Similarity-Based Compression of GPS Trajectory Data

Jeremy Birnbaum, Hsiang-Cheng Meng, Jeong-Hyon Hwang
Department of Computer Science
University at Albany, State University of New York
Albany, NY 12222
Email: {jbirn, hmeng, jhh}@cs.albany.edu

Catherine Lawson
Department of Geography and Planning
University at Albany, State University of New York
Albany, NY 12222
Email: lawsonc@albany.edu

***Abstract*—The recent increase in the use of GPS-enabled devices has introduced a new demand for efficiently storing trajectory data. In this paper, we present a new technique that has a higher compression ratio for trajectory data than existing solutions. This technique splits trajectories into sub-trajectories according to the similarities among them. For each collection of similar sub-trajectories, our technique stores only one sub-trajectory's spatial data. Each sub-trajectory is then expressed as a mapping between itself and a previous sub-trajectory. In general, these mappings can be highly compressed due to a strong correlation between the time values of trajectories. This paper presents evaluation results that show the superiority of our technique over previous solutions.**

I. Introduction

In recent years, the number of GPS-enabled devices has increased drastically. This surge has generated a plethora of spatio-temporal data that many companies and organizations rely on for tracking their staff, customers, shipments, and vehicles [1], [2], [3]. This data is typically produced at a high rate. For example, 4,000 moving objects can generate 1 GB worth of data per day with a collection interval of 10 seconds [4]. Fortunately, such data often contains large amounts of redundant information, yielding a great opportunity for compression.

The aforementioned data consists of GPS trajectories. Each trajectory is a list of 3-tuple entries including spatial information, as latitude and longitude, as well as temporal information, as a timestamp of the record. When compressing trajectories, there are two major classes of techniques to choose from: single trajectory compression (STC) and multiple trajectory compression (MTC). STC algorithms compress each trajectory individually, and therefore cannot benefit from the commonalities among trajectories [4], [5], [6], [7]. TrajStore [8], a recent MTC system, overcomes this limitation by sharing spatial data across similar trajectories and storing only the temporal data of each trajectory.

This paper presents a new approach for compressing a set of trajectories. This approach generally achieves a higher compression ratio than previous STC and MTC algorithms by effectively utilizing both the redundancy among different trajectories (inter-trajectory redundancy) and the redundancy within a trajectory (intra-trajectory redundancy). Our approach splits trajectories into sub-trajectories according to their similarity to others in the set. Then, it re-expresses each sub-trajectory T as a mapping T' between the time values of T and those of a similar trajectory R. For any time value on T, the corresponding position can be obtained from linear interpolations on both T' and R. This approach removes entries in T' as long as these removals do not cause an unacceptable degree of error. This scheme is especially effective when T and R reflect similar driving patterns (i.e., the time values of T and R are highly correlated and thus many of them can be removed safely).

In this paper, we make the following contributions:

1) We present a new approach for compressing trajectories by utilizing similarities among them.
2) We provide preliminary evaluation results that show the superiority of our approach over prior STC and MTC solutions.
3) We describe research challenges and our planned solutions for the construction of a usable trajectory data management and query system.

The rest of this paper is organized as follows: Section II summarizes related work. Section III presents our solution for efficiently compressing trajectories. Section IV discusses our preliminary evaluation results. Section V describes our future research plans. Section VI concludes this paper.

II. Related Work

There are two representative *performance metrics* for trajectory compression algorithms. *Compression time* refers to the amount of time that it takes to compress trajectory data. The *compression ratio* is defined as the size of the original data divided by the size of its compressed representation. There are also *error metrics* that measure the difference between a trajectory and its compressed representation. These metrics include linear distance [8], spatial error [5], and Synchronized Euclidean Distance (SED) [4]. For each point on the original trajectory T, these metrics find, using linear interpolation, a corresponding point on the compressed representation T' and then measure the distance between these two points. For each point on T, linear distance calculates the path length to that point and then finds the corresponding point with the same path length on T'. In contrast, spatial error finds the closest

This work has been supported by the National Science Foundation under CAREER award IIS-1149372 as well as the Research and Innovative Technology Administration of the U.S. Department of Transportation through the Region 2 - University Transportation Research Centers Program.

point on T' to each point on T. For a point on T, SED finds the point on T' with the same time value. In this paper, we use SED since the other error metrics have the limitation of not incorporating temporal data, and thus may not effectively reflect changes in speed.

There have been many techniques created for compressing individual trajectories [4], [5], [6], [7]. *Uniform sampling* downsamples a trajectory at fixed time intervals in order to achieve a desired compression ratio. This approach, however, cannot make any guarantee about the compression error. The *Douglas-Peucker* algorithm [5] initially inserts the two end points of the original trajectory into the compressed representation. It then adds the point with the largest spatial error to the compressed representation. This process is repeated until the largest spatial error becomes smaller than a user-specified bound. *TD-TR* [4] is a variant of the Douglas-Peucker algorithm which uses SED instead of spatial error. In general, uniform sampling and TD-TR outperform other STC algorithms in terms of compression time and compression ratio, respectively [7]. Compared to TD-TR, our recent SQUISH (Spatial QUalIty Simplification Heuristic) algorithm achieves a similar compression ratio, but with much lower computational overhead [6]. SQUISH iteratively adds points from a trajectory into a priority queue. Each point in the queue has an associated priority representing an upper bound of the error which would be introduced by its removal. In this way, it can quickly remove extraneous points while effectively bounding the growth of error caused by their removal. When the end of the input trajectory is reached, SQUISH outputs a trajectory consisting of the points kept in the priority queue.

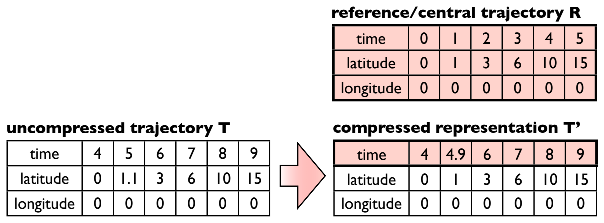
In contrast to STC algorithms, TrajStore compresses a set of trajectories by exploiting the redundancy among them [8]. It divides a map of trajectories into cells and then compresses the sub-trajectories within each cell. For each cell, trajectories are grouped according to their similarities with a central trajectory for each group. Each trajectory T in a group is re-expressed (e.g., T' in Figure 1) using the points of the central trajectory. Given the resulting trajectory T', only its temporal information is stored since its spatial information can be obtained from the central trajectory. T' can be compressed by removing entries whose removal does not increase the error beyond the user-specified bound. In Figure 1, where an error bound of 0.3 is assumed, no entry was removed from T' since any such removal would introduce an error beyond the bound. For example, if tuple $(8, 10, 0)$ were removed from T', then that tuple would be estimated as $(8, \frac{6+15}{2}, \frac{0+0}{2}) = (8, 10.5, 0)$ using interpolation with neighboring entries, thereby introducing an error of 0.5.

III. Multiple Trajectory Compression

This section describes our technique for efficiently compressing trajectories by taking advantage of commonalities among them. Sections III-A and III-B present the core ideas and implementation details of this technique, respectively.

A. Core Ideas

Given two similar trajectories R and T, it should be possible to achieve a higher compression ratio than STC algorithms by exploiting the similarities between them. We present a new solution to this problem. Our solution finds a mapping T' between the time values of T and those of R. Then, for any time value t_T on T, the corresponding time value t_R on R can be obtained from linear interpolation on T'. The position for time t_R (i.e., that for time t_T) can also be obtained from linear interpolation on R. Our solution then uses a STC algorithm to further remove entries from T'.


reference/central trajectory R

time	0	1	2	3	4	5
latitude	0	1	3	6	10	15
longitude	0	0	0	0	0	0

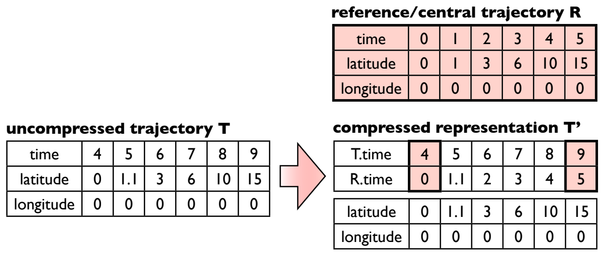
uncompressed trajectory T

time	4	5	6	7	8	9
latitude	0	1.1	3	6	10	15
longitude	0	0	0	0	0	0

compressed representation T'

time	4	4.9	6	7	8	9
latitude	0	1	3	6	10	15
longitude	0	0	0	0	0	0

Fig. 1. TrajStore. Uncompressed trajectory T is first transformed into T' using the points on R. Then, only the time values on T' are stored (see those highlighted). In this example, no entries are removed from T' since such a removal would cause an error beyond the user-specified bound of 0.3.


reference/central trajectory R

time	0	1	2	3	4	5
latitude	0	1	3	6	10	15
longitude	0	0	0	0	0	0

uncompressed trajectory T

time	4	5	6	7	8	9
latitude	0	1.1	3	6	10	15
longitude	0	0	0	0	0	0

compressed representation T'

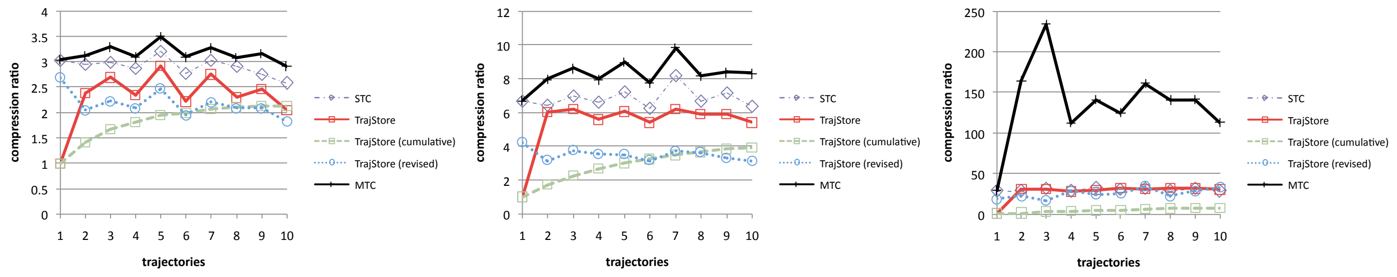
T.time	4	5	6	7	8	9
R.time	0	1.1	2	3	4	5

latitude	0	1.1	3	6	10	15
longitude	0	0	0	0	0	0

Fig. 2. Compression based on Time Mapping. A time mapping T' between T and R is constructed by finding, for each time value on T, the corresponding time value on R. Then, extraneous entries (i.e., those whose removal does not increase compression error beyond a bound) are eliminated from T'. In this example, only the first and last entries on T' are stored (see those highlighted).

Figure 2 illustrates our solution using an example where trajectories R and T are obtained from two cars which traveled along the same route. The car representing R, Car-R, started at time 0, while the car representing T, Car-T, started at time 4. In this example, both vehicles moved similarly with Car-T initially being slightly faster than Car-R. Furthermore, both vehicles accelerated during the first 5 seconds. Therefore, entries in R and T cannot be accurately estimated using linear interpolation with neighboring entries (i.e., STC cannot effectively compress R and T). For example, if tuple $(1, 1, 0)$ is removed from R, that entry will be estimated as $(1, \frac{0+3}{2}, \frac{0+0}{2}) = (1, 1.5, 0)$. Similarly, removing any entry from R except for the first and last entries would cause an error of 0.5. Given an error bound of 0.3, no entries on R can be removed.

In Figure 2, for each time value on T, our solution finds the corresponding interpolated time value on R. For example, for the time value 5 on T and its associated position $(1.1, 0)$, a linear interpolation on R obtains 1.1 as the corresponding time value on R (as shown by "R.time" in T'). After the time mapping T' between T and R is obtained, our solution can estimate the position on T for any time value t. For instance, the time value 5 on T corresponds to the time value 1.1 on R according to T', and then to position $(1.1, 0)$ according to interpolation on R for time 1.1. Let $T'(t)$ denote the estimated position on T for time value t. Then, T' can be treated as a



(a) Error Bound: 1 meter (b) Error Bound: 10 meters (c) Error Bound: 100 meters

Fig. 3. Impact of Error Bound on Compression Ratio

Algorithm 1: $add(T, \mathcal{C}, \mu)$

```
input : trajectory $T$, collection of compressed trajectories $\mathcal{C}$, error bound $\mu$
1 if $|\mathcal{C}| = 0$ then // $\mathcal{C}$ is empty
2     add the result of $stc(T, \mu)$ to $\mathcal{C}$; // STC
3 else
4     $R^* \leftarrow \arg\min_{R \in \mathcal{C}} |mtc(T, R, \mu)|$; // T's reference
5     add the result of $\min(mtc(T, R^*, \mu), stc(T, \mu))$ to $\mathcal{C}$;
```

Algorithm 2: $mtc(T, R, \mu)$

```
input  : trajectory $T$, reference trajectory $R$, error bound $\mu$
output : compressed representation $T'$ of trajectory $T$
1  $S \leftarrow$ null; // current sub-trajectory
2  for $(t, p) : T$ do // for each time-point pair in T
3      $(t', p') \leftarrow$ time-point pair indicating projection of $p$ on $R$;
4      if $distance(p, p') < \mu$ then // reference compression
5          if $S$ is null or not a relative trajectory then
6              set $S$ to a new relative trajectory referencing $R$;
7              add $S$ to $T'$;
8          add $(t, t')$ to $S$;
9      else // reference compression not possible
10         if $S$ is null or not an independent trajectory then
11             set $S$ to a new independent trajectory;
12             add $S$ to $T'$;
13         add $(t, p)$ to $S$;
14 for each sub-trajectory $S$ in $T'$ do
15     set $S$ to the result of $stc(S, \mu)$; // STC
16 return $T'$;
```

conventional trajectory and thus can be compressed using an STC algorithm. In Figure 2, the time values from T and R are highly correlated (the correlation coefficient is 0.9998). In this example, all of the entries between the first and last entries of T' can be safely removed because these removals introduce errors smaller than the user-specified bound of 0.3. As a result, the compressed representation T' in Figure 2 is smaller than that of Figure 1 (i.e., the result of the TrajStore approach). T' is also smaller than T, which is the compressed representation of T using STC since STC is unable to remove any entries given an error bound of 0.3.

B. Implementation Details

Algorithm 1 describes how our solution compresses each trajectory T while keeping the compression error under a bound μ. It also stores the compressed representation T' of T in a collection $\mathcal{C}$. When T is the first trajectory (line 1), T is compressed using an STC algorithm such as TD-TR or SQUISH (line 2). Otherwise (line 3), our solution finds a reference R^* that leads to the smallest compressed representation of T (line 4). In the worst case scenario, if MTC has a compression ratio lower than STC, then we simply use STC (line 5).

Algorithm 2 describes our approach for compressing T using a reference R given an error bound μ. This approach splits each trajectory T into (i) sub-trajectories referencing R (lines 4-8) and (ii) sub-trajectories containing a subset of points from T (lines 9-13). We refer to the former as *relative sub-trajectories* and the latter as *independent sub-trajectories*. The relative sub-trajectories are constructed using the points whose projections on R are sufficiently close (i.e., with a distance smaller than μ). For a time-point pair (t, p) with a close projection (t', p') on R (lines 3-4), (t, t') is inserted into a relative sub-trajectory S (line 8). If no such sub-trajectory is available (line 5), then a new sub-trajectory that references R is constructed (line 6). This new sub-trajectory is also added to trajectory T' (line 7), which tracks all of the sub-trajectories constructed from T (lines 7 and 12). After splitting T as above, each sub-trajectory is compressed using an STC algorithm (lines 15 and 16). As described in Section III-A, our solution provides an abstraction that allows relative trajectories to be compressed using any STC algorithm.

IV. Preliminary Evaluation Results

We conducted experiments to compare our compression technique to those summarized in Section II. For these experiments, we downloaded the public BerlinMOD data set with a scale factor of 0.05 [9]. From this data set, we obtained 10 similar trajectories, each of which contained approximately 1000 entries. Our experiments used meters as the measurement of distance and seconds as the measurement of time. We set the bound on tolerable error (μ) to 1 meter, 10 meters, and 100 meters. For a vehicle that moves at 100 kilometers per hour, these error bounds correspond to time errors of 0.036, 0.36, and 3.6 seconds, respectively (e.g., the vehicle can travel 1 meter during 0.036 seconds and 100 meters during 3.6 seconds).

Figure 3 shows the impact of the error bound on the compression ratios achieved by trajectory compression techniques. Figures 3(a), 3(b), and 3(c) illustrate the results obtained for the error bounds of 1 meter, 10 meters, and 100 meters, respectively. Each curve in these figures shows the performance of a compression technique with each point indicating the compression ratio for an individual trajectory.

In Figure 3, the curves labeled "STC" show the performance of TD-TR. In general, TD-TR achieves the highest compression ratio among STC algorithms, but incurs high computational overhead. For larger error bounds, it can be seen that TD-TR eliminates more points, leading to a higher compression ratio.

In our experiments, TrajStore selected the first of the 10 trajectories as the central trajectory. To the best of our knowledge, TrajStore does not compress central trajectories using an STC algorithm. For this reason, the first trajectory was uncompressed (i.e., the compression ratio was 1). We did not apply TrajStore's delta encoding scheme since it can also be used for all other compared compression techniques. The subsequent trajectories were compressed by referencing the central trajectory as described in Section II. In all of these cases, however, TrajStore did not have a significant benefit over TD-TR.

As Figure 3 shows, in the case of TrajStore, the compression ratio for each trajectory varied significantly. For this reason, we also calculated the cumulative compression ratio. When a set of trajectories are compressed, this ratio refers to the cumulative size of the original trajectories, divided by the cumulative size of their compressed counterparts. Our results demonstrate a relatively low cumulative compression ratio when the error bound is large (see "TrajStore (cumulative)" in Figure 3(c)). This is due to the fact that it does not compress the first trajectory. To overcome this limitation, we also examined the case where the first trajectory was compressed. In this case, the subsequent trajectories were less effectively compressed (see "TrajStore (revised)" where an error bound of 0.4μ was used to compress the first trajectory).

In all of our experiments, our approach (see "MTC" in Figure 3) outperformed other trajectory compression techniques. Our approach uses a time mapping to relate similar trajectories. This mapping, in general, can be effectively compressed due to a high correlation between time values from similar trajectories. Our method also has the unique benefit that for large error bounds, it achieves much higher compression ratios than other methods (Figure 3(c)).

V. Future Work

Our recent work on multiple trajectory compression raises various research challenges. We plan to address them as follows:

Indexing. Our approach compresses each trajectory by referencing another trajectory. Our current work iterates through all of the previously compressed trajectories to determine the reference that yields the smallest compressed representation. We plan on developing an indexing mechanism for quickly finding relevant references.

Multiple References. Compression with only one reference may not be ideal for situations where a trajectory spans multiple similar trajectories. For this reason, we intend to develop a framework that exploits these opportunities by allowing for one reference per sub-trajectory rather than for one trajectory as a whole.

Memory Hierarchy. We have been storing trajectories only in main memory. One potential issue is that the main memory may be too small to hold the entire set of trajectories. Therefore, we plan to develop a system for managing large amounts of trajectory data on disk. A key challenge in this research is to minimize disk I/O by collocating highly-related trajectory data within the same disk block and caching frequently accessed data in memory.

Distributed Trajectory Data Management. We envision situations where massive amounts of trajectory data necessitate the use of a large server cluster. We intend to develop techniques for distributing trajectory data over multiple servers in a manner that minimizes network overhead and balances the resource utilization of the servers.

Querying. We wish to develop a framework for processing various types of queries on trajectory data. This framework needs to utilize our compression technique not only to reduce storage costs, but also to improve query speed.

Experiments. In this paper, we compared the performance of compression techniques using the BerlinMOD data set (Section IV). We plan to extend our experiments by using more synthetic and real-world data sets.

VI. Conclusions

This paper presents a new technique for compressing multiple trajectories. This technique splits trajectories into sub-trajectories based upon their similarities. Each sub-trajectory is then re-expressed using only a time mapping between that sub-trajectory and another similar sub-trajectory. In general, such a time mapping can be highly compressed due to a strong correlation between the time values of similar trajectories. Our evaluation results demonstrate the advantage of our technique over previous trajectory compression solutions.

References

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

[2] J. F. Srour and D. Newton, "Freight-Specific Data Derived from Intelligent Transportation Systems: Potential Uses in Planning Freight Improvement Projects," *Transportation Research Record*, vol. 1957, pp. 66–74, 2006.

[3] E. McCormack and M. E. Hallenbeck, "ITS Devices Used to Collect Truck Data for Performance Benchmarks," *Transportation Research Record*, vol. 1957, pp. 43–50, 2006.

[4] N. Meratnia and R. A. de By, "Spatiotemporal Compression Techniques for Moving Point Objects," in *Proceedings of the 9th International Conference on Extending Database Technology (EDBT)*, 2004, pp. 765–782.

[5] 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*, vol. 10, no. 2, pp. 112–122, 1973.

[6] J. Muckell, J.-H. Hwang, V. Patil, C. T. Lawson, F. Ping, and S. S. Ravi, "SQUISH: An Online Approach for GPS Trajectory Compression," in *Proceedings of the 2nd International Conference and Exhibition on Computing for Geospatial Research & Application (COM.Geo)*, 2011, p. 13.

[7] J. Muckell, J.-H. Hwang, C. T. Lawson, and S. S. Ravi, "Algorithms for Compressing GPS Trajectory Data: An Empirical Evaluation," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*, 2010, pp. 402–405.

[8] P. Cudré-Mauroux, E. Wu, and S. Madden, "TrajStore: An Adaptive Storage System for Very Large Trajectory Data Sets," in *Proceedings of the 26th International Conference on Data Engineering (ICDE)*, 2010, pp. 109–120.

[9] BerlinMOD, http://dna.fernuni-hagen.de/secondo/BerlinMOD/BerlinMOD.html.