, Berlin/Heidelberg, 2004.

[12] J. Muckell, Q. Cao, P. Mackenzie, D. Messier, and J. Salvo. Toward an Intelligent Brokerage Platform: Mining Backhaul Opportunities in Telematics Data. *Transportation Research Record*, 2097:1–8, 2009.

[13] J. Muckell, J. Hwang, C. Lawson, and S. Ravi. Algorithms for Compressing GPS Trajectory Data: An Empirical Evaluation. Technical Report SUNYA-CS-10-06, Computer Science Department, University at Albany – SUNY, 2010.

[14] M. Potamias, K. Patroumpas, and T. Sellis. Sampling Trajectory Streams with Spatiotemporal Criteria. In *18th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'06)*, pages 275–284, 2006.

[15] H. Zhu, J. Su, and O. H. Ibarra. Trajectory Queries and Octagons in Moving Object Databases. In *Proc. 11th CIKM*, pages 413–421. ACM Press, 2002.