Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2020) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 11th International Conference on Ambient Systems, Networks and Technologies (ANT)
April 6-9, 2020, Warsaw, Poland

Multi-attribute reduction of GPS data trajectories: a new approach

Vlad Usyukov*

Ministry of Transportation of Ontario, 159 Sir William Hearst Ave, Toronto, ON M3M0B7

Abstract

In this paper, we investigate the multi-attribute compression of GPS paths. Most of the research in this area was done to reduce spatiotemporal details of travel paths that draw on the object's position in a time-space domain. Although these attributes are essential, the significance of other characteristics, such as instantaneous speed and altitude cannot be ignored, due to their importance to location-based applications. We proposed a new approach to tackle this task. Our approach uses an adaptation of the dead-reckoning algorithm for compressing a spatial component, and equal interval classification for reducing instantaneous speed and altitude attribute values.

*Keywords:*GPS/GLONASS; multi-attribute compression; data feed; dead-reckoning; equal interval classification

1. Introduction

The use of global navigation satellite systems, such as GPS and GLONASS, in mobile devices (e.g., smartphones, wearables, in-vehicle navigation), saw an impressive increase in trend over the past five years. In 2018, in Canada, 25.4M, or 86% of the population, were estimated to own a smartphone with navigational capabilities, which was an increase of 41% between 2013 and 2018 [9]. The popularity of mobile devices instigated a market for location-based services that are familiar to most as mobile applications. The data for these applications typically comes from the monitoring of mobile devices, while their location constantly changes [7]. We will refer to these locations hereafter as trajectories or paths.

The main observations about trajectories are that they are large and that their volume is continuously growing.

* Corresponding author: Vlad Usyukov
E-*mail address:*vlad.usyukov.at.gmail.com

For example, Uber, a ride-hailing company, in 2018 reported that, on average 14M trips were made daily using their mobile application [10]. By a crude estimate, this number of trips would need over 400GB of data storage every day, if data is collected at a rate of 2 seconds, with an average travel time per trip of 15 minutes. A massive data volume presents a range of problems, starting from storing and transmitting to querying it for various business purposes [6]. Nowadays, digital data storage is not of much of an issue as it was before due to substantial progress in this area. However, two other problems remain to be a challenge that can be resolved by data reduction algorithms.

Several trajectory minimization algorithms were proposed before [2-7]. Most of the algorithms are focused on reducing spatiotemporal details only. Considering that data trajectories have more attributes than just time and location, this problem requires special attention.

In addition to the spatiotemporal details, trajectories can have attributes such as instantaneous speed, altitude, quality of a satellite signal, temperature, and even a person's pulse if data comes from a fitness tracker. We should not ignore the importance of these attributes. For example, speed can be used to obtain acceleration and deceleration patterns. Car insurance companies can use this information to estimate insurance premiums for their customers. As for altitude, it provides a more accurate position of an object in space. Altitude, in addition to latitude and longitude, can be utilized to obtain cyclists' preferences for a route choice, and these preferences can be used for prioritizing cycling infrastructure [1]. The ability to minimize data of multi-attribute paths is essential, and it will become even more so in the future, as more sensors will be part of mobile devices.

In this paper, we propose an approach to minimize data of multi-attribute GPS paths. This method is using an adaptation of the dead-reckoning algorithm to reduce the spatial details of paths while having a better capture of non-linear sections, such as road turns. As for instantaneous speed and altitude attribute values, we proposed to use equal interval classification. This classification type was found suitable to reduce data size while preserving critical points of paths. Also, this type of classification provides a framework for reducing attribute values that show an unpredictable pattern that cannot be solved by traditional line-based algorithms. To summarize, our main contributions are:

- a modification to the dead-reckoning algorithm, allowing a better capture of the non-linear sections;
- an application of equal interval classification for reducing attribute values that have unpredictable patterns;

The next section describes the background behind the data paths reduction algorithms, along with utilized error metrics. In section 3, we present our methodology. The application of the dead-reckoning algorithm and the equal interval classification is described in section 4. In section 5, we provide results and recommendations for future work.

2. Background

Most of the data trajectory reduction algorithms work in either online or offline mode. Algorithms, working in an online way, are designed to process data on the fly, while algorithms working in the offline mode are used when an entire data trajectory is known. Our approach can be applied in both cases, and so the most common types of algorithms are reviewed here.

The most commonly used strategies for reducing trajectories are uniform sampling, dead-reckoning, opening windows, and SQUISH-E. Uniform sampling keeps every *i-th* point at fixed time intervals. The advantages of uniform sampling are ease of implementation, plus the ability to reduce most attribute values all at once, including speed and altitude. However, this compression type is not sensitive to idiosyncrasies present in data, and very often, key features, like road turns, can be missed.

The first application of the dead-reckoning (DR) algorithm to reduce data paths was found in [*7*]. This algorithm looks at the agreement between a mobile object and a server that receives regular updates about an object location. This specific set up reduces communication, as well as digital data storage and transmission costs, between a mobile object and a server due to the imperfection of the object's motion. This shortcoming can be expressed as an angular or a distance deviation δ away from the expected trajectory, where δ is a user-defined threshold. The process starts by reading the first point and its velocity vector. Next, these variables are combined to obtain the expected trajectory of the moving object. The expected path is then extended as an infinite ray, and the position of each following point is compared to the infinite ray to determine a difference between the two. If this difference is below a threshold, then a mobile object doesn't send any updates, and the point is rejected. However, if the difference is above the threshold,

then a point immediately before it is saved, and its velocity is used to obtain the subsequent expected trajectory. The process continues until all points are read. In comparison to a uniform method, this algorithm performs better in identifying spatial details, but it has certain limitations. First, the use of an infinite ray becomes tricky as an angular deviation at vast distances becomes less sensitive to register small spatial variations. Second, the use of a user-defined threshold opens a room for debates.

The next two data trajectory reduction algorithms are opening windows [3] and SQUISH-E [4]. Both methods represent an adaptation of the Douglas-Peucker algorithm and require one input parameter from a user to set a threshold for a resulting spatial error. As noted previously, a user-defined threshold can be questioned, when an optimum value can be obtained instead. Also, both algorithms reduce spatiotemporal details only, and therefore they have a limited application for minimizing extra attributes present in trajectories.

2.1. Metrics used to evaluate results

We used the following metrics to evaluate results: percentage of the actual captured length, compression loss ratio, and the optimum number of classes ratio. The rate of the actual captured length is a ratio of a compressed to the nominal trajectory length. The compression loss ratio represents a relationship between the difference of the nominal and the compressed number of records to the nominal number of records. The optimum number of classes is a ratio of the rate of change of the compressed number of records to the rate of change of the number of classes.

3. Methodology

Our main contributions are a modification to the DR algorithm and an application of the equal interval classification. The DR algorithm was modified by searching for an optimum threshold to allow better capture of the non-linear sections of paths, and we used it to reduce the spatial details only. Also, we used the equal interval classification to reduce speed and altitude values. The separate parts from both applications were then combined to obtain a final list of reduced paths. A diagram for the DR algorithm is shown in Figure 1.

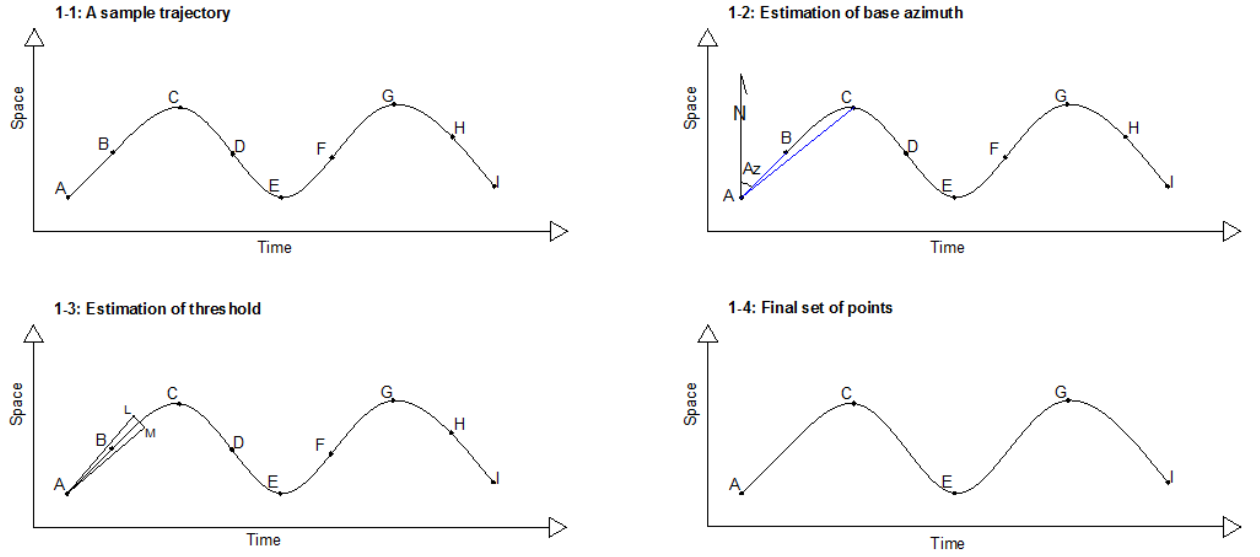

Fig. 1. Dead-reckoning algorithm

Figure 1 consists of four sub-plots, showing a sample trajectory (1-1), an estimation of a base azimuth (1-2), an evaluation of a threshold (1-3), and a final set of points (1-4). Figure 1-1 shows a sample path, consisting of nine points, A to I, where each point has latitude and longitude. The algorithm begins in sub-figure 1-2, by reading and saving point A, which is the first point. Then a second point B is read, and an azimuth AzAB between them is

computed. This angle is a base azimuth, also referred to as an infinite ray. (*Please note that azimuth is as an angular quantity measured clockwise from a reference North direction in the spherical coordinate system*). Then, we read the third point C, and compute azimuth AzAC between points A and C.

In sub-figure 1-3, an absolute difference is computed between two angles, AzAB and AzAC, to obtain a degree of linearity between rays AB and AC. If it's less than a threshold, then we save point C and remove point B. If it's above a threshold, then a point immediately before C, which is B, is selected, and a new base azimuth AzBC is stored, and the process continues until all the points are read. In our work, we searched for the optimum angular threshold using the metric percentage of the actual captured length. This metric allowed us to remove the bias from setting it manually. Also this approach allowed a better capture of the non-linear sections of roads. The final set of points is shown in sub-figure 1-4.

3.1. Equal interval classification

The concept behind the equal interval classification is shown in Figure 2. This method takes a range of values and then divide them into classes of equal size [11].

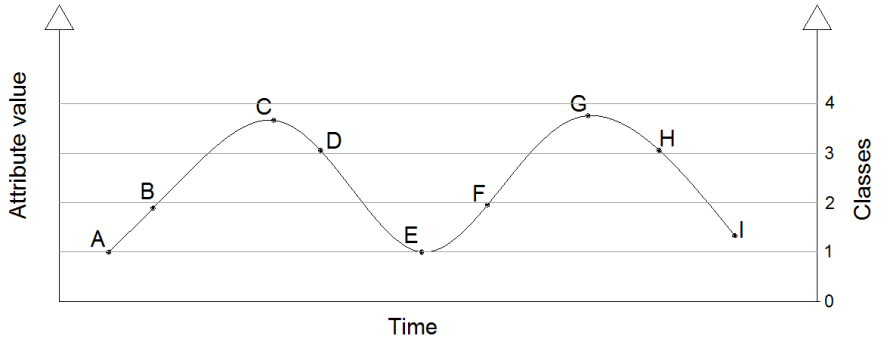

Fig. 2. Equal interval classification

This classification type allowed to reduce attribute values that have unpredictable patterns, that speed and altitude had, and where traditional line-based algorithms fall short. Typically, this classification method requires a user to specify a desired number of classes to perform classification. We changed this requirement from a user-defined variable to finding the optimum value using the metric optimum number of classes ratio. The rationale behind this metric was to select a class value to be large enough to capture data variability of attribute values and small enough to allow efficient data reduction to take place. For data reduction purposes, we elected to keep only the first occurrence of a class value as we move along a path. Thus, the maximum error was limited to the class width, where a class width is a ratio of data range of attribute values divided by the number of classes.

The algorithm begins by reading a first point attribute value to find its class and save it to the list, which is point A. Then, the next point B is read, and its class value is determined. If the class value of the next point is the same as the previous one, which is, then this point is removed, on the rationale that it doesn't bring any new information; otherwise, it is saved. Following this example, points A, C, E, G, and I were saved, and the other ones were removed.

4. Evaluation

This section presents the data requirements and evaluates the proposed data trajectory reduction algorithms.

4.1. Data requirements and analysis of data trajectories

This study builds on using GPS data feed collected from 108 cyclists in Waterloo, Ontario, in 2011. Cyclists were given low-cost mobile devices to record their activities for two weeks. Data were sampled at a rate of every 3-5 seconds and included latitude, longitude, altitude, timestamp, and instantaneous speed. Raw data feed contained

more than 2M points. Data feed was cleaned and then separated into individual paths. For more information on this study, as well as data cleaning and how to break data feed into trajectories, refer to [8].

Table 1. Data paths details.

Location	Mode	Org. # of trajectories	Selected #of trajectories	Size of clean data
Waterloo, Canada	Bicycle	8,788	7,216	129 Mb

Per Table 1, raw GPS data feed contained 8,788 trajectories, but only 7,216 were selected because we used paths that were larger than 1KB in size. We then analyzed their spatial characteristics, and a sample trajectory is shown in Figure 3. Per Figure 3, a reader can observe that spatial details follow closely the geometric design of roads, made of linear and curved elements, and the linear sections of trajectories are good candidates for data reduction.

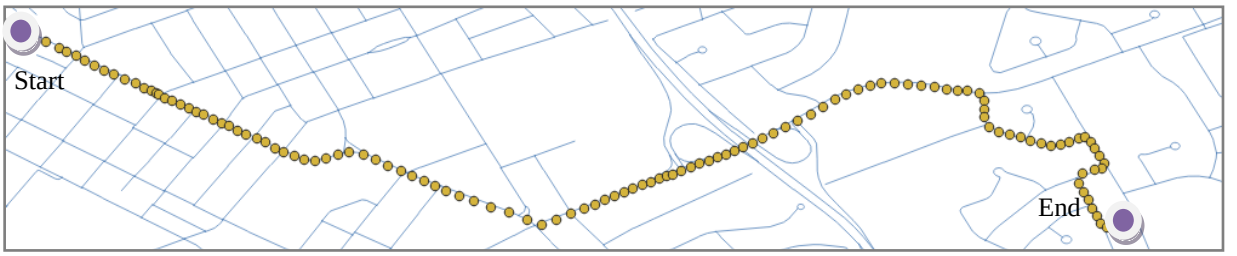

Fig. 3. An example of a GPS data trajectory

The typical profiles for altitude and instantaneous speed attributes are presented in Figure 4. The altitude values ranged between 200 to 400 meters, and this range corresponded to the acceptable elevation range in Waterloo. The instantaneous speed range was between 0 and 40 kph, and we selected it as the acceptable range. Per Figure 4, the shapes of both attribute values have very unpredictable profile. Thus, the use of conventional linear-based reduction algorithms was not suitable to capture the variability present in data. Instead, we used an algorithm based on the equal interval classification. We applied it separately for altitude and speed attribute values.

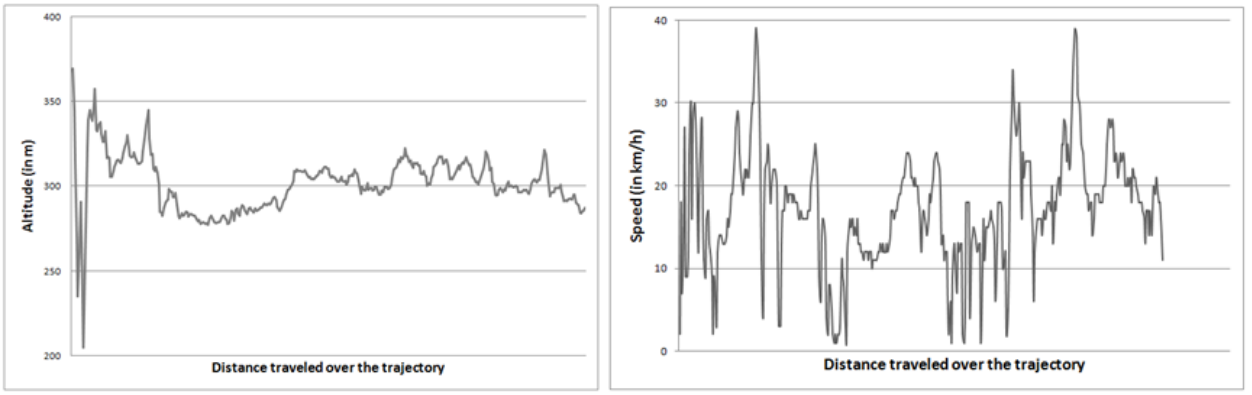

Fig. 4. A sample of altitude (left) and speed (right) profiles

4.2. The optimum threshold for the DR algorithm

Per Section 3.1, the DR algorithm requires an angular threshold to compress trajectories. To find the optimum value, we ran the algorithm from 0.1 to 10 degrees at a 0.1-degree step, thus creating 99 test cases for 7,216 paths. For each path of each case, we then computed two metrics: a percentage of the actual captured length and a percentage of the compression loss ratio. For each test case, we then averaged both metrics and plotted them in Figure 5.

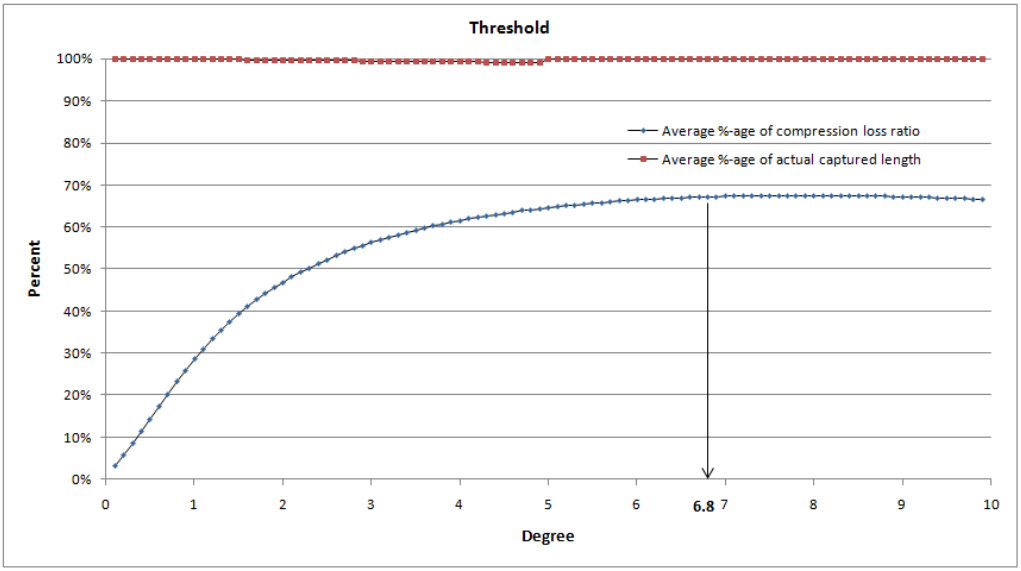

Fig. 5. A threshold for the DR algorithm

Per Figure 5, the optimum value is located at 6.8 degrees. At this value, a curve, made of the average percentage of compression loss ratio values, becomes asymptotic to the x-axis, which corresponds to a maximum attainable compression while gathering almost 100% of the average nominal length of trajectories.

4.3. The optimum number of classes for altitude and speed attribute values

The base form of the equal interval classification requires a user to specify the number of classes to perform classification. Instead of having a user-defined variable, we searched for the optimum number of classes, separately for altitude and speed values. The process was done iteratively, by taking a number of classes, and then computing a corresponding metric percentage compression loss ratio. If the metric did not change by adding one more class, then this number of classes was found to be the optimum.

Table 2. Number of classes for the altitude attribute

# of classes	Ave. % of loss ratio	Change in % of loss ratio	Optimum # of classes metric
1	99	-	-
5	83	16	0.03
10	69	14	0.03
15	59	10	0.02
20	52	8	0.02
25	46	6	0.01
30	41	5	0.01
35	37	4	0.01
40	34	3	0.01
45	32	3	0.00

The algorithm was run separately for the altitude and speed attribute values, and results are presented in Tables 2 and 3. For altitude, we varied the number of classes from 1 to 100 and found that having 45 classes was the optimum value. For this number of classes, we obtained a compression loss ratio of 32%, while having a maximum error of 4.4 meters. At 45 classes, the rate of change of the average percentage of compression loss to the difference in the

number of classes approached zero, which means that the compression loss ratio will not change by adding one more class.

Table 3. Number of classes for the speed attribute

# of classes	Ave. % of loss ratio	Change in % of loss ratio	Optimum # of classes metric
1	99	-	-
5	65	34	0.07
10	47	19	0.04
15	40	10	0.02
20	30	6	0.00

We used a similar approach to find the optimum number of classes for the speed. The process was run iteratively by changing the number of classes from 1 to 40. We found that the optimum number of classes was 20. By having 20 classes, we obtained a compression loss ratio of 31% while having a maximum error of 2 kph. At 20 classes, the rate of change of an average percentage of compression loss ratio to the change in the number of classes approached zero, which means that the compression loss ratio will not change by adding one more class.

4.4. Summary of results

Once we had all the required parameters, they were applied to process GPS data feed, to obtain final reduced paths. To automate these tasks, we developed a program in Python. A snapshot of the program's GUI is shown in Figure 6.

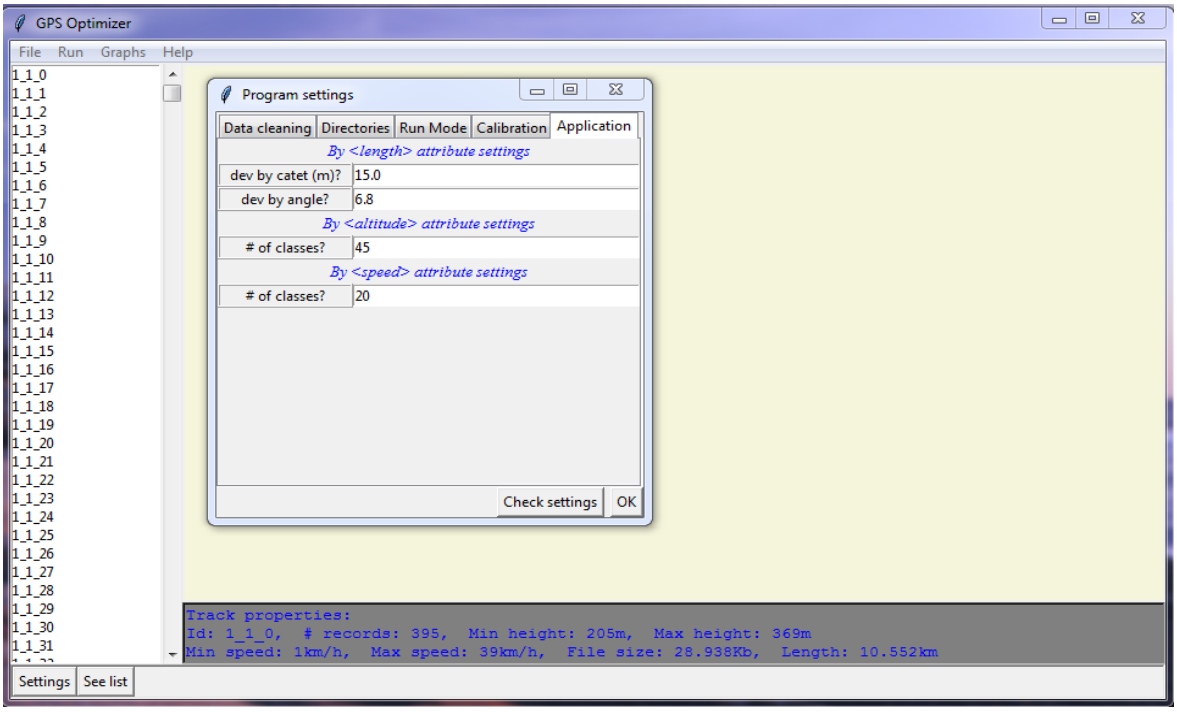

Fig. 6. Data automation program to compress GPS data trajectories

Per Table 4, the application of the DR algorithm alone, allowed to reduce the spatial details of paths by 3.42 times while keeping nearly 100% of the actual lengths. We achieved these results using a 6.8-degree threshold. Our results provided a slight improvement in the compression rate of 3.42 versus 3, published in the study by [7]. The

application of the equal interval algorithm allowed to reduce altitude and speed values by 1.59 and 1.35 times when they were applied separately.

Table 4. Summary of results

Attribute	Parameter	% of compression loss	Compression rate
Lat, long	6.8 degrees	67	3.42
Altitude	45 classes	32	1.59
Speed	20 classes	31	1.35

To find the combined effect from all algorithms, at first each one of them was applied separately. Then we pooled data from individual applications together and removed duplicates to obtain the final paths. Following this process, we reduced data by 1.17 times: from 129 to 110Mb. Regrettably, we cannot compare our results to other studies due to the lack of research in this area.

5. Experiment limitations and future work

For this project, we used two algorithms to reduce trajectories: the DR algorithm to minimize the spatial details, and the equal interval classification, to minimize altitude and instantaneous attribute values. The application of the DR algorithm allowed to reduce spatial features by 3.42 times while capturing nearly 100% of the actual length. The implementation of the equal interval algorithm allowed to reduce altitude and speed attribute values by 1.59 and 1.35 times accordingly. This compression result is a considerable achievement for large data volumes but not as striking as we observed using the DR algorithm alone.

Our work contains certain limitations which present an opportunity for future work. For example, the altitude values were processed separately from latitude and longitude due to the mathematical intricacies of working in three-dimensional space of the spherical coordinate system. We suggest that all spatial details (latitude, longitude and altitude) to be combined into one model, while in this work it was done separately.

Despite these limitations, we find our results positive and comparable with other papers in this area. The reduction of GPS paths is still an active area of research but the importance of having superior algorithms will become more critical in future as the amount of data will keep growing, and the application of location-based services will require more robust algorithms.

Acknowledgements

The author is thankful to Professor Jeff Casello for data feed files. Additionally, the author extends appreciation to a software company JetBrains for providing a community version of PyCharm and to numerous contributors of Python libraries, including Tkinter, geographiclib and geopy that allowed this project to happen.

References

[1] Casello, J., Usyukov, V. (2014), Modelling cyclists' route choice based on GPS data. Transportation Research Board, Volume: 2430, issue: 1
[2] Douglas, DH., Peucker, TK (1973), Algorithms for the reduction of the number of points required to present a digitized line or a caricature. The Canadian Cartographer 10:112-122
[3] Keogh, E., Chu, S., Hart, D, Pazzani, MJ (2001), An on-line algorithm for segmenting time series. In: Proceedings of the 2001 IEEE international conference on data mining (ICDM), pp 289–296
[4] Muckell, J., Olsen, P., Hwang, J-H., Lawson, C., Ravi, S.S (2014), Compression of trajectory data: comprehensive evaluation and a new approach. Geoinformatica 18:435-460 DOI 10.1007/s10707-013-0184-0
[5] Nibali, A., Zhen, H (2015), Trajic: An Effective compression system for trajectory data. IEEE Transactions on Knowledge and Data Engineering, Vol. 27, No.11
[6] Popa, I., Zeitouni, K., Oria, V., Kharrat, A. (2015), Spatio-temporal compression of trajectories in road networks. Geoinformatica 19:117-145 DOI 10.1007/s10707-014-0208-4
[7] Trajcevski, G., Cao, H., Scheuermann, P., Wolfson, O., Vacarro, D. (2006), On-line data reduction and the quality of history in moving objects databases
[8] Usyukov, V. (2017), Methodology for identifying activities from GPS data streams. The 8th International Conference on Ambient Systems, Networks and Technologies (ANT2017)
[9] https://www.statista.com/statistics/467190/forecast-of-smartphone-users-in-canada/ (Accessed on 1/15/2020)
[10] https://www.uber.com/newsroom/company-info/ (Accessed on 1/15/2020)
[11] https://pro.arcgis.com/en/pro-app/help/mapping/layer-properties/data-classification-methods.htm (Accessed on 1/15/2020)