# A Low-Dimensional Feature Vector Representation for Alignment-Free Spatial Trajectory Analysis

Martin Werner
Mobile and Distributed Systems Group
Ludwig-Maximilians University München
martin.werner@ifi.lmu.de

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

## ABSTRACT

Trajectory analysis is a central problem in the era of big data due to numerous interconnected mobile devices generating unprecedented amounts of spatio-temporal trajectories. Unfortunately, datasets of spatial trajectories are quite difficult to analyse because of the computational complexity of the various existing distance measures. A significant amount of work in comparing two trajectories stems from calculating temporal alignments of the involved spatial points. With this paper, we propose an alignment-free method of representing spatial trajectories using low-dimensional feature vectors by summarizing the combinatorics of shape-derived string sequences. Therefore, we propose to translate trajectories into strings describing the evolving shape of each trajectory, and then provide a sparse matrix representation of these strings using frequencies of adjacencies of characters (n-grams). The final feature vectors are constructed by approximating this matrix with low-dimensional column space using singular value decomposition. New trajectories can be projected into this geometry for comparison. We show that this construction leads to low-dimensional feature vectors with surprising expressive power. We illustrate the usefulness of this approach in various datasets.

## CCS Concepts

•**Information systems** → **Location based services; Geographic information systems;** *Data analytics; Nearest-neighbor search;*

## Keywords

Trajectory, Moving Objects, Multi-modal Trajectory, Big Data

## 1. INTRODUCTION

Every day, an enormous mass of positioning data is generated by smart phones, cars, and other mobile devices. However, without a way of structuring them, for example to say whether some of these travelled paths are similar or not, this huge data source is useless. Accordingly, there is a wide field of distance measures which are optimized for comparison of such spatial trajectories. Usually, trajectories are described by a set of waypoints each given by a spatial point and a time stamp, and each distance measure has its own way to process these ordered sets of waypoints. From a theoretical perspective, these many different distance measures emerge as spatial trajectories can be seen as the elements of the space of continuous mappings of the unit interval $[0, 1]$ into space. This space is infinite-dimensional and very complicated, hence, various different sensible definitions of distance exist. From a more practical perspective, the number of different distance measures can also be motivated by the general complexity of these objects: some of the measures are simple summaries over points, other measures use higher order geometry concepts such as polygonal lines, some ignore time, some treat time only with respect to the ordering of points induced by time, and some make direct use of time. From a very general perspective, we can also view many of the various distance measures as different forms of the same process: first, find correspondences between points and then summarize distances of corresponding points.

The motivation for this paper stems from an observation of human communication: *when humans describe a path or relate two paths with each other, they often do this in two isolated domains: in the spatial domain, we give a rough information about the trajectory. More detailed information is given by describing shapes.* As an example, consider the following description: "From here, go straight on for about 100 meters, then turn left and follow the street again for about 100 meters...". In this case the path is not described by a set of way-points or only by first order spatial features of points and distances. Instead a starting point is given followed by successive shape descriptions of the path (e.g., second order spatial information).

*This observation raised the question, how much information we can extract from shapes alone.* As a first step, we therefore transform a trajectory into a sequence (string) of discrete shape descriptors (characters). Note that these are basically two discretizations: we map the relations of specific pairs of points (e.g., neighboring points, points in relation to the first point, ...) into a character from a finite alphabet. Furthermore, we forget about the actual timestamps and only keep the ordering of points for building shape sequences. For the shape description by characters, geometrical characteristics like the length ("go on for 100 meters") or the direction ("go North" or "turn left by 90°") of sub-paths



*MobiGIS '16, October 31-November 03 2016, Burlingame, CA, USA*
 ISBN 978-1-4503-4582-8/16/10...$15.00
DOI: http://dx.doi.org/10.1145/3004725.3004733

can be used.

One of the most important advantages of this double discretization is the fact that we transforme the problem from the continuous spatio-temporal domain into a fully discrete, combinatorial domain. Similar to several approaches in bioinformatics and information retrieval, we can now try to compress the combinatorics of these sequences into a form which allows for alignment-free comparison [14, 3]. Therefore, we extract combinatorial knowledge about the relations of trajectories from sets of training trajectories and derive a distance measure for trajectories that captures this learned combinatorial knowledge.

Technically, we create low-dimensional feature vectors from trajectories and then use a matrix approximation technique based on singular value decomposition in order to separate the interesting effects in a given set of trajectories from the sampling error effects and show that this approximation leads to a translation invariant, noisy distance measure, which is able to capture surprisingly much about trajectory sets given that no spatial points or exact distances are used in the distance measure. Furthermore, this distance is very easy to compute as a matrix multiplication followed by a distance of vectors. Of course, these shape distances can be combined with spatial distances, for example the distance between start points, end points, or median points. And what is best: there is no temporal alignment to be computed neither implicitly nor explicitly reducing the complexity for larger sets of trajectories by a lot.

The remainder of the paper is organized as follows: Section 2 sums up existing distance measures for spatial trajectories. In Section 3, we presents the transformation of spatial trajectories into shape sequences and introduce the corresponding distance measure derived from latent semantic structure. Then, Section 4 presents classification examples showing the usefulness and descriptive power of the proposed methodology. Finally, Section 5 finalizes the paper and gives hints on future work.

## 2. RELATED WORK

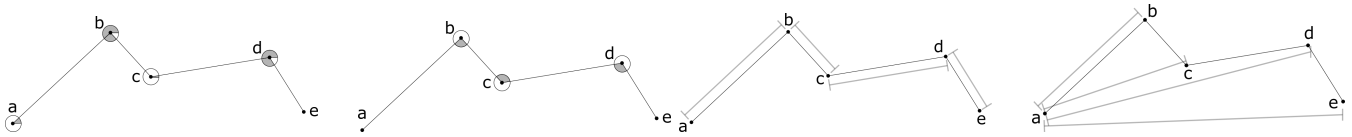
Trajectory Computing is a large field in spatial computing. The general question of trajectory computing is, how trajectory data should be stored, analyzed and retrieved in big data systems. Spatial trajectories are usually stored as sequences of spatial points with associated timestamps, e.g., $t = (p_1, t_1, p_2, t_2, \ldots, p_n, t_n)$. This set of waypoints is then usually assumed to be dense enough to allow for linear interpolation, that is, the trajectory is not given by the set of points, though it is stored in this way, it is rather given as the linear interpolation of all of these points. This, together with the fact that the points $p_i$ are spatial, also gives a good distinction from time series. It is possible to use other interpolation models such splines for trajectory interpolation, however, several algorithms exploit the geometric simplicity of linear segments.

Many distance measures for spatial trajectories have been proposed and successfully used. An overview is given in [4] and [12]. Among the most simple measures, we have the closest pair distance, which is simply the minimal distance between any pair of points from the two involved trajectories. This distance measure can be computed in quadratic time. Similarly simple, trajectories with the same number of samples sampled at the same points in time can be compared by the Sum of Pairs distance, e.g., the sum of the Euclidean distances of pairs of points. This distance measure only needs linear time. In most cases, however, trajectories are not in a suitable form for this distance measure and need to be transformed into this form by interpolation increasing the practical complexity into at least quadratic complexity. In a similar fashion, the Hausdorff distance of sets can be applied to trajectories. This distance is built from extending the distance between points to a distance between a point and a trajectory in the obvious way by taking the minimum[1]. This distance between a point and a set is then varied over the set to construct a distance between two sets. Another distance of trajectories is given by the Fréchet distance. It is very intuitive and can be efficiently calculated. It is defined to be the minimal length of a leash connecting the owner of a dog moving forward on the first trajectory with a dog following the second trajectory without ever going backwards. While this distance makes implicit use of the time domain by not allowing the dog and the owner to move backward, the Dynamic Time Warping (DTW) distance makes the time domain explicit: the distance is given as the sum over the distances of corresponding waypoints. The correspondence is calculated by associating the first points of both trajectories and then proceeding either one step in one of those trajectories or one step in both trajectories. From all of these possible choices, the one is chosen which minimizes the sum of the distances. This distance has proven to be very useful in a lot of applications and allows for a lower bound (LB_Keogh and variants) for speeding up similarity search in large databases [9].

A completely different approach is taken by three versions of trajectory edit distances. The first version, Edit Distance on Real Sequences (EDR), consists of transforming the idea of Edit Distance directly to trajectories. This distance measure counts the number of point insertions, deletions, or modifications needed to transform the one trajectory into the other. A modification (or insert, or deletion) in this context is needed if and only if the two matching points are farther away from each other than a predefined threshold $\epsilon$. This distance measure has proven useful, but does not fulfill all properties of a metric. However, Edit Distance with Real Penalties (ERP) repairs this defect by assigning a well-chosen penalty instead of an integer cost for edit operations. The third edit distance is known as Longest Common Subsequence (LCSS) and allows for skipping over some of the involved points while calculating the number of pairwise sufficiently near points.

A third class of approaches is given by transforming trajectories into a combinatorial object such as a set. For example, all points can be assigned to a spatial region (e.g., a Geohash cell) and trajectories can be represented as the set of cells which are hit by trajectory points. The Jaccard set similarity can be used as a similarity measure for trajectories and can well be approximated by Locality Sensitive Hashing (LSH) [7]. Furthermore, it is possible to compress these sets into tiny objects called Bloom Aggregated Cell Representation (BACR) whose Jaccard distance can be approximated and for which a lower bound is known [13].

[1]To be honest, one has to use the infimum as trajectories are infinite sets. In practice, however, trajectory measures of this style are approximated by taking minima and maxima over the waypoints only. These variants are usually called *discrete*.




(a) Global Direction (b) Local Direction (c) Length (d) Length Relative to Fixed Starting Point

**Figure 1: Shape features**

# 3. SHAPE SEQUENCES AND LATENT SEMANTICS

With this paper, we propose a method of comparing trajectories that lies between the last two categories: to some extent, the order relation induced by time is used, but in a mainly combinatorial setting.

Therefore, we first translate trajectories into sequences of letters where each letter represents a local shape feature of the trajectory. Then, the $m$-gram sequence of this sequence, that is, the sequence of overlapping substrings of length $m$ is summarized by counting the number of such $m$-grams. For a string encoding into an alphabet $\Sigma$, this creates a vector with $|\Sigma|^m$ entries. For a set of $N$ trajectories, these vectors form a matrix with $N$ columns and $|\Sigma|^m$ rows. This matrix is quite sparse as for each trajectory only few of the columns will be non-zero. We use a matrix generated in this way to identify useful shape features for comparison.

Basically, the approach is divided into two phases: a training phase, in which the approach learns how to represent specific aspects inside a trajectory dataset. At the end of this phase, each training trajectory has been transformed into a simpler form of a finite-dimensional feature vector. Secondly, there is an online phase in which unknown trajectories can be projected into the very same feature representation as derived in the first phase and where the Euclidean distance can be used to compare the resulting feature vector with other trajectories in this representation. The following sections explain the construction more carefully and give details on the steps involved in providing an alignment-free, low-dimensional feature vector representation of spatial trajectories.

## 3.1 String Encoding of Spatial Trajectories into Shape Feature Sequences

The idea of shape feature sequences is to encode the trajectories which are described as sequences of data points by local geometrical characteristics. For this initial study, we limit the discussion to two simple categories of shape features: those derived from orientation and those derived from length.

For features derived from orientation, a straightforward encoding of a trajectory into a string is given by discretizing the set of angles involved into finite regions as, for example, given in Table 1. We chose to use sectors of equal size, eight of these are used in the table. With this mapping from angles to characters, there are two obvious ways of extracting sequences of angles from trajectories: either globally (e.g., North, West, South, East) or locally (e.g., turning left, turning right, etc.). The two features are depicted in Figure 1(a) and Figure 1(b), respectively.

For features derived from length, we propose two different

| angle interval | mapping |
|---|---|
| $0° \leq \theta \leq 45°$ | A |
| $45° < \theta \leq 90°$ | B |
| $90° < \theta \leq 135°$ | C |
| $135° < \theta \leq 180°$ | D |
| $180° < \theta \leq 225°$ | E |
| $225° < \theta \leq 270°$ | F |
| $270° < \theta \leq 315°$ | G |
| $315° < \theta \leq 360°$ | H |

| interval (rel. $l_{\max}$) | mapping |
|---|---|
| $0\% \leq l \leq 12.5\%$ | A |
| $12.5\% < l \leq 25\%$ | B |
| $25\% < l \leq 37.5\%$ | C |
| $37.5\% < l \leq 50\%$ | D |
| $50\% < l \leq 62.5\%$ | E |
| $62.5\% < l \leq 75\%$ | F |
| $75\% < l \leq 87.5\%$ | G |
| $87.5\% < l \leq 100\%$ | H |

**Table 1: Orientation-, and length-based mapping relative to a predefined maximal length $l_{\max}$**

encodings: Either, we encode the length of each segment of a trajectory as illustrated in Figure 1(c) or we use the distance from the starting point of the trajectory to the end point of the current segment as depicted in Figure 1(d). The intuition for the last perspective on length is given by the fact that this measure captures a mixture of distance and orientation: If a character in the sequence is coming twice, we have been moving roughly along circle around the starting point. If the character "increases", we have been moving away and if the character "decreases" we are going towards the center. In this sense, this measure captures a mixture of orientation and distance.

There are arbitrary many additional ways of transforming a spatial trajectory into a string representing some partial, local geometry of the trajectory. For example, one can use first or second order derivative features (e.g., speed, turn rate, and similar). These can be chosen based on intuition depending on the application or on evaluation results of application-dependent measures of success. For the purpose of this paper, the simple features based on length and orientation, however, are sufficient.

In practice, we need to preprocess the given spatial trajectories in order to remove uninformative samples. Therefore, the Douglas Peucker algorithm is being used. Then, the trajectory is transformed into a sequence of characters modeling certain aspects of the temporal evolution of the trajectory's shape as described.

## 3.2 Feature Vector Representation of String Encodings

For this dataset of strings, we extract all overlapping sequences of $m$-grams and encode those in a sparse matrix by counting the number of occurences of a specific $m$-gram in each trajectory. Thereby, a trajectory is represented by a vector with $|\Sigma|^m$ entries, where $\Sigma$ denotes the alphabet used for string encoding.

This creates an $|\Sigma|^m$-dimensional vector for each trajectory. Note that this transformation reduced the situation from a high-dimensional problem into a problem of constant and comparably small dimension $|\Sigma|^m$. The global order-

ing information of the trajectory has been removed while local ordering information (e.g., $m$-gram frequencies) has been retained. Though this is a great step towards low-dimensionality, these vectors still contain too much "noise". Note, however, that if we can derive meaningful feature vectors from these intermediate vectors, a global alignment of trajectories is neither implicitly nor explicitly calculated resulting in better scalability as compared to many trajectory distance measures.

The situation that frequencies are important, but not all frequencies and patterns directly contribute to a result, is also observed in the field of information retrieval and bioinformatics. For information retrieval, the ordering of terms in a document provides information, but not each and every neighborship of words means something; for bioinformatics, the ordering of amino acids provides information, but not each and every pair of neighbors.

One way to amplify the effect of important frequencies and reduce the influence of the other frequencies is given by latent semantic indexing in its various forms. In general, the frequency vectors of a training set of trajectories are put into the columns of a sparse matrix $M$. For this matrix, a singular value decomposition provides a way to extract meaningful features. Essentially, a construction will be used which implicitly extracts associations of information from a rank-reduced version of the original matrix. Using the singular value decomposition, the matrix is reconstructed by a matrix $\hat{M}$ of lower rank that is as near as possible to the original matrix $M$, but with a column space of reduced dimensionality [1]. This reduction in dimensionality allows for concentrating a given model on finding the relevant aspects and associations between $m$-grams. The central advantage of this approach is that there is no alignment anymore. The sequences are transformed into vectors or matrices and similarity is further expressed in simple terms as a distance between vectors. For a new trajectory, there are only two things that need to be done: Extract the frequency vector and project it into a form suitable for the matrix approximation $\hat{M}$. Then, it can be compared to *all* other trajectories processed so far by a simple distance measure of vectors such as the Euclidean norm. As all dimensions are constant and chosen beforehand by selecting $m$ and $\Sigma$, this transformation is possible in linear time with respect to the length of the trajectory.

## 3.3 Training Phase

The vectors generated from $m$-gram frequencies of shape sequences are combined into a matrix $M$ with $|\Sigma|^m$ rows and $N$ columns. We could now try to use these frequency vectors for comparing trajectories and calculate the Euclidean distance between column vectors of this matrix $M$. However, this matrix does not only cover the interesting aspects for differentiating between the columns of this matrix and their associated trajectories, but also a lot of noise. The solution is here to control the amount of information that the linear map given by this matrix is allowed to use. Therefore, we perform a singular value decomposition of $M$:

$$M = USV^t$$

with $U$ and $V$ unitary matrices and $S$ a diagonal matrix of the singular values of $M$ in decreasing order [3]. From this information, we construct an approximation to $M$ with $k$-dimensional column space by setting $\hat{S} = (\sigma_1, \ldots, \sigma_k, 0, \ldots, 0)$ for some fixed small value $k$ and calculating

$$\hat{M} = U\hat{S}V^t$$

Note that this matrix has the same dimensionality as $M$, but only a $k$-dimensional column space and rank at most $k$. Along with this matrix, we can define a projection $P$ mapping a column of occurrences of $m$-grams into the reduced form suitable for column-wise comparison with the matrix $\hat{M}$:

$$P = U\hat{S}S^{-1}U^t$$

This now provides a model in which similarity between trajectories and training trajectories will be measured by the Euclidean distance of vectors of $m$-gram frequencies projected by $P$.

## 3.4 Choice of Parameters

The proposed methodology as described in the previous section needs three meta parameters: the $\epsilon$ error threshold for Douglas Peucker, the number $k$ of dimensions for the rank of the matrix approximation $\hat{M}$, and the number $m$ of adjacent characters in the $m$-gram counting.

The $\epsilon$ error threshold for Douglas Peucker simplification controlling how much we simplify a trajectory controlling the number of letters in the strings and the column space dimension $k$ of the matrix approximation $\hat{M}$.

The first value $\epsilon$ can be set from domain knowledge in many cases covering the expected measurement noise. However, for trajectory classification, we can set this value from cross-validation choosing the one value of $\epsilon$ for which the success rate (or another performance measure for classification) is best.

For the second parameter $k$, one should note that a small value of $k$ leads to less expressive models, while high values of $k$ lead to a model with sufficient capacity to also model a sampling error of the dataset and suffer from overfitting training examples.

While we can also try to set this value from cross-validation, Everitt and Dunne proposed the following heuristic, which greatly reduces the complexity of setting this parameter $k$ [2, 6]. In their formulation, we shall use a dimensionality $k$ such that their relative variance $v_i$ according to the following formula is smaller than $0.7/n$, where $n$ is the number of training examples [11].

$$v_i = \frac{\sigma_i^2}{\sum_i \sigma_i^2}$$

One should remark, that this can be set without an application-specific quality measure such as success rate in classification. For our example applications, however, much lower dimensionalities were sufficient. Consequently, we propose to set this parameter also from an application-dependent metric. For classification, for example, we used a simple success rate estimation.

## 3.5 Basic properties

The construction described so far leads to a system which is capable of transforming a spatial trajectory into a feature vector capturing a specific learned aspect of the trajectory shape. These feature vectors can then be compared using Euclidean distance, as we will show in the evaluation for two application examples.

Still, this construction has several useful theoretical properties, which we collect in this section. Firstly, the construction leads to a distance measure that is nearly a metric. It provides a non-negative, symmetric function. Only the identity of indiscernibles is broken, as different trajectories can lead to identical feature vectors.

From a geometrical perspective, this construction inherits geometric invariants of the shape sequence encoding. For all feature sequences described here, it will be translation invariant. So a trajectory will have zero distance to all of its translations. For local orientation, it will also be rotation invariant.

From a data quality perspective, we also see good features: The construction is quite stable with respect to outliers, as they only affect a limited amount of adjacencies including the outlier. The rest of the frequencies can dominate as long as there are enough inlier points. Partially, the construction is also stable with respect to noise, especially when the training matrix is extended by noise. For example, we can boost a given training set by copies of the training set perturbed by noise in order to combat noise in incoming trajectories.

From a scalability perspective, the system is by an order of magnitude faster than many other distance measures, as it removes the intermediate step of alignment and replaces it by matrix multiplication. Therefore, after training a matrix $M$, the comparison complexity is linear in the length of the trajectory. The most complicated operation for training is given by the singular value decomposition. However, randomized linear algebra (RandMLA) provides very fast approximate singular value decompositions, which only generate the largest $k$ singular values [5]. We used Randomized SVD using a specific form of column sampling for larger experiments [8].

# 4. EVALUATION

Distance measures for trajectories are difficult to compare with each other as every distance captures a different aspect of the overall relation. In this situation, distance-based applications such as classification are often used to show the usefulness of an approach. Therefore, we present two-class and many-class classification results to illustrate the descriptive power of the presented approach. In this section, we choose $m = 3$ (i.e., 3-grams) for all evaluations.

## 4.1 Player Team Detection

The first test for our model is based on a trajectory dataset sampled from five players playing the ego shooter Urban Terror in a map called "Prague". From these traces, we extracted the beginnings of trajectories for both teams by using the first 128 samples representing 6.4s of playing time.

Our goal for this sample problem is to identify the team from translation invariant shape sequences alone. Figure 2(a) depicts the dataset of trajectory beginnings. The dataset contains 276 different trajectories, 40% of the one team, 60% of the other team. We start by choosing a random subset of this trajectory dataset containing 150 trajectories from both teams, apply Douglas Peucker algorithm with varying values of $\epsilon$ and approximate the resulting frequency matrix using different column space dimensionalities $k$. In the sequel, Douglas Peucker algorithm is being applied using a threshold of 0.35 and the feature in use is *global direction* unless otherwise stated.

From the 150 training matrices, 109 of the generated sequences are long enough for analysis. Sequences shorter than 5 characters have been rejected. Figure 2 depicts distance matrices of the resulting feature vectors for the 109 training trajectories for different dimensionalities ordered by team.

One can clearly recognize the two teams represented by the four blocks of this distance matrix. Additionally, one sees that a column space dimensionality of $k = 1$ is not sufficient to extract enough information, that the teams in the middle matrix can be distinguished from each other quite good ignoring the noise and that for $k = 6$ the picture starts getting worse, again. This is in line with the expectation that a specific dimensionality is able to model enough of the situation but rejects noise.

The matrix $\hat{M}$ with 109 columns and 4096 rows has 50,826 entries, in other words 11.38% of the matrix are in use. In order to combat the noise in the feature vectors, we assign the class to an incoming trajectory from the remaining database to be the class for which the mean of the distances for the respective columns is smallest.

This results in a classification system capable of correctly classifying 107 of the remaining test trajectories. Given 9 wrong classifications, the overall success rate is 92.24%. Figure 2 depicts the result of this classification: The gray lines in the background show the training dataset, the blue and green lines depict correctly classified trajectories from both clusters and the red lines depict wrongly classified trajectories. It is surprising that even with a one-dimensional approximation, the success rate is 81.03% and that all dimensions between four and ten generate a success rate of 90.52%.

When replacing the feature *global direction* by *local direction*, we see that the problem gets harder as the feature is now also rotation invariant. Consequently, we report lower success rates of 56.03% for one dimension and still a respectable 76.72% for dimensions between two and ten. Beautifully, as depicted in Figure 2(b), the new additional errors are stemming from the fact that without a sense of global orientation, it is impossible to distinguish the red group of wrongly classified trajectories in the upper part of the figure from the longer vertical trajectories of the lower cluster.

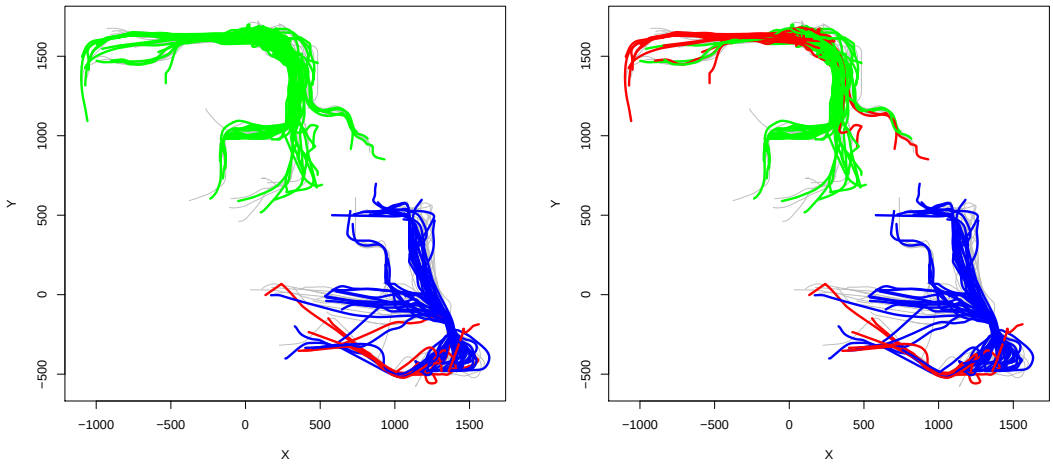
The feature *length*, interestingly, does not learn something useful. In fact, it converges into the trivial classifier assigning the upper class in any case for a wide range of parameters. In contrast, the feature *length relative to fixed starting point* is able to classify 72.4% correctly using four dimensions for the best choice of parameters.

In summary, all features except simple length were able to extract interesting patterns, the best results were able to be used from *global direction* with more than 90% success rate. However, *local direction* performed also well taking into account the rotation invariance of the feature and the self similarity of the misclassified examples up to rotation.

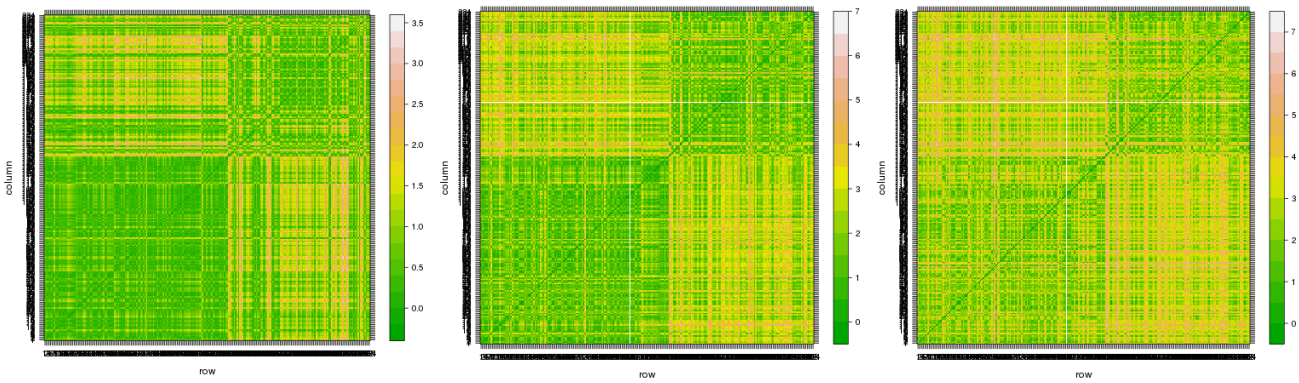
## 4.2 Character Classification

While the results of the previous section are convincing in that the proposed methodology is able to capture specific relevant shape features from shape sequences, it is not clear, whether this observation generalizes to problems with more classes. Therefore, we analyze the character trajectory dataset in this section.

The Character dataset, available from the UCI Machine Learning Repository, contains 2858 trajectories of 20 differ-




(a) *global direction*, $k = 3$ (b) *local direction*, $k = 3$




(c) $k = 1$ (d) $k = 3$ (e) $k = 6$

**Figure 2: Distance Matrix of the Prague Training Dataset for Various Dimensionalities and Classification Results ($\epsilon = 0.35$)**

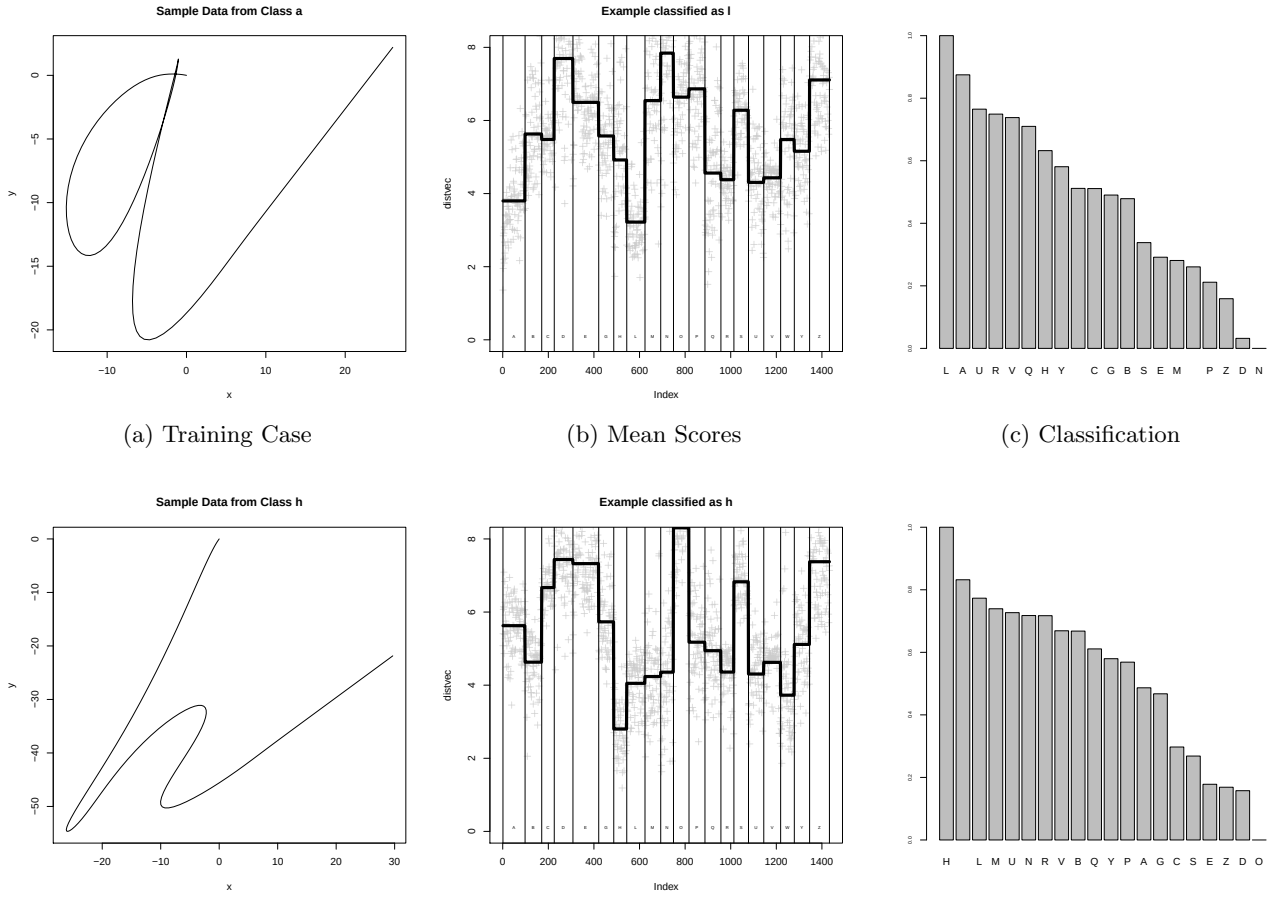
ent hand-written letters captured from a pen on a digital tablet [10]. The original dataset is given as the smoothed derivative of the recordings. For the evaluation in this paper, however, we integrate this dataset in order to capture the shapes of the letters. Due to this preprocessing, all letter shapes start at the origin. We use the first 1433 trajectories for building the training matrix and test classification accuracy on the remaining 1425 trajectories.

Figure 3 depicts two training cases, their classification scores as well as their classification results. While the first example is wrongly classified as letter 'l', this is perfectly reasonable taking into account that we are matching sequences of orientations after Douglas Peucker. You can see that the vertical line in the middle is slightly open, as would be for an 'l'. The second guess of the system was the correct letter 'a', and even the third guess 'u' makes sense given the shape, as the right part of the letter forms a 'u' and the left loop only needs to be made tiny in order to make this letter perfectly look like a handwritten 'u'. These results have been produced using a Douglas Peucker threshold of $\epsilon = 0.1$ , a dimensionality of $k = 7$ and the feature *global direction*. The second example shows that some quite ambiguous shapes can be distinguished. The letter is correctly classified as an 'h' and followed by 'w' according to the scores.

For the given configuration, Figure 4(a) depicts the distance matrix of the columns of the approximated training matrix $\hat{M}$. You can clearly see that some letters share shape structure such as 'a', 'u' and 'l'. However, the matrix covers quite well the distiction between the shape of 'o' and 'n' or 'b'. The letter 'o' is most similar (in this approximation) to the letter 'e', which is plausbile from nearly identical hand movement resulting in nearly identical shape sequences.

We performed a numerical evaluation on the testsets for varying parameters $\epsilon$ and $k$. For this evaluation, again, only correct predicitions are being taken into account. Hence, the case depicted in Figure 3 counts as a wrong classification. The results are depicted in Figure 4.

One can clearly see that for this problem with 20 classes, a small column space dimensionality degrades the ability of the system in finding good correlations for explaining shape




(a) Training Case (b) Mean Scores (c) Classification

(d) Training Case (e) Mean Scores (f) Classification

**Figure 3: Two character training cases, their mean scores and their classification result.**

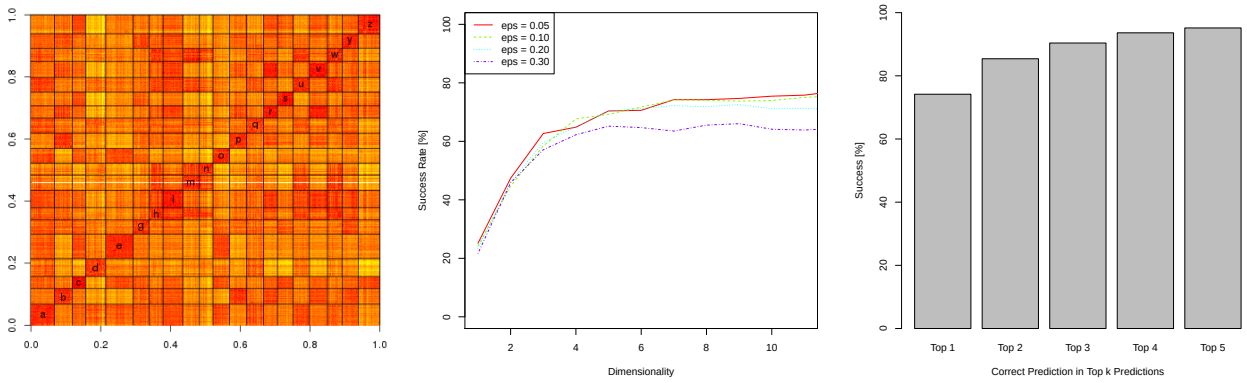
sequence differences. On the other hand, when dimensionality grows, the steepness reduces to a nearly flat shape. From $k = 6$ on, there is only marginal fluctuation in the success rate. This means, that the capacity of the approximation of $M$ by $\hat{M}$ is sufficient for $k = 6$ or $k = 7$ and that the remaining errors are due to other effects. Note that one should choose a smaller dimensionality in the range of good performance, as otherwise overfitting effects might destroy generalization capabilities. We have already seen this effect for the Prague dataset in Figure 2(e). It is worth noting that for small choices of $\epsilon$, higher success rates can be measured by using quite high dimensionality (e.g., more than 10 dimensions). This is interesting, but outside the scope of this paper with which we want to represent trajectories in low-dimensional geometry. Additionally, such a small value of $\epsilon$ would capture not the global shape of a letter but rather details in a millimeter scale. Therefore, we think that these high numbers are observed from overfitting to the specific writing style in the dataset.

A very good performance of 74.2% correct predictions for reasonable dimensionality ($k = 7$,$\epsilon = 0.1$) was reached. As already shown in the example, however, some of the wrong predictions occur due to similarity of shapes. Therefore, Figure 4(c) depicts the fraction of examples in which the correct class was among the top $k$ classes. For a considerable fraction of the 25.8% of wrong predictions, the correct prediction is among the top 3 predictions. Counting the fact that a letter is among the top 3 predictions of the system as a success, the success rate for the given configuration rises to 90.4%.

# 5. CONCLUSION

With this paper, we have proposed a highly scalable distance measure for spatial trajectories and illustrate its use in classification scenarios. The distance measure is based on transforming trajectories into strings and defined by analyzing their combinatorial structure. The system performed extremely well on a small problem with trajectories of similar shape from a computer game and was also able to extend to many classes (e.g., 20) in the character dataset.

The way of measuring distance is inspried by human path descriptions. This leads to highly scalable and efficient similarity measures for spatial trajectories. A central feature of this approach is the fact that no sequence alignment takes place. Consequently, the model can be applied over very large databases or even data streams without pairwise alignment. The most important drawback of this approach is the need of a training matrix containing examples of interesting trajectories. However, this can also be seen as a feature,




(a) Distance Matrix for the Character Training Matrix ($\epsilon = 0.1$,$k = 7$).

(b) Dimensionalty vs. Parameter

(c) Number of Times the Correct Prediction is in the Top $k$ Predictions

**Figure 4: Performance on character trajectories for varying paramters**

because it is impossible to describe "everything" that makes up a high-dimensional object such as a spatial trajectory in a low-dimensional feature space.

This observation opens up the approach for future research: for example, training data can be boosted by carefully adding noise to the samples. This is especially relevant, when trajectories are created from noisy measurements. We expect that this approach could reduce the amount of randomness in the training distance matrices a bit.

Another question is with respect to the proposed shape sequences: first of all, there are numerous other reasonable ways of transforming trajectories into strings, which could be able to capture other features in trajectory databases. For example, speed and relative measures to a fixed point (e.g., home) could be interesting in personalization and activity recognition contexts. Furthermore, several matrix approximations $\hat{M}$ for different shape sequences can be combined by voting or, possibly, by extending the vectors of $M$ by stacking the various matrices below each other before singular value decomposition.

Furthermore, one could train committees of approximation matrices $\hat{M}_i$ to resolve global ambiguitues. For the character example, one could use the presented model to weight the outcome of one against all other models or of models trained for often confused groups of characters.

It is futher worth noting that the low-dimensional column space can actually be used to project feature vectors into such low-dimensional space by using the matrix $U$ of the singular value decomposition. This is a useful way of spatially visualizing the capabilities of a training matrix in distinguishing objects as well as relating a novel objects to the training objects.

# 6. REFERENCES


[1] M. W. Berry, Z. Drmac, and E. R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM review*, 41(2):335–362, 1999.

[2] B. Couto, A. Ladeira, and M. Santos. Application of latent semantic indexing to evaluate the similarity of sets of sequences without multiple alignments character-by-character. *Genet Mol Res*, 6(4):983–999, 2007.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[4] K. Deng, K. Xie, K. Zheng, and X. Zhou. Trajectory indexing and retrieval. In *Computing with spatial trajectories*, pages 35–60. Springer, 2011.

[5] P. Drineas and M. W. Mahoney. Randnla: randomized numerical linear algebra. *Communications of the ACM*, 59(6):80–90, 2016.

[6] B. Everitt and G. Dunn. Applied multivariate data analysis, 2nd edn. arnold, london. Technical report, ISBN 0-340-54529-1, 2001.

[7] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[8] N. Halko, P. Martinsson, and J. Tropp. Finding structure with randomness: stochastic algorithms for constructing approximate matrix decompositions. *URL http://arxiv. org/abs/0909*, 4061, 2009.

[9] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

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

[11] M. E. Wall, A. Rechtsteiner, and L. M. Rocha. Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis*, pages 91–109. Springer, 2003.

[12] M. Werner. *Indoor Location-based Services - Prerequisites and Foundations*. Springer, 2014.

[13] M. Werner. Bacr: Set similarities with lower bounds and application to spatial trajectories. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2015.

[14] H.-J. Yu and D.-S. Huang. Normalized feature vectors: a novel alignment-free sequence comparison method based on the numbers of adjacent amino acids. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 10(2):457–467, 2013.