Improving persistence based trajectory simplification

Moritz Laass
Professorship for
Big Geospatial Data Management
Department of Aerospace and Geodesy
Technical University of Munich
Munich, Germany
moritz.laass@tum.de

Marie Kiermeier
Mobile and Distributed Systems Group
Ludwig-Maximilians-Universität
Munich, Germany
marie.kiermeier@ifi.lmu.de

Martin Werner
Professorship for
Big Geospatial Data Management
Department of Aerospace and Geodesy
Technical University of Munich
Munich, Germany
martin.werner@tum.de

***Abstract*—In this paper, we propose a novel linear time online algorithm for simplification of spatial trajectories. Trajectory simplification plays a major role in movement data analytics, in contexts such as reducing the communication overhead of tracking applications, keeping big data collections manageable, or harmonizing the number of points per trajectory. We follow the framework of topological persistence in order to detect a set of important points for the shape of the trajectory from local geometry information. Topological is meant in the mathematical sense in this paper and should not be confused with geographic topology. Our approach is able to prune pairs of non-persistent features in angle-representation of the trajectory. We show that our approach outperforms previous work, including multiresolution simplification (MRS) by a significant margin over a wide range of datasets without increasing computational complexity. In addition, we compare our novel algorithm with Douglas Peucker which is widely respected for its high-quality simplifications. We conclude that some datasets are better simplified using persistence-based methods and others are more difficult, but that the variations between the three considered variants of persistence-based simplification are small. In summary, this concludes that our novel pruning rule Segment-Distance Simplification (SDS) leads to more compact simplification results compared to β-pruning persistence and multiresolution simplification at similar quality levels in comparison to Douglas Peucker over a wide range of datasets.**

***Index Terms*—Trajectory Simplification; Movement Data Analysis; Spatial Computing**

I. Introduction

Movement data is becoming a more and more important resource for our modern societies [1]. For example, efficient transportation research is regularly based on mobility data [2]. Given the current megatrend of urbanization and increasing population densities, such data can pave the way to smart cities, reduced emissions, and sustainable urban growth. Furthermore, the collection of movement data has never been as simple as today. In addition, trajectory computing has been applied in a wide range of domains beyond transportation or movement data analytics including molecular biology [3], medicine [4], image analysis [5], [6], facial recognition [7] or handwriting recognition [8] and signature verification [9].

In this paper, we consider the problem of simplification of spatial trajectories. That is, we ask for a method to remove points from a given trajectory without affecting the spatial shape much. We consider only the spatial aspect of the set of points of the piece-wise linear interpolation of the original trajectory compared to the linear interpolation of the simplified trajectory consisting only of the selected subset of trajectory points.

In this paper, we present a novel method based on a filter and refine framework in which persistence is first applied to find a set of points that we can safely remove and to apply a geometric pruning rule to the remaining points. The method is related to multiresolution simplification [10], but reaches better accuracy with similar runtime performance. Note that this method provides a linear time algorithm which can easily be formulated on a data stream such that simplification can occur on a stream of incoming data.

The remainder of this paper is structured as follows: in Section II, we introduce relevant related work. We explain how persistence can be used for trajectory simplification and give hints on practical issues due to the discrete nature of our trajectory representation. We introduce β-pruning and multiresolution simplification (MRS) as two baseline simplification methods using topological persistence [10]. In Section III, we introduce our algorithm for noise removal from β-pruning and discuss some of its most important properties.

Section IV introduces quality measures and compare both the simplification quality and the speed of the proposed algorithm with its predecessors showing that we reach better quality simplification in comparable time. Finally, Section V concludes the paper.

II. Related Work

Trajectory simplification is a very important aspect of trajectory computing. When we reduce the number of points representing each trajectory, we also reduce the number of relations between two or more trajectories that might impact computation in trajectory data mining applications. This means that even a small reduction in the simplification phase of a trajectory computing pipeline can have orders of magnitude reduction of computational complexity further down the line.

Figure 1 depicts an original trajectory together with the three results of trajectory simplification computed by the algorithms compared in this paper, namely Douglas Peucker, Multiresolution Simplification (MRS), and Segment Distance Simplification (SDS). As one can clearly see, all three methods

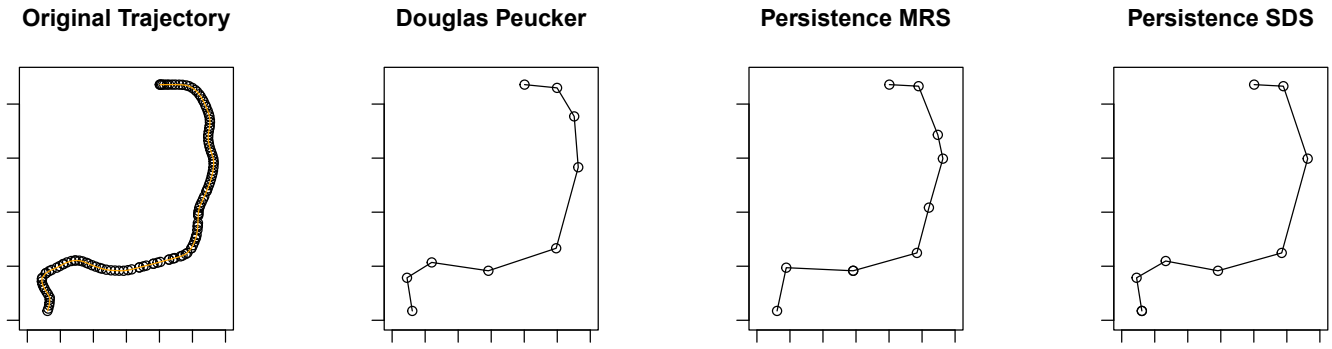


Fig. 1. Line Simplification Examples

significantly reduce the number of points used to represent a trajectory, but all of them use slightly different sets of points leading to differences in the details. Note that trajectory simplification is based on the interpolation assumption and usually formulated with linestrings, that is, with trajectories sampled at points where the interpolation between two consecutive points is given by linear interpolation. The most common setting for trajectory simplification is a purely spatial one, that is, a setting of shape simplification ignoring additional attributes such as time or travel mode.

This is important, because some more advanced trajectory simplification algorithms have been proposed and engineered to protect not only the shape, but parameters such as speed or aspects such as staypoints as well [11], [12].

However, there are many applications such as visualization, path clustering, target detection, or smart routing that do not actually relate to speed information. The presented algorithms in this paper are meant for such applications.

In general, two types of simplification algorithms need to be distinguished: those that simplify a given complete trajectory called *batch processing algorithms* or those algorithms that are applicable to an incoming data stream called *online algorithms*. The persistence algorithm discussed in this paper is capable of performing *online processing*.

We compare our results against the widely-respected Douglas Peucker line simplification algorithm which is known for its excellent objective and subjective quality [13].

A. Douglas Peucker Algorithm

The Douglas Peucker algorithm proceeds in iterations starting with approximating the trajectory just by its first and last point. The algorithm then searches the point of the original trajectory with the maximal distance. If this point is farther away than a predefined tolerance threshold ϵ, the algorithm adds this point to the simplification and proceeds recursively with two new segments formed by the new point and the two segment points. This process is iterated until no point is found with a distance greater than ϵ and the simplification consists of all points that have been added in the process.

The recursion leads to an overall time complexity of $O(n \log n)$ such that very long trajectories can be difficult to simplify.

While the non-linear time complexity is an issue, it should be seen in the light that common online simplification algorithms have cubic time complexity such as the Opening Window Algorithm [14], [15].

Therefore, the search for effective (in terms of quality) and efficient (in terms of time complexity as well as real-world implementation efficiency) trajectory simplification algorithms is still ongoing [10]–[12] and we add a novel method to this vibrant research area.

B. The Persistence Algorithm

The persistence algorithm is given in Algorithm 1. Note that we formulate it now as a batch algorithm for a more intuitive separation of concerns. However, all operations can be interleaved and applied in a streaming setting to an incoming point quite naturally, we will give some remarks on this at the end of this section.

The first step in persistence-based algorithms is to represent the trajectory as a real-valued function mapping time to some real number. This is essential as persistence is going to use the total odering of real numbers in order to formulate a domination rule of features, namely, how persistent they are. For spatial trajectories, ignoring time, a good representation can be the angle function: we represent the trajectory $T = (p_i)$ as $C = (\sphericalangle(p_i, p_{i+1}))$. This function is then analyzed to enumerate local minima and maxima, see line 1 in Algorithm 1. Note that while this sounds easy, it is not that obvious for sampled data and one has to take care to generate a consistent set of minima and maxima in this step. Then, for each minimum, a component is initialized to start at this minimum. A component is a data structure holding a start and end index in the trajectory, respective angle function values, and a flag whether this component is finished, initialized to false. In addition, a flag is associated with each local maximum to track whether this maximum has been used in a component so far. The loop (line 3) now ensures that all maxima are being used (e.g., bound to a component, that is related to a beginning point inside a topological component). Therefore, a

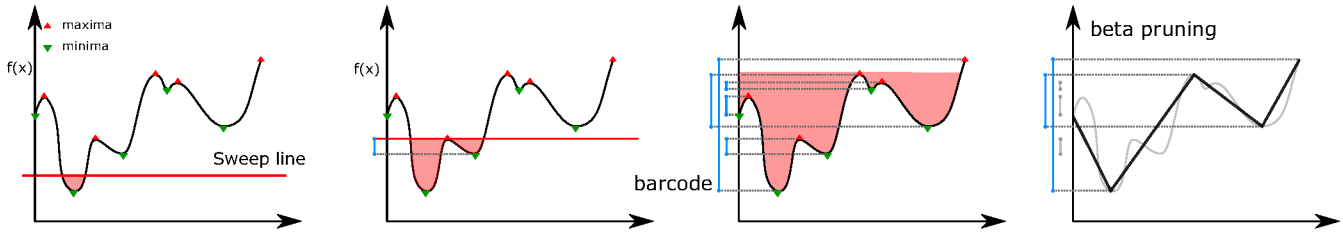


Fig. 2. The process of persistence simplification. A line sweep from smallest to largest local minimum is performed and components (e.g., bars) are initiated at each local minimum and merged with neighboring components during the sweep as soon as they merge together into a single connected component. This leads to a barcode structure which is then used to prune irrelevant, short-living water components for a simplfified curve.

Algorithm 1: Persistence Algorithm with β Pruning

Data: A trajectory T [list of points]
Result: The result S [list of points]
1 *Minima,Maxima* $\leftarrow$ `GetExtrema(GetAngles(`T`))`;
2 Components $\leftarrow$ `InitComponents(`$Minima$`)`;
3 **while** *not all* Maxima *used* **do**
4 **for** *unfinished* C *in Components* **do**
5 P $\leftarrow$ `GrowComponent(`C`)`;
6 **if** $P \in$ $Maxima$, *but used by other Component* C_{other} **then**
7 `MergeComponents(`C, C_{other}`)`;
8 **else**
9 `CloseComponent(`C, P`)`
10 **end**
11 **end**
12 **end**
13 Filtered = $\{c \in Components : \alpha(c_{end}) - \alpha(c_{start}) \leq \beta\}$;
14 **return** $S \leftarrow (p_i \in T : i \in Filtered)_i$

growing algorithm `GrowComponent` (line 5) is being used, where a given component is extended to the left or to the right choosing the step that leads to a point with smaller angle α. If this point is a local maximum, two cases need to be distinguished (line 6): if it is part of another component, we need to merge the current component with other component (line 7), otherwise, we can close the current component using this maximum and storing that this maximum is now used for later nearby merge steps. After generating all components, we remove components shorter than a parameter β (line 13). This is known as β-pruning and reduces irrelevant features of the trajectory. Then (line 14), we build the resulting trajectory as the trajectory built from all points that are part of a component.

For more of such technical details, we refer to the open-sourced implementation of this approach on our web page [1].

Figure 2 depicts the whole process of trajectory simplification using persistence in a visual way.

[1]https://www.bgd.lrg.tum.de/resources/persistence.html

C. Online Algorithm Formulation

For didactic reasons, we gave the persistence algorithm as a batch algorithm by first extracting all extrema, then building all components, and then pruning components. However, it is easy to turn this algorithm into a stream algorithm by noting that the definition of minima and maxima are possible with a lag of 1 (i.e., a point is a local minimum if it is smaller than the predecessor and successor), a bar is opened at any minimum, a bar is closed just based on the set of open bars (which constitutes the state of the algorithm and needs to be held in memory). In addition, growing and merging are local operations and we can implement a streaming algorithm with worst-case linear memory (track open bars, track unused points for GrowComponent). In practice, the number of bars to track is rather small and one can enforce a constant memory online algorithm by limiting the number of elements (points, bars) in the cache. If this reservoir is full, one just removes one of the bars. Therefore it is possible to implement persistence algorithm as a constant-memory online algorithm.

D. Distance Pruning and Multi-Resolution Simplification

In this context, distance pruning has been proposed as an augmentation of the algorithm by [10]. In distance pruning, an additional spatial threshold is used to reject points in the simplification that are too near to other points in the simplification hoping to remedy the situation that β-pruning creates clusters of points at corners reducing efficiency.

Due to the online nature of this algorithm, pure distance pruning would keep the first and last point of a curved subsegment due to a distance constraint of the points and would have difficulty to find a good corner point. Therefore an exponential growing multiresolution approach (MRS) has been proposed.

The MRS algorithm gets a trajectory T, a parameter β for β-pruning, a spatial distance ϵ for distance pruning and a number of rounds R. It applies persistence simplification with beta pruning in rounds, but prunes in terms of space (T^*) with pruning rule $d(p_i, p_{i+1}) < \epsilon$. Note that the pruning measures in space but affects the curve. Between rounds, ϵ is doubled.

III. Persistence with Segment Distance Simplification (SDS)

When analyzing the behaviour of the MRS algorithm, we identified situations in which it does not work very well.

Abstractly speaking, these situations occur in cases where the clear separation of spatial pruning and curvature-based pruning of MRS fails to prune some points, because they are part of longer persistent structures (e.g., not prunable by β-pruning) and far away from each other (e.g., not prunable by spatial distance).

Our contribution is a novel pruning rule which works similar to MRS in very dense clusters, but is able to resolve colinear points in later optimization stages where both the simple distance pruning and the β-pruning fail to remove certain redundant points.

Algorithm 2: SDS Algorithm

Data: Trajectory T
Input: β, ϵ, R
Result: Trajectory S

$C \leftarrow Curve(T)$;
for $i = 1 \dots R$ **do**
  $C \leftarrow \texttt{Persistence}(C, \beta)$;
  $T^* = \texttt{ExtractTrajectory}(C)$;
  foreach (p_i, p_{i+1}) *segment in* T^* **do**
    if $d\left(p_i, \texttt{Line}(p_{i-1}, p_{i+1})\right) < \epsilon$ **then**
      remove p_i from C
    end
  end
  $\epsilon \leftarrow 2\epsilon$
end

The algorithm proceeds similar to MRS by alternating between applying persistence with β-pruning and spatial pruning, but we replace the spatial pruning rule with a triangle rule: we remove points if and only if the point line distance between a given point and the predecessor and successor point is smaller than a threshold ϵ, which is as well grown in rounds.

Note that we expect this approach to be no worse than MRS for the reason that in dense clusters, the line between successor and predecessor is very short such that the point line distance is similar to the distance between a point and its successor. But, we can prune some additional points where this distance is very large and the curvature information does not allow for pruning.

When looking at our novel pruning rule from this angle, one can conclude that it is not purely spatial in terms of distances, but that it rather includes a notion of curvature, because the pruning measure is small if the middle point is almost linear and grows with the angle reaching a maximum at orthogonal corners.

IV. Evaluation

In this section, we evaluate our novel simplification algorithm on a wide range of datasets and for a wide range of parameters. We compare our algorithm with MRS showing that SDS outperforms MRS on average as well as in the vast majority of cases. We identify root causes of how SDS is able to outperform MRS and give a runtime comparison between Douglas Peucker, Persistence, MRS, and SDS.

Name	Size	Type	Source
Character	2,858 trajectories	handwriting	[16]
Geolife	25 mio. points	gps	[17]
Prague	250,000 points	ego-shooter	[18]
Roma	122 mio. points	gps	[19]
T-Drive	15 mio. points	gps	[20]
San Francisco	5 mio. points	synthetic	[21]

A. Datasets

For each of the datasets, we sampled consecutive subtrajectories of 100 points.We did not perform any preprocessing and data gaps are left inside.

B. Evaluation Protocol and Performance Measure

We discuss the power of a simplification algorithm by comparing the number of points used for the simplification to the error introduced in the simplification procedure measured by the Fréchet distance between the original and the simplified trajectory.

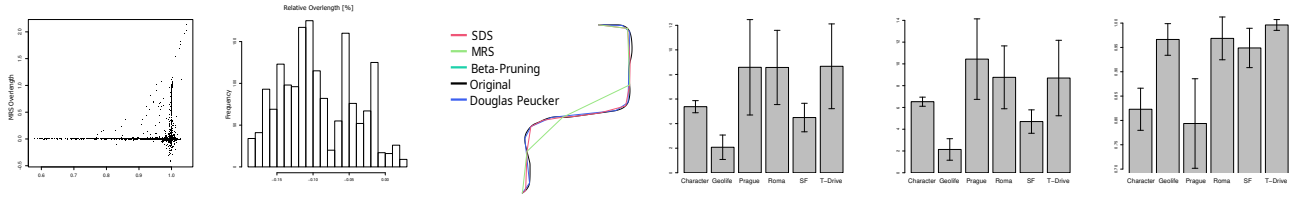
In order for this trajectory distance to be sensible, we first normalize all datasets to contain 100 points per trajectory, choose a number of points uniformly from $[10, 20]$ and search for simplification parameters such that the mean resulting trajectory length across the whole dataset equals this chosen number. If we are able to find such parameters, error measures of trajectories can be safely compared with each other as all algorithms have used the same capacity for representing trajectories.

For the two pruning rules leading to MRS and SDS, we randomly choose an additional parameter κ. We then first calibrate β-pruning to produce κN points and rely on the pruning subsystem to reduce these κN points further to N points. In this paper, κ is chosen as a uniform floating point number between 1 and 4. In this way, we can frequently expect to come up with a sensible intermediate results as $\kappa \in [\min(\kappa)\min(N), \max(\kappa)\max(N)] = [10, 80]$. That is, the first phase of simplification can prune between 20 and 90 points and the second phase the remaining points to reach the target N.

Choosing $\kappa = 1$, of course, reduces both MRS and SDS to the persistence algorithm as there is no "room" for improving towards the target N. The number of rounds for SDS and MRS is chosen between 3 and 5 and is the same across both algorithms.

It is worth noting that all algorithms can fail to create an exactly prescribed number of points: for Douglas Peucker, we choose not not terminate the algorithm at the given number of points, instead we let it run completely with a given fixed parameter ϵ.

We object to choosing parameters per trajectory, as trajectory simplification is in practice applied to unknown trajectories from a certain data source and, therefore, trajectory simplification parameters need to be chosen in advance to knowing the trajectory.



(a) (b) (c) (d) (e) (f)

Fig. 3. Evaluation Results: 3(a) Relative Fréchet Error between SDS and MRS vs. MRS Overlength 3(b) Statistics of the vertical cluster of comparable performance using average Overlength of SDS compared to MRS. 3(c)Illustrative Cases where MRS has difficulties 3(d)SDS vs. DP 3(e)MRS vs. DP 3(f)MRS vs. SDS

In order to evaluate the simplification quality, we employ the Fréchet distance of trajectories, which is a sound distance measure in which time is correctly ignored. As simplification changes the sampling structure quite significantly, the stability of the Fréchet distance with respect to sampling structure is an essential property. We used the open-sourced implementations stemming from the ACM SIGSPATIAL GIS Cup 2017 [22][2].

C. Results

Figure 3(a) depicts the overall result distribution across all random parameters and all datasets comparing the relative performance gain of SDS compared to MRS (X-axis) to the average number of points the MRS result is longer than the SDS result. Two visual blocks can be distinguished: the horizontal block near $Y = 0$ is a set of examples, where both algorithms result in the same number of points yet SDS is often better than MRS in terms of error ($X < 1.0$). The second block is the vertical block near $X = 1$. Here, MRS and SDS lead to comparable error yet MRS has used more points in many cases $Y > 0$.

For the vertical cluster of comparable performance, we give the distribution of how many points shorter the SDS simplification has been observed in Figure 3(b). Note that the X axis is scaled to percentages, hence, it is fair to say that for many cases of observed comparable error, MRS used 5-15% more points to model the trajectory.

There is no sensible error measure for examples in the vertical cluster as it would be obscure to choose a tradeoff between number of points and Fréchet distance for the vertical cluster, hence we stick with the qualitative result that in these cases *SDS outperforms MRS in the sense of comparable error reached with less points*.

For the remainder of the evaluation, we concentrate on those parameter choices where all algorithms where able to produce the essentially same average number of points making the a trajectory distance to the original trajectory a fair performance metric.

First, a qualitative example is depicted in Figure 3(c). This example occured in the Prague dataset and nicely illustrates that the intuition of that spatial pruning might remove points in a way such that not the "middle" point of a cluster survives, but an "early" point is affirmed. In this image, it is obious that the MRS result is not acceptable for this instance and that one would prefer one of the others. You can see that even plain persistence with β-pruning without MRS simplficiation is able to find a better simplification in this case. Though the algorithms are using a different number of points here, they are successfully calibrated to emit the same number of points on average across the whole Prague dataset.

Figure 3(d), 3(e), 3(f) compare the Fréchet distance of given algorithm pairs as the quotient. Therefore, we compute for algorithms A_1 and A_2 the quotient

$$\frac{d_{\text{Fréchet}}(T, A_1(T))}{d_{\text{Fréchet}}(T, A_2(T))},$$

where A_i is one out of SDS, MRS, and Douglas Peucker.

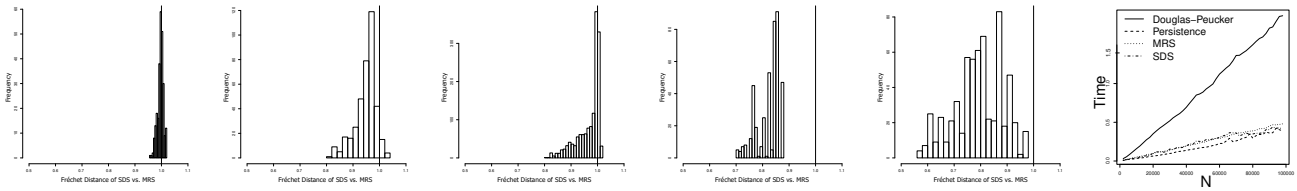
The direct comparison between MRS and SDS is depicted in Figure 3(f). Here, depending on the dataset, *SDS outperforms MRS significantly across all datasets with only very few examples where MRS found better simplifications*.

Figure 4 depicts the comparison of SDS and MRS as a histogram of the observed performances to further illustrate the behaviour that SDS outperforms MRS in many cases quite by a huge margin and that for some cases, the results are comparable. Note again, that all of these statistics include cases where $\kappa \approx 1$ where neither MRS nor SDS are given room for pruning.

You can see similar structures of the distributions across all datasets: there are two short tails to the distribution: one where SDS outperforms MRS by a wide margin and a short one to the right where MRS outperforms SDS. A vertical bar at one simplifies understanding: to the left of one are the cases where SDS is better, to the right are those where MRS outperformed. Notably, for Character and Prague we don't have a single case where MRS outperforms SDS.

Figure 4(f) depicts a scalability experiment with our current implementation illustrating the fact that simplification time of all algorithms grows with the size of the input trajectory. The figures relate to simplifying a boundary of Bavaria with more than 100,000 points. The figure shows the elapsed time of running each algorithm on subtrajectories of this polygon boundary. One clearly sees that persistence-based methods outperform Douglas Peucker by a margin and that SDS and MRS have comparable runtime. The conclusion from this figure is that - within our expectations - simple β-pruning is fastest, MRS and SDS are similar and even in our non-optimized

[2]see https://www.github.com/mwernerds/frechetrange



(a) T-Drive (b) San Francisco (c) Roma (d) Character (e) Prague (f) Wallclock Time

Fig. 4. (a) - (e) Performance Distributions of Fréchet Distance of SDS vs. MRS

implementation faster with a high number of five pruning rounds than an optimized Douglas Peucker implementation. We also confirm that the additional complexity of SDS over MRS is not significant to the runtime.

V. Conclusion

Trajectory simplification is a challenging and important primitive in movement data analysis. In this paper, one philosophy of simplification based on representing trajectories through a time-evolution function of their heading is evaluated carefully in the light of existing work.

We confirm that these linear-time online algorithms are suitable to replace Douglas Peucker in many applications opening up real-time streaming trajectory data simplification with a manageable penalty in simplification error per point. In addition, we confirm that persistence simplification with β-pruning leads to clusters of points in curved areas leading to unneeded redundancy and defects. We evaluate the previously proposed multiresolution pruning strategy known as the MRS algorithm [10] and identify a weakness occurring when the multiresolution pruning ends with colinear points of large distance.

Based on this, we design a novel pruning rule implicitly taking care that both curvature (e.g., local colinearity) and distance are taken into account and show that the novel SDS algorithm outperforms the MRS baseline significantly.

Finally, we provide an open-source implementation as part of libtrajcomp [23]. For future work, we envision to increase the performance of this approach by increasing memory locality and cache-friendliness of the data access pattern to further increase performance.

References

[1] S. Chauhan, N. Agarwal, and A. K. Kar, "Addressing big data challenges in smart cities: a systematic literature review," *info*, 2016.

[2] F. Wu, R. E. Stern, S. Cui, M. L. Delle Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, B. Piccoli, B. Seibold *et al.*, "Tracking vehicle trajectories and fuel rates in phantom traffic jams: Methodology and data," *Transportation Research Part C: Emerging Technologies*, vol. 99, pp. 82–109, 2019.

[3] B. Legrand, C. Chang, S. Ong, S.-Y. Neo, and N. Palanisamy, "Chromosome classification using dynamic time warping," *Pattern Recognition Letters*, vol. 29, no. 3, pp. 215–222, 2008.

[4] T. Syeda-Mahmood, D. Beymer, and F. Wang, "Shape-based matching of ecg recordings," in *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2007, pp. 2012–2018.

[5] A. Marzal, V. Palazón, and G. Peris, "Contour-based shape retrieval using dynamic time warping," in *Current Topics in Artificial Intelligence*. Springer, 2005, pp. 190–199.

[6] I. Bartolini, P. Ciaccia, and M. Patella, "Warp: Accurate retrieval of shapes using phase of fourier descriptors and time warping distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 142–147, 2005.

[7] L. E. M. López, R. P. Elías, and J. V. Tavira, "Face localization in color images using dynamic time warping and integral projections," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2007, pp. 892–896.

[8] R. Niels, L. Vuurpijl *et al.*, "Using dynamic time warping for intuitive handwriting recognition," in *Proc. IGS*. Citeseer, 2005, pp. 217–221.

[9] W.-D. Chang and J. Shin, "Modified dynamic time warping for stroke-based on-line signature verification," in *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, vol. 2. IEEE, 2007, pp. 724–728.

[10] P. Katsikouli, R. Sarkar, and J. Gao, "Persistence based online signal and trajectory simplification for mobile devices," in *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2014, pp. 371–380.

[11] W. Cao and Y. Li, "Dots: An online and near-optimal trajectory simplification algorithm," *Journal of Systems and Software*, vol. 126, pp. 34–44, 2017.

[12] H. Cao, O. Wolfson, and G. Trajcevski, "Spatio-temporal data reduction with deterministic error bounds," *The VLDB Journal*, vol. 15, no. 3, pp. 211–228, 2006.

[13] M. Visvalingam and J. D. Whyatt, "The douglas-peucker algorithm for line simplification: Re-evaluation through visualization," in *Computer Graphics Forum*, vol. 9, no. 3. Wiley Online Library, 1990, pp. 213–225.

[14] Y. Zheng, "Trajectory data mining: an overview," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, p. 29, 2015.

[15] K. Deng, K. Xie, K. Zheng, and X. Zhou, "Trajectory indexing and retrieval," in *Computing with spatial trajectories*. Springer, 2011, pp. 35–60.

[16] M. Lichman, "UCI machine learning repository," http://archive.ics.uci.edu/ml, 2013.

[17] Y. Zheng, X. Xie, and W.-Y. Ma, "Geolife: A collaborative social networking service among user, location and trajectory." *IEEE Data Engineering Bulletin*, vol. 33, no. 2, pp. 32–39, 2010.

[18] M. Werner, "Dataset containing trajectories of four players playing one hour of urban terror capture and hold on the map prague," Available at http://trajectorycomputing.com/datasets/prague-ego-shooter-dataset/, 2014.

[19] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, "CRAWDAD dataset roma/taxi (v. 2014-07-17)," Downloaded from http://crawdad.org/roma/taxi/20140717, Jul. 2014.

[20] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 316–324.

[21] M. Werner, "GISCUP 2015: Notes on Routing with Polygonal Constraints," in *SIGSPATIAL GIS CUP 15, in conjunction with 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2015)*, 2015.

[22] M. Werner and D. Oliver, "ACM SIGSPATIAL GIS Cup 2017 - Range Queries Under Fréchet Distance," *ACM SIGSPATIAL Newsletter*, 2018.

[23] M. Werner, "libtrajcomp," 2017, http://www.github.com/mwernerds/trajcomp.