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
We chose to "stabilize" PCA since its principal component gives stable whole temporal ranges, that is, they can be adapted to be stable. We conclude that stable dimensionality reduction outperforms existing methods on stability, without sacrificing spatial quality or efficiency. However, they directly extend to three or higher dimensions.

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.

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 method [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, particularly well suited for data sets that exhibit clear clusters. For this reason, we describe our approaches in two-dimensions, 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 dimensiona-
when the point set is not stretched. The intuition is as follows. If 

\[ t = 0 \]

the projection vector \( PV \) for all time steps onto the computed projection vectors \( PV \). Finally, we can project the point sets for all 

\[ \alpha \]

consistent with \( PV \) not. For the time steps 

\[ v \]

if a point set is stretched, we use the corresponding eigenvalues 

\[ \sigma \]

mented as follows (see Algorithm 1 for an overview). To determine 

\[ \sigma \]

orderings of points within a cluster are then placed around the pro-

\[ \sigma \]

a vector in the direction along which the point set 

\[ \sigma \]

has most variance. Projecting onto this vector maxi-

\[ \sigma \]

unstable behavior when the point set is not stretched, that is, the 

\[ \sigma \]

spatial quality and stability of orientation-based shape descriptors, 

\[ \sigma \]

are consistent with the variance along the second principal component, then the 

\[ \sigma \]

than the variance along the first principal component, then the 

\[ \sigma \]

is clearly higher than the variance along the second principal component, then the 

\[ \sigma \]

direction is very discriminative: the point set is clearly stretched 

\[ \sigma \]

direction is straightforward to determine the most consistent direction. 

\[ \sigma \]

that is most 

\[ \sigma \]

we use the projection vector used by the majority of the points in the 

\[ \sigma \]

set. Based on this technique, we describe two algorithms that make 

\[ \sigma \]

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] 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 inter-

\[ \sigma \]

Concretely, the Stable Principal Component algorithm is imple-

\[ \sigma \]

mented as follows (see Algorithm 1 for an overview). To determine 

\[ \sigma \]

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 

\[ \sigma \]

For the time steps \( t \) where the point set is stretched (including 

\[ \sigma \]

we simply compute the principal component as projection vector \( PV[t] \). Note that \( -PV[t] \) is equally good as projection vector, but results in a mirrored representation. To avoid 

\[ \sigma \]

if following, we therefore use the direction \( (PV[t] \) or \( -PV[t] \)) that is most consistent with \( PV[t-1] \) (computed using the dot product). For time steps \( t \) where the point set is not stretched we also first compute the 

\[ \sigma \]

We use these vectors to keep track of the signed angle \( \alpha \) 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 \( \alpha \) to uniquely interpolate the projection vector. Finally, we can project the point sets for all 

\[ \sigma \]

Algorithm 1 StabilePrincipalComponent(P, \( \sigma \))

Input: Point set \( P \) over \( T \) time steps, and \( \sigma \in [0,1] \)

Output: Visual summary \( S \) for \( P \)

1: Set \( PV[1] \) 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 \( PV[t] \) to the first principal component of \( P(t) \) and compute eigenvalues \( v_1, v_2 \)
5: Add the signed angle between \( PV[t] \) and \( PV[t-1] \) to \( \alpha \)
6: if \( v_2/v_1 \leq \sigma \) or \( t = T \) then
7: \( \alpha_t = t' + 1 \) to \( t \) - 1 do
8: Set \( PV[t'] \) rotated over \( \alpha \). \( t' = t \)
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 \( PV[t] \)
12: return \( S \)

Since the eigenvalues and principal components of \( n \) points in 

\[ \sigma \]

2D can be computed in \( O(n) \) time, it is easy to see that the entire 

\[ \sigma \]

algorithm runs in \( O(nT) \) time. The explicit trade-off between spatial quality and stability can be configured via parameter \( \sigma \). If \( \sigma \) 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 \( \sigma \) 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] 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 Stabe-

\[ \sigma \]

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 and a clustering algorithm (such hybrids have also been explored in [24]).

Intuitively, this algorithm performs SPC on the separate clusters. More specifically, for every frame we first perform Complete Link-

\[ \sigma \]

age 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 on the individual clusters, with two small adaptations, resulting in projection vectors \( PV{C[t]} \) for a cluster \( C \). First, we end the linear interpolation of \( PV{C[t]} \) 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 (\( PV{C[t]} \) or \( -PV{C[t]} \)) 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
We also include a baseline algorithm [FXD] that is solely focused on 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 linear order is computed from the cluster hierarchy as follows.

### 4 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 clustering. Fig 3 [A] 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 Appendix [A].

**[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, [ZOR] Z-order curve, [PQR] Point Quadtree, and [RTR] R-tree.

**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.

### 4.1 Algorithms

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.

**4.1.1 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 [32], 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].

### 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) ∈ P denote the jth nearest neighbor of p_i in P, for each j with 1 ≤ j ≤ 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

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

where w(i,j) denotes the weight or importance of maintaining the jth 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 Appendix [A] 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, $S_i$ and $S_{i+1}$ for each $i$ with $1 \leq i < T$. 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.

\[ \text{[JMP]} \text{ Jump distance.} \]

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

\[ \text{JMP}(P, S) = \sum_{p_i \in P} |S_i(p_i) - S_{i+1}(p_i)| \]

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

\[ \text{[CRS]} \text{ Crossings.} \]

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 $S_i(p_i) < S_i(p_j)$ and $S_j(p_i) > S_j(p_j)$. The metric $\text{CRS}(P, S)$ lies between 0 (perfectly stable) and $\frac{n(n-1)}{2}$ (complete inversion of the order).

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

\[ \text{[KSte]} \text{ Temporal Keys Similarity.} \]

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 $\text{KSte}(P, S)$, 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 [36] 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. 18 in the appendix 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” [4] for certain ordering methods, most notably HIL, PQR, and SNEp (see Fig. 9). 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 [35] 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 Pijlje [24] is public.

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

5 EXPERIMENTAL RESULTS

Here we report on the results of the quantitative experiments described in Section 3. Tables 1 and 2 in Appendix C 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. 3 and 7 show visual summaries for all algorithms for Fish 1. The MotionRugs are accompanied by a visualization of the mean Kstdi 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 Kstdi, 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 Kstdi has a minimal edge over KSra. As expected
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 $\sigma$ 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. 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 $\sigma$ of SPC and its effect on the results. We run SPC for 101 different values for $\sigma$ 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 $\sigma$ is 0.95, while the lowest is 0.29. Values above

![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.](image)

![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.](image)

![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.](image)
and below these extremes are identical to results with 0.95 and 0.29 respectively. The $\sigma$ 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 $\sigma$ 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 $\sigma$ 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 $\sigma$ drops below 0.3, 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 $\sigma$ 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. 8 specifically shows an instability that occurs in Fish 1, using the same intermediate values for $\sigma$ 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 $\sigma$ 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 $\sigma$ becomes too low: when we interpolate over too many frames at once, spatial quality will drastically deteriorate, as seen for $\sigma = 0$.

**Trade-offs.** Our main goal is to investigate the trade-off between spatial quality and stability. Fig. 8 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
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. 9 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 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 Appendix E).

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 Appendix E.

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 $\sigma$ 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 $\sigma$ 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 $\sigma$. While this is counter-intuitive in general, it can be explained for this data set. As $\sigma$ 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
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 $\sigma$ 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 $0.45 < \sigma < 0.59$. Increasing $\sigma$ 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 why more interpolation (for decreasing $\sigma$) 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 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 this. 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
think of as “MotionLines”, which combine ideas from MotionRugs [2] and Story Lines [33]. As shown in Fig. [3] 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 Netherlands eScience Center (NLeSC); grant 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 Transactions on Visualization and Computer Graphics*, 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. In *Proc. 11th EuroVis Workshop on Visual Analytics (EuroVA)*, 2020.

[4] M. Burch, C. Vehlow, S. Diehl, and D. Weiskopf. Parallel Edge Splatting for Scalable Dynamic Graph Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 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 Pacific Visualization Symposium*, 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 Transactions on Visualization and Computer Graphics*, 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. Gutman. 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 Linie 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 Transactions on Visualization and Computer Graphics*, 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 Visualizing of Evolving Trees. *IEEE Transactions on Visualization and Computer Graphics*, 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 Transactions on Visualization and Computer Graphics*, 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 preprint arXiv:1802.03426*, 2018.

[20] W. Meulemans, K. Verbeek, and J. Wulms. Stability Analysis of Kinetic Orientation-based Shape Descriptors. *arXiv preprint arXiv:1903.11445*, 2019.

[21] C. Mueller, B. Zhu, W. Chen, H. Zhang, and K. Ma. Visual analysis of cloud computing performance using behavioral lines. *IEEE Trans. Vis. Comput. Graph.*, 22(6):1694–1704, 2016.

[22] J. Orford. Implementation of criteria for partitioning a dendrogram. *Journal of the International Association for Mathematical Geology*, 8(1):75–84, 1976.

[23] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 11(5):559–572, 1901.

[24] I. Piljek. VisSwarmR: Visual Analytics tool for the generation of Collective Movement Datasets, 2020. [https://github.com/piljek/VisSwarmR](https://github.com/piljek/VisSwarmR).

[25] P. Rauber, A. Falcão, and A. Telea. Visualizing Time-Dependent Data Using Dynamic t-SNE. In *Proc. 18th Eurographics Conference on Visualization (EuroVis)*, pp. 73–77, 2016.

[26] C. W. Reynolds. Steering Behaviors For Autonomous Characters. In *Proceedings of Game Developers Conference. San Francisco, California*, pp. 763–782. Miller Freeman Game Group, 1999.

[27] B. Rieck and H. Leitte. Persistent homology for the evaluation of dimensionality reduction schemes. *Computer Graphics Forum*, 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] A. Telea. Personal communication, March 2019.

[30] J. Tenenbaum, V. De Silva, and J. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.

[31] S. van den Elzen, D. Holten, J. J. van Wijk. Dynamic Network Visualization With Extended Massive Sequence Views. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1087–1099, 2014.

[32] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[33] T. van Dijik, M. Fink, N. Fischer, F. Lipp, P. Markfelder, A. Ravskey, S. Suri, and A. Wolff. Block Crossings in Storyline Visualizations. *Journal of Graph Algorithms and Applications*, 21(5):873–913, 2017.

[34] 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 Transactions on Visualization and Computer Graphics*, 24(1):131–141, 2018.

[35] U. Wilensky. Netlogo flocking model. [http://ccl.northwestern.edu/netlogo/models/Flocking](http://ccl.northwestern.edu/netlogo/models/Flocking), 1998.

[36] U. Wilensky. Netlogo. [http://ccl.northwestern.edu/netlogo](http://ccl.northwestern.edu/netlogo), 1999.
Figure 13: A MotionLines and a MotionRug of the clustered data set, using 1D representations produced by our CPC algorithm ($\sigma = 0.5$).

A  COMPUTING 1D ORDERINGS

In addition to dimensionality reduction, various other techniques can be found in literature that are used to compute linear orders for a set of points. We compare dimensionality-reduction techniques and our new adaptations not only to each other, but also to several of such existing algorithms. Most of these algorithms are not designed to be stable and typically consider different time steps in isolation. To select suitable algorithms for our comparison, we choose the algorithms that performed best in the experiments by Guo and Gahegan [10] and Buchmüller et al. [2]. We may classify these remaining algorithms based on how they compute a linear order: (1) via spatial subdivisions; (2) via clustering. Finally, we also include a baseline algorithm that is solely focused on stability. Figure 14 shows an example of the orderings generated by a selection of the algorithms, including the dimensionality-reduction techniques, for one time step of our test data for reference.

Figure 14: Orderings for one time step generated using linear (SPC) and nonlinear (SNE) dimensionality reduction, space-filling curves (HIL) and clustering (CLC).

[HIL] Hilbert curve and [ZOR] Z-order curve.
The Hilbert curve [12] is a continuous space-filling curve. It can be applied to cover a spatial region in arbitrary precision by repeating the construction pattern recursively. A set of points in space can then be linearized by sampling the curve and noting the order in which the points are encoded on the curve. Another representative of space-filling curves is the Z-order curve, which differs from the Hilbert curve in its geometrical construction pattern resembling a Z shape, where the space is partitioned in four quadrants in the order NW, NE, SW, SE (see figure on the right). Both approaches differ in neighborhood retention and construction complexity, as Lu and Ooi describe [18]. In their comparison, Guo and Gahegan [10] found that Hilbert curves avoid long jumps better than the Z-Order curve, which in turn outperforms the Hilbert curve in the average of the compared metrics. Since both produce visually different outcomes, we include both strategies in the comparison.

[PQR] Point Quadtree.
Quadtrees [8] partition space recursively in four parts, until each part contains only a single point. Consequently, sparse areas cause fewer splits than dense areas. Standard quadtrees divide the space in equal parts, while Point Quadtrees split at an input point and thus potentially unevenly in terms of area. To derive the 1D ordering, a depth-first tree-iteration strategy is used; given the neighborhood structure in the tree, this is more suitable than a breadth-first strategy. See [7] for details on tree-iteration strategies. The standard quadtree essentially reflects a Z-Order curve linearization if the same quadrant iteration is applied. Hence, we use the point quadtree variant which produces different orderings due to the intermittent partition.

[RTR] R-tree.
In R-Trees [11] objects are stored recursively in minimum bounding rectangles (MBR). Each MBR can hold at most a predefined number of objects, thus ensuring a minimum fill. In comparison to quadtrees, more complex balancing is necessary, recomputing the MBRs, when the object limit is reached. Note that MBRs can overlap. Again, a depth-first iteration strategy is used to order points in an R-Tree.

A.1 Spatial Subdivisions

Several well-known linearization approaches, which are primarily used for spatial-indexing applications, are based on the principle of iterating through some spatial subdivision. These approaches encompass tree data structures and space-filling curves. We focus on four established, representative techniques from this area, though many variations exist; see [18] for an overview.

[FXD] Fixed Order.
This algorithm outputs the same arbitrary linear order for every time step and hence serves as reference baseline for our experiments. With FXD, each horizontal line always represents the same moving entity.

[18]
algorithms we consider below, we always use the following method to compute the linear order from the clustering.

The hierarchical clustering is represented by a tree with the individual points stored in the leaves. We aim to order to leaves of such a tree without changing the cluster structure: we can change only the order of the children of any internal node. We follow the algorithm by Bar-Joseph et al. [1] to compute the order that minimizes the length of the path formed by visiting the input points in that order. The algorithm uses dynamic programming to efficiently find the optimal order for every subtree placing two specific points at the first and last position in the order.

**[CLC] Complete Linkage Clustering.** Initially, every point is considered as a separate cluster to be hierarchically merged in a bottom-up fashion [2]. We do so by repeatedly merging the closest two clusters, until we obtain a single cluster. Distance between clusters is measured as the distance between their farthest points.

**[SNN] Shared Nearest Neighbors.** This clustering algorithm [14] works the same as CLC, but it uses a different metric than Euclidean distance to measure the dis(similarity) between two points. For two points \(p\) and \(q\), we first count the number of points \(x\) that are in the set of \(k\) nearest neighbors for both \(p\) and \(q\). We then define the shared nearest neighbor (SNN) distance between \(p\) and \(q\) as \(1/(x+1)\). The SNN clustering is computed using SNN distance instead of Euclidean distance. In case of ties in SNN distance, we use Euclidean distance to break ties. In our experiments we use \(k = 10\).

### A.3 Gradient-descent methods for dimensionality reduction

Many dimensionality-reduction techniques define a cost function to describe the spatial quality of the resulting representation, and aim to minimize that function. For example, Sammon mapping uses a function that measures how well distances are preserved, while t-SNE uses a function that measures how well local neighborhoods are preserved. Since finding the global minimum of such a cost function is typically hard, they often use local search heuristics (usually gradient descent) to find a good solution. In our experiments we consider two such dimensionality-reduction techniques: Sammon mapping and t-SNE. There are other dimensionality-reduction techniques, such as MDS [16] and Isomap [30], but based on their cost functions we believe that they give similar results (in fact, in the Euclidean plane, classical MDS is equivalent to PCA). We first review Sammon mapping and t-SNE for a static point set, before explaining our adaptations for improved stability.

**[SAM] Sammon Mapping.** Sammon mapping [28] aims to preserve distances. Let \(d_{ij}\) denote the Euclidean distance between points \(p_i\) and \(p_j\), denote the resulting (1D) coordinates by \(x_i\), and let \(\delta_{ij} = |x_i - x_j|\). Sammon mapping computes coordinates \(x_i\), attempting to minimize this cost function:

\[
C = \frac{1}{\sum_{i \leq l \leq j} d_{ij} \sum_{1 \leq l \leq j} (d_{ij} - \delta_{ij})^2} d_{ij}
\]

The cost \(C\) is then minimized using a gradient descent starting from a random initial solution.

**[SNE] t-Distributed Stochastic Neighbor Embedding.** The goal of t-SNE [12] is to preserve local neighborhoods in the dimensionality reduction. Again, let \(d_{ij}\) denote the Euclidean distance between points \(p_i\) and \(p_j\). Similarities between points are captured by a probability distribution:

\[
P_{ij} = \frac{\exp(-d_{ij}^2/(2\sigma_i^2))}{\sum_{k \neq i} \exp(-d_{ik}^2/(2\sigma_i^2))}
\]

The values \(\sigma_i\) are chosen depending on the predefined perplexity \(\kappa\) (see [22] for details); in our experiments we use \(\kappa = 40\). We further define \(Q_{ij} = \frac{1}{d_{ij}} (P_{ij} + P_{ji})\) and we set \(P_{ij} = 0\) if \(i = j\). Denote the resulting (1D) coordinates by \(x_i\), and define \(\delta_{ij}\) as

\[
\delta_{ij} = (1 + |x_i - x_j|^2)^{-1}
\]

The cost function is defined by the Kullback-Leibler divergence as:

\[
C = \sum_{i \neq j} P_{ij} \log \frac{P_{ij}}{Q_{ij}}
\]

Finally, the cost \(C\) is again minimized using a (momentum-based) gradient descent starting from a random initial solution.

**Stability improvements.** Both Sammon mapping (SAM) and t-SNE (SNE) start the gradient descent with a random initial solution, which may result in low 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 a random other local minimum.

Recently, Rauber et al. [25] described Dynamic t-SNE: a more explicit way of making t-SNE stable over multiple time steps. Their approach performs a global optimization over all time steps simultaneously, using a separate copy of each point for each time step. They enforce temporal coherence by adding a term to the optimization function depending on the distance between two copies of the same point at consecutive time steps. For two reasons we were not able to include this algorithm in our experiments. First, it is very slow. The paper reports a running time of about 6 minutes per time step. Although a single time step of our data consists of only hundreds of points, we consider thousands of time steps, making their algorithm prohibitively slow for our experiments. Second, the implementation of Dynamic t-SNE rarely gives meaningful output when run on our data [1], which changes much faster than the data experimented on in [25]. We further believe that Dynamic t-SNE would converge slowly on a time-varying data set with many time steps: it would take at least \(T\) gradient descent iterations for frames that are \(T\) time steps apart to affect each other. Since t-SNE is already known to converge quite slowly, the combination may simply require too many iterations to obtain a reasonable solution. Thus, Dynamic t-SNE exacerbates the usual downsides of t-SNE, namely black-box parameter tuning and slow convergence.

### B Metrics

**[KSra] Rank-weighted Keys Similarity.** We define \(w(i, j) = 1/j\) inversely proportional to the rank, such that maintaining the closest neighbors is considered more important than the more distant neighbors. We used the existing implementation at https://github.com/lejon/T-SNE-Java to compute the t-SNE mapping. This implementation uses approximations to speed up the computation, which lead to artifacts in our results. We therefore implemented the default version of t-SNE ourselves. See Appendix F for more details.

\footnote{We verified that this is a known problem with the implementation.}
neighbors. With $H_k$ denoting the $k^{th}$ harmonic number, we obtain:

$$KS_{ra}(P, S) = \frac{\sum_{p_i \in P} \sum_{j=1}^{k} r(i, j)/j}{\sum_{p_i \in P} \sum_{j=1}^{k} 1/j} = \frac{\sum_{p_i \in P} \sum_{j=1}^{k} r(i, j)/j}{n \cdot H_k}$$

$[KS_{di}]$ 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 considered more important than distant neighbors. In contrast to KSra, this variant does not treat neighbors at (nearly) identical distances differently.

$$KS_{di}(P, S) = \frac{\sum_{p_i \in P} \sum_{j=1}^{k} r(i, j)/\|p_i - n(i, j)\|}{\sum_{p_i \in P} \sum_{j=1}^{k} 1/\|p_i - n(i, j)\|}$$

C  EXPERIMENTAL EVALUATION Fish 2

This section gives an in-depth explanation of the results found in the experimental evaluation for Fish 2. In this data set, the fish start moving as a single cluster, after which they split into two separate clusters, and eventually merge into a single cluster again. We investigate the spatial quality and stability in isolation, followed by an exploration of the parameter for SPC. We conclude this section with discussion of the trade-off between spatial quality and stability. All results can be found in Figure 15.

Spatial quality. In Figure 15a we can see the spatial quality measurements for Fish 2, which are nearly identical to the results for Fish 2. Even though the movement patterns of the fish are quite different in the two excerpts, this does not seem to affect the overall spatial quality of the resulting orderings. One small but notable difference in the results is that SNEp no longer has a higher KSra than KSd. We can see KSd measurements for Fish 2.

Stability. A similar overview of the stability for this data set can be found in Figure 15b. Again we see very similar results in comparison to Fish 1. Differences can be seen for SAM, and the spatial subdivision and clustering techniques. These have both higher absolute values for all stability measures, as well as higher values relative to the other techniques.

Parameter experiment. The results of the parameter experiment can be found in Figures 15c and 15d. We choose the intermediate values of the parameter $\sigma$ to be $a = 0.35$, $b = 0.49$ and $c = 0.65$ for Fish 2. The cut-off values are 0.30 and 0.86, meaning all values below and above the respective cut-off results in the same summaries. While the plotted measurements for all values of $\sigma$ look very similar to those of Fish 1, there are some small differences. We choose the intermediate values to be in similar positions in the plot, so that they are nicely spread over all 101 measurements. However, the values are different than for Fish 1, which indicates that we need different parameter values to achieve similar spatial quality and stability. Overall we see that the maximum values for KSte are also a lot higher for Fish 1. This is probably due to the fact that in Fish 1 the fish becomes very rounded, hence forcing the first principal component changes quickly. The difference in intermediate values is therefore not unexpected: Even though the stability measurements of Fish 2 are very similar to the measurements for Fish 1, individual instabilities require different parameter settings to result in stable orderings.

Trade-offs. The trade-off between spatial quality and stability measures is shown in Figures 15d and 15e. As we already saw in the overall stability, SAM performs a lot worse on Fish 2, compared to Fish 1. Furthermore, CPC performs a lot worse on maximum stability. This is due to the fact that the fish in Fish 2 split into different clusters. The changes in the clustering of CPC will cause big changes in the ordering, leading to instability at the few frames where the clustering changes. Finally we can see that SNE performs better, relative to its performance on Fish 1, especially for maximum values of KSte.

Visual summary. In our experiments we use two excerpts from a large fish data set. To put the two excerpts in perspective, we show a visual summary of the full data set in Figure 18. The visual summary uses 1D orderings and is generated using SPC with $\sigma = 0.5$. Fish 1 is located at the start of the data set, while Fish 2 is taken close to the end of the data set.

D  EXPERIMENTAL EVALUATION Netlogo

We examine the statistics for the Netlogo data set used in our experiments in more detail in this section. Again, we first consider the spatial quality and stability separately, followed by the parameter exploration for SPC. Finally we discuss the trade-off between spatial quality and stability as observed on Netlogo. Charts of all statistics can be found in Figure 16.

Spatial quality. As can be seen in Figure 16a, while the absolute values for Netlogo are higher than for Fish 1 and Fish 2, the relative values are very similar. The fixed order gives very bad results on spatial quality, followed by SNEp and RTR. The spatial subdivision techniques all perform similarly, and are slightly better than the previously mentioned techniques. Of the remaining algorithms, the clustering techniques (CLC and SNN) perform slightly worse than all remaining dimensionality reduction techniques (SAM, SAMp, SNE, SPC and CPC). For Netlogo we have chosen different parameter values for SPC, specifically $a = 0.40, b = 0.59$ and $c = 0.62$.

Stability. In Figure 16b we plot the stability statistics for Netlogo. While the chart looks quite different from the stability chart for Fish 1 and Fish 2, this is mostly due to the fact that JMP and CRS count the absolute number of changes in the orders, whereas KSte is normalized. Since Netlogo behaves less stable than Fish 1 and Fish 2, all metrics show higher values. However, Netlogo also contains more moving entities, which increases the absolute number of changes even further.

Comparing the statistics of JMP and CRS, we see very similar performances of all algorithms, with SNEp being the least stable, while SAMp is the most stable. On Fish 1 and Fish 2 SNE was the least stable method overall, and while it still has the highest number of absolute changes on Netlogo, it performs a lot better according to KSte, meaning neighborhoods are preserved relatively well over time. Now we see that RTR performs worst on KSte, followed by the space-filling curves (HIL and ZOR) and the clustering algorithms (CLC and SNN) together with PQR. The SPC variants perform relatively worse than on Fish 1 and Fish 2, with parameter values close to but lower than 1 being optimal for stability. Finally SAM, SAMp, SNEp, and CLC are the most stable according to KSte.

Parameter experiment. As already explained in the main text, the results of the parameter experiment are slightly different for Netlogo. In Figures 16c and 16d the results are plotted. For Netlogo, the cut-off values are 0.76 and 0.32, so everything above 0.76 uses exactly the first principal component per frame, and similarly for values below 0.32 we always interpolate between the first and last frame. The parameter values that are indicated by labels in the figures are the values we used in our other experiments ($a = 0.40, b = 0.59$, and $c = 0.62$). Note that there are two blue labels, which represent other values of interest that we will use in the analysis below.

We will consider the results between the values indicated by black labels in the figures. Starting from the lowest parameter value 0.32 we see that increasing $\sigma$ has chaotic effects on both spatial quality and stability up to 0.40 where this fickle behaviour ends. On closer inspection, the values between 0.32 and 0.40 constantly pick up more frames where the entities are stretched enough to use the actual first principal component. Since the intervals between which interpolation happens, constantly change, the results do not steadily change, but are quite erratic. Parameter value 0.38 shows the worst
combination, having both bad spatial quality and stability (max and mean).

From 0.40 to 0.45 we see a steady decrease in stability and increase in spatial quality, as expected when increasing $\sigma$. Further increasing the parameter to 0.59 has negative effects on both the spatial quality and stability. On closer inspection, this increase in spatial quality can be attributed to properties of Netlogo. During instabilities we can 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. While we expect projections to the first principal component to have high spatial quality, it is not always the case, as we see here. In Netlogo this occurs when the cluster of points changes direction and shortly does not form a convex shape. It is therefore not unreasonable that interpolating less (and using first principal component more) when increasing $\sigma$ from 0.46 to 0.56 can negatively effect spatial quality.

At 0.61 SPC splits the interpolation over the two consecutive instabilities that were seen as one big instability. This split improves both spatial quality and stability, up to 0.62 where there are a couple of non-interpolation frames between the instabilities. Increasing the parameter further leads to certain instabilities not being interpolated over any longer, which negatively affects the maximum KSte values observed for those runs of SPC.

**Trade-offs.** For Netlogo the trade-offs between spatial quality and stability can be found in Figures [16c] and [16d]. When considering the mean values for KSte and KStd, we see a similar spread as before: SPC and SPC variants along with CPC are in the bottom left corner, SNEp and FXD have worse spatial quality but good stability, while the remaining techniques (spatial subdivision, clustering and SNE) have relatively good spatial quality but bad stability. However, SNE is more stable than we have seen for Fish 1 and Fish 2, outperforming all subdivision and clustering techniques, except for PQR and CLC. For the maximum values of KSte, we again see CPC perform worse than on Fish 1. Netlogo mostly consists of a single cluster, but the occasional outlier can trigger the clustering in CPC to find different clusters, resulting spikes of instability.

## E Experimental evaluation Reynolds

The results of our experimental evaluation for Reynolds are presented in this section. The structure of this section is similar to the previous two sections, first considering spatial quality and stability in isolation, followed by the parameter experiment. We end the section by considering the trade-off between spatial quality and stability. Charts of all statistics can be found in Figure [17].

**Spatial quality.** The chart in Figure [17a] shows the spatial quality for all techniques. Familiar patterns can be found, FXD, RTR, and SNEp are the worst performers, but are this time joined by PQR as another technique that gives relatively worse spatial quality. The other algorithms all score relatively close on spatial quality, in order of increasing spatial quality (and decreasing measure values), the other spatial subdivision techniques are followed by SPC variants, the clustering techniques and finally SNE, SAM, SAMp and CPC. Since there are multiple cluster in this data, which do not interact with each other, it is not surprising that clustering techniques and CPC perform so well.

**Stability.** The results on the stability of Reynolds are shown in Figure [17b]. These results are again very similar to the results on Fish 1 and Fish 2, with the exception that clustering techniques and SNE perform better in comparison to the other techniques. Since the clusters do not interact and contain the same points in every frame, the clustering techniques also perform very well on stability. Especially when considering the KSte we see 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.

**Parameter experiment.** The parameter experiment also gave some surprising results for Reynolds, as already explained in the main text. Figures [17c] and [17d] show 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 exactly the 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 and Fish 2.

Between values 0.59 and 0.50 we see that lowering $\sigma$ improves both the spatial quality as well as the stability, just as we saw for some parameters in Netlogo. 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 $\sigma$ further results in worse spatial quality and stability, as we saw in all other data sets.

Finally between 0.98 and 0.86 we see better stability when increasing $\sigma$. While this is counter intuitive in general, it can be explained for this data set. As $\sigma$ increases we interpolate less and 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 many changes in the neighborhood of all the points in those two clusters. If we interpolate less, this behaviour is prominent and contained in a few frames, leading to less instability according to KSte.

**Trade-offs.** The trade-offs between spatial quality and stability for Reynolds can be observed in Figures [17d] and [17e]. 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 $\sigma$ values, SAMp, SNEp and CLC, when it comes to stability. For $\sigma = 0.50$ SPC performs particularly well, even better than SAMp and CLC on maximum KSte. However SAMp and CLC also have very good spatial quality, making them the best techniques for this data set.

## F Summary Statistics

The two tables on the next page provide the statistics over all time steps, for each metric and on every algorithm in our experiment. Each table shows the results for one of the data sets, the fish data set and Netlogo data set respectively. The data in the tables is used throughout the paper in a variety of charts and diagrams.

In Appendix A.3 we explained that t-SNE was implemented from scratch, using the simplest form of the algorithm. There are extensions that approximate the gradient during gradient descent to improve run times. These extensions are integrated in most libraries that are currently available. When using the libraries we indeed saw faster run times for both SNE and SNEp, but stability was influenced by the approximations used in the libraries. We therefore chose to implement t-SNE from scratch to show its true capability to produce stable visual summaries. While this lead to slower run times than one would expect from state-of-the-art t-SNE, the run times of the libraries still greatly exceeded the run times observed for the other algorithms in our experiments.
Figure 15: Quantitative evaluation of Fish 2.
(a) The two quality metrics: mean for KSra (left) and KSdi (right column) for all algorithms over all Netlogo data set frames.

(b) The three stability metrics: mean for JMP, CRS (left), and KSra (right) for all algorithms over all Netlogo data set frames.

(c) A comparison between the mean for KSra and for KSdi, for uniformly distributed \( \sigma \) of SPC on the Netlogo data set.

(d) A comparison between the mean for KSra and for KSdi for all algorithms on the Netlogo data set.

(e) A comparison between the max for KSra and the mean for KSdi, for uniformly distributed \( \sigma \) of SPC on the Netlogo data set.

(f) A comparison between the max for KSra and the mean for KSdi for all algorithms on the Netlogo data set.

Figure 16: Quantitative evaluation of the Netlogo data set.
(a) The two quality metrics: mean for KSra (left) and KSdi (right column) for all algorithms over all Reynolds frames.

(b) The three stability metrics: mean for JMP, CRS (left), and KSte (right) for all algorithms over all Reynolds frames.

(c) A comparison between the mean for KSte and for KSdi, for uniformly distributed σ of SPC on Reynolds.

(d) A comparison between the mean for KSte and for KSdi for all algorithms on Reynolds.

(e) A comparison between the max for KSte and the mean for KSdi, for uniformly distributed σ on Reynolds.

(f) A comparison between the max for KSte and the mean for KSdi for all algorithms on Reynolds.

Figure 17: Quantitative evaluation of Reynolds.
Figure 18. Visual summary of the whole fish data set, cut into 6 pieces to fit on a single page. Fish 1 is taken at the start of the first line, while Fish 2 can be recognized near the end of the last line.
Table 1: Statistics on Fish 1 for all algorithms and all metrics, including run time in seconds.

| Run time (s) | KNe ·   | CBS ·   | MIP ·   | Spatial Quality · |
|--------------|---------|---------|---------|-------------------|
|              | Mean    | Min     | Max     | Mean              |
| 1.09         | 1.15    | 0.48    | 0.27    | 1.12              |
| 1.14         | 1.16    | 1.11    | 1.18    | 1.15              |
| 2.8        | 3.42    | 2.81    | 3.47    | 3.39              |
| 1.17         | 1.18    | 1.17    | 1.18    | 1.16              |
| 2.47         | 3.37    | 2.62    | 3.54    | 3.46              |
| 2.17         | 2.19    | 2.16    | 2.26    | 2.21              |
| 2.18         | 2.19    | 2.16    | 2.26    | 2.21              |
| 2.18         | 2.19    | 2.16    | 2.26    | 2.21              |
| 2.18         | 2.19    | 2.16    | 2.26    | 2.21              |
| 2.18         | 2.19    | 2.16    | 2.26    | 2.21              |
Table 2: Statistics on Fish 2 for all algorithms and all metrics, including run time in seconds.

| Run time (s) | KNearest | CBS* | Stability | IMP* | Spatial Quality | KNN* |
|--------------|-----------|------|-----------|------|-----------------|------|
| 0.0          | 0.0       | 0.0  | 0.0       | 0.0  | 0.0              | 0.0  |
| 0.1          | 0.1       | 0.1  | 0.1       | 0.1  | 0.1              | 0.1  |
| 0.2          | 0.2       | 0.2  | 0.2       | 0.2  | 0.2              | 0.2  |
| 0.3          | 0.3       | 0.3  | 0.3       | 0.3  | 0.3              | 0.3  |
| 0.4          | 0.4       | 0.4  | 0.4       | 0.4  | 0.4              | 0.4  |
| 0.5          | 0.5       | 0.5  | 0.5       | 0.5  | 0.5              | 0.5  |

Note: The table includes statistical data for Fish 2 across various algorithms and metrics, with run times measured in seconds.
Table 3: Statistics on Netlogo for all algorithms and all metrics, including run time in seconds.

| RUN TIME(s) | SNSN | SMDP | SMD | CCF | CTFP | CTFH | CTFH + | CTFH + + | CTFH + + + |
|-------------|------|------|-----|-----|------|------|--------|---------|-----------|
| 0.000       |      |      |     |     |      |      |        |         |           |
| 0.050       |      |      |     |     |      |      |        |         |           |
| 0.100       |      |      |     |     |      |      |        |         |           |
| 0.150       |      |      |     |     |      |      |        |         |           |
| 0.200       |      |      |     |     |      |      |        |         |           |
| 0.250       |      |      |     |     |      |      |        |         |           |
| 0.300       |      |      |     |     |      |      |        |         |           |
| 0.350       |      |      |     |     |      |      |        |         |           |
| 0.400       |      |      |     |     |      |      |        |         |           |
| 0.450       |      |      |     |     |      |      |        |         |           |
| 0.500       |      |      |     |     |      |      |        |         |           |
| 0.550       |      |      |     |     |      |      |        |         |           |
| 0.600       |      |      |     |     |      |      |        |         |           |
| 0.650       |      |      |     |     |      |      |        |         |           |
| 0.700       |      |      |     |     |      |      |        |         |           |
| 0.750       |      |      |     |     |      |      |        |         |           |
| 0.800       |      |      |     |     |      |      |        |         |           |
| 0.850       |      |      |     |     |      |      |        |         |           |
| 0.900       |      |      |     |     |      |      |        |         |           |
| 0.950       |      |      |     |     |      |      |        |         |           |
| 1.000       |      |      |     |     |      |      |        |         |           |
Table 4: Statistics on Reynolds for all algorithms and all metrics, including run time in seconds.

| Run time (s) | Kne * | CBS * | MIP * | Knop * | Spatial Quality * |
|-------------|-------|-------|-------|--------|-------------------|
| 0.00        | 0.00  | 0.00  | 0.00  | 0.00   | 0.00              |
| 0.01        | 0.02  | 0.03  | 0.04  | 0.05   | 0.06              |
| 0.02        | 0.04  | 0.06  | 0.08  | 0.10   | 0.12              |
| 0.03        | 0.06  | 0.09  | 0.12  | 0.15   | 0.18              |
| 0.04        | 0.08  | 0.12  | 0.16  | 0.20   | 0.24              |
| 0.05        | 0.10  | 0.15  | 0.20  | 0.25   | 0.30              |
| 0.06        | 0.12  | 0.18  | 0.24  | 0.30   | 0.36              |
| 0.07        | 0.14  | 0.21  | 0.28  | 0.35   | 0.42              |
| 0.08        | 0.16  | 0.24  | 0.32  | 0.40   | 0.48              |
| 0.09        | 0.18  | 0.27  | 0.36  | 0.45   | 0.54              |
| 0.10        | 0.20  | 0.30  | 0.40  | 0.50   | 0.60              |

Note: Table entries are placeholders for actual statistical data.