Trajectory Similarity using Compression

Gabriel Dax
Professorship for Big Geospatial Data Management
Department of Aerospace and Geodesy
Technical University of Munich
Munich, Germany
gabriel.dax@tum.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 present a novel approach for trajectory similarity based on Kolmogorov complexity approximated by a lossy compression of the original trajectory data using selected features compressed into a concise memory representation by means of a Bloom filter. Given the importance of trajectory data, a linear-time distance measure with all theoretical guarantees implied by a proper metric is very powerful if it is capturing enough detail for important trajectory mining tasks. This stack of feature extraction combined with feature embedding in a Bloom filter constitutes a lossy compression for trajectory data which can easily be extended with other discrete data like travel mode. In addition, this compression has the needed properties for efficient calculation of a normalized compression distance (NCD) which approximates Kolmogorov complexity. We evaluate this novel trajectory distance measurement using very simple features and k-nearest-neighbor classification on selected real-world datasets with remarkable classification accuracies. Furthermore, we argue that the distance measure is very suited to geospatial big data applications as each trajectory is first transformed into few bits using the lossy compression stack. At time of comparison, the original trajectory geometry is not needed, instead, the sketches suffice. Despite very compressible parameters (equal or less than 1024 bit per trajectory) and very simple features, we already reach classification accuracies for real-world trajectory classification tasks of more than 80% across various datasets.**

***Index Terms*—Compression Distance, Kolmogorov Complexity, Bloom Filter, Similarity Measure, Trajectories**

I. Introduction

Trajectory data comprises an important novel source of geospatial information and is routinely used in many applications such as biology [1], map generation [2], or urban analysis [3]. However, many trajectory distance functions have nonlinear time complexity including the most important distance measures of dynamic time warping (DTW) [4] is Fréchet distance [5]. For some of them, localized versions have been proposed that reduce the time complexity to linear time essentially by ignoring relations between two trajectories that are unlikely to affect similarity such as between temporally distant points. Still, the high quality distance measures are time-consuming to apply and a lot of work has been dedicated to trajectory simplification and dataset filtering for limiting the amount of data that is fed into trajectory distance computations. The ACM SIGSPATIAL GIS Cup 2017, for example, was dedicated to one of these problems, namely to perform a range search in a large database of trajectories relative to Fréchet distance [6]. All winning submissions employ powerful filtering mechanisms based on sketches of the trajectories such as simplifications [7] or bounding box representations [8], [9].

With this paper, we merge these two phases of the typical filter-refine framework of traditional trajectory analysis using a technique related to Kolmogorov complexity. The Kolmogorov complexity is a theory in which the complexity of a string is measured as the length of the shortest program producing this string and is tightly linked to Shannon entropy. Unfortunately, it is not computable [10]. However, it can be approximated by compression. This is at least plausible when realizing that, for a given random string, a program generating this string must be longer than the string containing all the unpredictable sequence in the source code. To the contrary, when a string is compressible, the compression result together with a decompression algorithm constitutes a comparably short program.

In this paper, we extract simple features related to orientation and global location of trajectory points, encode them into a Bloom filter, which we interpret as a compression of the trajectory. Note that this is a lossy compression, hence, we need to empirically evaluate that the information loss during compression is related to mainly uninteresting aspects and that the resulting compressed data blocks are sufficient to successfully compare trajectories.

In the context of Kolmogorov complexity, one can introduce the relative Kolmogorov complexity $K(x, y)$ as the length of the shortest program taking x as the input and producing output y. With this additional notion, a distance measure is extracted from the framework by carefully normalizing this relative complexity.

As a consequence of the non-computability of the Kolmogorov complexity, this metric is not computable as well, yet we can replace the Kolmogorov complexity with real-world compressors and get a working approximation to the distance. Note that we need to compute the joint compression of two trajectory sketches in this setting and that this is possible based on the sketches only as Bloom filters allow for the computation of the union without accessing original set information. That is, given two compressed trajectories, because their joint compression is directly accessible via a binary OR operation.

Finally, we evaluate the approach on several trajectory classification problems and conclude that this distance measure is effective. The remainder of this paper is structured as follows: In Section III we discuss the related work and introduce the basic principles. In Section III we introduce the novel approach of calculating the similarity between two trajectories by using Bloom filter as compressors. Then, Section IV presents the classification examples. Finally, Section V discusses the results and the future work.

II. Related Work and Basic Principles

The Bloom filter is a well-known probabilistic data structure. The usages of these filters cover multiple scientific fields, including classification using the machine learning algorithm k-NN. The experiment performed in [11] showed that it is possible to store trajectories in the data structure and classify those trajectories only based on those filters. Based on this experiment we use this approach to approximate the Kolmogorov complexity to classify trajectories. To our knowledge, little research has been done on the impact of Bloom filters in combination of the Kolmogorov complexity.

A. Trajectory Representation as Strings of Discrete Orientations

The most common data representation of spatial trajectories is a sequence of points, such as GPS coordinates and their corresponding timestamps. While the coordinates construct the trajectory by linear interpolation, the additional attributes can have various relations to the data. However, an alternative representation of trajectories is to store the angles between two or more segments and their lengths. This method adds a rotational invariance to the trajectories. Furthermore, the angles and distances within the cartesian coordinate system of one trajectory can be encoded into a sequence of letters.

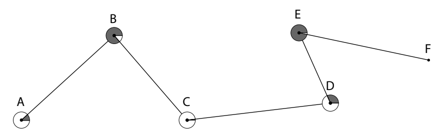


Fig. 1. Visualisation of the shape feature global orientation.

For this representation, two choices of reference frames are possible: a global one, like *North* or *South* and a local one, which is relative to the previous segment of the trajectory, like *turn left* or *turn right*. We discretize this representation in both the angular and distance dimensions and assign a character to each of the discrete values thereby generating a string [11]. For long segments, it is feasible to subsegment those segments and thereby produce repeated characters representing a discrete notion of length. While Figure 1 depicts an example using global orientation, Figure 2 represents an example using local direction.

In order to calculate the orientation with a global direction the angle between 0 and 2π is split into N equal parts and each trajectory point constitutes an instance of the character assigned to the split of the value range.

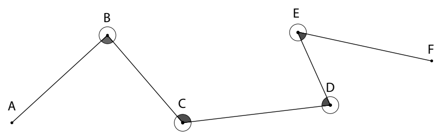


Fig. 2. Visualisation of the shape feature local orientation.

B. Bloom Filter

Bloom filter is a probabilistic data structure, introduced in [12], it represent sets of objects in a freely chosen size of memory, which is done using hash functions. Furthermore, the structure of a Bloom filter includes k pairwise independent uniform hash functions, which are mapping all sets of objects to a non-negative number smaller than the filter size m. The empty set is represented by having the bit array $F = 0$. The filter is able to perform two operations. For inserting an element e, the k hash functions are invoked leading to a set of numbers $h_i(e), i = 1 \dots k$ and the slots are set to one, e.g.,$F[h_i(e)] = 1, i = 1 \dots k$ Reversely, to test if an element e is considered to be in the filter, we test the same locations. An important property of this construction is that the test function does not provide false negative values. Essentially, this error is probabilistically controlled by choosing parameters m and k in relation to the number of elements to be held in the filter n.

The fraction of zeros within a filter can be calculated by using Equation 1.

$$foz = \left(1 - \frac{k}{m}\right)^n \approx e^{-kn/m} \tag{1}$$

C. Normalized Compression Distance

The Kolmogorov complexity $K(s)$ of a string s is defined as the length of the shortest computer program that prints the string s [13]. Additionally, the relative Kolmogorov complexity $K(x, y)$ is the length of the shortest program that gets an input string x and outputs the string y. Obviously, the Kolmogorov complexity of a string without a possibility of a short algorithmic description is higher than the Kolmogorov complexity of a string with a short representation [10].

The Normalized Information Distance (NID) is an approach of a universal similarity metric to calculate the difference of two objects, like strings, using the Kolmogorov complexity K. Note that this construction leads to a metric, yet is not computable as K is not calculable [10]. Within the formula, the complexity of an object x is defined as $K(x)$. Furthermore, the complexity of the concatenated objects x and y is defined with a logarithmic precession as $K(x, y) = K(xy) = K(yx)$ [14]. Equation 2 represents the NID of an object x and y.

$$NID(x, y) = \frac{K(x, y) - \min\{K(x), K(y)\}}{\max\{K(x), K(y)\}} \tag{2}$$

The Normalized Compression Distance (NCD) is an approximation of the NID in which the non-computable Kolmogorov complexity K is replaced with a computable number, namely, the size of a real-world compression function output C and is defined in Equation 3.

$$\mathit{NCD}(x, y) = \frac{C(x, y) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} \tag{3}$$

C represents a real word compressor (such as gzip or bz2) and $C(x)$ represents the length of the compressed object x. $C(x, y)$ represents the "joint" compression of objects x and y, for example, by first concatenating x and y. A compressor will find redundancies across x and y and therefore, this measures relative complexity of x and y. Furthermore, NCD is normalized to $0 \leqslant \mathit{NCD}(x, y) \leqslant 1 + \varepsilon$ with error ε related to the discrepancy between real-world compressors and information-theoretic bounds such as entropy [14].

III. Compression Distance of Trajectories

In this section, we apply the framework of NCD on trajectories. However, it is not easily possible to find a measure of trajectory complexity, where a relative complexity of one trajectory with respect to another one, that is $C(x, y)$ can be defined as well. In this situation, we decided to adopt a Bloom filter as a form of a lossy compression together with a set of simple features to translate trajectories into sequences of features. In summary, our framework first translates a trajectory into a set of words (n-gram) which are then modeled using a Bloom filter. The relative complexity is then given in a canonical way as the complexity of the union of the set of features of both trajectories and Bloom filters directly support the needed computations without going back to the uncompressed trajectory information.

A. Bloom Filter and Trajectory Features

The embedding of trajectory features, such as the global orientation, into the Bloom filter is a central task within the calculation of the novel trajectory similarity. However, our approach uses two types of features, the geohashes and the global orientations of a trajectory encoded into a set of symbols (see Section II-A). Because of the fact that a trajectory can point more than one time into the same direction, like East, and can visit the same Geohash grid cell more than once, it can lead to an information overlay in case of the global orientations. So, it is necessary to consider the frequencies of the symbols to be able to represent more than just a set of all features that appeared. Hence, storing the set of features ignoring their frequencies leads to a distance measure which might not be able to capture enough geometric information for the envisioned trajectory data mining tasks. To mitigate the problem of an information overlay and an enumeration of the features, an incrementing suffix is needed. For example, a sample trajectory has been encoded to the sequence *AGAH* of characters. After adding the frequency to each symbol the new sequence of symbol $\{A_1, G_1, A_2, H_2\}$ allowing a lossless representation as a set.

An advantage of the Bloom filter is that the size of the filter is independent of the number of added elements. This suppresses the quadratic complexity of common trajectory comparisons. This is because of the linear complexity of the filters, related to the fixed length of the filter, which linearly depends on the length of the trajectory for our selected local features. Furthermore, this does not include the additional overhead, such as storing features in the filter.

B. Compression using Bloom Filter

The normalized compression distance (NCD) approximates the Kolmogorov complexity $K(x)$ by using the length of the output of a real-world compressor $C(x)$, such as gzip. Furthermore, the relative Kolmogorov complexity $K(x, y)$ is the size of a joint compression $C(x, y) = C(x \cup y)$, where the , $\cdot \cup \cdot$ can be chosen as the concatenation of the inputs. Furthermore, the joint complexity $C(x, y)$ is computationally expensive as it accesses the original uncompressed values x and y. In a previous project, the authors showed that in case of satellite images the computational burden is limiting the applicability [15].

In this paper, we therefore replace the compressor of NCD with an alternative complexity measure. This measure is related to how far the Bloom filter is already filtered. Taking advantage of the fact that the number of zeros in the filter shrinks with the number of different elements inserted into a Bloom filter. To exploit this observation, we compute the Shannon entropy $H(x) = -x log_2(x) - (1 - x) log_2(1 - x)$ of the bit field underlying the Bloom filter of enumerated features as the information-theoretic complexity of a trajectory. To summarize, we use $B(x) := H(\mathit{foz}(\mathit{BF}(x)))$, where H stands for the entropy and *foz* represents the fraction of zero of the Bloom filter *BF*.

Similarly, we propose a notion of joint complexity approximating $K(x, y)$ by using the union of the Bloom filters which is directly computable as the binary OR of the bit fields. Concretely, using $\mathit{BF}(x \cup y) = \mathit{BF}(x) \vee \mathit{BF}(y)$ we define $B(x, y) = H(\mathit{foz}[\mathit{BF}(x) \vee \mathit{BF}(y)])$. This definition is plausible since it resembles the complexity for identical trajectories $B(x, x) = B(x)$ and increases with the difference of x and y.

Furthermore, this approach reduces the computational complexity and memory footprint as it is sufficient to keep only the Bloom filters of individual trajectories during computation instead of needing to actually compress pairs of trajectories jointly to estimate $C(x, y)$. Additionally, the approximation of K using real- world compressors, such as gzip or bz2, can be avoided.

C. NCD Using Bloom Filter

The previous discussion hides a subtlety of the construction of the NCD using Bloom filters. The presented information measure $B(x) = H(\mathit{foz}(\mathit{BF}(x)))$ is monotonously increasing only as long as the fraction of zeros remains smaller than 0.5. Higher fraction of zeros lead to smaller numbers and, thus, the distance could become negative.

There are three possible solutions to this situation: the first solution would be to increase the number of bits. However, this involves a recomputation of all trajectory sketches and might not be practical or efficient as this binds the configuration to the most complex example instead of to the average situation leading to a waste of main memory. A second solution would be to omit the entropy scaling and directly use the fraction of zeros $B(x) = \mathit{foz}(\mathit{BF}(x))$. This approach is not covered within this work because of using information-theoretic techniques on a compressor with an intentionally wrong unit.

What can be observe is that negative values can only occur if the union of the features extracted from the two trajectories is significantly larger as opposed to the individual complexities. Therefore, the third solution is to add an absolute value to the numerator and give the following definition of the Bloom Compression Distance:

$$\mathit{NBD}(x, y) = \frac{|B(xy) - \min\{B(x), B(y)\}|}{\max\{B(x), B(y)\}} \tag{4}$$

By improving the NBD with the absolute value of the numerator, the range becomes $0 \leqslant \mathit{NBD}(x, y) \leqslant 1 + \varepsilon$, where a value of zero represents that the compared objects are similar and a value of one indicates a maximal dissimilarity. The small number ϵ takes care of the fact that the entropy of the fraction of zeros suffers from sampling problems for small filters (only a few fractions of zeros can be attained by a finite bit array) and distribution problems when using real-world hash functions with non-uniform hash collision patterns. Experiments affirm that this approach of repairing the Normalized Bloom Distance is able to increase performance as compared to using a more direct measure extracted from the fraction of zeros directly. Furthermore, this not only corrected a deficiency, but also has improved the results of the experiments.

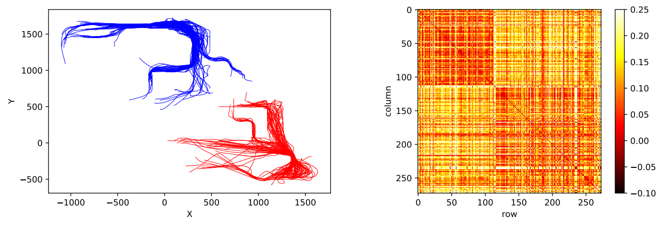
IV. Experiment

The calculation of a distance-based similarity metric is difficult for trajectories. Many aspects, as well as characteristics of single segments have a big influence on the classification result using a specific distance metric. The introduced approach enables to approximate the incomputable Kolmogorov complexity using trajectory feature extraction and representation using Bloom filters. The procedure by which the experiments were performed can be divided into several parts. When reading in the dataset, the coordinates must be assembled into trajectories. It is necessary to taken care that not two or more are put together, because this can cause large jumps between the points. Another step of the processing chain of the experiment is to clean each trajectory, such as the removing of sequences where the number of points is below a minimum. The third step is to extract the selected features from the trajectory, like the global directions in the form of symbols. The last step before classification is to store the extracted features into the trajectory specific Bloom filters. From those filters a similarity matrix is calculated by using the introduced metric. Additionally, a classification using the K-Nearest Neighbors algorithm is done. To find the optimal parameters for the number of global directions, the length and the number of hash functions for the Bloom filter and the parameters for a K-NN classifier, the brute force method is used.

In order to evaluate the described approach and its descriptive power, we present results on a range of both binary and a multiclass trajectory classification tasks.

A. Results

Prague-Teams. The first experiment with the introduced similarity metric NBD is conducted on the trajectory dataset *Prague-Teams* [11]. This dataset has been sampled from a computer game, called Urban Terror, where five players are playing a map called Prague. From the paths of each player spawning in the level the first 128 location points have been extracted, which represents 6.4 seconds of playing time. Furthermore, each trajectory within the game is assigned to a team. The dataset contains a number of 273 individual trajectories, where 40% is assigned to one team and 60% to the other team. This dataset is represented in Figure 3(a).



(a) Representation of the Prague-Teams dataset, where the coloring is the teams membership.

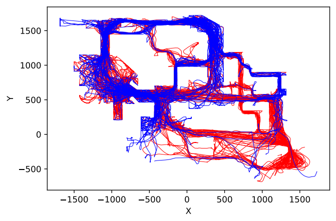
(b) Visualisation of the similarity matrix calculated from the Prague-Teams dataset.

Fig. 3. The Prague-Teams dataset.

As we don't want to exploit the obvious spatial separation of both teams, we apply global orientation with 16 directions. Furthermore, we build filters of 256 bits using two hash functions leading to an average fraction of zeros of 47.51%. The resulting set of Bloom filters occupies about 8kB of main memory (this amounts to a representation of the whole trajectory dataset with only two bits per trajectory point).

Evaluating k-NN, where the number of neighbors has been set to 19, leads to an overall success rate of 91.59% for assigning a trajectory the right team. This is near to the results reported for a matrix factorization approach employing n-gram structures in [11]. It is interesting to see that the results reported there for a n-gram-free version, yet with only eight directions, is reported as 81.03%. In addition, within the similarity matrix depicted in Figure 3(b), one can clearly see two darker regions representing the two teams and a noise pattern related to the randomized nature of the whole approach. In summary, the classification with the k-NN shows that the noise is not harmful.

Prague Ego-Shooter. The second experiment has been created using the same game than before. In this situation, the trajectories are self-redundant due to the nature of the game including lots of loops and the joint objective for both teams to "capture and hold" a certain flag location on the map. The dataset contains several 275 individual trajectories, with $244,675$ samples of locations. Figure 4(a), represents the *Prague Ego-Shooter* dataset, where the coloring represents the team membership of each individual trajectory.


(a) Representation of the Prague Ego-Shooter dataset, where the coloring is the teams membership.

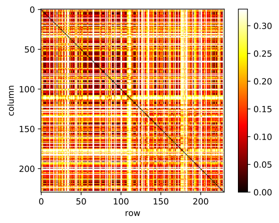
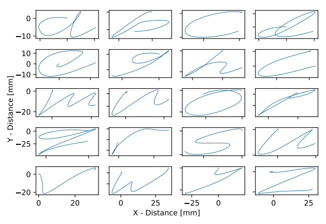

(b) Visualisation of the similarity matrix calculated from the Prague Ego-Shooter dataset.

Fig. 4. The Prague Ego-Shooter dataset.

As in the previous dataset the number of global orientations has been set to 16. Furthermore, we have built Bloom filters of 1024 bits using three hash functions leading to an average fraction of zeros of 73.89%. The evaluation using a k-NN, where the number of neighbors has been set to 13, leads to an overall success rate of 90.52% for assigning a trajectory the right team. In addition, the calculated distance matrix is visualized in Figure 4(b). The Figure shows the same pattern of the two teams, but now with a very small difference and not very convincing at first sight. This is not surprising as beyond the structure near the spawn points, the game mode of capture and hold leads to similar trajectories in both teams. This makes it a very hard classification problem as only the fact that spawn points remain slightly safer zones for players and that teams might regroup near the spawn points induces structures. In summary, the result shows that the approach of the introduced similarity metric NBD is able to predict accurately even complex two class classification problem.

Characters. The third experiment using the introduced similarity metric NBD, is performed on the trajectory dataset *Character Trajectories*. This dataset has been provided by the UCI Machine Learning Repository and contains 2858 Trajectories characters, written by hand with a pen. Furthermore, 20 classes are included, with characters which are possible to write with a single pen stroke. The dataset has three dimensions, where the first two represent the coordinates and the last is the pen tip force [16]. Because of the fact that the dataset has been numerically differentiated and Gaussian smoothed, the preprocessing includes additional steps in order to get the shape of the letters. The distribution of trajectories in the dataset is unbalanced, where each class contains between 125 and 175 trajectories. Figure 5(a) illustrates the *Character Trajectories* dataset.

We apply the presented framework by first integrating the given trajectory data into shapes as depicted in Figure 5(a), encoding each trajectory using only global orientation with eight directions, and putting the resulting features into a Bloom filter of 256 bits and two hash functions. This result to an


(a) Representation of the Characters dataset, where the coloring is the teams membership.

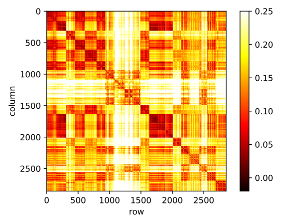

(b) Visualisation of the similarity matrix calculated from the Characters dataset.

Fig. 5. The Characters dataset.

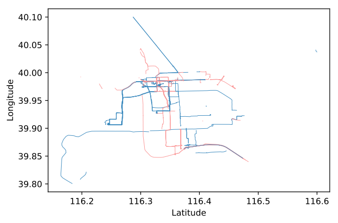
average FOZ of 74.40%. In this context, our approach reaches an overall success rate of 81.52% using NBD and k-NN classification, where the number of neighbors has been set to 15. With the same parameters and features (e.g., number of directions), a matrix factorization approach reached an accuracy of only 74.2% [11].

From the Bloom filter representation, we compute the similarity matrix depicted in Figure 5(b) using the similarity metric NBD with axes ordered by classes. One can clearly see the darker regions representing the classes of letters as well as a noise pattern related to the randomized nature of the whole approach. In addition, the patterns are in a few regions not very convincing at first sight. This is not surprising as beyond the structure of certain letters, such as the difference between the letters *U* and *V* is handwritten smaller than in case of other letters and leads to similar trajectories in different classes.

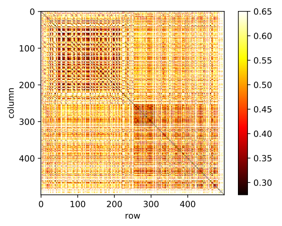
In summary, the results show that the approach of NBD provides accurate results not just in two class classification problems, but that the approach is also able to classify a complex multiclass classification problem accurately.

Geolife. The last experiment with the use of the similarity metric NBD, is with the trajectory dataset *Geolife*, which has been provided from [3]. The dataset contain trajectories collected from 182 users using GPS in the timespan between April 2007 and August 2012. Furthermore, the full dataset includes $17,621$ labeled trajectories with more than 24.8 million points. The classes represent different types of mobility, such as *Bike* and *Taxi*. This experiment uses a subset of the dataset, which includes the classes *Bus* and *Taxi* in the area of Beijing. This subset contains a number of 500 individual trajectories, equally distributed over the two classes. Figure 6(a), represents the balanced *Geolife* dataset, where the coloring represents the class membership of each individual trajectory.

For the feature extraction, we use global orientation with 16 different directions as well as a geohash of 7 characters. The geohash is not interpolated, but pulled for each sample in the trajectory such that a weak notion of speed or stays is represented as well. Using 1024 bits for the Bloom filters, this results in filters with an average fraction of zeros of 93.11%. With a ten-fold cross validation and a k-NN classifier, where $k = 4$, we gain a success rate of 82.00%. From the Bloom filter representation of this dataset, we computed a distance



(a) Representation of the Geolife dataset, where the coloring shows the type of mobility.



(b) Visualisation of the similarity matrix calculated from the Geolife dataset.

Fig. 6. The Geolife dataset.

matrix depicted in Figure 6(b) using the similarity metric NBD with axes ordered by classes. One can clearly see two darker regions representing the two classes and a noise pattern related to the randomized nature of the whole approach.

V. Conclusion

In this paper, we introduced a similarity metric inspired by the Kolmogorov complexity as well as by the idea of approximating Kolmogorov complexity with compression. In order to apply this framework to trajectories, we first select feature sequences and encode features with multiplicities in a Bloom filter to obtain a compressed representation of trajectories.

By deriving an information measure from a Bloom filter instance in main memory, we are able to provide a novel distance function similar to normalized compression distance in which Kolmogorov complexity is approximated by the size of compressions of data. One central advantage of this approach is that we are able to compute the joint information complexity $C(x, y)$ without compressing x and y together thereby actually not needing the original data during comparison not even for newly incoming trajectories. Instead, we exploit properties of the Bloom filter to estimate the information of the joint compression from the compressed data directly. This leads to a very memory-efficient representation with only few bits per point or even less than one bit per trajectory point.

Given this compressible nature, it is surprising, how powerful even simple features like orientation and global location can be in classifying real-world trajectory data. In addition, many trajectory distances have quadratic complexity including DTW and Fréchet distance and are, therefore, difficult to apply in a big data setting. With compression distance, we can provide an alternative based on randomized algorithms in which we might loose a few guarantees due to the random nature of the underlying data structures yet come up with good results from heavily compressed representations. This opens up trajectory computing to application on mobile and resource-constrained devices or in extremely large collections otherwise impractical today.

For future work, we envision to apply the framework of randomized algorithms combined with algorithmic information theory to a wider range of spatial objects beyond trajectories. In addition, there are many more features possible to represent trajectories which should be discussed in more detail. There is a continuum of methods ranging from lossless representations to heavy information loss and we need to find a better understanding of which information is relevant to a given spatial computing task and which information can safely be removed.

References

[1] A. Karbalayghareh, U. Braga-Neto, and E. R. Dougherty, "Classification of single-cell gene expression trajectories from incomplete and noisy data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 1, pp. 193–207, 2019.

[2] T. Alsahfi, M. Almotairi, R. Elmasri, and B. Alshemaimri, "Road map generation and feature extraction from gps trajectories data," in *Proceedings of the 12th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, ser. IWCTS'19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3357000.3366140

[3] Y. Zheng, X. Xie, and W.-Y. Ma, "Geolife: A collaborative social networking service among user, location and trajectory," *IEEE Data(base) Engineering Bulletin*, June 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/geolife-a-collaborative-social-networking-service-among-user-location-and-trajectory/

[4] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, Oct. 2007.

[5] T. Devogele, L. Etienne, M. Esnault, and F. Lardy, "Optimized discrete fréchet distance between trajectories," in *Proceedings of the 6th ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data*, ser. BigSpatial'17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 11–19. [Online]. Available: https://doi.org/10.1145/3150919.3150924

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

[7] K. Buchin, Y. Diez, T. van Diggelen, and W. Meulemans, "Efficient trajectory queries under the fréchet distance (gis cup)," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2017, pp. 1–4.

[8] J. Baldus and K. Bringmann, "A fast implementation of near neighbors queries for fréchet distance (gis cup)," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2017, pp. 1–4.

[9] F. Dütsch and J. Vahrenhold, "A filter-and-refinement-algorithm for range queries based on the fréchet distance (gis cup)," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2017, pp. 1–4.

[10] P. M. B. Vitányi, F. J. Balbach, R. L. Cilibrasi, and M. Li, "Normalized information distance," in *Information Theory and Statistical Learning*, F. Emmert-Streib and M. Dehmer, Eds. Boston, MA: Springer US, 2009, pp. 45–82.

[11] M. Werner and M. Kiermeier, "A Low-Dimensional Feature Vector Representation for Alignment-Free Spatial Trajectory Analysis," in *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems (MobiGIS'16)*, 2016.

[12] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970. [Online]. Available: https://doi.org/10.1145/362686.362692

[13] T. M. and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006.

[14] R. L. Cilibrasi and P. M. B. Vitányi, "Clustering by compression," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1523–1545, Apr. 2005.

[15] C. O. Dumitru, G. Dax, G. Schwarz, C. Cazacu, M. C. Adamescu, and M. Datcu, "Accurate monitoring of the danube delta dynamics using copernicus data," in *SPIE Remote Sensing*, 2019, pp. 1–13. [Online]. Available: https://elib.dlr.de/129121/

[16] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml