klcluster: Center-based Clustering of Trajectories

Kevin Buchin[*]
Eindhoven Technical University
Eindhoven, the Netherlands

Anne Driemel[*]
Hausdorff Center for Mathematics
University of Bonn
Bonn, Germany

Natasja van de L'Isle[*]
Eindhoven Technical University
Eindhoven, the Netherlands

André Nusser[*]
Max Planck Institute for Informatics
Graduate School of Computer Science
Saarbrücken, Germany

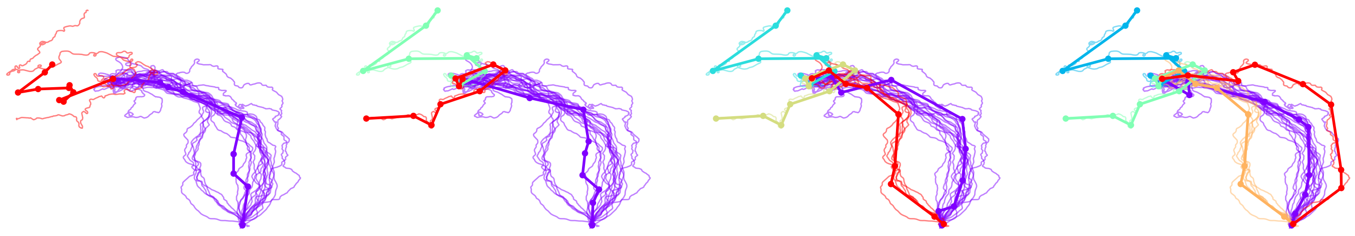

Figure 1: Example of a (k, ℓ)-clustering for the flight paths of a pigeon with the number of clusters k increasing from 2 (left) until 5 (right) and the complexity of the clusters being $\ell = 10$. Trajectories belonging to the same cluster are shown in the same color. For each cluster, a center trajectory generated by the algorithm is shown using thick lines of the same color.

ABSTRACT

Center-based clustering, in particular k-means clustering, is frequently used for point data. Its advantages include that the resulting clustering is often easy to interpret and that the cluster centers provide a compact representation of the data. Recent theoretical advances have been made in generalizing center-based clustering to trajectory data. Building upon these theoretical results, we present practical algorithms for center-based trajectory clustering.

CCS CONCEPTS

• **Information systems → Geographic information systems**;
• **Theory of computation → Computational geometry**.

KEYWORDS

Computational Geometry, Algorithms and Data Structures, Trajectories, Clustering

ACM Reference Format:
Kevin Buchin, Anne Driemel, Natasja van de L'Isle, and André Nusser. 2019. klcluster: Center-based Clustering of Trajectories. In *27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '19), November 5–8, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3347146.3359111

[*]Authors listed in alphabetical order. All authors contributed equally to this research.

1 INTRODUCTION

Clustering is a fundamental task in data analysis that allows to partition the data into groups according to inherent similarities, inferring subpopulations and other hidden structures in the data. Standard clustering algorithms, such as k-means, are designed for point data (i.e., points in a feature space or geographic locations) and are not suitable for clustering trajectories. A trajectory is a series of time-stamped locations tracking the movement of an object recorded by a GPS sensor or other positioning technology. Similar trajectories may have variable length and small temporal deviations, which make the Euclidean distance largely unsuitable for capturing the inherent similarity. For a survey on common approaches to time series clustering we refer to [14].

Generally speaking, we can distinguish two types of clustering tasks for trajectories: to find clusters of similar subtrajectories [6, 8, 15] and, secondly, to group whole trajectories [18, 21]. In this paper we focus on the latter. In particular, we focus on center-based clustering, that is, the clustering partitions the data into groups in such a way that within each group (or cluster) all objects are close to one central object, the *center* of the cluster. To this end, we need to be able to define and measure the similarity between different trajectories. Frequently used similarity measures for trajectories are dynamic-time warping [22], longest common subsequence [21], the Fréchet distance [7], and the discrete Fréchet distance [10].

Center-based clustering can be contrasted with hierarchical clustering under the single- or complete-linkage criterion, which has also been applied to trajectory data in the past [18, 21]. In this approach, the instances are iteratively merged into clusters according to some linkage criterion. In single-linkage clustering two clusters are merged based on the smallest distance between two instances, one from each cluster. In complete-linkage, clusters are only merged if all pairwise distances across the cluster boundary are small. Thus, the latter criterion aims to minimize the *diameter* (the largest distance between two instances) in each cluster. In our setting, we aim to minimize the *radius* (resp., the variance) of a cluster. To this end, we need to generate cluster centers. The center of a cluster should again be a trajectory. An advantage of center-based clustering is that the centers provide a compact representation of the data and the resulting clustering is often easy to interpret.

When clustering points there is a natural center of a cluster: For k-means clustering the center of a cluster is the *mean*, which is the point minimizing the sum of squared Euclidean distances. However, choosing a center from the input is problematic on trajectory data. First, none of the input trajectories might be a good representation for the cluster, for instance, because of local variations in the trajectories. Second, the input trajectories might be of high complexity and thus are not a concise representation. Consequently, using non-input centers and limiting them to a certain complexity (denoted by ℓ), potentially increases the center quality significantly.

This is closely related to the problem of trajectory simplification. However, we need to find a simplification that is valid with respect to *multiple* trajectories. A similar problem was studied by Bereg et al. [4] and later Fan et al. [12], in the context of analyzing protein backbones. Conceptually our approach is similar to the approach by Petitjean and Gançarski [19]. However, while their algorithm is designed for discrete univariate time series, our methods work for continuous trajectories. This difference is crucial: Because the parameter ℓ enforces very compact centers, dynamic time warping becomes less suited as measure for computing such an average. A comparison of the different methods was done by van de L'Isle [20]. One of the outcomes of the study was that a DTW average of two trajectories running in parallel tends to result in the center curve zig-zagging back and forth between the two trajectories.

Our methods are based on recent algorithmic work on center-based clustering of curves and time-series data. Driemel et al. [11] studied the problem of time series clustering under the Fréchet distance. They present algorithms for (k,ℓ)-clustering of time series that run in near-linear time in the input size for constant k, ℓ and approximation factor ε. However, the dependency on these constants is exponential, which makes the algorithm unsuitable for practical purposes. Recently, Buchin et al. [9] extended these results further to multi-dimensional trajectories. In particular, they present a simple and fast approximation scheme.

2 OUR CONTRIBUTIONS

Building upon the work of Buchin et al. [9] and Bringmann et al. [5], we present practical algorithms for (k, ℓ)-clustering of trajectories. Our main algorithm is based on the algorithm proposed by Buchin et al. but is optimized for speed and scalability. In particular it integrates the recent fast Fréchet distance computation by Bringmann et al. [5]. Moreover, we develop heuristics to improve the quality of the centers that the (k, ℓ)-clustering algorithm outputs. To this end, we develop a new method, which we call *Fréchet centering*. This method aims at computing a central curve, based on aligning the curves of the cluster with an initial center curve. Figure 1 shows examples of the computed clusterings.

3 DEFINITIONS

In the following, we define (k, ℓ)-clustering and the (continuous) Fréchet distance following Driemel et al. [11]. Let P and Q denote two polygonal curves, each defined by an ordered sequence of points in the plane. That is, we linearly interpolate between consecutive points in each sequence and obtain piecewise linear parametrized curves $P : [0, 1] \to \mathbb{R}^2$ and $Q : [0, 1] \to \mathbb{R}^2$. We call the points of the initial sequence the *vertices* of the curve and the linear pieces the *edges* of the curve. We call the number of vertices the *complexity* of the curve. The Fréchet distance between two such curves is defined as

$$d_F(P, Q) := \inf_{f:[0,1]\to[0,1]} \sup_{t\in[0,1]} \|P(t) - Q(f(t))\|,$$

where f ranges over the set of continuous and monotone functions $f : [0, 1] \to [0, 1]$ with $f(0) = 0$ and $f(1) = 1$. We refer to such a mapping f as *alignment* between P and Q.

We say a curve P' is an *ℓ-simplification* of a curve P if P' has complexity ℓ and $d_F(P, P')$ is minimal. We define the following variants of (k, ℓ)-clustering. Let $\mathcal{P}$ be a set of input curves. The *(k, ℓ)-center* cost of a clustering with centers $\boldsymbol{Q}$ is defined as

$$\phi_\infty(\boldsymbol{Q}) := \max_{P\in\mathcal{P}} \min_{Q\in\boldsymbol{Q}} d_F(P, Q).$$

Similarly, the (k, ℓ)-*median* cost function ϕ_1 and the (k, ℓ)-*means* cost function ϕ_2 are defined as

$$\phi_1(\boldsymbol{Q}) := \sum_{P\in\mathcal{P}} \min_{Q\in\boldsymbol{Q}} d_F(P, Q), \quad \phi_2(\boldsymbol{Q}) := \sum_{P\in\mathcal{P}} \min_{Q\in\boldsymbol{Q}} (d_F(P, Q))^2.$$

In each of the three clustering variants (center, median, means), the optimal solution to the (k, ℓ)-clustering problem for input $\mathcal{P}$ is defined as the set $\boldsymbol{Q}$ consisting of k polygonal curves, each of complexity ℓ, which minimizes the cost. Note that, when $k = 1$ and $|\mathcal{P}| = 1$, each of the above clustering problems is equivalent to computing an ℓ-simplification of the input curve, since we restrict the complexity of the cluster center(s) and minimize the distance. Not surprisingly, curve simplification will turn out to be a crucial element of the algorithms we use.

4 ALGORITHMS

Our algorithms combine different elements used for center-based clustering in Euclidean space and in metric spaces, and generalizes these to trajectories. The general setup follows Lloyd's popular k-means algorithm [16]:

(1) Compute a clustering (see Section 4.1)
(2) Improve centers (see Section 4.2)
(3) Update clusters and go to 2) unless there was no change

4.1 Initial Clustering

A careful choice of the initial clustering can provide a guarantee on the quality of the resulting clustering. For instance, for the k-means problem in Euclidean space, k-means++ in this way guarantees an $O(\log k)$-approximation [3]. In metric spaces –like the space of trajectories with the Fréchet distance as metric– Gonzalez' algorithm [13] guarantees a 2-approximation for k-center clustering. Both these algorithms select (initial) cluster centers by first choosing one of the input objects at random, and then iteratively choosing cluster centers that have a large distance to all previous centers.

These algorithms do not directly apply to (k, ℓ)-clustering, since the resulting centers would not have complexity ℓ. Buchin et al. [9] recently showed that a 3-approximation for (k, ℓ)-center can be achieved by intertwining Gonzalez' algorithm with simplification algorithms. We implemented their algorithm, but to achieve a clustering algorithm that scales to large data sets we use a fast greedy algorithm for simplification [1] together with a binary search to (approximately) minimize the Fréchet distance between the simplification and the original trajectory for a given ℓ. For (k, ℓ)-center this results in a 6-approximation [6].

In summary, we compute our initial clustering for the input set of trajectories $\mathcal{P}$ by the following algorithm:

(1) For all trajectories $P \in \mathcal{P}$: set $min\_dist\_to\_centers_P$ to ∞
(2) Choose one of the trajectories uniformly at random, and take its ℓ-simplification as the first center
(3) For all trajectories $P \in \mathcal{P}$: compute the distance to the new center, and update $min\_dist\_to\_centers_P$ if necessary
(4) Pick the trajectory P with the largest $min\_dist\_to\_centers_P$, take its ℓ-simplification as the new center and continue at (3) until k centers have been chosen.

Fast Fréchet distance computation. The main bottleneck of our algorithm is the frequent computation of the Fréchet distance. To make this part as fast as possible, we use the state-of-the-art implementation of Bringmann et al. [5]. Their implementation is based on the classical free-space diagram [2], but they use a divide and conquer approach combined with a pruning strategy which can decide large chunks of the diagram.

4.2 Improving Centers

After the initial clustering, we alternate between computing new centers and updating the clustering. A simple approach to compute a new center for a cluster is to compute ℓ-simplifications for all curves in the cluster, and to select the simplification that minimizes the maximum distance (or for k-median/means the sum or sum of squared distances) within the cluster. However, as argued in the introduction, this limits the choices for the center too much.

We therefore propose *Fréchet centering*. As input we are given a set of trajectories C, i.e., the trajectories within a cluster, and an initial center trajectory T^c. First, we compute the Fréchet distance for each trajectory $T \in C$ to T^c. By that we obtain an alignment between T and T^c. For each vertex v of T_c this gives us a matching to a point on each of the trajectories in C. We call this set P_v. For each $v \in T^c$ we now compute the minimum enclosing circle of P_v. We then move v to the center of this minimum enclosing circle. Doing this for all vertices of T^c, we obtain a new center curve. See Figure 2 for an illustration of one step of Fréchet centering. We stop this process when the new center does not improve the induced radius compared to the previous center (i.e., the new center has at least the maximum distance to any curve in the cluster as before).

4.3 Single- and Complete-Linkage Clustering

For the sake of completeness, we briefly sketch the clustering algorithm that we compare to in our experiments. *Complete-linkage* is a hierarchical clustering algorithm, i.e., it produces a series of clusters which are nested. At the beginning of the algorithm, each curve starts out being its own cluster. Then we iteratively merge

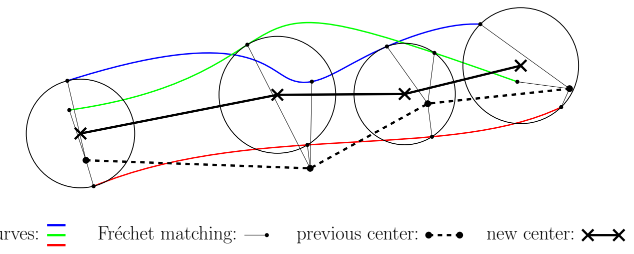

Figure 2: A schematic illustration of Fréchet centering. The colored curves are the input curves for which we want to find a center. We already computed a preliminary center, which is the dashed polyline. To compute a new center, we first compute the Fréchet traversal and thereby a matching from the old center vertices to points on all three curves. By taking the centers of the minimum enclosing circle of those matched points, we obtain our new center, which is the fat non-dashed polyline.

the two clusters that minimize the maximum distance between any two curves in the respective clusters. In other words, we greedily minimize the diameter of the clusters. We stop the process when exactly k clusters are remaining, where k is an input parameter. *Single-linkage* clustering works the same as complete-linkage clustering, except that it merges the two clusters which minimize the closest distance of any two curves in the respective clusters.

5 EXPERIMENTS

5.1 Implementation

To test our new approach, we implemented it in modern C++ only depending on the C++ Standard Library. For a fast computation and decision of the Fréchet distance and to obtain valid traversals, we rely on the implementation of [5]. We conducted our experiments on a laptop with an Intel i5-6440HQ processor with 4 cores.

In total we implemented the following methods[1]: For clustering we implemented Complete-Linkage, Single-Linkage and our adaption of Gonzalez' algorithm as described in Section 4.1. To compute and improve centers, we implemented k-median, k-means, k-center, and Fréchet centering as introduced in Section 4.2. In preliminary experiments single-linkage appeared to be dominated by complete-linkage clustering for our scenario, and we therefore restrict to the latter, using it as a baseline comparison to our clustering algorithm. For center improvement, k-median, k-means, and k-center were far from being competitive with Fréchet centering regarding running time and no significant quality improvement was observed in preliminary experiments. Summarizing, we restrict to a comparison with complete-linkage clustering, using k-means centers.

5.2 Comparison with Complete-Linkage

We are not aware of any similar practical work on clustering with respect to the Fréchet distance. However, a comparison to a baseline approach is necessary for judging the running time and quality of our new approach. Therefore, we compare our work with a complete-linkage clustering (which stops when only k clusters are

[1]Our code is available at https://gitlab.com/anusser/klcluster-sigspatial19

	time (s)	diam.	radius
complete-linkage + k-means center	21.83	0.076	0.0104
Gonzalez + Fréchet centering	2.04	0.076	0.0092

Table 1: Comparison of our approach with complete-linkage clustering and a mean clustering (averaged over 10 runs).

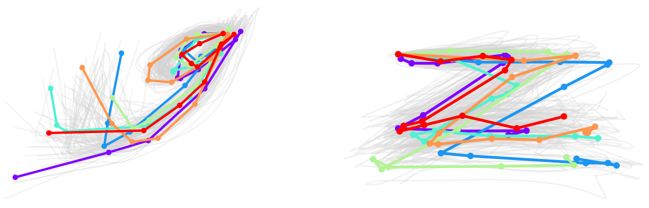

Figure 3: Two examples for a character curve set (gray) and their center curves (colorful). The centers adapt to different shapes of the characters (left) and to scalings (right).

left). As centers we use the best ℓ-simplified curve from each cluster which minimizes the sum of squared distances. To compare the quality of the two clusterings, we compare their

- *diameter,* the largest distance of any curve pair which are in the same cluster.
- *average radius,* the average over all cluster radii (the maximal distance from a curve in the cluster to the cluster center).

We chose these two measures as the first one is what complete-linkage is aiming to minimize, while the second one is what (k, ℓ)-clustering aims to minimize. We applied our algorithm to flight paths of homing pigeons [17]. In Table 1, we show the result of a brief performance and quality comparison measured on all the curves of the Bladon Heath release site of this data set. The experiment is on 168 trajectories with an average hop length of 601.54 and $k = 9, \ell = 8$. On this data, our new approach clearly dominates the baseline approach. Further experiments lead to similar results.

5.3 Examples

Figure 1 shows clusterings of the data of one pigeon for $k \in \{2, \ldots, 5\}$. A small number of data points ($\ell = 10$) is sufficient to capture the shapes of the clusters. For $k = 5$, two main clusters are detected while there are 3 outlier trajectories, defined by the clusters of size 1. The center trajectories (visually) are truthful representations of their cluster.

In Figure 3, we show example clusters from applying (k, ℓ)-clustering to a handwritten characters data set which was already used in [5] as benchmark set for fast Fréchet distance computation. Our experiments showed that our implementation can classify handwritten characters with high accuracy on this data set.

6 CONCLUSIONS

We showed that (k, ℓ)-center clustering combined with Fréchet centering is an efficient approach yielding qualitatively high results. Our implementation is about 10 times faster when compared to the commonly used complete-linkage clustering while yielding clusterings with the same or better fit, not being bound to choose the center from the input set. Moreover, (k, ℓ)-center clustering computes cluster centers that simplify the input trajectories, thereby providing a compact and interpretable representation of the data, making it additionally suitable to be used for visualization of trajectory data.

REFERENCES

[1] Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. 2005. Near-Linear Time Approximation Algorithms for Curve Simplification. *Algorithmica* 42, 3-4 (2005), 203–219.

[2] Helmut Alt and Michael Godau. 1995. Computing the Fréchet Distance between Two Polygonal Curves. *Int. J. Comput. Geometry Appl.* 5 (1995), 75–91.

[3] David Arthur and Sergei Vassilvitskii. 2007. *k*-means++: The advantages of careful seeding. In *Proc. 18th ACM-SIAM Symposium on Discrete Algorithms.* 1027–1035.

[4] Sergey Bereg, Minghui Jiang, Wencheng Wang, Boting Yang, and Binhai Zhu. 2008. Simplifying 3D Polygonal Chains Under the Discrete Fréchet Distance. In *Proc. 8th Latin American Symposium on Theoretical Informatics.* 630–641.

[5] Karl Bringmann, Marvin Künnemann, and André Nusser. 2019. Walking the Dog Fast in Practice: Algorithm Engineering of the Fréchet Distance. In *Proc. 35th Internat. Symposium on Computational Geometry.*

[6] Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristan, Rodrigo I Silveira, Frank Staals, and Carola Wenk. 2017. Clustering trajectories for map construction. In *Proc. 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* ACM, 14.

[7] Kevin Buchin, Maike Buchin, and Joachim Gudmundsson. 2010. Constrained free space diagrams: a tool for trajectory analysis. *International Journal of Geographical Information Science* 24, 7 (2010), 1101–1125.

[8] Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. 2011. Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry & Applications* 21, 03 (2011), 253–282.

[9] Kevin Buchin, Anne Driemel, Joachim Gudmundsson, Michael Horton, Irina Kostitsyna, Maarten Löffler, and Martijn Struijs. 2019. Approximating (k, ℓ)-center clustering for curves. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms.* SIAM, 2922–2938.

[10] Thomas Devogele, Laurent Etienne, Maxence Esnault, and Florian Lardy. 2017. Optimized Discrete Fréchet Distance between trajectories. In *Proc. 6th ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data.* ACM, 11–19.

[11] Anne Driemel, Amer Krivosija, and Christian Sohler. 2016. Clustering time series under the Fréchet distance. In *Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms.* 766–785.

[12] Chenglin Fan, Omrit Filtser, Matthew J. Katz, and Binhai Zhu. 2016. On the General Chain Pair Simplification Problem. In *41st Internat. Sympos. Mathematical Foundations of Computer Science.* 37:1–37:14.

[13] Teofilo F. Gonzalez. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science* 38 (1985), 293–306.

[14] Dimitrios Kotsakos, Goce Trajcevski, Dimitrios Gunopulos, and Charu C. Aggarwal. 2013. Time-Series Data Clustering. In *Data Clustering: Algorithms and Applications.* 357–380.

[15] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. 2007. Trajectory clustering: a partition-and-group framework. In *Proc. ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007.* 593–604.

[16] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137.

[17] R. P. Mann, R. Freeman, M. Osborne, R. Garnett, C. Armstrong, J. Meade, D. Biro, T. Guilford, and S. Roberts. 2011. Objectively identifying landmark use and predicting flight trajectories of the homing pigeon using Gaussian processes. *Journal of The Royal Society Interface* 8, 55 (2011), 210–219. arXiv:http://rsif.royalsocietypublishing.org/content/8/55/210.full.pdf

[18] Nikos Pelekis, Ioannis Kopanakis, Gerasimos Marketos, Irene Ntoutsi, Gennady Andrienko, and Yannis Theodoridis. 2007. Similarity search in trajectory databases. In *Proc. 14th Internat. Sympos. Temporal Representation and Reasoning (TIME'07).* IEEE, 129–140.

[19] François Petitjean and Pierre Gançarski. 2012. Summarizing a set of time series by averaging: From Steiner sequence to compact multiple alignment. *Theoretical Computer Science* 414, 1 (2012), 76–91.

[20] Natasja van de L'Isle. 2018. *Algorithms for center-based trajectory clustering.* Master's thesis. Eindhoven Technical University, the Netherlands. https://pure.tue.nl/ws/portalfiles/portal/125739911/thesis_NatasjaVanDeLIsle.pdf_2.pdf

[21] Michail Vlachos, Dimitrios Gunopoulos, and George Kollios. 2002. Discovering Similar Multidimensional Trajectories. In *Proc. 18th Internat. Conf. Data Engineering.* IEEE, 0673.

[22] Michail Vlachos, Marios Hadjieleftheriou, Dimitrios Gunopulos, and Eamonn Keogh. 2003. Indexing multi-dimensional time-series with support for multiple distance measures. In *Proc. 9th ACM SIGKDD Internat. Conf,. Knowledge Discovery and Data Mining.* ACM, 216–225.