Stable Visual Summaries for Trajectory Collections

Jules Wulms[*] TU Wien
Juri Buchmüller[†] University of Konstanz
Wouter Meulemans[‡] TU Eindhoven
Kevin Verbeek[‡] TU Eindhoven
Bettina Speckmann[‡] TU Eindhoven

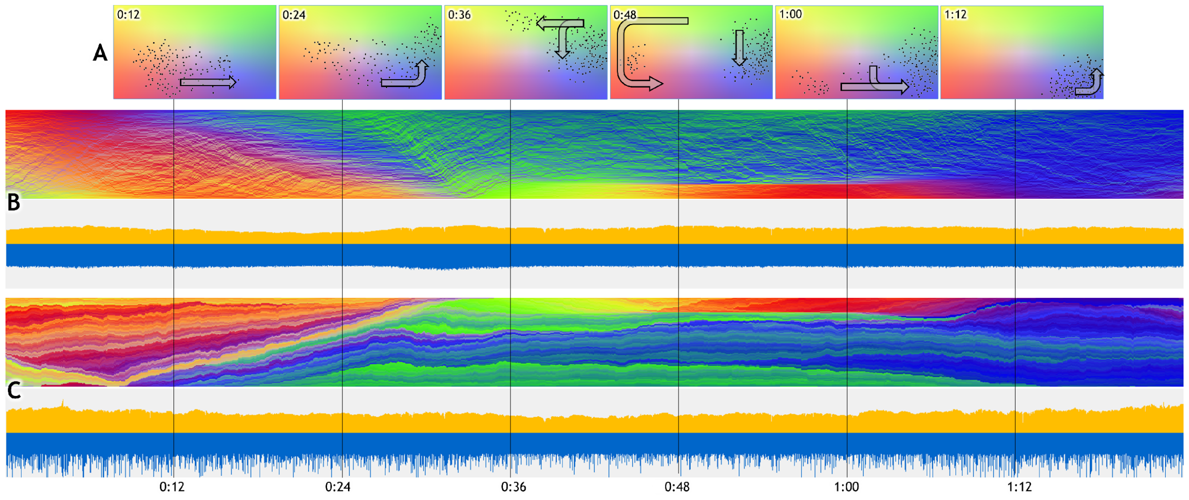


Figure 1: (A) Sample frames of movement; background indicates spatial coloring. (B) Our Stable Principal Component method translates collections of moving entities at each time step into a vertical array of pixels; color indicates spatial location. Chart below indicates spatial quality (yellow) and stability (blue) per time step; high values indicate low quality. (C) Unstable ordering using Hilbert curve, note the artificial split between the green locations.

Abstract

The availability of devices that track moving objects has led to an explosive growth in trajectory data. When exploring the resulting large trajectory collections, visual summaries are a useful tool to identify time intervals of interest. A typical approach is to represent the spatial positions of the tracked objects at each time step via a one-dimensional ordering; visualizations of such orderings can then be placed in temporal order along a time line. There are two main criteria to assess the quality of the resulting visual summary: *spatial quality* – how well does the ordering capture the structure of the data at each time step, and *stability* – how coherent are the orderings over consecutive time steps or temporal ranges?

In this paper we introduce a new Stable Principal Component (SPC) method to compute such orderings, which is explicitly parameterized for stability, allowing a trade-off between the spatial quality and stability. We conduct extensive computational experiments that quantitatively compare the orderings produced by ours and other stable dimensionality-reduction methods to various state-of-the-art approaches using a set of well-established quality metrics that capture spatial quality and stability. We conclude that stable dimensionality reduction outperforms existing methods on stability, without sacrificing spatial quality or efficiency; in particular, our new SPC method does so at a fraction of the computational costs.

Index Terms: Human-centered computing—Visualization—Visualization techniques; Mathematics of computing—Probability and statistics—Statistical paradigms—Dimensionality reduction

[*]e-mail: jwulms@ac.tuwien.ac.at
[†]e-mail: juri.buchmueller@uni-konstanz.de
[‡]e-mail: [w.meulemans | k.a.b.verbeek | b.speckmann]@tue.nl

1 Introduction

Over the past years the availability of devices that track moving objects — GPS satellite systems, mobile phones, radio telemetry, surveillance cameras, RFID tags, and more — has increased dramatically, leading to an explosive growth in trajectory data. Objects being tracked range from animals (for behavioral studies) and cars (for traffic prediction), to hurricanes, sports players (for video analysis of games), and suspected terrorists. When exploring the resulting large trajectory collections, visual summaries are a useful tool to identify time intervals for further consideration. A typical approach is to represent the spatial positions of the tracked objects at each time step via a one-dimensional ordering; visualizations of such orderings can then be placed in temporal order along a time line.

MotionRugs by Buchmüller *et al.* [2] (Fig. 1D) translate orderings of 2D moving objects into a compact grid-based visual summary; in a MotionRug each vertical strip of n grid cells encodes n objects at a specific time step. This representation is arguably as compact as possible and served as the initial motivation for our work. MotionRugs, and other previous work [2, 10] which computes orderings for moving entities, use either spatial subdivisions or clustering techniques. As a result, entities which are close in the ordering are also close in the input. However, the converse does not necessarily hold: elements which are separated in the ordering can be close in the input. This causes unfortunate and visually salient artifacts (the so-called "phantom splits" in MotionRugs). Let us emphasize that these artifacts are not caused by the visual encoding in MotionRugs, but they are an inherent consequence of using spatial subdivision or clustering techniques to create orderings; any visualization which uses the resulting orderings will exhibit the same artifacts.

The algorithmic problem to be solved is in fact low-dimensional dimensionality reduction: how to adequately represent higher-dimension data in 1D? In this paper we hence propose to use actual dimensionality-reduction techniques to compute orderings of high *spatial quality*: objects which are close in the input are also close in

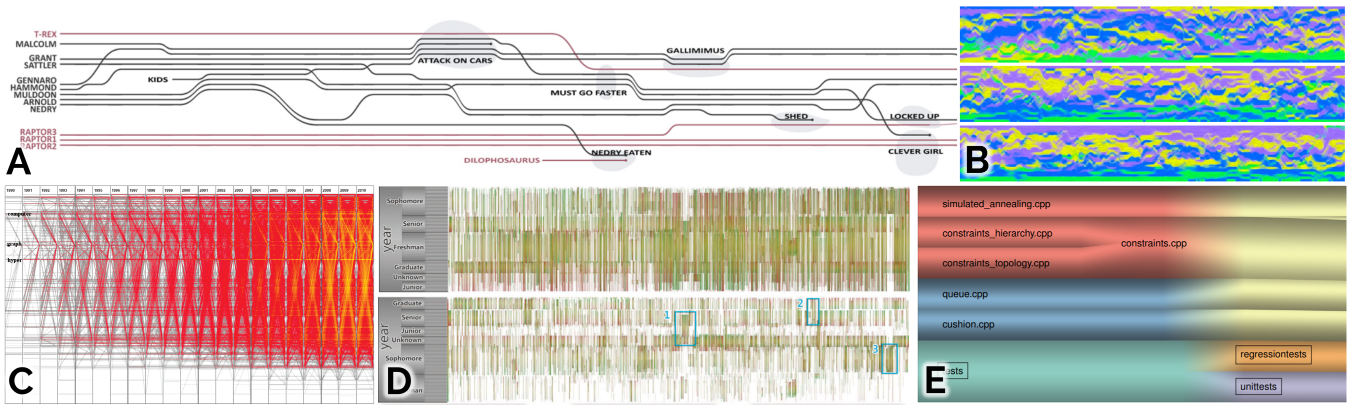


Figure 2: Examples of visual summaries in existing work: (A) Dynamic StoryLine graph [32], (B) Let It Flow for dynamic graphs [5], (C) Parallel Edge Splatting [4], (D) Extended Massive Sequence Views [30] and (E) Temporal Treemaps [15].

the ordering, avoiding the aforementioned artifacts. We also show that these techniques can produce coherent representations over whole temporal ranges, that is, they can be adapted to be *stable*.

Formal problem statement. Our input is a set $P = \{p_1, \ldots, p_n\}$ of n point objects. We sample their positions at T consecutive time steps; each object p_i is a sequence of T locations. We use $p_i(t)$ to denote the location of p_i at time t, $1 \leq t \leq T$, and, correspondingly, $P(t)$ to denote the complete point set at time t. A visual summary S of P is a sequence of orderings of the points in P, one per time step. We denote the ordering at time t by S_t, $S_t(p_i)$ denotes the rank of point object p_i in the ordering at time t. The quality of a visual summary S is determined by two criteria:

Spatial quality. How well does S_t capture the spatial structure of $P(t)$? We characterize the spatial structure via local neighborhoods: we say that an ordering has high spatial quality if points that are spatially close in the input are also close in the ordering.

Stability. How consistent are the orderings over time? Here we can consider absolute changes between orderings or changes in local neighborhoods, as captured by nearest neighbors in the ordering. Both types of measures can be considered for consecutive time steps or over temporal ranges.

Clearly, a visual summary that uses the same ordering for all time steps is maximally stable. However, the spatial quality of this representation will typically be low. Conversely, optimizing spatial quality for each time step in isolation tends to result in unstable summaries which make it more difficult for the user to track objects.

Contributions. In this paper we explore the use of dimensionality-reduction techniques to create orderings from trajectory collections. Our contributions are twofold. **(1)** We introduce a new Stable Principal Component method [SPC] which is explicitly parametrized for stability, allowing a trade-off between spatial quality and stability. We chose to "stabilize" PCA since its principal component gives us an explicit representation of the shape of a trajectory collection at any point in time. We can interpolate between the first components of different time steps to achieve stability. We also describe a stable Clustered Principal Component [CPC] method which is particularly well suited for data sets that exhibit clear clusters. For ease of explanation we describe our approaches in two-dimensions, however, they directly extend to three or higher dimensions. **(2)** We conduct extensive computational experiments, which allow us to conclude that stable dimensionality reduction outperforms existing methods on stability, without sacrificing spatial quality or efficiency. In particular, our new SPC methods does so at a fraction of the computational costs.

We discuss related work in Section 2 and describe our Stable Principal Component method in Section 3. Section 4 explains our experimental setup, including the ordering methods which we compare against (Section 4.1), the quality metrics we use to capture spatial quality and stability (Section 4.2), and our real-world and synthetic data sets (Section 4.3). In Section 5 we report on the results of our experiments. In Section 6 we close with a discussion of our results, as well as current limitations and directions for future work.

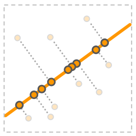
2 Related Work

Visual summaries have been used for various different types of time-varying data. For example, there are several methods that summarize time-varying graphs, such as Parallel Edge Splatting [4] (Fig. 2C) and Extended Massive Sequence Views [30] (Fig. 2D), which show the temporal evolution by drawing the graph at each time step in a narrow vertical strip. Similarly, Temporal Treemaps [15] (Fig. 2E) encode hierarchies via (essentially) one-dimensional intervals and show the temporal evolution by placing these intervals consecutively along a line. Also, Storyline Visualizations [17, 32] (Fig. 2A) use a compact representation at each time step (essentially a pixel per protagonist); these representations must be coherent between consecutive time steps and as such trace a trajectory for each actor. An interesting variation of Storylines is presented by Muelder et al. [21], who create so-called Behavioral Lines, which consist of a collection of features as 1D time series turned into comparable 2D lines, allowing users to identify similar and deviating behavior of the observed feature sets. Jäckle et al. [13] introduce another technique which is based on one-dimensional orderings, namely window-based 1D MDS plots, which are arranged horizontally, grouping together similar events. The aforementioned techniques all rely on variations of similarity measures to determine the ordering of entities and thus, the visual outcome. Our approach takes order quality into account, as well as the stability of the visualization as a function of the changes in the entity orders.

Another set of techniques exploits one-dimensional mappings to produce dense representations of temporal data to generate insightful visual patterns. MotionRugs by Buchmüller *et al.* [2] (Fig. 1C and D) computes orders from spatial locations for entities moving in 2D, whereas Cui *et al.* [5] (Fig. 2B) use node degree to order dynamic graph data. Due to the packed representation of the ordering of the displayed data points, the visual outcomes of such techniques are specifically sensitive to ordering quality and, thus, could benefit directly from our approach.

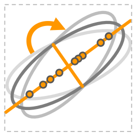
In this paper we focus on computing orderings using dimensionality-reduction techniques. We perform experiments with PCA, Sammon mapping, and t-SNE. There are other dimensionality-reduction techniques, such as MDS [16], Isomap [29], and UMAP [19], but given the cost functions that they minimize, we believe that they give similar results (in fact, in the Euclidean plane, classical MDS is equivalent to PCA). Recently, Rauber *et al.* [25] also described Dynamic t-SNE: a more explicit way of making t-SNE stable over multiple time steps. Unfortunately, various issues prevented the inclusion of this method in our experiments; see Section A of the supplementary material for details.

3 Stable Principal Component Analysis



PCA was first introduced by Pearson [23] and can be used for dimensionality reduction to 1D by projecting points onto the first principal component: a vector in the direction along which the point set has most variance. Projecting onto this vector maximally preserves spatial relations in the original point set. Based on this technique, we describe two algorithms that make such projections stable for moving entities.

Meulemans, Verbeek and Wulms [20] study the trade-off between spatial quality and stability of orientation-based shape descriptors, including PCA, from a theoretical point of view. Their results show that the principal components of a set of moving points in 2D exhibit unstable behavior when the point set is not stretched, that is, the variance along the first and second principal component is similar. Our approach leverages this result by explicitly enforcing stability when the point set is not stretched. The intuition is as follows. If the variance along the first principal component is clearly higher than the variance along the second principal component, then the direction is very discriminative: the point set is clearly stretched in this direction and sorting the points along this vector tends to lead to high spatial quality. If this is not the case, then the point set is "round" and the spatial quality is roughly equivalent for other directions as well. Our goal is to smoothly interpolate the projection vector in those cases.



[SPC$_\sigma$] Stable Principal Component. To create a stable version of PCA, we use the optimal direction (first principal component) as projection vector for any t where $P(t)$ is stretched, as well as for the first and last time step. For all time steps in between (when the point set is not stretched) we linearly interpolate the orientation of the projection vector. We use a parameter σ ($0 \leq \sigma \leq 1$) to control when we consider a point set as stretched.

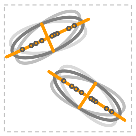
Concretely, the Stable Principal Component algorithm is implemented as follows (see Algorithm 1 for an overview). To determine if a point set is stretched, we use the corresponding eigenvalues v_1 and v_2 of the first and second principal components, respectively. If $v_2/v_1 \leq \sigma$, then the point set is stretched, and otherwise it is not. For the time steps t where the point set is stretched (including $t = 1$ and $t = T$), we simply compute the principal component as projection vector $\text{PV}[t]$. Note that $-\text{PV}[t]$ is equally good as projection vector, but results in a mirrored representation. To avoid flipping, we therefore use the direction ($\text{PV}[t]$ or $-\text{PV}[t]$) that is most consistent with $\text{PV}[t-1]$ (computed using the dot product). For time steps t where the point set is not stretched we also first compute the (consistent) first principal component. We use these vectors to keep track of the signed angle α describing how the orientation of the first principal component has changed since the last time t' the point set was stretched (or $t' = 1$). Once we reach another time t'' where the point set is stretched (or $t'' = T$), we can linearly interpolate the orientation of the projection vector for all times t with $t' < t < t''$. Although linear interpolation of orientations is not unique in general, we can use the accumulated signed angle α to uniquely interpolate the projection vector. Finally, we can project the point sets for all time steps onto the computed projection vectors $\text{PV}[t]$.

Algorithm 1 StablePrincipalComponent(P, σ)

Input: Point set P over T time steps, and $\sigma \in [0,1]$
Output: Visual summary S for P

```
 1: Set $\text{PV}[1]$ to the first principal component vector for $P(1)$
 2: Set $t'$ to 1 and $\alpha$ to 0
 3: for $t = 2$ to $T$ do
 4:     Set $\text{PV}[t]$ to the first principal component of $P(t)$ and compute eigenvalues $v_1, v_2$
 5:     Add the signed angle between $\text{PV}[t]$ and $\text{PV}[t-1]$ to $\alpha$
 6:     if $v_2/v_1 \leq \sigma$ or $t = T$ then
 7:         for $t_s = t'+1$ to $t-1$ do
 8:             Set $\text{PV}[t_s]$ to $\text{PV}[t']$ rotated over $\alpha \cdot \frac{t_s - t'}{t - t'}$
 9:         Set $t'$ to $t$ and $\alpha$ to 0
10: for $t = 1$ to $T$ do
11:     Define $S[t]$ by projecting $P(t)$ onto $\text{PV}[t]$
12: return $S$
```

Since the eigenvalues and principal components of n points in 2D can be computed in $O(n)$ time, it is easy to see that the entire algorithm runs in $O(nT)$ time. The explicit trade-off between spatial quality and stability can be configured via parameter σ. If σ is set to a value close to 1, the focus of the algorithm is on spatial quality, and only when the point set is very "round", stability will be enforced; $\sigma = 1$ eliminates interpolation and always uses the first principal component in every time step. However, if σ is set closer to 0, the focus will be on stability and even for moderately stretched point sets, linear interpolation can occur, thereby sacrificing spatial quality for stability; $\sigma = 0$ causes one interpolation, from the first principal component at $t = 0$ to the first principal component at $t = T$.



[CPC$_\sigma$] Clustered Principal Component. If a point set is strongly clustered, then we would expect an ordering of this point set with high quality to separate the different clusters. However, in the Stable Principal Component algorithm described above, two clusters may be interleaved if their projections happen to overlap. Therefore, we also propose the Clustered Principal Component algorithm, which is essentially a hybrid between SPC$_\sigma$ and a clustering algorithm (such hybrids have also been explored in [33]).

Intuitively, this algorithm performs SPC$_\sigma$ on the separate clusters. More specifically, for every frame we first perform Complete Linkage Clustering [9] [CLC] on the point set, resulting in a hierarchical clustering. CLC is agglomerative and repeatedly merges the two clusters that are closest, where the distance between two clusters is determined by the farthest two points in different clusters. To obtain a partitioning of the points, we stop the process when the closest distance between clusters doubles with respect to the previous iteration. While this heuristic suffices to find salient clusters in our data sets, many other techniques exist to find a good partitioning in a hierarchical clustering [22].

Next, we perform SPC$_\sigma$ on the individual clusters, with two small adaptations, resulting in projection vectors $\text{PV}_C[t]$ for a cluster C. First, we end the linear interpolation of $\text{PV}_C[t]$ when the clustering changes and there is no longer a cluster with exactly the same points as C (basically treating the time step as $t = T$). Second, it is no longer straightforward to determine the most consistent direction ($\text{PV}_C[t]$ or $-\text{PV}_C[t]$) for a cluster when the clustering changes. Here we use the projection vector used by the majority of the points in the cluster at time $t-1$ to determine the most consistent direction.

To find the global ordering at time t, we use the first principal component of the whole set P to project the cluster centers. The orderings of points within a cluster are then placed around the projection of its cluster center. Although this approach may still result in overlap between two clusters in the projection, we can easily separate the clusters in the eventual ordering: first, we order the clusters according to their cluster centers, and then we order the points within a cluster according to their internal ordering.

4 Experimental Setup

We aim to quantitatively evaluate methods for computing 1D orderings, based on the resulting spatial quality and stability, to understand the trade-offs that are likely to exist between these methods, as well as to understand how our own parametrized methods make a trade-off between spatial quality and stability. Below, we briefly describe the methods we compare, the measures we use to compare them, and the data that is used in this evaluation.

4.1 Algorithms

Previous work by Guo and Gahegan [10] and Buchmüller *et al.* [2] which compute 1D orderings for moving entities in 2D use either spatial subdivisions or clustering techniques. For our experiments, we chose the algorithms that performed best in their experiments. We also include a baseline algorithm [FXD] that is solely focused on stability. Fig. 3 shows examples of the orderings generated by some of the algorithms, including dimensionality-reduction techniques, for one time step of our test data. We give a short overview here; details can be found in Section A of the supplementary material.

[FXD] Fixed order. Outputs the same arbitrary linear order for every time step; each horizontal line represents one moving entity.

Spatial subdivisions. Several well-known 1D ordering approaches used for spatial indexing, such as tree data structures and space-filling curves, are based on iterating through some spatial subdivision. Many variations exist; see [18] for an overview. Here we focus on four established, representative techniques from this area: **[HIL]** Hilbert curve [12], **[ZOR]** Z-order curve, **[PQR]** Point Quadtree [7, 8] and **[RTR]** R-tree [11].

Clustering. Another approach is to first compute a hierarchical clustering on the point set, and then order the points in such a way that clusters stay together. These algorithms are defined by how the points are clustered, and how the linear order is computed from the clustering. We use the aforementioned **[CLC]** Complete Linkage Clustering [9] and **[SNN]** Shared Nearest Neighbors [14] to cluster points and derive an ordering from the cluster hierarchy as follows.

The hierarchical clustering is represented by a tree with the individual points stored in the leaves. We aim to order the leaves of such a tree without changing the cluster structure, that is, by only changing the order of the children of any internal node. We follow the algorithm by Bar-Joseph *et al.* [1] to efficiently compute the optimal order that minimizes the length of the path formed by visiting the input points in that order.

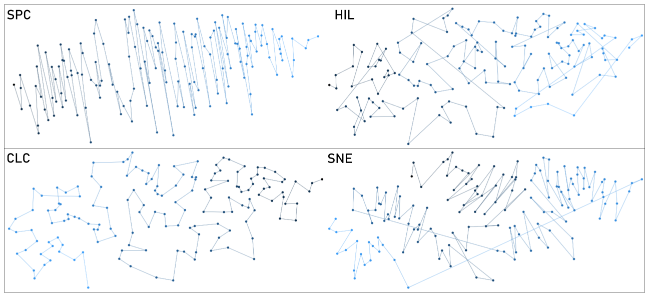


Figure 3: Visualization of orderings generated for one data frame using dimensionality reduction (SPC and SNE), space-filling curves (HIL) and clustering (CLC). Note the very different ways in which space is transformed into an ordering.

Dimensionality reduction. We also consider dimensionality reduction techniques to compute 1D ordering. In our experiments we specifically consider **[SAM]** Sammon mapping [28] and **[SNE]** t-SNE [31], next to the PCA-based techniques described in Section 3.

Both Sammon mapping and t-SNE use gradient descent to minimize a cost function. Normally, this gradient descent is started with a random initial solution, but this may result in poor stability over time. To improve the stability of both algorithms, we initialize them with the solution of the previous time step, resulting in two stable versions, **[SAMp]** and **[SNEp]**. The rationale is that, if the local minimum found in the previous time step still exists, but has slightly shifted, then this approach will likely find this local minimum again rather than any other local minimum.

4.2 Metrics

In this section we discuss the quality metrics we use to capture spatial quality and stability. We choose measures that focus on the preservation of local neighborhoods. For applications where other types of measures are preferred, we refer to the survey in [6].

4.2.1 Spatial Quality

Spatial quality measures the correspondence between $P(t)$ and the linear order S_t. We capture this by considering the local neighborhood of a point, as characterized by its nearest neighbors. One way to measure changes in local neighborhoods is using an evaluation of dimensionality reduction via persistent homology as introduced by Rieck and Leitte [27]. However, we choose not to use this type of measure. While this approach is more recent than the measure we are using, it does not compare to older results, it is more complex, and most importantly it does not directly relate input to output, but measures through an intermediate topological structure. Hence, we use the *Keys Similarity* measures as described by Guo and Gahegan [10] to directly measure the changes in nearest neighbors.

To simplify notation, we omit dependencies on time step t, as the metrics consider each time step in isolation. Thus, P denotes a point set in the plane, and S denotes a linear order. Let $n(i,j) \in P$ denote the j^{th} nearest neighbor of p_i in P, for each j with $1 \leq j \leq k$ for some constant k. We use $r(i,j)$ to denote the neighbor rank in S between p_i and $n(i,j)$. However, the difference in rank $|S(n(i,j)) - S(p_i)|$ is not unique. There are two neighbors at rank difference 1, two at rank difference 2, until we reach one end of a linear order. To avoid arbitrariness, we do not break ties but rather consider each pair with the same rank difference to have the same value for $r(i,j)$. Thus, there are two nodes with $r(i,j) = 1$ (rank difference 1), two nodes with $r(i,j) = 3$ (rank difference 2), etc.

Generally, Keys Similarity at time t is then defined as

$$KS(P,S) = \frac{\sum_{p_i \in P} \sum_{j=1}^{k} w(i,j) \cdot r(i,j)}{\sum_{p_i \in P} \sum_{j=1}^{k} w(i,j)},$$

where $w(i,j)$ denotes the weight or importance of maintaining the j^{th} nearest neighbor of p_i at time t – note that these weights need not be the same at every time step. We use two variants of Keys Similarity, see the supplementary material for the exact formulas.

[KSra] Rank-weighted Keys Similarity. We define $w(i,j) = 1/j$ inversely proportional to the rank; hence maintaining the closest neighbors is more important than maintaining the distant neighbors.

[KSdi] Distance-weighted Keys Similarity. We define $w(i,j) = 1/\|p_i - n(i,j)\|$ inversely proportional to the Euclidean distance, such that maintaining close neighbors is more important than maintaining distant neighbors. In contrast to KSra, this variant does not treat neighbors at (nearly) identical distances differently.

Other facets. Our metrics focus on combinatorial aspects of the position of the point objects. Spatial structure in general knows many other facets, such as distances and directions between points, as well as density. For projections into a single dimension, distances and density can factor into spatial quality. However, a linear order inherently does not lend itself to represent such concepts.

4.2.2 Stability

Stability or temporal coherence measures the similarity between consecutive orders in S. In our evaluation, we use the following three measures for stability. The first two are based on absolute changes in the order and match the measures used by Buchmüller *et al.* [2] to evaluate MotionRugs. The latter uses neighborhoods, based on the concepts by Guo and Gahegan [10].

We aim to compare the similarity between two linear orders, S_t and S_{t+1} for each t with $1 \leq t < T$. We could easily use the same metrics to compare nonconsecutive orders, but this provides little insight for such inherently sequential data. To consider the stability over a temporal range $[t, t']$, we use standard summary statistics (e.g., average, minimum, or maximum) over all consecutive pairs.

[JMP] Jump distance. We quantify the jump distance for a single point object p_i as the difference between its ranks in the two orders, that is, $|S_t(p_i) - S_{t+1}(p_i)|$. The jump distance between two orders is then the sum over all jump distances for each point object.

$$JMP_t(P,S) = \sum_{p_i \in P} |S_t(p_i) - S_{t+1}(p_i)|$$

The value for $JMP_t(P,S)$ lies between 0 (perfectly stable) and $\frac{n(n-1)}{2}$ (complete inversion of the order).

[CRS] Crossings. Whereas JMP penalizes any change in the order, many points moving up together may not constitute much change. Instead we may count the number of inversions or crossings in the order, that is, the pairs p_i, p_j for which $S_t(p_i) < S_t(p_j)$ and $S_t(p_i) > S_t(p_j)$. The metric $CRS_t(P,S)$ lies between 0 (perfectly stable) and $\frac{n(n-1)}{2}$ (complete inversion of the order).

Buchmüller et al. [2] also use Kendall's τ coefficient to evaluate stability. We choose to omit this, as it is equivalent to $1 - 2 \cdot CRS_t(P,S)/(n(n-1)/2)$. That is, Kendall's τ is the same as CRS up to normalization to the range $[-1, 1]$.

[KSte] Temporal Keys Similarity. We may also take the same approach as for spatial similarity and consider the similarity of local neighborhoods in both orders. As distances are not inherently meaningful in the combinatorial order and simply correspond to ranking differences, we use only the rank-weighted version of Keys Similarity. Also for this metric $KSte_t(P,S)$, we do not break ties in either order, but rather give them the same rank.

4.3 Data

For comparability, we use the same data as MotionRugs [2] along with two synthetic data sets, one generated using Netlogo [35] and another generated with the well known Reynolds model [26]. Note that we use MotionRugs solely to visualize the output of our algorithms. The algorithms as well as the quantitative evaluation are independent of the visualization that uses the resulting ordering.

The first data set tracks 151 fish of the Notemigonus crysoleucas species (Golden shiner). Golden shiner fish live in large groups called "shoals", moving in coordination at almost any given time. The 151 fish were tracked optically while moving through a 2.1m by 1.2m shallow water tank, thus avoiding movement in the third dimension. The tank did not feature any obstacles or hindrances besides the side walls. Different movement patterns can be observed in the data, which allows us to test quality in different situations. Among these patterns are uniform group movements, partial and complete changes of direction, circular movement patterns, splitting off in separate clusters, and changes in group density, speed, and acceleration. This data set is quite large, hence we use two excerpts of 2000 frames of movement, which were recorded at a rate of 25 frames per second, each resulting in 80 seconds of available collective movement data. For each frame, the spatial coordinates of each fish are recorded in a Cartesian coordinate system. Fig. 17 in the supplementary material is a visual summary of the full data set.

Fish 1. In the first excerpt, the fish first move around the boundary of the tank and finally enter a so-called milling formation, moving in a circular shape. The fish always form a single cluster.

Fish 2. In contrast, the second excerpt shows the fish splitting in separate clusters, as can be seen in Fig. 1A.

In addition to analyzing Fish 1 in full, we also zoom in on a small part of the movement of the fish, which triggers so-called "phantom splits" [2] for certain ordering methods, most notably HIL, PQR, and SNEp (see Fig. 6). The shoal of fish appears to split, but this is purely an artifact of the method and not reflecting the data.

Netlogo. The Netlogo data set is generated using the Flocking model [34] from the openly available Models Library within the Netlogo application. Minimal adaptations were made to the model to ensure the boundaries of the canvas do not wrap around, and the trajectories of the moving entities could be extracted easily.

Reynolds. The final data set, which we use to demonstrate clustered movement, is generated by an adaptation from the well known Reynolds model [26], where between the movers of the three visible clusters only repulsion forces apply, but no attraction, keeping the clusters separate. The generator code by Piljek [24] is public.

Since the results are similar across all data sets, we mainly focus on Fish 1 and Reynolds, while highlighting notable difference in the results of the other data sets. A full analysis for the remaining data sets is given in the supplementary material.

5 Experimental Results

Here we report on the results of the quantitative experiments described in Section 4. Tables 1–4 in the supplementary material provide summary statistics over all time steps and for each metric, for all data sets. We first determine the most effective measures for spatial quality and stability. We then explore how the parameter for SPC effects a trade-off between spatial quality and stability. Finally we briefly discuss the computational efficiency.

Our experiments explore how these methods and quality criteria relate. To illustrate the resulting orderings, we use MotionRugs using a 2D RGB colormap as introduced in [3]. As can be seen in Fig. 1, in the orderings (B) and (C) the data points are colored according to their 2D position in (A). In visual summaries of high spatial quality, data points that are close in 2D should be close in 1D, hence similar colors should end up close to each other. Furthermore, in stable summaries the neighborhoods do not change much in the orderings, and hence the colors should smoothly change over time. Thus, this graphical representation of the orderings allows us to also visually assess spatial quality and stability.

5.1 Quality Results – Fish 1

Fig. 6 and 7 show visual summaries for all algorithms for Fish 1. The MotionRugs are accompanied by a visualization of the mean KSdi and KSte values for each frame, cut off slightly above the mean values of most algorithms. This ensures that the differences between the average behavior of the algorithms becomes visible at a glance. Below, we first discuss spatial quality and stability statistics separately, along with a discussion on phantom splits. We follow up with an exploration into the effects of the parameter value on the outcome of SPC and finally consider the trade-off between spatial quality and stability for all methods.

Spatial quality. Fig. 4 compares the spatial-quality measures KSra and KSdi, as measured on all algorithms for Fish 1. For both measures lower values indicate higher spatial quality. Overall, we see that the KSra measurements are slightly lower for all algorithms, except SNEp where KSdi has a minimal edge over KSra. As expected

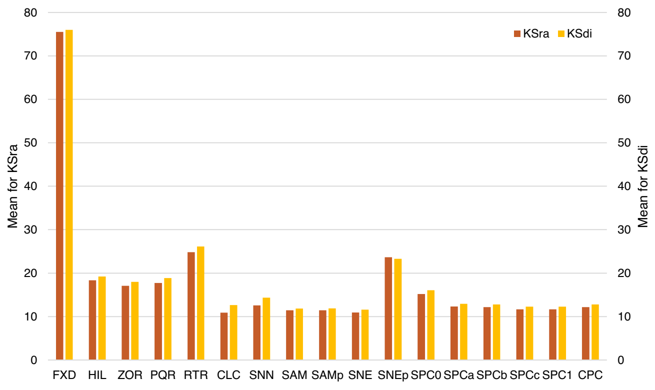


Figure 4: Spatial-quality metrics: mean KSra (left) and KSdi (right) for all algorithms over all frames of Fish 1. Lower numbers indicate better overall spatial quality of the ordering.

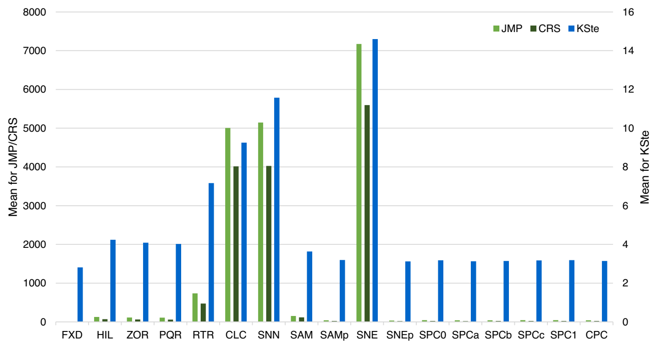


Figure 5: Stability metrics: mean JMP, CRS (left axis), and KSte (right axis) for all methods over all frames of Fish 1. Lower numbers indicate better overall ordering stability.

FXD achieves the worst spatial quality. Furthermore, SNEp and the algorithms using spatial subdivisions are outperformed by the clustering algorithms, and other dimensionality-reduction techniques. Comparing the spatial quality of SPC and CPC to the algorithms that perform best on spatial quality, we see that SPC and CPC both achieve comparable spatial quality. The choices for parameter σ of SPC on Fish 1 are 0, 1, and variables $a = 0.35, b = 0.53, c = 0.78$, while for CPC we chose $\sigma = 0.53$. The choice for the intermediate values a, b and c is different for the various data sets and will be justified in the parameter exploration. For CPC we choose a parameter value that according to the parameter experiment performs well on both spatial quality and stability, again different for every data set. Due to the strong correlation of both spatial quality measures, we focus on only KSdi in the remainder.

Stability. Fig. 5 compares the stability measures: JMP, CRS and KSte. While JMP and CRS measure absolute changes between orders, KSte captures changes in local neighborhoods. For each measure lower values indicate higher stability. We see that CRS results in lower values than JMP, which is expected: two entities can jump to different positions in the next frame without crossing, but they cannot cross each other without jumping. We do see some differences between the two data sets, as opposed to the results for spatial quality. For FXD the result is again obvious: all measures are at their minimum. While JMP and CRS are generally low, CLC, SNN and SNE show very high numbers. Those three algorithms also perform worst according to the KSte metric. Another outlier that performs poorly on KSte is RTR, which also performs comparatively poorly on JMP and CRS. Of the remaining algorithms, the spatial subdivisions perform worst on KSte. The SAM, SAMp and SNEp algorithms, the SPC variants and CPC show similar and very low mean values of KSte. Again, we observe a strong correlation between the metrics, and thus consider only KSte in the remainder.

Phantom splits. In Fig. 6 each frame of movement data is represented by a column of pixels, where each fish corresponds to a pixel. The pixels are colored according to the position of the fish in 2D, as shown in Fig. 1. The ordering method clearly defines the resulting visual patterns. An anomaly can be identified in the subdivision methods (HIL, ZOR, PQR, RTR) and SNEp, the so called phantom splits [2]. These visual summaries suggest that the shoal of fish somehow splits, but this is not the case. Such patterns are hence undesirable, as they convey false information. Other algorithms do not seem to be prone to these kind of visual artifacts or generally produce visual results too fuzzy for such patterns to appear. Some algorithms, such as CLC and SNE, result in cluttered visuals despite having good spatial quality. This clutter is caused by instability: the summaries fail to convey patterns over time despite individual frames being objectively good.

SPC parameter. We now investigate the parameter σ of SPC and its effect on the results. We run SPC for 101 different values for σ from 0 to 1 with increments of 0.01. As discussed before, we use KSdi to measure the spatial quality of the visual summaries, and specifically we use the mean over all frames. For stability we use the mean as well as the max KSte to quantify stability. As we saw before, mean KSte captures cohesion over time, while max KSte should be low to prevent visual artifacts from disrupting temporal patterns. The results for Fish 1 are shown in Fig. 7 and 8. Note that the highest plotted value of σ is 0.95, while the lowest is 0.29. Values above

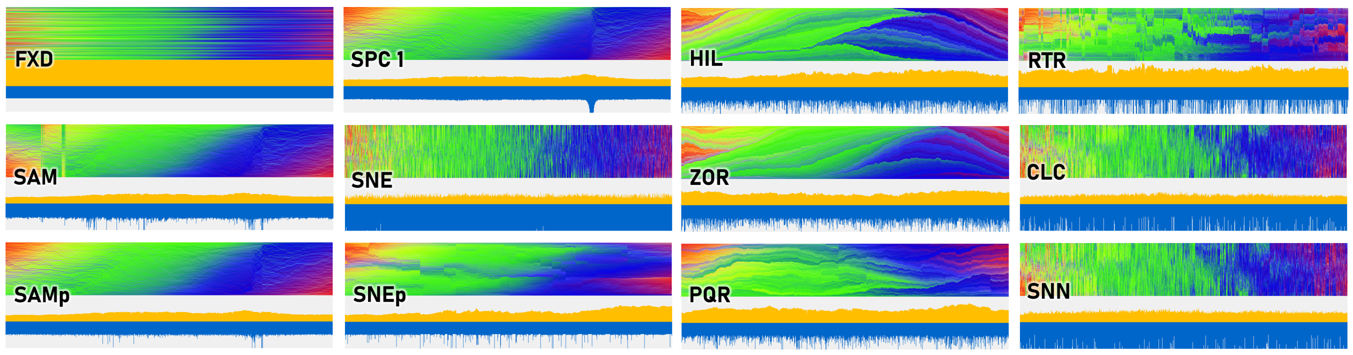


Figure 6: Visual summaries for Fish 1, focused on so called phantom splits for SNEp, HIL, ZOR, PQR and RTR. Below each we show KSdi (yellow) and KSte (blue), capped at 37.5 and 6.25, respectively. More gradual color changes in the MotionRug relate to better quality.

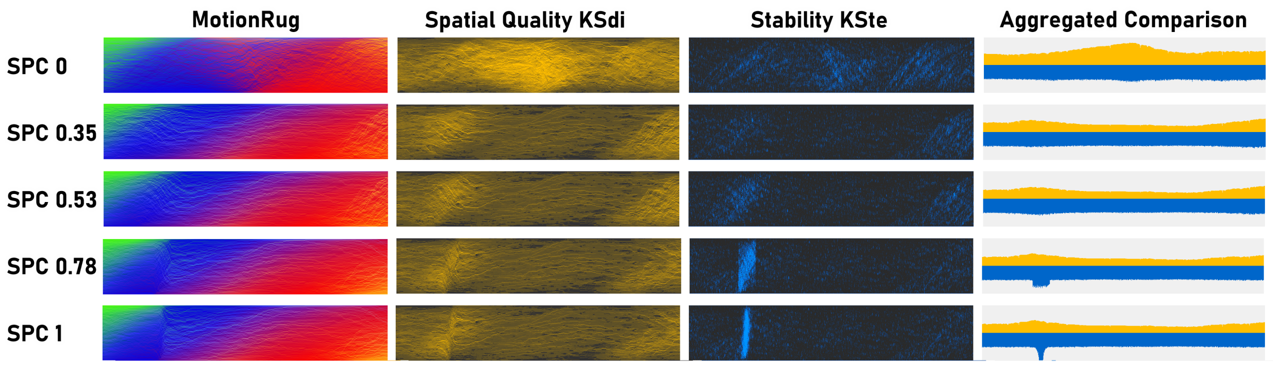


Figure 7: Visual summaries for Fish 1, focused on instabilities to show how σ affects interpolation in SPC. The summaries in the second and third columns show how each data point adds to the KSdi (yellow) and KSte (blue) measures. Brighter colors indicate worse spatial quality and stability. The measures are aggregated per frame in bar charts on the right.

and below these extremes are identical to results with 0.95 and 0.29 respectively. The σ values indicated by labels in the figures are chosen as representatives, and used in our other experiments.

Overall, we see an inverse relation between stability and spatial quality. Values of σ closer to 1 result in better spatial quality, while values closer to 0 sacrifice some spatial quality for more stability. This is to be expected, as SPC_1 always projects the fish to the first principal component; this will likely lead to the best spatial quality that can be achieved for any parameter value.

As σ is decreased, SPC increasingly uses interpolated lines for projection instead. This interpolation smooths changes in angle of the line, but the projection reflects spatial relations less accurately as a result. When σ drops below 0.30, the interpolation happens purely between the first and last frame of the data set. Contrary to expectation, this negatively affects both spatial quality and stability: the first principal component rotates both clockwise and counter-clockwise at varying speeds, not matching the uniform interpolation over such a long time period; as a result, the interpolated lines do not correspond at all to the first principal components, neither in angles nor in rotation direction. This mismatch in angles leads to poor spatial quality per frame, while the mismatch in rotation direction also decreases stability.

Finally, we explicitly show the effects of changing σ on the resulting visual summaries using Fig. 7. In this figure, we visualize spatial quality in yellow and stability in blue. On the left we show how much every point contributes to the measures, with brighter colors indicating worse spatial quality/stability, while darker colors show placements in the ordering of high spatial quality or stability. On the right the aggregated values over all points in the data are visualized in a histogram. Fig. 7 specifically shows an instability that occurs in Fish 1, using the same intermediate values for σ as before. When the first principal component is used without introducing stability ($\sigma = 1$), we see a short burst of instability, along with slightly elevated measurements in spatial quality. As σ decreases and more stability is introduced by interpolating the direction of the first principal component over larger time frames, we see that the instability is distributed over more frames and decreases. The spatial quality is not negatively impacted by this, until σ becomes too low: when we interpolate over too many frames at once, spatial quality will drastically deteriorate, as seen for $\sigma = 0$.

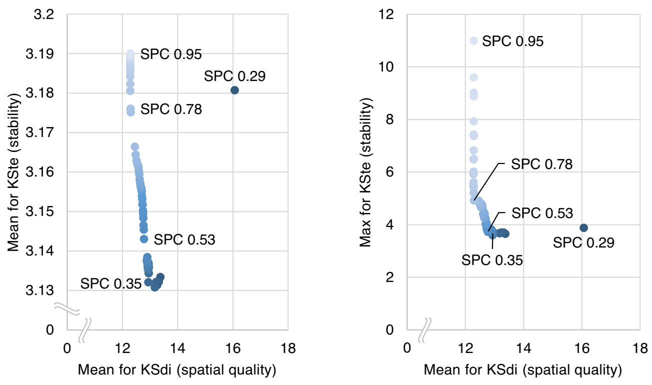


Figure 8: Comparing the mean and mean (left) as well as max and mean (right) for KSte (stability) and for KSdi (spatial quality), for uniformly distributed parameter settings of SPC on Fish 1.

Trade-offs. Our main goal is to investigate the trade-off between spatial quality and stability. Fig. 9 shows a scatterplot on the means of KSdi and KSte of all algorithms. Since lower values indicate better quality for both, methods in the bottom-left corner perform well on both aspects. In both figures SPC variants and CPC are colored in shades of red, SAM variants in blue and SNE variants in green. The "best" variants have fully opaque colors, while unstable variants or those of worse spatial quality have a lighter shade.

The results for Fish 1 clearly show that methods based on spatial subdivisions (ZOR, HIL) and space-filling curves (PQR, RTR), albeit fast to compute, perform poorly on spatial quality and stability. The clustering methods (CLC and SNN) as well as SNE, on the other hand, perform well on spatial quality, but exhibit very poor stability.

The fixed order (FXD) and SNEp are on the other extreme, having good stability, but very poor spatial quality. Furthermore, the strong influence of initialization for t-SNE stands out. When initialized with random coordinates (SNE), the spatial quality is very good, but the stability is extremely poor. On the other hand, initializing t-SNE with the embedding of the previous time step (SNEp) greatly improves stability, but spatial quality suffers greatly.

That leaves SAM, SAMp, SPC variants, and CPC, which perform well on both aspects. We note that SAM and SAMp perform very similarly on KSdi (difference of 0.03), but SAMp performs significantly better in terms of stability. SPC variants also strike a good balance between spatial quality and stability. All SPC variants have slightly worse spatial quality than SAM variants, but improve stability. However, recall that SPC is significantly faster to compute than the Sammon mapping algorithms SAM and SAMp. Finally, CPC performs similarly to SPC, which is expected since the fish stay grouped in a single cluster, hence CPC and SPC have very similar outcomes on this data set.

It is also important for stability to be consistently low, to avoid

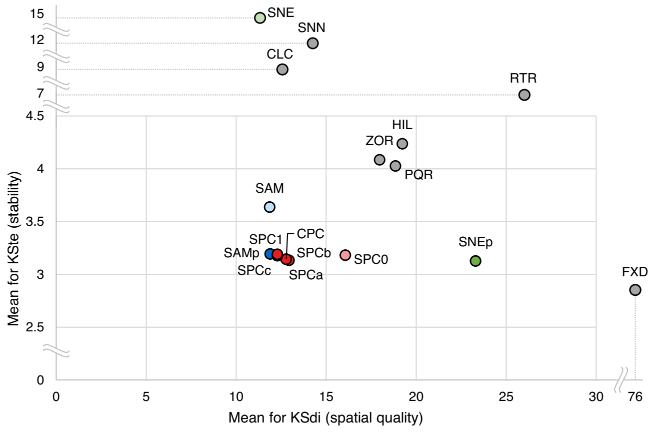


Figure 9: Comparing the mean for KSte (stability) and for KSdi (spatial quality) for all algorithms on Fish 1.

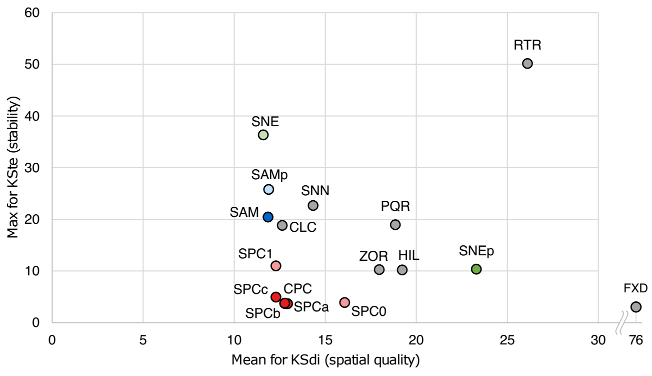


Figure 10: Comparing between the max for KSte (stability) and the mean for KSdi (spatial quality) for all algorithms on Fish 1.

visual artifacts and ensure that visual patterns in the summary point to actual movement patterns. As such, bursts of high instability are undesirable. We hence also consider the maximum value of KSte; see Fig. 10 for a scatterplot. The overall composition remains similar, but differences in stability are highlighted. Note that SAM, SAMp and SNEp are deteriorating with respect to other methods; we can also see clear bursts of instability in Fig. 6 for these methods.

Interestingly, SAMp performs worse here on stability than SAM, unlike for other data sets. This shows that, although initializing the gradient descent with the solution of the previous time step generally improves stability, there is no guarantee that it will always do so; there may be outliers, as is the case here.

Fig. 10 also highlights stability differences between SPC variants. SPC_1 always uses the first principal component, which can behave erratically for round point sets, decreasing stability. The other variants of SPC overcome this problem by interpolating over these bursts of instability. Indeed, SPC is largely unaffected for lower parameter values, having the smallest standard deviation overall (see Table 1 in the supplementary material).

5.2 Quality results – Reynolds

In this section we present the results of our experimental evaluation for the Reynolds data set. For spatial quality and stability we only sketch our results. A more elaborate analysis, including figures, can be found in Section E in the supplementary material.

Spatial quality & stability. Overall, the relative performance of most algorithms on both spatial quality and stability is similar for Reynolds and Fish 1. However, on spatial quality measures clustering techniques and CPC perform better relative to the other algorithms, making CPC one of the best algorithms to achieve spatial quality. Since there are multiple cluster in this data, which do not interact with each other, it is not surprising that these techniques perform so well. In terms of stability, lower values are also measured for the absolute changes for clustering techniques, and CPC performs relatively better than on other data sets. Especially when considering the max values for KSte we see that the clustering techniques and SNE perform better than on the other data sets. The clustering technique CLC even beats some spatial subdivision techniques (HIL and ZOR) on this measure. All of these results are to be expected: since the clusters do not interact and contain the same points in every frame, the clustering techniques also perform very well on stability. Finally, SNE is less of an outlier for this data set, as SNN also performs worse on the mean KSte.

Parameter experiment. The parameter experiment gave some surprising results for Reynolds. Fig. 11 shows charts containing the results. The cut-off values are 0.98 and 0.42 for this data set, meaning that every value above 0.98 and below 0.42 uses the same projection vectors as the visual summaries using the cut-off values. The parameter values that are indicated by labels in the figures are the values we used in our other experiments. As intermediate values we choose $a = 0.50$, $b = 0.59$, and $c = 0.86$.

First we consider the parameter values between 0.86 and 0.59. These values show the inverse relation between spatial quality and stability: as the parameter value decreases, the stability increases while the spatial quality deteriorates. This is the expected behaviour, which we already saw for Fish 1.

Between values 0.59 and 0.50 we see that lowering σ improves both the spatial quality as well as the stability. The first principal component does not seem to be the vector that results in the best spatial quality here, hence interpolating more can give better spatial quality, while improving stability. Lowering σ further results in worse spatial quality and stability, as we saw before.

Finally between 0.98 and 0.86 we see better stability when increasing σ. While this is counter-intuitive in general, it can be explained for this data set. As σ increases we interpolate over less frames and interpolate over configurations where the point set is very rounded. This has a positive effect on the stability in this data set, since it prevents 2 clusters from overlapping a lot: When interpolating, we get a lot of frames where the projected points of two clusters interleave, while the points move in opposite directions. This causes

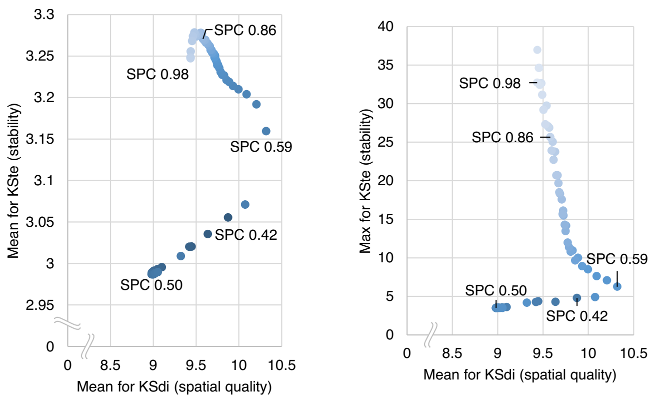


Figure 11: Comparing the mean and mean (left) as well as max and mean (right) for KSte (stability) and for KSdi (spatial quality), for uniformly distributed parameter settings of SPC on Reynolds.

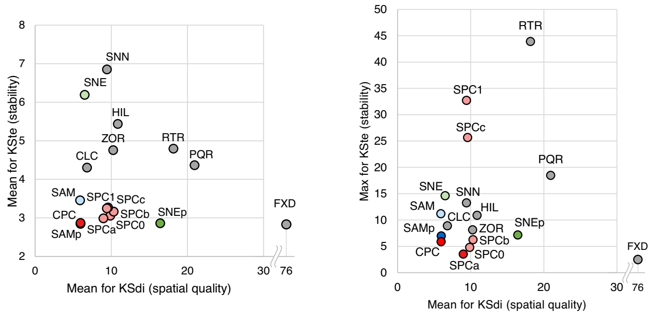


Figure 12: Comparing the mean for KSdi (spatial quality) and the mean (left) and max (right) for KSte (stability), for all algorithms on Reynolds.

many changes in the neighborhood of all the points in those two clusters. If we interpolate less, this behaviour is less prominent and contained in a few frames, resulting in better stability.

Trade-offs. The trade-offs between spatial quality and stability for the clustered data set can be observed in Fig. 12. As we already observed when considering stability in isolation, clustering techniques and SNE perform really well on this data set, especially when considering maximum values for KSte. These techniques end up in the bottom left corner, making them viable techniques for data sets that are clustered. However, they are still outperformed by SPC variants for low σ values, SAMp, SNEp and CPC, when it comes to stability. For $\sigma = 0.50$ SPC performs particularly well, even better than SAMp and CPC on maximum KSte. However, SAMp and CPC also have very good spatial quality, making them the best techniques for this data set.

5.3 Quality results – Fish 2 & Netlogo

For the remaining data sets, the values for spatial quality and stability are very similar to the previous data sets. We therefore give a short overview of the parameter experiment and trade-offs.

Parameter experiment. For both data sets there is mostly an inverse relation between stability and spatial quality. However, for the Netlogo data set this relation is absent for $0.45 \leq \sigma \leq 0.59$. Increasing σ in this interval leads to both worse stability and worse spatial quality, like we saw in the Reynolds data set. During instabilities we observe that at certain frames where SPC projects to an interpolated line, the spatial quality is better than when we project to the first principal component. This explains how more interpolation (for decreasing σ) can improve spatial quality. For the Netlogo data set we further observe that both spatial quality and stability change erratically for $\sigma < 0.40$, caused by large intervals of interpolation.

Trade-offs. For both data sets we see similar results as before, except for CPC, which seems to perform worse. In general, whenever the outcome of the clustering in CPC changes, this significantly changes the ordering and results in a burst of instability. Fish 1 and Reynolds are not affected by this, as the clustering does not change in those data sets.

5.4 Running Time

We implemented and executed all algorithms in Java 11 on a workstation with two Intel Xeon E5-2687W CPUs at 3.10GhZ, 16 Cores, 128GB Ram and an NVidia Quadro M600 GPU, running Windows 10. We measure running times only for computing the orderings, excluding reading input, color mapping and rendering. The running times range from a few milliseconds for the Z-Order curve (ZOR) to just over 8 hours for t-SNE (SNE). General observations include comparably good performance for the subdivision methods (ZOR, HIL, PQR, RTR), with values under one second. Only SPC variants are on par with this speed.

5.5 Conclusion

Overall, stable dimensionality reduction methods such as SAMp, SNEp, and SPC for parameter values lower than 1 perform very well in terms of average and worst-case stability, while only marginally sacrificing spatial quality in the case of SAMp and SPC. SPC does so at a fraction of the computational cost necessary for more complex dimensionality reduction techniques. Considering all the above, we conclude that stable dimensionality reduction methods are the best for computing visual summaries of time-varying data.

6 Discussion & Future Work

Here we discuss our results and future work in the context of data properties, algorithm performance, and visual summaries.

Movement characteristics. Our results show that algorithm efficacy is influenced by the characteristics of the moving entities. SPC works particularly well for a a single, roughly convex cluster (Fish 1) or with only a few such clusters (Fish 2), even though the method emphasizes cluster order and uses a suboptimal axis within each (Fish 2). We can consider many clusters with only a few entities to be effectively the same as a single cluster, as the order within a cluster has little to no influence on the spatial quality.

With multiple, reasonably sized clusters (Reynolds), separating the different clusters in the linear order can be desirable. By their nature, clustering-based methods will perform better in this regard. But our experiments show that such methods nonetheless struggle to find a good, stable order within the clusters. Our hybrid CPC method combines the advantages of clustering with SPC. However, when the cluster composition changes, stability is now harder to achieve as CPC does not interpolate between the different cluster compositions. We leave to future work how such a hybrid method can be turned into a “clairvoyant” algorithm [20] that already aligns the SPC axes of clusters before a change in clusters is actually occurring.

For a complexly shaped cluster, we face yet another issue. Cluster detection might not find an adequate structure. Neither does a single, straight projection axis necessarily capture proximity or neighborhood structure well and is hence likely to give unsatisfactory results as well. Perhaps methods from topological persistence can play an important role in identifying the structures of these clusters. We leave the development and evaluation of algorithms for more complex data as future work. Our results show the potential here, for adapting existing methods to explicitly consider stability.

Beyond spatial data. Our stable methods can be used in any situation with time-varying data in at least two (numeric) dimensions, to determine the ordering and construct visual summaries from these. In other words, our techniques may thus be useful for providing an overview also for nonspatial data. However, we expect it to be primarily useful when proximity (or more generally, neighborhoods) of items are meaningful and of interest. Investigating precise conditions under which this approach is effective is left to future work.

Enriching summaries. We can augment a visual summary with indicators of its spatial quality and stability. In this paper we used juxtaposed bar charts (Fig. 1 and 6) or color (Fig. 7). Various other encodings could also be considered, e.g. reducing the saturation of the colors or underlining the summary with two lines where the pixel colors indicate the spatial and temporal quality. Such augmentations carry information about uncertainty introduced by the dimensionality reduction. How to best visually convey such information, and how this affects interpretation are left to future work.

If the graphical space and the number of elements allow for it, we can also use not just the order but the actual resulting 1D representation of dimensionality reduction. Connecting the different positions for each object with a single line then gives us what we

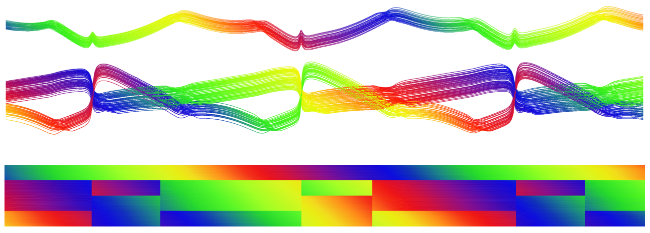


Figure 13: A MotionLines and a MotionRug of the clustered data set, using 1D representations produced by our CPC algorithm ($\sigma = 0.5$).

think of as "MotionLines", which combine ideas from MotionRugs [2] and Story Lines [32]. As shown in Fig. 13, the additional space can be used to communicate more information, such as relative distances or cluster structure. This hints at a possible trade-off between the compactness and expressiveness of visual summaries, which can be further investigated in future work.

Overview-first. Visual summaries are primarily an overview-first tool. They give an analyst a rough idea of what happens during the motion of the entities, as a first entry point to find time spans or sets of entities to further investigate. It is thus important to understand how movement patterns relate to patterns visible in the summary and vice versa. To ensure that *collective* movement of subgroups leads to observable patterns in a visual summary, we need the attribute used for coloring to be similar for spatially close entities. Without a relation between spatial proximity and attribute value, the colors may jump and it becomes difficult to follow entities or subgroups.

Generally, an efficacious visual summary is coherent: its observable visual patterns are meaningful in the actual movement and vice versa. Though we leave the visual saliency and meaning of patterns to future work, our results point towards algorithms that have good measured performance in the spatial and temporal domain.

Acknowledgments. The authors wish to thank Prof. Dr. Iain Couzin and Dr. Alex Jordan of the Max-Planck-Institute for Ornithology, Radolfzell, Germany, and Dominik Jäckle for helpful insights and discussions, and for providing the fish data set.

W. Meulemans and J. Wulms were partially supported by the Netherlands eScience Center (NLeSC); grant no. 027.015.G02. J. Wulms was partially supported by the Austrian Science Fund (FWF), grant P 31119. B. Speckmann and K. Verbeek were partially supported by the Dutch Research Council (NWO); project no. 639.023.208 and no. 639.021.541, respectively.

References

[1] Z. Bar-Joseph, E. Demaine, D. Gifford, A. Hamel, T. Jaakkola, and N. Srebro. K-ary Clustering with Optimal Leaf Ordering for Gene Expression Data. *Bioinformatics*, 19(9):1070–8, 2003.
[2] J. Buchmüller, D. Jäckle, E. Cakmak, U. Brandes, and D. A. Keim. MotionRugs: Visualizing Collective Trends in Space and Time. *IEEE TVCG*, 25(1):76–86, 2019.
[3] J. Buchmüller, U. Schlegel, E. Cakmak, D. A. Keim, and E. Dimara. SpatialRugs: Enhancing Spatial Awareness of Movement in Dense Pixel Visualizations. *Proc. 11th EuroVA*. pp. 1–5, 2020.
[4] M. Burch, C. Vehlow, S. Diehl, and D. Weiskopf. Parallel Edge Splatting for Scalable Dynamic Graph Visualization. *IEEE TVCG*, 17(12):2344–2353, 2011.
[5] W. Cui, X. Wang, S. Liu, N. Henry Riche, T. M. Madhyastha, K. Ma, and B. Guo. Let It Flow: A Static Method for Exploring Dynamic Graphs. In *Proc. 7th PacificVis*, pp. 121–128, 2014.
[6] M. Espadoto, R. M. Martins, A. Kerren, N. S. Hirata, and A. C. Telea. Towards a Quantitative Survey of Dimension Reduction Techniques. *IEEE TVCG*, 2019.
[7] S. Even and G. Even. *Graph Algorithms, Second Edition*. Cambridge University Press, 2012.
[8] R. Finkel and J. Bentley. Quad Trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, 4:1–9, 1974.
[9] A. Gordon. A review of hierarchical classification. *Journal of the Royal Statistical Society: Series A (General)*, 150(2):119–137, 1987.
[10] D. Guo and M. Gahegan. Spatial ordering and encoding for geographic data mining and visualization. *Journal of Intelligent Information Systems*, 27(3):243–266, 2006.
[11] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *Proc. 1984 SIGMOD*, pp. 47–57, 1984.
[12] D. Hilbert. Über die Stetige Abbildung Einer Line auf ein Flächenstück. *Mathematische Annalen*, 38(3):459–460, 1891.
[13] D. Jäckle, F. Fischer, T. Schreck, and D. A. Keim. Temporal MDS plots for analysis of multivariate data. *IEEE TVCG*, 22(1):141–150, 2016.
[14] R. Jarvis and E. Patrick. Clustering Using a Similarity Measure Based on Shared Near Neighbours. *IEEE Transactions on Computers*, 22(11):1025–1034, 1973.
[15] W. Köpp and T. Weinkauf. Temporal Treemaps: Static Visualization of Evolving Trees. *IEEE TVCG*, 25(1):534–543, 2019.
[16] J. Kruskal. Multidimensional Scaling by Optimizing Goodness of fit to a Nonmetric Hypothesis. *Psychometrika*, 29(1):1–27, 1964.
[17] S. Liu, Y. Wu, E. Wei, M. Liu, and W. Liu. StoryFlow: Tracking the Evolution of Stories. *IEEE TVCG*, 19(12):2436–2445, 2013.
[18] H. Lu and B. C. Ooi. Spatial Indexing: Past and Future. *IEEE Data Engineering Bulletin*, 16(3):16–21, 1993.
[19] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426*, 2018.
[20] W. Meulemans, K. Verbeek, and J. Wulms. Stability Analysis of Kinetic Orientation-based Shape Descriptors. *arXiv:1903.11445*, 2019.
[21] C. Muelder, B. Zhu, W. Chen, H. Zhang, and K. Ma. Visual analysis of cloud computing performance using behavioral lines. *IEEE TVCG*, 22(6):1694–1704, 2016.
[22] J. Orford. Implementation of criteria for partitioning a dendrogram. *Journal of the IAMG*, 8(1):75–84, 1976.
[23] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.
[24] I. Piljek. VisSwarmR: Visual Analytics tool for the generation of Collective Movement Datasets, 2020. `https://github.com/piljek/VisSwarmR`.
[25] P. Rauber, A. Falcão, and A. Telea. Visualizing Time-Dependent Data Using Dynamic t-SNE. In *Proc. 18th EuroVis*, pp. 73–77, 2016.
[26] C. W. Reynolds. Steering Behaviors For Autonomous Characters. In *Proc. Game Developers Conference*, pp. 763–782, 1999.
[27] B. Rieck and H. Leitte. Persistent homology for the evaluation of dimensionality reduction schemes. *CGF*, 34(3):431–440, 2015.
[28] J. Sammon. A Non-linear Mapping for Data Structure Analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.
[29] J. Tenenbaum, V. De Silva, and J. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.
[30] S. van den Elzen, D. Holten, J. Blaas, and J. J. van Wijk. Dynamic Network Visualization With Extended Massive Sequence Views. *IEEE TVCG*, 20(8):1087–1099, 2014.
[31] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.
[32] T. van Dijk, M. Fink, N. Fischer, F. Lipp, P. Markfelder, A. Ravsky, S. Suri, and A. Wolff. Block Crossings in Storyline Visualizations. *Journal of Graph Algorithms and Applications*, 21(5):873–913, 2017.
[33] J. Wenskovitch, I. Crandell, N. Ramakrishnan, L. House, S. Leman, and C. North. Towards a systematic combination of dimension reduction and clustering in visual analytics. *IEEE TVCG*, 24(1):131–141, 2018.
[34] U. Wilensky. Netlogo flocking model. `http://ccl.northwestern.edu/netlogo/models/Flocking/`, 1998.
[35] U. Wilensky. Netlogo. `http://ccl.northwestern.edu/netlogo/`, 1999.
[36] L. Zhou, C. R. Johnson, and D. Weiskopf. Data-driven space-filling curves. *arXiv:2009.06309*, 2020.