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 how well they preserve the spatio-temporal information across numerous real-world datasets, execution times and various error metrics. Such comparisons are useful in identifying the most effective algorithms for various situations. We also provide recommendations for a hybrid algorithm which can leverage the strengths of various algorithms while mitigating their drawbacks.

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 [6]. 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 [7].

Data generated from GPS units are commonly used in a variety of business and public section applications, such as supply-chain management and traffic modeling [11, 13, 23, 8, 16]. These efforts are being hampered by the sparse nature of some 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 By [17] 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, 2, 29, 17].

Compression strategies can be classified into two forms, namely lossless and lossy compression. Lossless compression enables an exact reconstruction of the original data; that is, no information is lost due to compression. In contrast, lossy compression introduces inaccuracies when compared to the original data. The primary advantage of lossy compression is that it can often reduce the storage requirements drastically while maintaining an acceptable degree of error.

The most native method of lossy compression, namely uniform sampling, offers the advantages of time efficiency and reduced storage requirements for geo-spatial data. This approach down-samples a stream of GPS data at fixed time

intervals; that is, from the original data consisting of a series of time-stamped points (x, y, t), every ith point is kept in the compressed version, for some suitable integer i.

Uniform sampling can reduce the storage requirements, but often results in significant loss of information. The aim of compression techniques is to minimize storage requirements for the GPS data, while maintaining as much useful information as possible. Various algorithms exist in the literature to balance the tradeoff between accuracy and storage size. These algorithms can be logically grouped based on criteria such as batch vs. online processing and the error metrics that measure information loss.

Transportation mode is defined as the method of traveling between locations (e.g, walking, bus, rail or airplane). The characteristics of a GPS trace (changes in direction, accuracy, speed and acceleration) differ substantially based on the transportation mode of moving object. For example, pedestrians walking in an urban setting exhibit a drastically different movement pattern when compared to buses traveling along a predefined route. Pedestrian data typically contains frequent changes in direction and less accurate location readings, based on the urban canyon effect that causes inaccuracies. The effectiveness of a compression technique varies with the characteristics of the data being compressed. Frequent, often unpredictable changes in movement are difficult to compress with high accuracies; conversely, traces that contain high degrees of redundancy (e.g. a vehicle traveling at a consistent speed on a highway) can be compressed significantly without introducing large amounts of error.

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 described 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. As discussed later, our empirical study also provides some guidelines for developing hybrid algorithms that intelligently identify segments of GPS traces that are best suited for particular algorithms.

Our main contributions can be summarized as follows:

- We carry out a comprehensive empirical comparison of a number of known compression algorithms using real-world GPS traces that represent various travel modes (pedestrian, bus and multimodal).
- Our evaluation of the compression algorithms is based on a number of different error metrics such as synchronized Euclidean distance, speed and heading (defined in Section 2) as well as actual execution times.
- Our results provide guidelines for choosing a compression algorithm that is best suited to meet organizational and technical requirements based on the characteristics of the GPS data collected.
- Our empirical results also allow us to develop broad recommendations for developing a hybrid algorithm which can leverage the strengths of various algorithms while mitigating their drawbacks.

The remainder of this paper is organized as follows. The next section describes the various metrics found in the literature for measuring the information loss due to compression. Section 3 provides an overview of the compression algorithms considered in this paper. Section 4 describes the characteristics of the data used in this research. Section 5 uses the experimental results to carry out a detailed comparison of the various algorithms. Section 6 provides additional discussion of the results, recommendations for developing hybrid compression schemes and choosing the best available compression technique to meet organizational goals. Directions for future work are provided in Section 7.

2. TRAJECTORY-SPECIFIC INFORMATION

GPS trajectories do not consist only of spatial data (latitude, longitude); crucially, the temporal component t, is also stored along with the spatial location (x, y). The three components define a series of time-stamped positions (x, y, t). GPS devices also can record the speed and **heading** (defined as the direction of travel) of a moving object. We use the word "asset" to refer to a moving object following the practice in transportation literature. When evaluating the effectiveness of each algorithm, both spatial and temporal accuracy must be measured. Applications using stored GPS data, often require the preservation of the spatial component (where was the asset?), the temporal component (when was the asset at that location?), as well as speed and heading information (how fast and in which direction was the asset traveling?). Preserving the speed and heading information is particularly important in some applications. For example, in fleet monitoring applications, logistics managers often want to know how safely the drivers are operating tractor-trailers. The information needed by the managers includes the following: whether the drivers are operating above the speed limit, whether they are hitting the brakes hard (acceleration) and whether they are driving erratically (heading). Logistics firms have been the most eager to use this new technology, since they often have hundreds or thousands of vehicles on the road; such firms have a vested interest in flagging drivers who are subjecting the trucks to unnecessary strain and causing safety problems for other motorists.

Additionally, certain queries that use trajectories as input, often require accurate approximations of speed. These data mining applications include detecting travel mode [28] and trip purpose [25]. Determining travel mode relies heavily on speed, since many modes can be eliminated based on the speed of travel. For instance, it is obvious that an individual is not walking if the estimated speed is 50 mph. Another

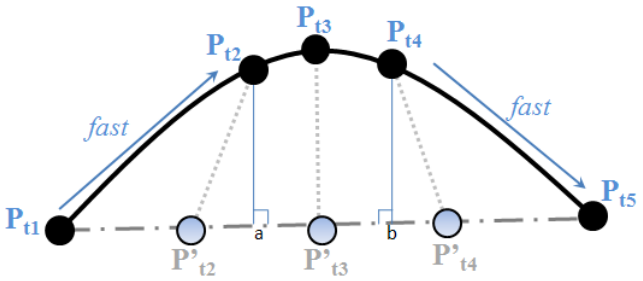

Figure 1: Synchronized Euclidean Distance measures the distance between the original and compressed trace at the same time. In contrast, line generalization algorithms ignore the temporal component and use simple perpendicular distance.

example is trip purpose detection, which is defined as the reason or activity performed by an individual at a specific location (e.g. shopping, work or school). Since speed is very useful in inferring the mode of travel to a destination, determining trip purpose often relies on good speed estimates.

Queries that involve detecting patterns between multiple trajectories sometimes use the heading information to establish relationships and connections. For example, detecting *single file* movement [5], uses heading information to determine whether or not a group of entities is moving in single file (i.e., if they are following each other, one behind the other). This is a movement pattern that occurs often among animals, vehicles and humans.

One way of measuring the difference between a GPS trace and its compressed version is to measure the perpendicular distance (Figure 1). Since the perpendicular distance does not incorporate temporal content, a more effective approach involves measuring the distance at synchronized points. For instance, Figure 1 shows the distances at synchronized time points t_2, t_3 and t_4. (At time t_3, the time synchronized distance and the perpendicular distance coincide.)

2.1 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. 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. Details regarding these algorithms are presented in the next section; here, a description of the error metrics is provided.

As will be seen in Section 3, the algorithms of Douglas-Peucker and Bellman are forms of line generalization algorithms that are often used in computer graphics applications for reducing the complexity and storage requirements of curves. Additional line generalization algorithms include Bezier curves, NURBs curves or cubic splines. A common characteristic of all these algorithms is that error is measured only in terms of the spatial distance between the original curve and the approximation.

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

Synchronized Euclidean distance (sed) measures the distance between two points at identical time stamps [21]. In Figure 1, five time steps (t_1 through t_5) are shown. The simplified line (which can be thought of as the compressed representation of the trace) is comprised of only two points (P_{t_1} and P_{t_5}); thereby, it does not include points P_{t_2}, P_{t_3} and P_{t_4}. To quantify the error introduced by these missing points, distance is measured at the identical time steps. Since three points were removed between P_{t_1} and P_{t_5}, the line is divided into four equal sized line segments using the three points P'_{t_2}, P'_{t_3} and P'_{t_4} for the purposes of measuring the error. The total error is measured as the sum of the distance between all points at the synchronized time instants, as shown below. (In the following expression, n represents the total number of points considered.)

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

Another trajectory metric is the Meratnia-By time-distance ratio [17], which 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 Meratnia-By time-distance ratio in order to select points for compression.

3. OVERVIEW OF COMPRESSION ALGORITHMS

3.1 Douglas-Peucker Algorithm

The Douglas-Peucker Algorithm [9] is a popular 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. A common application is in reducing the number of points required to store state and county boundaries.

Douglas-Peucker is a line generalization algorithm, that recursively selects points from the original set of GPS trajec-

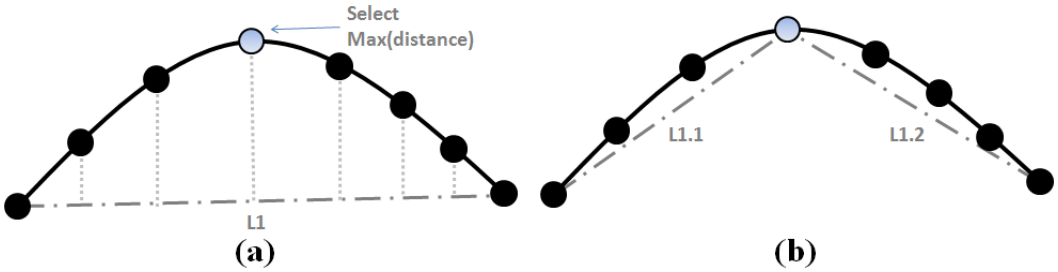

Figure 2: Douglas-Peucker Line Generalization Algorithm recursively fits a series of line segments to the original curve by selecting the furthest point.

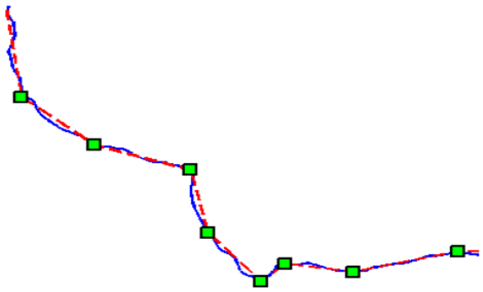

Figure 3: Douglas-Peucker Line Generalization Algorithm approximating a GPS Trajectory.

tory points. A series of line segments is fitted to the original curve based on new points that are selected. The execution of the algorithm proceeds along the following steps. To begin with, the first and last points in the trajectory are stored in the compressed version. Together, these two points form a line segment, as shown in Figure 2(a). Next, a point with the maximum distance from the line segment $L1$ is selected; this point, along with the start and end points form the compressed representation at the end of the first iteration. In Figure 2(b), the newly added point is used along with the start and end point to form two line segments labeled $L1.1$ and $L1.2$. This process is repeated using recursion on each line segment. The algorithm halts when the maximum distance between the original trajectory and the line segments is below a user defined tolerance.

If the above algorithm is implemented in a straightforward manner, its worst-case running time is $O(n^2)$, where n is the number of original points. The running time can be improved to $O(n \log n)$ using a more complex approach involving convex hulls [12].

3.2 Modified Douglas-Peucker (TD-TR)

Despite the popularity of line generalization algorithms for a wide range of applications in cartography and computer graphics, Meratnia and By [17] indicate that line generalization algorithms such as Douglas-Peucker (shown in Figure 3) are not suitable for GPS trajectory data since both spatial and temporal data should be taken into account.

To measure the error more accurately, Meratnia and By [17] defined the time-distance ratio metric discussed in Section 2. In this metric, the difference between the original trace and compressed trace is evaluated using the differences in both distance and time. To fit the dataset more accurately, an algorithm called TR-TD [17], modifies the distance formula of the Douglas-Peucker algorithm to utilize the temporal component of the trajectory data stream. The modified distance is used in compressing the trace in a manner similar to the Douglas-Peucker algorithm described in the previous section.

3.3 Opening Window Algorithms

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, than 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.

The standard implementation of Opening Window Algorithm is unsuitable for compressing GPS traces because essential information such as temporal data is ignored. To effectively use a version of the opening window algorithm, the error metric needs to be modified to incorporate spatiotemporal information and not just the spatial characteristics. The algorithm OPW-TR [17] 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 (also discussed in [17]) 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.

3.4 Bellman's Algorithm

Bellman's algorithm, based on dynamic programming [4, 3], 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 $O(n^2)$ [14].

The input to Bellman's dynamic programming algorithm is a series of latitude and longitude points that can be taken directly from the GPS loggers. An additional positive value C, which represents the penalty for introducing a new line segment into the compressed representation, is also needed for the algorithm. More line segments imply better accuracy; however, the corresponding compressed representation needs more storage. Bellman's algorithm creates an optimal fitting of the GPS trace using line segments (where optimality is defined as minimizing the RMS error), taking into account the penalty factor C.

In order to optimality fit the line segments to the curve, Bellman's algorithm assumes that the input data is a valid (i.e., single-valued) function; thus, the trajectory cannot contain no loops. However, due to inaccurate measurements by GPS devices, loops are often present in trajectory data. In this research, the input data stream was scanned to detect if the next point in the series would violate the rules of a valid function. When this occurs, Bellman's algorithm would be called on a subset of the original data stream, with each subset containing only points that form valid functions. Therefore, our trajectory-based version of Bellman's Algorithm, may execute the original algorithm multiple times for a single trajectory, with each call executing on a segment of the original GPS trace. The smaller dataset sizes improves execution time (since it is running on smaller segments of the original trace); however, this affects the accuracy since global optimization is not performed.

3.5 STTrace

The STTrace algorithm [21] 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.

The two input parameters are used to construct a "safe area" polygon to adjust the degree to which the speed and heading can vary. If the third point in the GPS trajectory is inside this polygon, then it is considered to be within the predicted range and therefore not stored. Otherwise, the point is thought of as exhibiting unexpected behavior. Since such a point may contain information that is vital to the GPS trace, it is included in the compressed representation.

If the STTrace algorithm is implemented as described above, error propagation can be a major problem. This occurs when

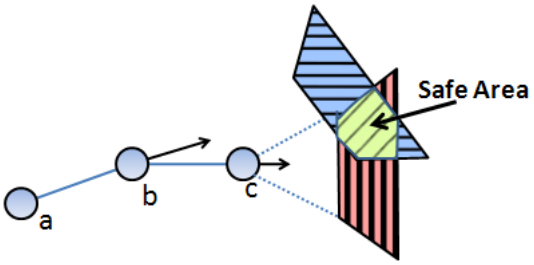

Figure 4: The STTrace Algorithm predicts the behavior of the next points based on previous GPS trajectory points. Points within the predicted Safe Area are not stored.

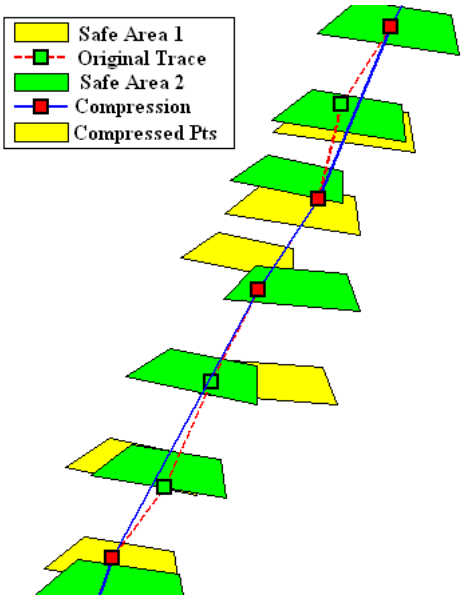

Figure 5: STTrace approximating a GPS trajectory. The original trajectory is shown as the red dashed line; the compressed trace is shown as the blue solid line. Points that reside outside the intersection of both the yellow and green polygons are added to the compressed trajectory.

a point falls on the outer edge of each polygon; thus, it is possible for the error to accumulate, causing the compressed trace to become more and more inaccurate. Small changes at each time stamp over a long enough time period result in major deviations that need to be stored in the compressed version.

In order to resolve the issue of error propagation, STTrace uses a second safe area polygon, shown using horizontal slashes in Figure 4. (This figure is a simplified version of Figure 4 in [21].) This safe area is defined by using two GPS points: the point three time-steps back, and the point two time-steps back (points a and b). The direction and speed are extrapolated over a two time-stamp distance to determine a predicted range at a given time. Only points that are within both safe areas, shown using diagonal slashes, are determined to be within the predicted range. If a point is outside the intersection of the two polygons, it is stored in

the compressed version. This eliminates the problem of error propagation, at the cost of some additional computation.

Figure 5 illustrates the execution of STTrace in Matlab on a subset of a GPS trajectory. The two colored polygons show the two safe areas defined at each time step. Points that are in the intersection of both polygons (shown in green) are considered to be predicted and redundant based on previous points, and therefore are not included. Points that are outside the intersection of both polygons (shown in red) are stored in the compressed form.

3.6 Additional Algorithms

Clustering and statistical aggregation methods that compress many GPS traces were not compared in this study due to the difficulty of appropriately measuring the effectiveness of such vastly different techniques. While the algorithms used in this study compress a single GPS trace, the algorithms in the literature that were not compared in this study operate on a GPS trajectory collection comprised of potentially hundreds or thousands of traces. A brief description of these other approaches is included below to provide a thorough literature review of available techniques.

Statistical methods are primarily focused on determining appropriate aggregation levels. These techniques are based on comparisons of statistical measures, and more recently, on wavelet decomposition. Data aggregation involves determining the proper sampling interval for storing the most significant spatial-temporal information. By exploiting the characteristics of the data, properly determined aggregation levels can lead to a drastic reduction in storage space. The optimal decomposition level is determined by finding the time interval with the most similarity and hence the least amount of variability during at each time step.

Implementations of statistical techniques utilize the minimal amount of sufficient statistics necessary to capture the full information contained within a parameter distribution. Some statistical solutions use a cross-validated mean square error, while others utilize an F-statistic computation to obtain the optimal aggregation level [10]. The decision as to which approach works best is often based on the most significant statistical approach, in conjunction with market-driven parameters such as the value of the data. Recent work using wavelet decomposition [22, 26, 27] is most often applicable for capturing significant information in intelligent transportation systems. Based on Shannon's Theorem in information theory [15], wavelet analysis is useful for identifying trends.

Other compression methods use a suitable data structure for storing the most recent information. The AM-Tree data structure [19, 20] is primarily based on the assumption that information value decays over time. As time progresses, the AM-Tree stores past information regarding GPS trajectories at coarser and coarser resolutions. The root of the tree contains information about every time-stamped message received, with each additional level from the root containing half the number of time-stamped messages over a given time interval.

Another class of algorithms exploits redundancy across a large set of trajectories. One such example is the Scalable ClUster-Based Algorithm (SCUBA) [18] which compresses many GPS trajectories together, instead of reducing the storage requirements for a single trace. This algorithm clusters similar GPS trajectories; this process can substantially reduce the space complexity by minimizing the storage of similar information.

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 New York Metropolitan Transportation Council (NYMTC dataset).

The Public-Transit dataset was obtained from GPS units onboard 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 the 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.

Three groups of traces (trajectories) were created from the Public-Transit and NYMTC datasets, with each group representing one travel mode (pedestrian, bus and multimodal). Each trajectory contains 5,000 points. The bus dataset contains four trajectories, one for each route tracked. The other two groups (pedestrian and multimodal) have five trajectories each. (The classification of trajectories into the groups was done by matching the travel mode indicated on the travel diary to the trace.) To select representative samples, five traces for each of these two modes were chosen from different individuals traveling in separate regions across New York City. An additional dataset consisting of a single trace comprised of 30,000 points was selected for each transportation mode. This larger dataset was used to measure execution times of the seven algorithms considered in this study.

5. PERFORMANCE COMPARISON

Seven algorithms (Uniform sampling, Douglas-Peucker, TD-TR, OPW-TD, OPW-SP, STTrace and Bellman's algorithm) were run on the collection of traces obtained as described in the previous section. This section discusses the performance

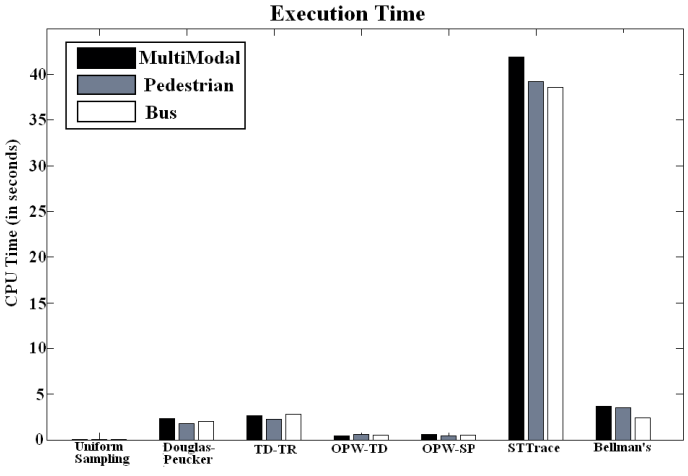

Figure 6: Algorithm Execution Time across different compression algorithms and travel modes. Input GPS trajectories consist of 30,000 points.

of these algorithms with respect to execution times and various error metrics.

5.1 Comparison Based on Execution Times

The actual execution time of each algorithm across the three travel modes are shown in Figure 6. 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. However, datasets with unpredictable behavior contain little redundancy, and typically lead to higher error rates. (This is discussed further in the next subsection.) In practice, it is logical to compress datasets with frequent changes in direction and speed to a smaller compression ratio compared to datasets that demonstrate more predictable behavior. Therefore, differences in execution times for different travel modes becomes more apparent, since they are not all benchmarked to a common compression ratio.

Substantial difference in run-time performance between 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 simply every ith point is taken from the original trace. (In our experiments, i was set to 7 to achieve a compression ratio of 7.)

Significant differences with respect to run-times were not apparent in algorithms that are slight modifications of each other. For example, Douglas-Peucker and TD-TR, had execution times of about 2.2 seconds and 2.5 seconds respectively. The Opening Window Algorithms OPW-TR and OPW-SP had median run-times of 0.4 seconds and 0.5 seconds respectively. The slight decrease in performance is noticeable for algorithms that compute more complicated metrics. This explains the slight difference between Douglas-Peucker and TD-TR and that between OPW-TR and OPW-SP. The performance of OPW-SP depends heavily on the input parameters given. A low value for the speed tolerance results in poorer performance, as the speed component is more computation intensive compared to synchronized Euclidean distance.

Bellman's algorithm had a mean run-time of about 3.2 seconds. The performance of Bellman's Algorithm depends heavily on the number of times the algorithm is called. When a trajectory contains loops, Bellman's algorithm was run for each segment after eliminating the loops as described in Section 3.4. The more loops, the better is the run-time performance; however, this inversely impacts the overall accuracy of the compression.

5.2 Comparison Based on Error Metrics

A comparison of the algorithms with respect to the median synchronized Euclidean distance error metric (shown in Figure 7) demonstrates significant differences between the various algorithms, as well as the three travel modes: pedestrian, multimodal and bus. 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, causing the 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 reasoning 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 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, such as driving a car or traveling in trains. Travel on major roadways and trains leads to less fluctuation in movement, speed and heading.

In comparing the seven algorithms with respect to the sed error, substantial differences are observed. Most striking is the *sed* error of algorithm OPW-SP even compared to uniform sampling. This error is high due to the parameter selection, since a trade-off is made between spatiotemporal accuracy and speed accuracy. Setting a low speed tolerance and a high time-distance ratio tolerance results in poor performance with respect to the *sed* error metric but smaller speed residuals. Accordingly, this was the param-

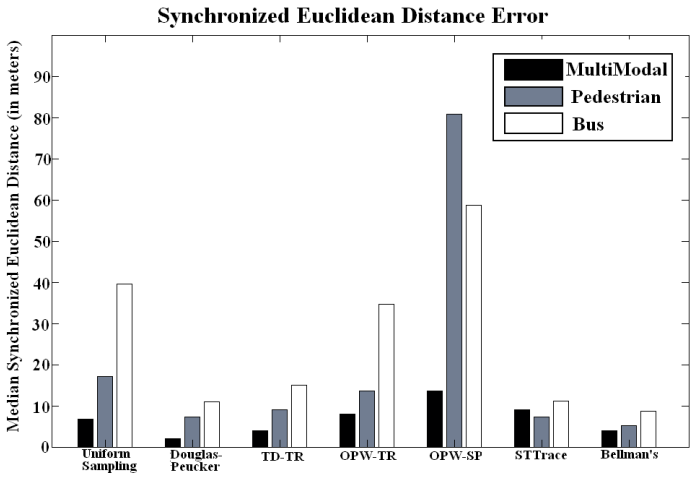

Figure 7: Difference in spatiotemporal error across different compression algorithms and travel modes. Error is measured using median synchronized Euclidean distance.

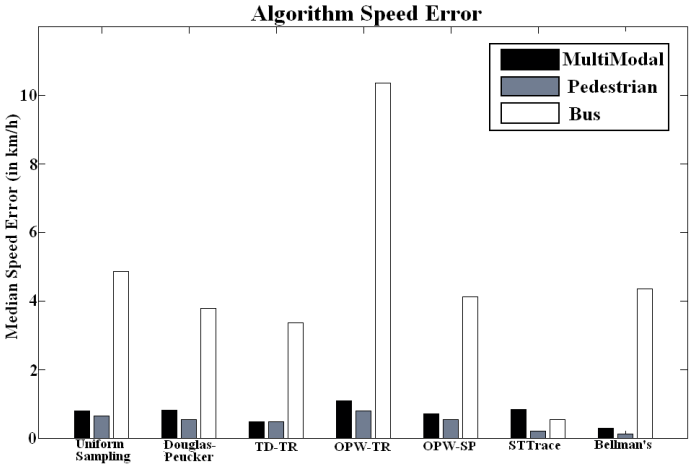

Figure 8: Difference in median speed error across different compression algorithms and travel modes.

eter selection used in this study, in order to demonstrate the difference between the two opening-window algorithms. In contrast, Bellman's algorithm was the best choice for low compression ratios, but was not suitable for high compression ratios and datasets that contain a large number of loops. One unexpected result was that Douglas-Peucker performed better than the modified version of TD-TR, which uses the time-distance ratio in the error calculation instead of maximum spatial distance. Overall, Douglas-Peucker also outperformed the Opening Window Algorithm (OPW-TR), with a median *sed* error of 5 meters compared to an average rate of 10 meters respectively. However, it is important to remember that Douglas-Peucker is a batch algorithm, while OPW-TR is online.

For the seven algorithms, the results of the median difference in speed between the original trajectory and the compressed trajectory are shown in Figure 8. Similar to the *sed* error metric results, the bus dataset had the highest error rate compared to the 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.

STTrace obtained the most consistent speed results, since sharp changes in speed are readily detected by the algorithm; such points lie outside the intersection of the two safe area polygons. However, the median speed error for the pedestrian dataset was slightly worse than that of Uniform Sampling. The algorithms of Bellman and Douglas-Peucker both have low speed errors. The median speed error for OPW-SP was slightly better compared to OPW-TR. Our data shows that the error rate for OPW-SP is dependent on the speed tolerance parameter. Changing this parameter can lead to results that are better or worse than OPW-TR.

Measuring the error in direction or heading introduced by compression algorithms yields some distinctly different findings compared to the *sed* and median speed metrics. One difference is the degree of error between the different transportation modes. With respect to *sed* or speed error, the bus dataset had significantly higher error than the pedestrian and multimodal datasets. However, when measuring the difference in heading, on average the bus dataset has the lowest error (shown in Figure 9). This is most likely due to simple, predictable bus routes that often run in straight lines down major roads; thus, the dataset has less changes in direction.

The difference in median heading error between the pedestrian dataset and multimodal was negligible. For minimizing heading error, the best performance was exhibited by Bellman's Algorithm followed by STTrace. However, both these algorithms are only effective when the compression ratios are fairly small. For larger compression ratios, there was no significant advantage in heading accuracy between Douglas-Peucker, TD-TR, OPW-TR and OPW-SP compared to using Uniform Sampling.

6. DISCUSSION

We believe that this empirical study fills a gap in the literature, by comparing numerous algorithms on the same dataset using different error metrics. In comparing these different algorithms side-by-side, the 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. Table 2 shows recommendations for the choosing the most appropriate algorithm based on the characteristics of

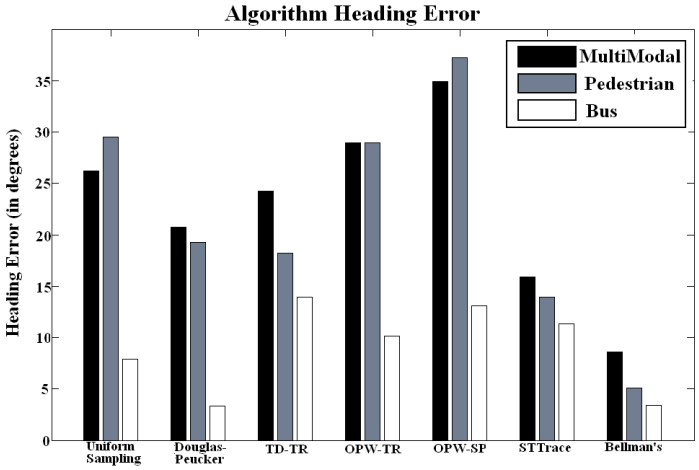

Figure 9: Difference in median heading error across different compression algorithms and travel modes.

Table 2: Preferred Algorithm based on Travel Mode

Travel Mode	Data Characteristics	Recommendations
Multi-Modal	Disjoint Traces, Infrequent Changes in Speed and Direction	Uniform Sampling, Bellman's, OPW-TR, TD-TR, Douglas-Peucker
Pedestrian	Random, Slow	Uniform Sampling, Douglas-Peucker, Bellman's
Bus	Frequent Changes in Speed, Infrequent Changes in Direction	Uniform Sampling, STTrace, TD-TR

the dataset.

Overall, uniform sampling is a viable alternative if the application requires a quick and simple compression with reasonable error rates. Other algorithms used in this study can result in substantially better compression; however, they also require substantially more CPU time and memory. Specifically, the Opening Window Algorithms execute fairly quickly, but may not lead to a large decrease in median *sed* and speed errors compared to uniform sampling. Indeed, for a number of trajectories, the Opening Window algorithms may have a higher error.

With GPS traces that appear random (i.e., traces that contain many rapid changes in direction/speed), our study indicates that sophisticated algorithms may not yield consistently better performance across numerous error metrics compared to uniform sampling. Segments of GPS traces that exhibit frequent changes in heading are often the result of noise; such segments can be omitted from the compressed representation with minimal information loss. Thus, our study suggests that uniform sampling may offer the best trade-off in terms of execution time and accuracy for GPS traces that exhibit high variability in direction and/or speed. Traces that are information-rich, are best compressed using the Bellman's algorithm or the Douglas-Peucker algorithm. The development of a hybrid approach to GPS trace compression requires knowledge of the scenarios in which an algorithm works well as well as those in which the algorithm performs poorly. The trajectory-based execution of Bellman's algorithm described in Section 3.4 performs very well for small compression ratios. However, for most input trajectories, very high compression ratios are unattainable. This is due to the segmenting of the original GPS trajectory into numerous pieces in order to prevent execution of Bellman's algorithm on trajectories with loops. Each segment requires a minimum of two points, namely the start and end points of the segment. Thus, the size of the compressed representation depends on the number of (loop-free) segments used.

7. FUTURE WORK

One possible approach for improving the compression of GPS trajectory data would be to use knowledge of the road network. Deviations from the road network would need to be stored, along with additional information needed to capture the speed and acceleration of the vehicle. This could allow for a drastically smaller compressed representation with minimal compute time and substantially reduced error. Map matching algorithms [24] are currently used on mobile devices to determine where on a road network an asset most likely resides. These algorithms can be used in conjunction with compression schemes to determine when an object deviates from a specific road segment.

GPS devices are unable to pinpoint the exact location of an object; instead, they provide an approximation. The accuracy of the location data derived from a GPS device depends on a few major factors, including the hardware, location of the satellites, atmospheric conditions and obstructions that can reflect signals. Future work that accounts for this inherent noise can reduce the storage requirements, without any significant loss of information. Smoothing that occurs from this de-noising process, if implemented correctly, could result in more useful traces for data analysis and query processing.

The sheer diversity of applications that collect and retrieve information from GPS trajectory data makes it difficult to obtain a comprehensive data set that encompasses this wide range of possible applications. Extending this work to allow larger datasets and additional travel modes would allow a more comprehensive comparison.

An additional factor that is related to optimal parameter selection is the ease-of-use of an algorithm. Some of the algorithms compared in this study use sophisticated techniques to compress the data. Users may find it difficult to understand these algorithms. As a consequence, they may find it difficult to choose an appropriate set of input parameters for such algorithms. Therefore, simplicity and knowledge regarding typical users of these algorithms may influence the trade-offs and overall practicality of the different compression methods.

A related opportunity for future work is to devise methods

for optimal parameter selection to achieve the best trade-off between accuracy and compression ratio for a given trajectory. For example, algorithms such as OPW-SP have two parameters, one for the spatiotemporal component, and one to indicate a maximum deviation in speed. Often, it is difficult to manually determine the appropriate values of these parameters that yields a desired compression ratio.

In this work, the effectiveness of the compression techniques was measured using metrics that indicate how well spatiotemporal information is preserved and the actual runtimes 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.

8. ACKNOWLEDGMENTS

Special thanks to Professor Siwei Lyu (Computer Science Department, University at Albany) for assistance in algorithm development and for providing valuable suggestions.

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.

9. REFERENCES

[1] M. Abdelguerfi, J. Givaudan, K. Shaw, and R. Ladner. The 2-3tr-tree, a trajectory-oriented index structure for fully envolving valid-time spatio-temporal datasets. In *10th ACM-GIS*, pages 29–34. ACM Press, 2002.

[2] 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.

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

[4] R. E. Bellman. On the approximation of curves by line segments using dynamic programming. *CACM*, 4(6):284, 1961.

[5] K. Buchin, M. Buchin, and J. Gudmundsson. Detecting single file movement. In *GIS '08: Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, pages 1–10, New York, NY, USA, 2008. ACM.

[6] Canalys. Worldwide mobile navigation device market more than doubles. Technical report, Canalys Research Release, 2007.

[7] Canalys. North america overtakes emea as largest satellite navigation market. Technical report, Canalys Research Release, 2009.

[8] N. R. Council. A concept for national freight data program, 2003.

[9] 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.

[10] B. J. Gajewski, S. M. Turner, W. L. Eisele, and C. H. Spiegelman. Intelligent transportation system data archiving: Statistical techniques for determining optimal aggregation widths for inductive loop detector speed data. *Transportation research record.*, 1719:85, 2000.

[11] S. P. Greaves and M. A. Figliozzi. Commerical vehicle tour data collection using passive gps technology: Issuesand potential applications. *Transportation Research Record*, 2049:158–166, 2008.

[12] J. Hershberger and J. Snoeyink. Speeding up the douglas-peucker line simplification algorithm, 1992.

[13] C. Jones and J. Sedor. Improving the reliability of freight travel. *Public Roads*, 70(1), 2006.

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

[15] R. J. Marks. *Introduction to Shannon sampling and interpolation theory*. Springer texts in electrical engineering. Springer-Verlag, New York, 1991.

[16] E. McCormack and M. E. Hallenbeck. Its devices used to collect truck data for performance benchmarks. *Transportation Research Record*, 1957:43–50, 2006.

[17] N. Meratnia and R. A. d. By. *Spatiotemporal Compression Techniques for Moving Point Objects*, volume 2992. Springer Berlin / Heidelberg, 2004.

[18] R. Nehme and E. Rundensteiner. Scuba: Scalable cluster-based algorithm for evaluating continuous spatio-temporal queries on moving objects. In *EDBT*, pages 1001–1019, 2006.

[19] M. Potamias, K. Patroumpas, and T. Sellis. Amnesic online synopses for moving objects. In *Proceedings of Conference on Information and Knowledge Managment (CIKM'06)*, 2006.

[20] M. Potamias, K. Patroumpas, and T. Sellis. Online amnesic summarization of streaming locations. In *Proc. 10th international conference on Advances in spatial and temporal databases*, 2006.

[21] M. Potamias, K. Patroumpas, and T. Sellis. Sampling trajectory streams with spatiotemporal criteria. In *18th International Conference on Scientific and Statistical Database Management (SSDBM'06)*, pages 275–284, 2006.

[22] F. Qiao, X. Wang, and L. Yu. Optimizing aggregation level for intelligent transportation system data based on wavelet decomposition. *Transportation Research Record: Journal of the Transportation Research Board*, pages 10–20, 2003.

[23] J. F. Srour and D. Newton. Freight-specific data derived from intelligent transportation systems: Potential uses in planning freight improvement projects. *Transportation Research Record*, 1957:66–74, 2006.

[24] C. E. White, D. Bernstein, and A. L. Kornhauser. Some map matching algorithms for personal navigation assistants. *Transportation Research*, 8(1-6):91–108, 2000.

[25] J. Wolf, R. Guensler, and W. Bachman. Elimination of the travel diary: Experiment to derive trip purpose from global positioning system travel data. *Transportation Research Record: Journal of the Transportation Research Board*, 1768:125–134, 2001.

[26] H. Xiao. Fuzzy-neural network traffic prediction framework with wavelet decomposition. *Transportation research record.*, 1836:16–20, 2003.

[27] P. Yi, L. Sheng, and L. Yu. Wavelet transform for feature extraction to improve volume adjustment factors for rural roads. *Transportation research record.*, 1879:24–29, 2004.

[28] Y. Zheng, L. Liu, L. Wang, and X. Xie. Learning transportation mode from raw gps data for geographic applications on the web. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 247–256, New York, NY, USA, 2008. ACM.

[29] H. Zhu, J. Su, and O. H. Ibarra. *Trajectory queries and octagons in moving object databases*. ACM Press, 2002.