Dimensionality Reduction of Collective Motion by Principal Manifolds

Kelum Gajamannage[a], Sachit Butail[b], Maurizio Porfiri[b], Erik M. Bollt[a,*]

[a]*Department of Mathematics, Clarkson University, Potsdam, NY-13699, USA.*
[b]*Department of Mechanical and Aerospace Engineering, New York University Polytechnic School of Engineering, Brooklyn, NY-11201, USA.*

Abstract

While the existence of low-dimensional embedding manifolds has been shown in patterns of collective motion, the current battery of nonlinear dimensionality reduction methods are not amenable to the analysis of such manifolds. This is mainly due to the necessary spectral decomposition step, which limits control over the mapping from the original high-dimensional space to the embedding space. Here, we propose an alternative approach that demands a two-dimensional embedding which topologically summarizes the high-dimensional data. In this sense, our approach is closely related to the construction of one-dimensional principal curves that minimize orthogonal error to data points subject to smoothness constraints. Specifically, we construct a two-dimensional principal manifold directly in the high-dimensional space using cubic smoothing splines, and define the embedding coordinates in terms of geodesic distances. Thus, the mapping from the high-dimensional data to the manifold is defined in terms of local coordinates. Through representative examples, we show that compared to existing nonlinear dimensionality reduction methods, the principal manifold retains the original structure even in noisy and sparse datasets. The principal manifold finding algorithm is applied to configurations obtained from a dynamical system of multiple agents simulating a complex maneuver called predator mobbing, and the resulting two-dimensional embedding is compared with that of a well-established nonlinear dimensionality reduction method.

Keywords:
Dimensionality reduction, algorithm, collective behavior, dynamical systems

1. Introduction

With advancements in data collection and video recording methods, high-volume datasets of animal groups, such as fish schools [1, 2], bird flocks [3, 4], and insect and bacterial swarms [5, 6], are now ubiquitous. However, analyzing these datasets is still a nontrivial task, even when individual trajectories of all members are available. A desirable step that may ease the experimenter's task of locating events of interest is to identify coarse observables [7, 8, 9] and behavioral measures [10] as the group navigates through space. In this context, Nonlinear Dimensionality Reduction (NDR) offers a large set of tools to infer properties of such complex multi-agent dynamical systems.

Traditional Dimensionality Reduction (DR) methods based on linear techniques, such as Principal Components Analysis (PCA), have been shown to possess limited accuracy when input data is nonlinear and complex [11]. DR entails finding the axes of maximum variability [12] or retaining the distances between points [13]. Multi Dimensional Scaling (MDS) with Euclidean metric is another DR method which attains low-dimensional representation by retaining the pairwise distance of points in low dimensional representations [13]. However, Euclidean distance calculates the shortest distance between two points on a manifold instead of the genuine manifold distance, which may lead to difficulty of inferring low-dimensional embeddings. The isometric mapping algorithm (Isomap) resolves the problem associated with MDS by preserving the pairwise geodesic distance between points [14]; it has recently been used to analyze group properties in collective behavior, such as the level of coordination and fragmentation [15, 16, 17].

*Corresponding author
Email addresses: dineshhk@clarkson.edu (Kelum Gajamannage), sb4304@nyu.edu (Sachit Butail), mporfiri@poly.edu (Maurizio Porfiri), bolltem@clarkson.edu (Erik M. Bollt)

Within Isomap, however, short-circuiting [18] created by faulty connections in the neighborhood graph, manifold non-convexity [19, 20] and holes in the data [21] can degrade the faithfulness of the reconstructed embedding manifold.

Diffusion maps [22] have also been shown to successfully identify coarse observables in collective phenomena [23] that would otherwise require hit-and-trial guesswork [24]. Beyond Isomap and diffusion maps, the potential of other NDR methods to study collective behavior is largely untested. For example, Kernel PCA (KPCA) requires the computation of the eigenvectors of the kernel matrix instead of the eigenvectors of the covariance matrix of the data [25] but this is computationally expensive [11]. Local Linear Embedding (LLE) embeds high-dimensional data through global minimization of local linear reconstruction errors [11]. Hessian LLE (HLLE) minimizes the curviness of the higher dimensional manifold by assuming that the low-dimensional embedding is locally isometric [26]. Laplacian Eigenmaps (LE) perform a weighted minimization (instead of global minimization as in LLE) of the distance between each point and its given nearest neighbors to embed high dimensional data [27].

Iterative NDR approaches have also been recently developed in order to bypass spectral decomposition which is common in most of NDR methods [28]. Curvilinear Component Analysis (CCA) employs a self-organized neural network to perform two tasks, namely, vector quantization of submanifolds in the input space and nonlinear projection of quantized vectors onto a low dimensional space [29]. This method minimizes the distance between the input and output spaces. Manifold Sculpting (MS) transforms data by balancing two opposing heuristics: first, scaling information out of unwanted dimensions, and second, preserving local structure in the data. The MS method, which is robust to sampling issues, iteratively reduces the dimensionality by using a cost function that simulates the relationship among points in a local neighborhoods[28]. The Local Spline Embedding (LSE) is another NDR method which embeds the data points using splines that map each local coordinate into a global coordinate of the underlying manifold by minimizing the reconstruction error of the objective function [30]. This method reduces the dimensionality by solving an eigenvalue problem while the local geometry is exploited by the tangential projection of data. LSE assumes that the data is not only unaffected by noise or outliers, but also, sampled from a smooth manifold which ensures the existence of a smooth low dimensional embedding.

Due to the global perspective of all these methods, none of them provide sufficient control over the mapping from the original high-dimensional dataset to the low-dimensional representation, limiting the analysis in the embedding space. In other words, the low-dimensional coordinates are not immediately perceived as useful, whereby one must correlate the axes of the embedding manifold with selected functions of known observables to deduce their physical meaning [24, 14]. In this context, a desirable feature of DR that we emphasize here is the regularity in the spatial structure and range of points on the embedding space, despite the presence of noise.

With regard to datasets of collective behavior, nonlinear methods have limited use for detailed analysis at the level of the embedding space. This is primarily because the majority of these methods collapse the data onto a lower dimensional space, whose coordinates are not guaranteed to be linear functions of known system variables [31]. In an idealized simulation of predator induced mobbing [32], a form of collective behavior where a group of animals crowd around a moving predator, two degrees of freedom are obvious, namely, the translation of the group and the rotation about the predator (center of the translating circle). This two-dimensional manifold is not immediately perceived by Isomap, even for the idealized scenario presented in Figure 1, where a group of twenty individuals rotate about a predator moving at a constant speed about a line bisecting the first quadrant. Specifically, the algorithm is unable to locate a distinct elbow in the residual variance vs. dimensionality curve, notwithstanding substantial tweaking of the parameters. Thus, the inferred dimensionality is always 1 (Fig. 1b). For a two-dimensional embedding (Fig. 1c), visual comparison of the relative scale of the axes indicates that the horizontal axis represents a greater translation than the vertical axis. It is likely that the horizontal axis captures the motion of the group along the translating circle. The vertical axis could instead be associated with (i) motion about the center of the circle, or (ii) noise, which is independent and identically distributed at each time step. The ambiguity in determining the meaning of such direction indicates a drawback of Isomap in provide meaningful interpretations of the low-dimensional coordinates.

An alternative approach to DR, one that does not require heavy matrix computations or orthogonalization, involves working directly on raw data in the high-dimensional space [33, 34]. We propose here a method for DR that relies on geodesic rather than Euclidean distance and emphasizes manifold regularity. Our approach is based on a spline representation of the data that allows us to control the expected manifold regularity. Typically, this entails conditioning the data so that the lower dimensions are revealed. For example, in [33], raw data is successively clustered through a series of lumping and splitting operations until a faithful classification of points is obtained. In [34], one-dimensional parameterized curves are used to summarize the high-dimensional data by using a property called self-consistency,

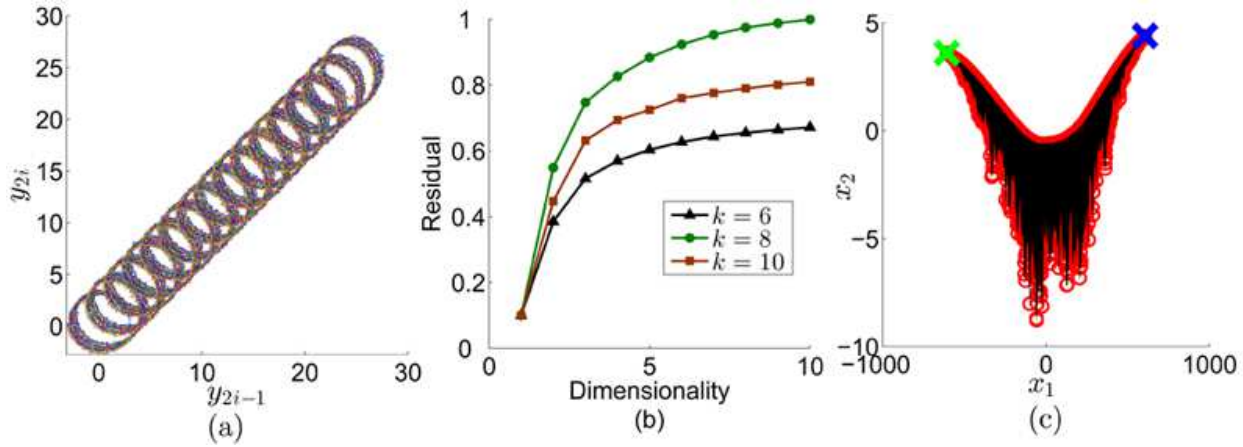

Figure 1: Using Isomap to create a two-dimensional embedding of a simulation of collective behavior. (a) Predator mobbing of twenty agents moving on a translating circular trajectory on a plane (enclosing a predator moving at constant speed at a 45° angle), axes y_{2i-1}, y_{2i} generally represent coordinates for the i-th agent. (b) Scaled residual variance of candidate low-dimensional embeddings produced by Isomap using different nearest neighbor values k (green-circle, brown-square, and black-triangle), and (c) two-dimensional representation of the data for five nearest neighbors (black-triangle). Green and blue crosses mark the start and end points of the trajectory.

in which points on the final curve coincide with the average of raw data projected on itself. In a similar vein, we construct a PM of the high-dimensional data using cubic smoothing splines. Summarizing the data using splines to construct a PM of the data has two advantages: (i) it respects the data regularity by enabling access to every point of the dataset, and (ii) it gives direct control over the amount of noise rejection in extracting the true embedding through the smoothing parameter. We illustrate the algorithm using the standard swiss roll dataset, typically used in NDR methods. Then we validate the method on three different datasets, including an instance of collective behavior similar to that in Figure 1.

This paper is organized as follows. Section 2 describes the three main steps of the algorithm using the swiss roll dataset as an illustrative example. In Section 3, we validate and compare the algorithm on a paraboloid, a swiss roll with high additive noise, and a simulation of collective animal motion. Section 4 analyzes the performance of the algorithm by comparing topologies in the original and embedding space as a function of the smoothing parameter, noise intensity, and data density. We conclude in Section 5 with a discussion of the algorithm performance and ongoing work. Individual steps of the algorithm are detailed in Appendix A. Computational complexity is discussed in Appendix B.

2. Dimensionality reduction algorithm to find principal manifold

Dimensionality reduction is defined as a transformation of high-dimensional data into a meaningful representation of reduced dimensionality [11]. In this context, a d-dimensional input dataset is embedded onto an e-dimensional space such that $e << d$. The mapping from point $\boldsymbol{x} \in \mathbb{R}^e$ in the embedding space to the corresponding point $\boldsymbol{y} \in \mathbb{R}^d$ in the original dataset is

$$\Phi : \mathbb{R}^e \rightarrow \mathbb{R}^d \tag{1}$$

Differently from other approaches, the proposed DR algorithm is based on a PM of the raw dataset, obtained by constructing a series of cubic smoothing splines on slices of data that are partitioned using locally orthogonal hyperplanes. The resulting PM summarizes the data in the form of a two-dimensional embedding, that is, $e = 2$. The PM finding algorithm proceeds in three steps: clustering, smoothing, and embedding. The clustering step partitions the data using reference points into non-overlapping segments called slices. This is followed by locating the longest geodesic within each slice. A cubic smoothing spline is then fitted to the longest geodesic. A second set of slices, and corresponding splines, are created in a similar way resulting in a two-dimensional PM surface. The PM is constructed

Table 1: Nomenclature

Notation	Description
$\mathcal{D}$	Input dataset
d	Input dimension
$\boldsymbol{y}$	A point in the input dataset
n	Number of points in the input dataset
e	Output embedding dimension (equal to two)
$\boldsymbol{x}$	A point in the embedding space
$\boldsymbol{z}$	An external input point for the embedding
$\boldsymbol{\mu}$	Mean (centroid) of the input dataset
p	Smoothing parameter
k	Number of neighbors in the nearest neighbor search
$\boldsymbol{O}$	Origin of the underlying manifold
$\boldsymbol{q}_{1,2}$	Reference points 1 and 2
$C_j^{1,2}$	j-th cluster for reference points 1 and 2
$n_C^{1,2}$	Number of clusters for reference points 1 and 2
$\boldsymbol{S}_j^{1,2}$	j-th cubic spline for reference points 1 and 2
$\mathcal{G}_j^{1,2}$	j-th geodesic for reference points 1 and 2
$\boldsymbol{t}^{l,m}$	Intersection point in d-dimensional space between splines l and m
$(\boldsymbol{u}_{\boldsymbol{z}}^1, \boldsymbol{u}_{\boldsymbol{z}}^2)$	Tangents to splines at the point $\boldsymbol{z}$
$(\boldsymbol{v}_1, \sigma_1), (\boldsymbol{v}_2, \sigma_2)$	First two Principal Components (PCs) of the input dataset, where $\boldsymbol{v}_i$ is the d-dimensional coefficients and σ_i is the eigenvalue of the i-th PC

in the form of intersection points between pairs of lines, one from each reference point. Any point in the input space is projected on the PM in terms of the intersection points and their respective distances along the cubic splines. Table 1 lists the notation used in this paper.

2.1. Clustering

Consider an input dataset $\mathcal{D}$ described by n d-dimensional points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$. Clustering begins by partitioning the data into non-overlapping clusters. The partitioning is performed by creating hyperplanes orthogonal to the straight line joining an external reference point $\boldsymbol{q} \in \mathbb{R}^d$ through the mean $\boldsymbol{\mu} \in \mathbb{R}^d$ to some point $\tilde{\boldsymbol{q}} \in \mathbb{R}^d$. Although any reference point may be selected, we use Principal Components (PCs) to make the clustering procedure efficient in data-representation. In particular, Principal Component Analysis (PCA) of the matrix $\mathcal{D} = \left[\boldsymbol{y}_1 | \boldsymbol{y}_2 | \ldots | \boldsymbol{y}_n\right]$ is performed to obtain two largest principal components $(\boldsymbol{v}_1, \sigma_1)$ and $(\boldsymbol{v}_2, \sigma_2)$, where $\boldsymbol{v}_i$ is the d-dimensional coefficients and σ_i is the eigenvalue of the i-th PC. The first reference point is chosen such that $\boldsymbol{q}_1 = \boldsymbol{\mu} + \boldsymbol{v}_1\sigma_1$. To cover the full range of the dataset, the line joining the points $\boldsymbol{q}_1$ and the point $\tilde{\boldsymbol{q}}_1 = \boldsymbol{\mu} - \boldsymbol{v}_1\sigma_1$ is divided into n_C^1 equal parts using the ratio formula in a d-dimensional space [35]. The j-th point $\boldsymbol{a}_j \in \mathbb{R}^d$ on this line is

$$\boldsymbol{a}_j = \boldsymbol{\mu} + \frac{\boldsymbol{v}_i\sigma_i(2j - n_C^i)}{n_C^i}; \quad j = 0, \ldots, n_C^i \tag{2}$$

where $i = 1$ for this case of dealing with the first reference point. All points between hyperplanes through points s_{j-1} and s_j are assigned to the cluster C_j^1, $j = 1, \ldots, n_C^1$, by using the following inequality for $k = 1, \ldots, n$ with $i = 1$

$$\cos^{-1}\left(\frac{\left(\boldsymbol{y}_k - \boldsymbol{a}_{j-1}\right)^{\mathrm{T}}\left(\boldsymbol{q}_i - \boldsymbol{a}_{j-1}\right)}{\|\boldsymbol{y}_k - \boldsymbol{a}_{j-1}\|\|\boldsymbol{q}_i - \boldsymbol{a}_{j-1}\|}\right) < \pi/2 \quad , \quad \cos^{-1}\left(\frac{\left(\boldsymbol{y}_k - \boldsymbol{a}_j\right)^{\mathrm{T}}\left(\boldsymbol{q}_i - \boldsymbol{a}_j\right)}{\|\boldsymbol{y}_k - \boldsymbol{a}_j\|\|\boldsymbol{q}_i - \boldsymbol{a}_j\|}\right) \geq \pi/2 \tag{3}$$

(Fig. 2a). The clustering method is adapted to detect gaps or holes in the data by setting a threshold on the minimum distance between neighboring points based on a nearest neighbor search [36]. If a set of points do not satisfy the threshold, they are automatically assigned another cluster giving rise to sub-clusters [37] (Fig. 2b).

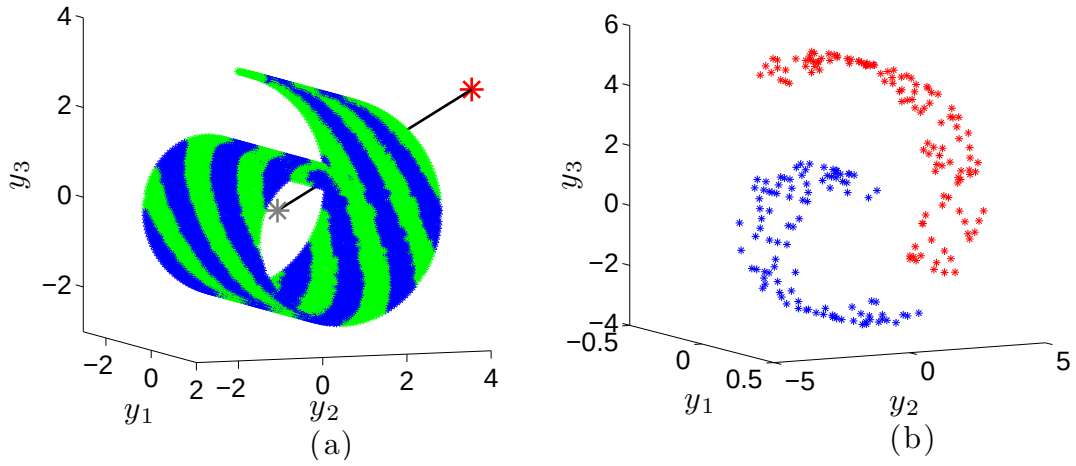

Figure 2: Clustering the raw data into slices. (a) Using a single reference point (red) located far from the data points, the dataset can be sliced into non-overlapping sections which are perpendicular to the line joining the centroid (gray star) and the reference point (red star), then sections are used to draw a grid pattern using splines. (b) A hole in the data will result in multiple sub-clusters (shown in different colors red and blue) within a cluster.

The above procedure is repeated for the second reference point $\boldsymbol{q}_2 = \boldsymbol{\mu} + \boldsymbol{v}_2\sigma_2$ to partition the dataset along the line joining $\boldsymbol{\mu} \pm \boldsymbol{v}_2\sigma_2$ using the equation (2) with $i = 2$. Then, equation (3) is used for $k = 1, \ldots, n$ with $i = 2$ to make another set of clusters C_j^2; $j = 1, \ldots, n_C^2$. The resulting clustering algorithm inputs the data $\mathcal{D}$, and the number of clusters $n_C^{1,2}$ with respect to each reference point, which can be set on the basis of data density in each direction. The set of clusters for each reference point are stored for subsequent operations. The clustering algorithm is detailed in Appendix A as algorithm 1.

2.2. Smoothing

The clustering step is followed by smoothing, where cubic splines are used to represent the the clusters. This approach of using a spline to approximate the data is similar to forming a principal curve on a set of points [34, 38]. Briefly, the principal curve runs smoothly through the middle of the dataset such that any point on the curve coincides with the average of all points that project orthogonally onto this point [34]. The algorithm in [34] searches for a parameterized function that satisfies this condition of self-consistency. In the context of smoothing splines on a cluster with m points, $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m$, the principal curve $\boldsymbol{g}(\lambda)$ parameterized in $\lambda \in [0, 1]$ minimizes over all smooth functions $\boldsymbol{g}$ [34]

$$\sum_{i=1}^{m} \|\boldsymbol{y}_i - \boldsymbol{g}(\lambda_i)\|^2 + \kappa \int_0^1 \|\boldsymbol{g}''(\lambda)\|^2 d\lambda, \tag{4}$$

where κ weights the smoothing of the spline and $\lambda_i \in [0, 1]$ for $i = 1, \ldots, m$. Equation (4) yields d individual expressions, one for each dimension in the input space.

Since the spline is created in a multi-dimensional space, the points are not necessarily arranged to follow the general curvature of the dataset. In order to arrange the points that are input to the equation (4), we perform an additional operation to build geodesics within the cluster. The geodesics are created using a range-search[1] with range distance set as the cluster width $2\sigma_1/n_C^1$ or $2\sigma_2/n_C^2$ depending on the reference point. The longest geodesic within the cluster is used to represent the full range and curvature of the cluster. Using n_g number of d-dimensional points $\left\{\boldsymbol{y}_i = [y_{i,1}, \ldots, y_{i,d}]^{\mathrm{T}} \,|\, i = 1, \ldots, n_g\right\}$ on the longest geodesic, the d-dimensional cubic smoothing spline $\boldsymbol{S}(\lambda) =$

[1] A neighbor search which choses all neighbors within a specific distance.

$[S_1(\lambda)|S_2(\lambda)|\ldots|S_d(\lambda)]$ parameterized in $\lambda \in [0,1]$ is computed as a piecewise polynomial that minimizes the quantity [34, 39]

$$p\sum_{i=1}^{n_g}|y_{i,j}-S_j(\lambda_i)|^2+(1-p)\int_{\lambda_1}^{\lambda_{n_g}}\left[S_j''(\lambda)\right]^2 d\lambda, \tag{5}$$

where $p \in [0,1]$, in each dimension $j = 1,\ldots,d$. Thus, p weights the sum of square error and $(1-p)$ weights the smoothness of the spline; conversely $p = 0$ gives the natural cubic smoothing spline and $p = 1$ gives an exact fit (Fig. 3a). This procedure is repeated for all clusters for each reference point (Fig. 3b) resulting in a grid-like surface of smoothing splines. We denote cubic smoothing splines made with respect to the first reference point by $\boldsymbol{S}_j^1; j = 1,\ldots,n_C^1$, and second reference point by $\boldsymbol{S}_j^2; j = 1,\ldots,n_C^2$. The smoothing algorithm is described in Appendix A as algorithm 2.

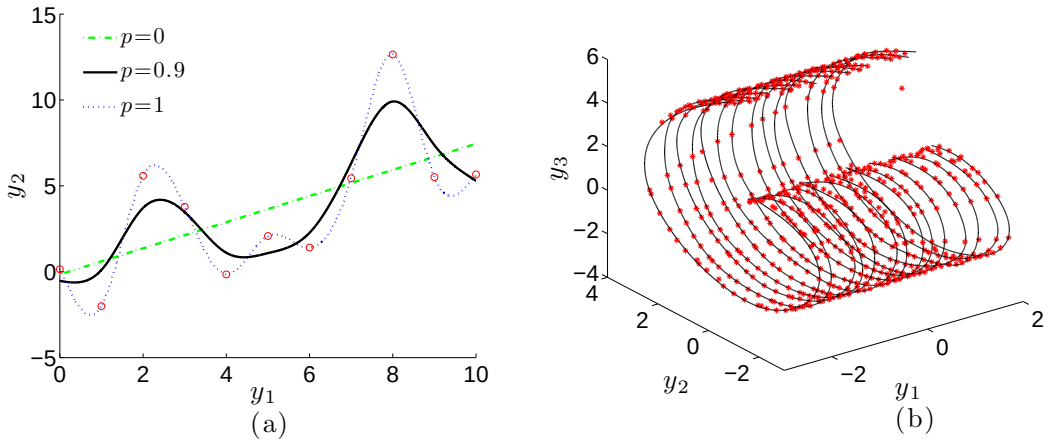

Figure 3: Approximating clusters with smoothing splines. (a) A cubic smoothing spline is fit onto two-dimensional data using different values of the smoothing parameter p. (b) Smoothing splines fit clusters of a swiss roll dataset from a single reference point.

2.3. Embedding

Once constructed, intersection points between all pairs of splines, one from each reference point are located as landmarks to embed new points onto the PM. Since the splines from two reference points may not actually intersect each other unless $p = 1$ for all splines, virtual intersection points between two non-intersecting splines are located at the midpoint on the shortest distance between them. The embedding algorithm loops through all pairs of splines to locate such points, and once located they are denoted as $\boldsymbol{t}^{l,m} \in \mathbb{R}^d$ where $(l,m) \in \boldsymbol{S}_l^1 \times \boldsymbol{S}_m^2$.

In order to define the embedding coordinates, an intersection point is selected randomly and assigned as the manifold origin O of the PM. The set of smoothing splines corresponding to the origin are called axis splines. The embedding coordinates of a point $\boldsymbol{z} \in \mathbb{R}^d$ are defined in terms of the spline distance of the intersecting splines of that point from the axis splines (Fig. 4c). For that, first we find the closest intersection $\tilde{\boldsymbol{z}}$ for $\boldsymbol{z}$ in Euclidean distance as

$$\tilde{\boldsymbol{z}} = \text{argmin}_{l,m}\|\boldsymbol{t}^{l,m}-\boldsymbol{z}\| \tag{6}$$

and tangents to splines at this point are called the local spline directions and denoted by $(\boldsymbol{u}_{\boldsymbol{z}}^1, \boldsymbol{u}_{\boldsymbol{z}}^2)$. Distance from O to $\tilde{\boldsymbol{z}}$ is computed along the axis splines by

$$[\tilde{x}_1,\tilde{x}_2]^{\mathrm{T}} = \left[d_l(\tilde{\boldsymbol{z}}-O), d_m(\tilde{\boldsymbol{z}}-O)\right]^{\mathrm{T}} \tag{7}$$

where $d_l(b_1 - b_2)$ is the distance between points b_1 and b_2 along the spline l. We project the vector $\tilde{\boldsymbol{z}} - \boldsymbol{z}$ on the local tangential directions $(\boldsymbol{u}_{\boldsymbol{z}}^1, \boldsymbol{u}_{\boldsymbol{z}}^2)$ as

$$\begin{aligned} \delta x_1 &= (\tilde{\boldsymbol{z}} - \boldsymbol{z})^{\mathrm{T}} \boldsymbol{u}_{\boldsymbol{z}}^1 \\ \delta x_2 &= (\tilde{\boldsymbol{z}} - \boldsymbol{z})^{\mathrm{T}} \boldsymbol{u}_{\boldsymbol{z}}^2 \end{aligned} \tag{8}$$

at the intersection point $\tilde{\boldsymbol{z}}$. The final embedding coordinates are obtained by summing the quantities in equations (7) and (8) as

$$[x_1, x_2]^{\mathrm{T}} = [\tilde{x}_1 + \delta x_1, \tilde{x}_2 + \delta x_2]^{\mathrm{T}} \tag{9}$$

The embedding algorithm is detailed in Appendix A as algorithm 3.

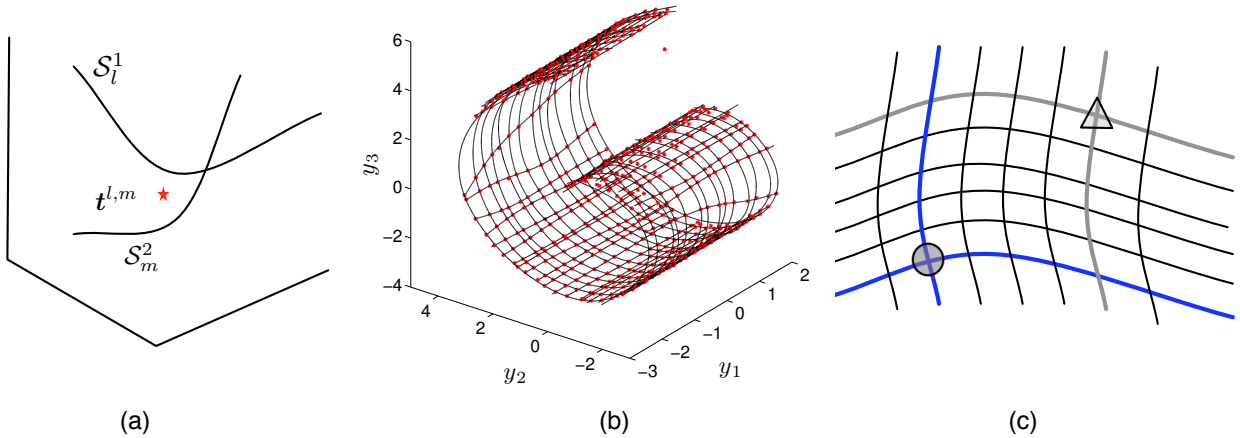

Figure 4: Embedding the data into two-dimensional coordinates. (a) The intersecting point between two splines lies midway on the shortest distance between them (b) intersecting points for a swiss roll and (c) the two-dimensional coordinate of a point (triangle) is the geodesic distance along the intersecting splines (grey) to the axis splines (blue) from an origin (circle).

2.4. Mapping between the manifold and raw data

The mapping between the manifold and the raw data is created by inverting the steps of the embedding algorithm. In particular, a two-dimensional point $\boldsymbol{x} = \left[x_1, x_2\right]^{\mathrm{T}}$ in the embedding space is located in terms of the embedding coordinate $\boldsymbol{x}^{l,m}$ of the closest intersection point $\boldsymbol{t}^{l,m} \in \mathbb{R}^d$. We then complete the local coordinate system by finding the two adjacent intersection points. The resulting three points on the PM, the origin and the two adjacent points denoted by $\{\boldsymbol{x}^{l,m}, \boldsymbol{x}^{l,m+1}, \boldsymbol{x}^{l+1,m}\}$, are used to solve for ratios α and β such that

$$\begin{aligned} \alpha &= \frac{x_2 - x_2^{l,m}}{x_2^{l,m+1} - x_2^{l,m}} \\ \beta &= \frac{x_1 - x_1^{l,m}}{x_1^{l+1,m} - x_1^{l,m}}. \end{aligned} \tag{10}$$

The ratios are then propagated to the high-dimensional space (Fig. 5) by using the same relation on the corresponding points in the higher dimension $\{\boldsymbol{t}^{l,m}, \boldsymbol{t}^{l,m+1}, \boldsymbol{t}^{l+1,m}\}$ to obtain the high-dimensional point

$$\boldsymbol{y} = \boldsymbol{t}^{l,m} + (\boldsymbol{t}^{l,m+1} - \boldsymbol{t}^{l,m})\alpha + (\boldsymbol{t}^{l+1,m} - \boldsymbol{t}^{l,m})\beta. \tag{11}$$

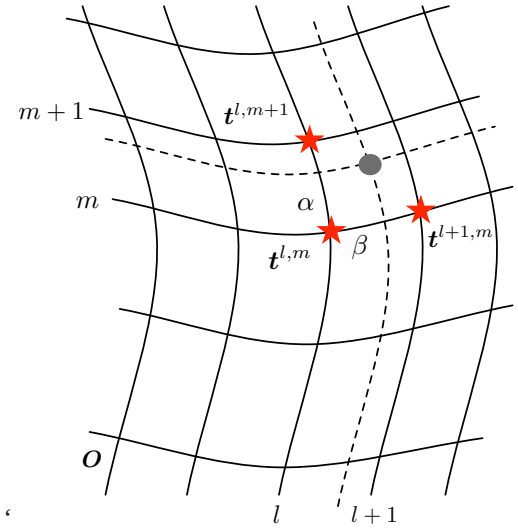

Figure 5: The mapping from the embedding manifold to the original high-dimensional space is found by finding the projections α, β on the local coordinate space and propagating the same into the high dimensions. Local coordinate space can be created using adjacent splines $(l, l+1)$, and $(m, m+1)$.

3. Examples

In this section, we evaluate and compare the PM finding algorithm with other DR methods on three different datasets: a three-dimensional paraboloid, a swiss roll constructed with high additive noise (noisy swiss roll), and a simulation of collective behavior. Specifically, we use a standard version of eight different DR methods implemented in the Matlab Toolbox for DR [40] with default parameters listed in Table 2. Note that we run each method in an unsupervised manner using the standard default values available in that toolbox.

Table 2: Default parameters of DR methods implemented in Matlab Toolbox for DR [40] for the paraboloid and noisy swiss roll examples.

Method	Parameters
MDS	none
Isomap	nearest neighbors = 12
Diffusion maps	variance = 1 and operator on the graph = 1
KPCA	variance = 1 and Gaussian kernel is used
LSE	nearest neighbors = 12
LLE	nearest neighbors = 12
HLLE	nearest neighbors = 12
LE	nearest neighbors = 12 and variance = 1

3.1. Paraboloid

The three-dimensional paraboloid (Fig. 6a) is generated in the form of 2000 points using

$$y_3 = y_1^2 + y_2^2 + \epsilon, \tag{12}$$

where $\boldsymbol{y}$ is a point in the three-dimensional space with $y_1, y_2 \in [-2, 2]$, and $\epsilon \sim \mathbb{U}(-0.05, 0.05)$ is a noise variable sampled from a uniform distribution ranging between -0.05 and 0.05. Since the data is generated along the y_1, y_2 axes, the two reference points are selected along the same just outside the range for each value. We run the clustering

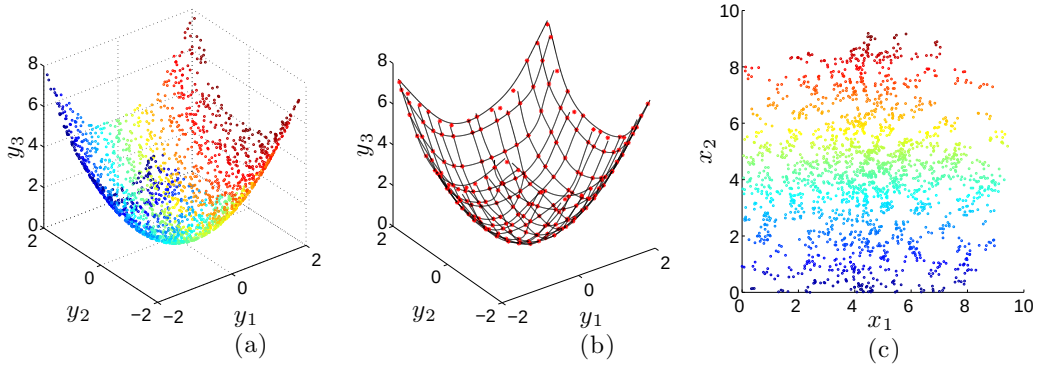

Figure 6: Embedding data from a three-dimensional paraboloid with 2000 points (a) of raw data after smoothing (b) into two dimensional embedded space with intrinsic coordinate of the manifold (c). Points are colored to highlight the relative configuration.

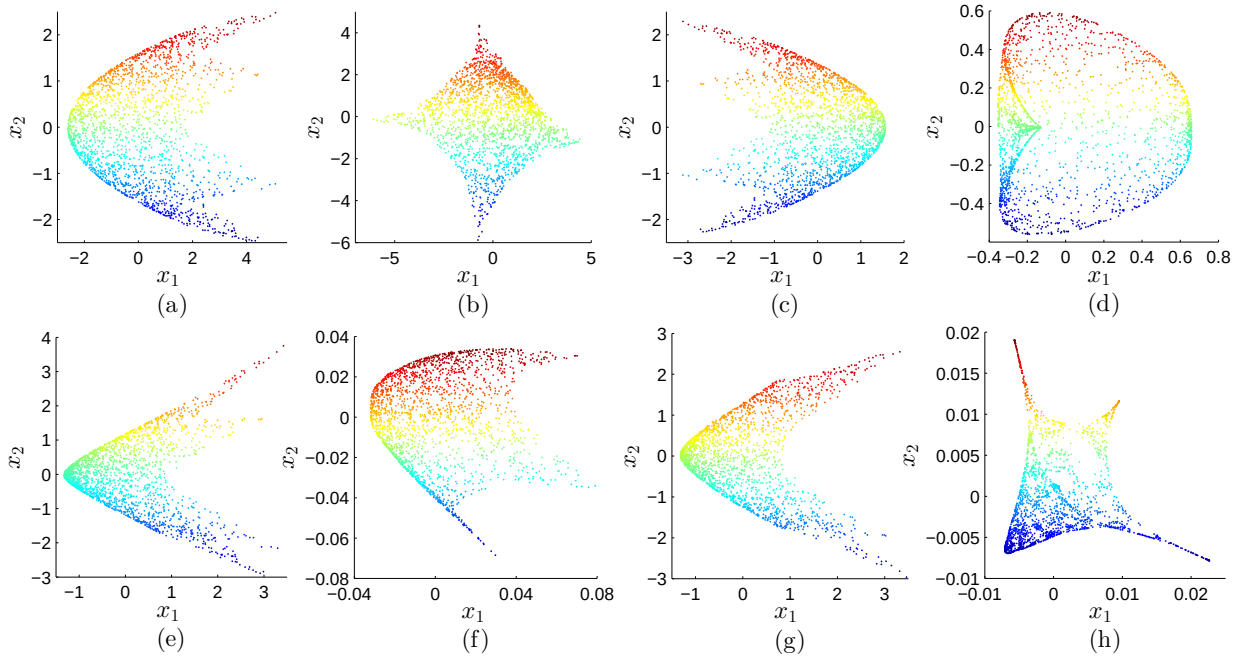

Figure 7: Two-dimensional embeddings of the paraboloid by MDS (a), Isomap (b), Diffusion maps (c), KPCA (d), LSE (e), LLE (f), HLLE (g), and LE (h)

algorithm with the value of $n_C = 14$ for each reference point to generate corresponding sets of clusters with algorithm 1. Next, we run the smoothing algorithm with a smoothing parameter $p = 0.9$ to produce a set of representative splines for the data (Fig. 6b). Finally, we embed the manifold by first finding the intersection points and, then, computing the distance of each point from the axis splines (Fig. 6c).

The two-dimensional embedding manifold generated using our algorithm preserves the topology of the data as shown in Figure 6c. Furthermore, the axes in the embedding manifold retain the scale of the original data in terms of the distance between individual points. For a comparison, we run other NDR methods on this data set with parameters specified in table 2 (Fig.7). Results show that except Isomap and the proposed method, none of the NDR methods are able to preserve the two-dimensional square embedding after dimensionality reduction. Between Isomap and the PM approach, the PM approach retains the scale of the manifold as well.

3.2. A noisy swiss roll

In a second example, we run our algorithm on a 2500 point three-dimensional noisy swiss roll dataset with high additive noise (Fig. 8a). The dataset is generated using

$$\begin{aligned} y_1 &= \theta^{0.8}\cos\theta + \epsilon \\ y_2 &= \theta^{0.8}\sin\theta + \epsilon \\ y_3 &= \phi + \epsilon \end{aligned} \tag{13}$$

where $\theta \in [0.25\pi, 2.5\pi]$, $\phi \in [-2, 2]$, and $\epsilon \sim \mathbb{U}(-0.4, 0.4)$ is a noise variable sampled from a uniform distribution ranging between -0.4 and 0.4. Two reference points are chosen along two major principal component directions $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ at a distance of σ_1 and σ_2 units away from the data centroid $\boldsymbol{\mu}$. We run the clustering algorithm with the value of $n_C = 15$ for each reference point, however, due to folds and therefore gaps in the data, the clusters along $\boldsymbol{v}_2$ are further split into 42 subclusters. The value of smoothing parameter $p = 0.75$. Figure 8b and 8c show the smoothing splines in a grid pattern that are used to embed the manifold, and the resulting embedding. For a comparison, we run other NDR methods on this data set with parameters specified in table 2 (Fig. 9). Comparison between existing NDR methods show that while only Isomap and KPCA are able to flatten the swiss roll into two dimensions, Isomap preserves the general rectangular shape of the flattened swiss roll. A majority of NDR methods suppress one component out of the two principal components in the raw data revealing an embedding that resembles a one-dimesional curve. In contrast, the PM approach is able to embed the swiss roll into a rectangle while preserving the scale.

We note that by changing the value of the smoothing parameter we control for the level of noise-rejection in the original data, a feature that is not available in the existing DR algorithms [11]. Although the splines form illegal connections, our algorithm is still able to preserve the two-dimensional embedding; this is due to the inherent embedding step which is able to overcome such shortcuts while computing the low-dimensional coordinates. Furthermore, we note that our method is able to better exclude noisy data from the embedding automatically in the form of outliers; neighborhood based algorithms would be mislead to attempt to include these as valid points [41]. For example, the two-dimensional embedding obtained using Isomap shows an unforeseen bend demonstrating that, despite fewer outliers, the general topological structure is compromised more heavily. This global anomaly is a direct consequence of the Isomap's attempt to include noisy points in the embedding; points that are otherwise ignored by the principal manifold.

3.3. Collective behavior: simulation of predator mobbing

In a third example of low-dimensional embedding, we generate a dataset representing a form of collective behavior. We simulate point-mass particles revolving on a translating circle to represent a form of behavior called predator induced mobbing [42]. In prey animals, predator mobbing, or crowding around a predator, is often ascribed to the purpose of highlighting the presence of a predator [42]. In our idealized model, we assume that N agents ($N = 20$) are revolving on the circumference of a half circle whose center marks the predator and translating on a two-dimensional plane which is orthogonal to the axis of the revolution. To generate a two-dimensional trajectory with ρ revolutions around a translating center we represent the two-dimensional position of an agent i at k, $r_i[k] = \begin{bmatrix} y_{2i-1}[k], & y_{2i}[k] \end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{r}_i[k] \in \mathbb{R}^2$

$$\boldsymbol{r}_i[k] = s_i[k]\begin{bmatrix}\cos(2\pi\rho k/n_k + \pi i/N) \\ \sin(2\pi\rho k/n_k + \pi i/N)\end{bmatrix} + k\mathbf{v}_p[k] + \boldsymbol{\epsilon}[k], \tag{14}$$

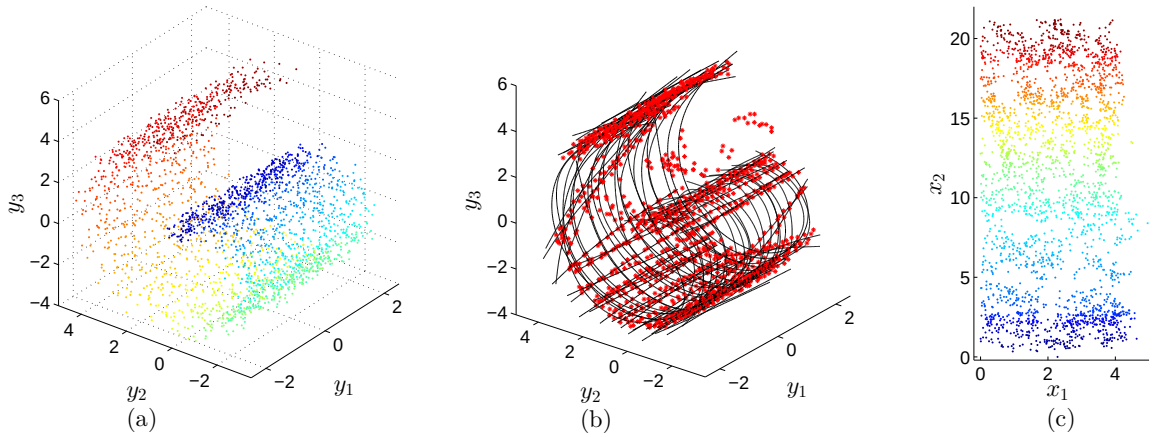

Figure 8: Two-dimensional embedding of noisy swiss-roll with 2500 points, raw data (a) of the noisy swiss roll is clustered and smoothened in order to obtain the principal manifold (b) and then embedded in two dimensions with intrinsic coordinates denoting the distance from the center of the manifold coordinate (c).

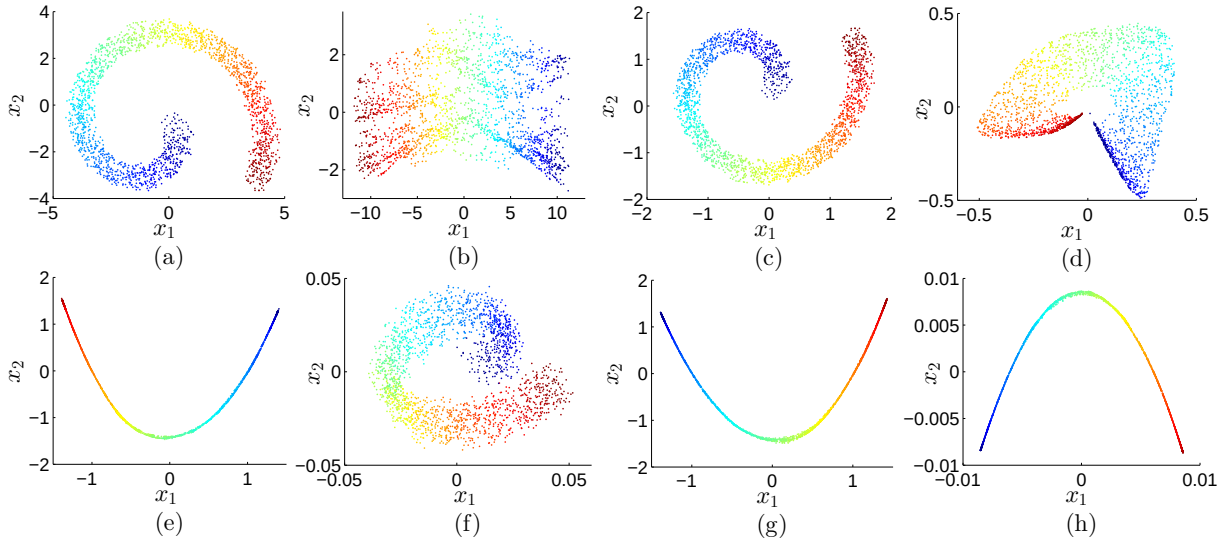

Figure 9: Two-dimensional embeddings of the noisy swiss-roll by MDS (a), Isomap (b), Diffusion maps (c), KPCA (d), LSE (e), LLE (f), HLLE (g), and LE (h).

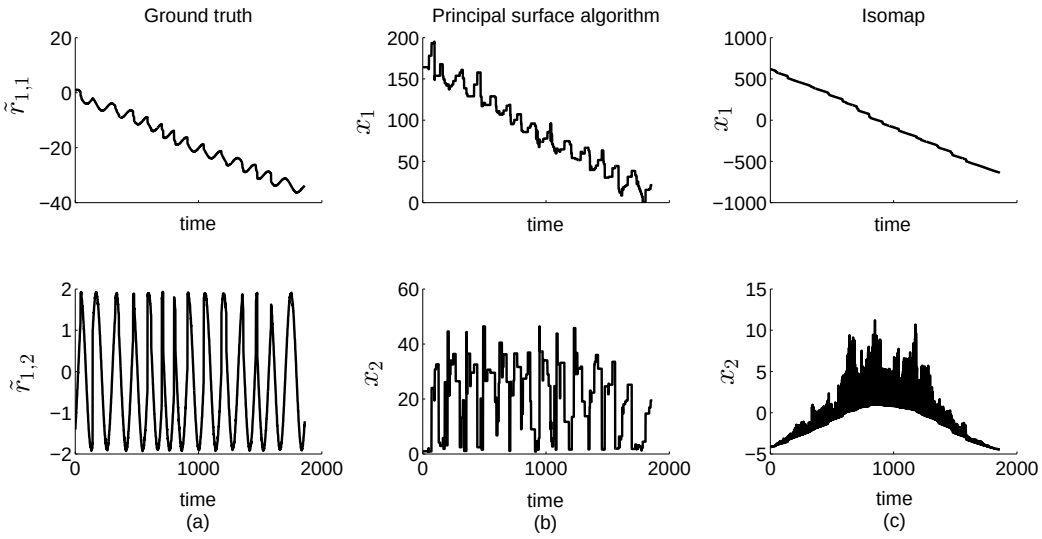

Figure 10: Embedding a forty dimensional dataset of an idealized simulation of predator mobbing where twenty particles move around a translating circle. A comparison of individual axes of the embedding produced by the principal surface algorithm described in this paper and those produced using Isomap shows that our algorithm performs distinctly better in terms of preserving the time signature as well as range of the axes. The nearest neighbor parameter used for Isomap is equal to 10.

where $\mathbf{v}_p[k] = [1, 1]^{\mathrm{T}}/80$ is the velocity of the predator $s_i[k] = 3$ is the speed of the agent i, n_k is the number of total time-steps, and $\boldsymbol{\epsilon}[k] \sim \mathbb{N}(\mathbf{0}, \mathbf{1}/100)$ is the Gaussian noise variable with mean $\mathbf{0} \in \mathbb{R}^2$ and standard deviation $\mathbf{1}/100 \in \mathbb{R}^2$. The first term of Eqn. (14) assigns the relative positions of N agents onto equally spaced points on the circumference of a half circle of units $s_i[k]$, and the second term gives the velocity of the virtual center (predator) of the translating circle. The agent-dependent phase $\pi i/N$ creates spatial separation of individual members around the virtual center. Figure 1a shows the resulting trajectories for 2000 time-steps with $\rho = 14$ in a d-dimensional space (note that $d = 2N$ in this case, where N is the number of agents). The interaction between the agents is captured in the form of noise $\boldsymbol{\epsilon}$. A high value means that the agents interact less with each other.

To cluster the dataset, we use $n_C = 3$. The value of the smoothing parameter $p = 0.9$. The embedding manifold is shown in Fig. 10b with the start and end points of the trajectory. For comparison, we also run the Isomap algorithm on this dataset with a neighborhood parameter value set to 10. To ensure that the embedding is consistent and robust to our choice of neighborhood parameter, we run the algorithm with neighborhood parameters 5 and 15 and verify that the output is similar. Figure 10c shows the resulting two-dimensional embedding coordinates each as a function of time.

Unlike the paraboloid and the noisy swiss roll, the predator mobbing dataset represents a dynamical system with a characteristic temporal evolution. Therefore, we use cross-correlation between the embedding of each method and a reference ground truth to compare the performance of our algorithm to Isomap (Fig. 1). For ground truth, we transform the original data by a clockwise rotation of $\pi/4$ such that

$$\tilde{\boldsymbol{r}}_i[k] = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \boldsymbol{r}_i[k] \tag{15}$$

and use the position of the first agent $\tilde{\boldsymbol{r}}_1[k] = \begin{bmatrix} \tilde{r}_{1,1}[k], & \tilde{r}_{1,2}[k] \end{bmatrix}^{\mathrm{T}}$ (this value serves as a descriptive global observable of the predator mobbing agents separated by a constant phase, than, for example, the group centroid, where the revolving motion is suppressed due to the instantaneous two-dimensional arrangement). We then cross-correlate each component of the two-dimensional embedding signal $\begin{bmatrix} x_1, x_2 \end{bmatrix}^{\mathrm{T}}$ with the corresponding component of the ground-truth and add the two values. We compute correlations of the embedded data of our algorithm and Isomap with the ground

truth along the oscillating directions as 0.3987 and 0.0074. Thus, the correlation of our algorithm with the ground truth is fifty times higher than the correlation of Isomap with the ground truth. Correlation p-values under our method and Isomap with the ground truth are 0.0000 and 0.3888 respectively, showing that under 95% of confidence interval, the values from our method are related to the ones available from the ground truth. Figure 1 visually demonstrates the same where x_2 from the principal surface follows the same trend as $\tilde{r}_{1,2}$. In addition, the range of values of the embedding coordinates is similar to the original values.

4. Performance analysis

To analyze the performance of the algorithm, we use a distance-preserving metric between the original and the embedding data in terms of adjacency distance of graph connectivity. For both the original and the embedding data, we first search k-nearest neighbors through the algorithm given in [43] and then produce two individual weighted graphs based on points connectivity in the data sets. A graph constructed through nearest neighbor search is a simple graph[2] that does not contain self-loops or multiple edges. We compute the adjacency distance matrix A for the original data as

$$A_{i,j} = \begin{cases} d(i,j) & : \exists \text{ an edge } ij \text{ in the graph of the original data} \\ 0 & : \text{otherwise} \end{cases} \tag{16}$$

and the adjacency matrix $\tilde{A}$ for the embedding data as

$$\tilde{A}_{i,j} = \begin{cases} d(i,j) & : \exists \text{ an edge } ij \text{ in the graph of the embedding data} \\ 0 & : \text{otherwise} \end{cases} \tag{17}$$

Here, A_{ij} and $\tilde{A}_{ij}$ are the $(i,j)-$th entries of the adjacency matrices A and $\tilde{A}$, and $d(i,j)$ is the Euclidean distance between nodes i and j. For n points, this metric is computed as the normalized number of pairwise errors between entries of the adjacency distance matrices as

$$\Delta(\tilde{A}, A) = \frac{1}{nk} \sum_{i,j} \mid \tilde{A}_{ij} - A_{ij} \mid, \tag{18}$$

Equation (18) is a modification of the structure preserving metric in [45] where the connectivity is replaced by the distance and the denominator n^2 with the number of edges nk. We study the dependence of $\Delta(\tilde{A}, A)$ on the smoothing parameter, the noise in the dataset, and the data density.

4.1. Error dependence on smoothing

The primary input to the algorithm described in this paper is the smoothing parameter that controls the degree of fitness and noise-rejection by the cubic smoothing splines. Figure 11 shows the dependence of the distance preservation error Δ in term of adjacency distance on the smoothing parameter p. Using a noise-free swiss roll dataset comprising 3000 points in three dimensions, we vary the smoothing parameter between 0 and 1 with an increment of 0.01 while keeping the location of reference points and cluster width consistent. The error dependence shows a linear decay with R^2-value [3] of 0.9728 and becomes nearly constant at a $p = 0.88$, beyond which the change in error for an increase in the value of p is negligible. A higher value of p improves the data-fit but a low value improves smoothness. Since the graph connectivity is dependent on the relative point configuration, it is expected that Δ will rise with increasing smoothness. On the other hand, the error dependence shows that a given degree of smoothing can be attained beyond the value of $p = 0.88$ without an accompanied increase error. Thus, as a design parameter, the value $p = 0.9$ may be used as a starting point for all datasets to investigate the embedding manifold.

[2] A simple graph is an undirected graph that does not contains loops (edges connected at both ends to the same vertex) and multiple edges (more than one edge between any two different vertices) [44].

[3] In a fit of a data set, R^2-value (coefficient of determination) ranges between 0 and 1, and measures the goodness of the fit so that 1 is the best while 0 is the worst. For a set of points $\{y_i\}_{i=1}^n$ associated with fitted values $\{f_i\}_{i=1}^n$, coefficient of determination or R^2-value is defined as, $1 - \frac{\sum_i^n (y_i - f_i)^2}{\sum_i^n (y_i - \bar{y})^2}$ where $\bar{y} = \sum_i^n y_i / n$.

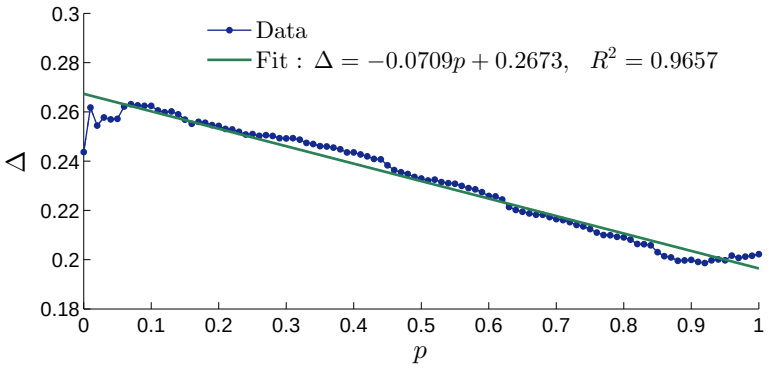

Figure 11: Variation of Δ with respect to p and corresponding linear fit for a noise free swiss-roll. Δ is computed using Eq. (18) on raw and embedded data.

4.2. Error with noise

To analyze the change in pairwise error with increase in noise, we create multiple same-sized swiss-roll datasets generated using (13) with noise value ϵ ranging between 0 and 1.035 with an increment of 0.015. The number of points in the dataset are 3000 and the value of smoothing parameter $p = 0.9$. Figure 12 shows the variation of Δ with respect to ϵ. The error Δ increases quadratically with R^2-value 0.9983 when noise increases. This relation shows that, the error of the embedding is affected quadratically by the intense of the associated noise in the data.

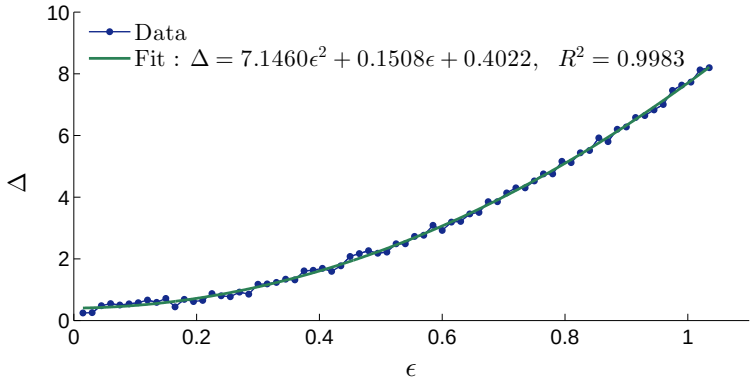

Figure 12: Variation of Δ with respect to noise (ϵ) and corresponding quadratic fit Δ is computed over adjacency matrices for the raw data and embedded data by Eq. (18).

4.3. Data density

To analyze the dependence of Δ on data density we sub-sample the swiss roll dataset such that the number of points are systematically decreased. The amount of noise is set at $\epsilon = 0.2$ and a sequence of data sets are generated with number of points between 500 and 3500 with increment of 40. The value of the smoothing parameter p is fixed at 0.9. We see that the pairwise error (Δ) decreases exponentially with R^2-value 0.9713 from an initial high value (Fig. 13).

5. Conclusion

In this paper, we introduce a PM finding dimensionality reduction algorithm that works directly on raw data by approximating the data in terms of cubic smoothing splines. We construct the splines on two sets of non-overlapping

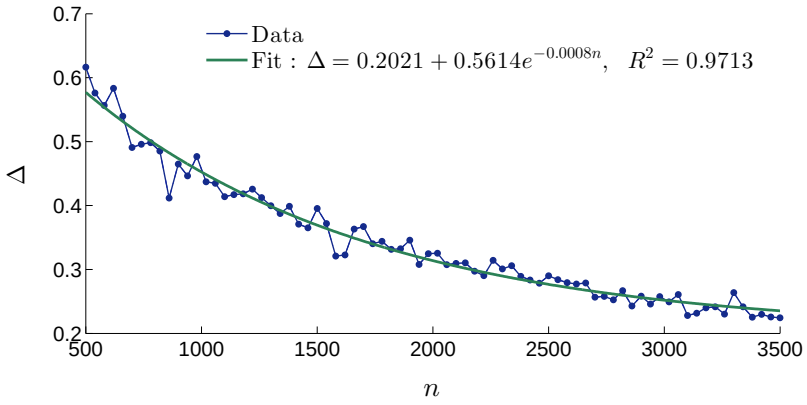

Figure 13: Variation of Δ with respect to number of points (n) and corresponding exponential fit with $p = 0.9$ and $\epsilon = 0.2$. Δ is computed over adjacency matrices for the raw data and embedded data by Eq. (18).

slices of the raw data created using parallel hyperplanes that are known distance apart. The resulting PM is a grid-like structure that is embedded on a two-dimensional manifold. The smoothness of the splines can be controlled via a smoothing parameter that selectively weighs fitting error versus the amount of noise rejection. The PM is represented by the intersection points between all pairs of splines, while the embedding coordinates are represented in the form of distance along these splines.

The spline representation improves regularity of an otherwise noisy dataset. We demonstrate the algorithm on three separate examples including a paraboloid, a noisy swiss roll, and a simulation of collective behavior. Upon embedding these datasets on two-dimensional manifolds, we find that in the case of the paraboloid, the algorithm is successful in preserving the topology and the range of the data. In the case of the noisy swiss roll, we find that the algorithm is able to reject high noise, thereby preserving the regularity in the original dataset. Unlike Isomap, our algorithm for finding PM is able to maintain a general structure. In other words, the algorithm fails gracefully giving the user enough indication of the trend of increasing error differently than Isomap, which fails abruptly and dramatically [46]. This property has a distinct advantage over other methods that are more sensitive to input parameters and therefore make it difficult to identify when the algorithm is embedding correctly. In contrast, the proposed algorithm on the swiss-roll dataset shows that in its current form, is inefficient to embed the dataset with holes faithfully onto a two-dimensional space, since it is highly focused on revealing a smooth underlying manifold.

We used a simulation of twenty particles moving in a translating circle as a representation of a two-dimensional data embedded in a forty-dimensional state. In collective behavior, this form of motion approximates predator mobbing. Cross-correlation between the embedded signals and the ground-truth available in the form of a transformed signal shows that our algorithm is able to replicate the time-history on a low-dimensional space than for example Isomap. Furthermore, we see that the range of axes in the embedding manifold available from our algorithm is closer to that in the original data.

The analysis of the distance preserving ability of our algorithm with respect to the smoothing parameter reveals that an increase in the smoothing parameter reduces the error in the structure. This is expected because a high value of p implies that the cubic smoothing spline weights data fit more than smoothing. However, we also note that the amount of error saturates near $p = 0.88$, showing that values close to but not equal to one may also be used to represent the data. We observe a similar trend with respect to the amount of noise introduced in a swiss-roll dataset, where we find that the amount of error increases quadratically with an increase in noise. Finally, we find that the algorithm is able to work on sparse datasets up to a representation with five hundred points.

Operating and representing raw data for dimensionality reduction provides an opportunity for extracting true embeddings that preserve the structure as well as regularity of the dataset. By demanding a two-dimensional embedding we also provide a useful visualization tool to analyze high-dimensional datasets. Finally, we show that this approach is amenable to high-dimensional dynamical systems such as those available in dataset of collective behavior.

6. Acknowledgements

Kelum Gajamannage and Erik M. Bollt have been supported by the National Science Foundation under grant no. CMMI- 1129859. Sachit Butail and Maurizio Porfiri have been supported by the National Science Foundation under grants nos. CMMI- 1129820.

Appendix A. Algorithms

Here, we state the algorithms of dimensionality reduction by principal manifold by three components as, clustering, smoothing, and embedding.

Algorithm 1 *Clustering* : Data matrix $\mathcal{D}$ and number of clusters with respect to the first (n_C^1) and second (n_C^2) reference points are three inputs in the clustering algorithm. As the output, this produces two sets of clusters $C_j^i\ ;\ j = 1, \ldots, n_C^i$ for $i = 1, 2$ from $\mathcal{D}$ such that one for each reference point.

procedure CLUSTERING($\mathcal{D}, n_C^1, n_C^2$)
  Compute the mean $\boldsymbol{\mu} \in \mathbb{R}^d$ of the input data $\mathcal{D}$.
  Perform Principal Component Analysis (PCA) [47, 48] on the input data matrix $\mathcal{D} = \left[\boldsymbol{y}_1|\boldsymbol{y}_2|\ldots|\boldsymbol{y}_n\right]$ to obtain two largest principal components $(\boldsymbol{v}_1, \sigma_1)$ and $(\boldsymbol{v}_2, \sigma_2)$, where $\boldsymbol{v}_i$ is the d-dimensional coefficients and σ_i is the eigenvalue of the i-th PC for $i = 1, 2$
  In order to assure that two reference points are in the directions of two PCs from the mean, compute the first reference point $\boldsymbol{q}_1 = \boldsymbol{\mu} + \boldsymbol{v}_1\sigma_1$, and the second $\boldsymbol{q}_2 = \boldsymbol{\mu} + \boldsymbol{v}_2\sigma_2$.
  for each reference point $\boldsymbol{q}_i; i = 1, 2$ **do**
    Segment the line joining the reference points $\boldsymbol{q}_i = \boldsymbol{\mu} + \boldsymbol{v}_i\sigma_i$ and the point $\tilde{\boldsymbol{q}}_i = \boldsymbol{\mu} - \boldsymbol{v}_i\sigma_i$ into n_C^i parts with equal width using the ratio formula given in equation (2) to obtain a set of points $\boldsymbol{a}_j; j = 0, \ldots, n_C^i$ [35], where $\boldsymbol{a}_0 = \tilde{\boldsymbol{q}}_i$ and $\boldsymbol{a}_{n_C^i} = \boldsymbol{q}_i$
    For $j = 1, \ldots, n_C^i$, choose data between hyperplanes, which are made through points $\boldsymbol{a}_{j-1}$ and $\boldsymbol{a}_j$ normal to the line joining $\tilde{\boldsymbol{q}}_i$ and $\boldsymbol{q}_i$, into the cluster C_j^i by satisfying the inequalities in (3).
  end for
end procedure

Algorithm 2 *Smoothing* : Smoothing algorithm produces two sets of cubic smoothing splines with respect to both reference points to represents the data in clusters. This algorithm inputs spline smoothing parameter p and two sets of clusters $C_j^i\ ;\ j = 1, \ldots, n_C^i$ for $i = 1, 2$ made in the clustering algorithm, and outputs two sets of cubic smoothing splines $\boldsymbol{S}_j^i\ ;\ j = 1, \ldots, n_C^i$ for $i = 1, 2$.

procedure SMOOTHING($p; C_j^i, j = 1, \ldots, n_C^i$ for $i = 1, 2$)
  for each reference point $\boldsymbol{q}_i; i = 1, 2$ **do**
    for all clusters $C_j^i; j = 1, \ldots, n_C^i$ **do**
      Compute the neighborhood graph using range-search with the distance set as the cluster width $2\sigma_i/n_C^i$.
      Compute the longest geodesic $\mathcal{G}_j^i$ using Dijkstra's algorithm [49, 50]. $\mathcal{G}_j^i$ is a set of points, where each point is in $\mathbb{R}^d$.
      For points in $\mathcal{G}_j^i$, use (5) and the value of the smoothing parameter p to produce a smoothing spline $\boldsymbol{S}_j^i \in \mathbb{R}^d$ representation for data in the cluster C_j^i.
    end for
  end for
end procedure

Algorithm 3 *Embedding* : This produces a grid structure by approximating the intersections of splines, followed by doing the embedding of a new point based on this structure. Embedding algorithm inputs a new point $z \in \mathbb{R}^d$ to embed and two sets of cubic smoothing splines $\boldsymbol{S}_j^i$; $j = 1, \ldots, n_C^i$ for $i = 1, 2$ produced by smoothing algorithm, and this outputs two dimensional embedding coordinates $[x_1, x_2]^{\mathrm{T}}$ of $\boldsymbol{z}$.

procedure EMBEDDING($\boldsymbol{S}_j^i$, $j = 1, \ldots, n_C^i$ for $i = 1, 2$; a point $\boldsymbol{z} \in \mathbb{R}^d$ for embedding)
  for all pairs (l, m) of smoothing splines $\boldsymbol{S}_l^1 \times \boldsymbol{S}_m^2$ belonging to reference points 1 and 2 **do**
    Approximate the minimum distance between the two splines after discretizing each spline.
    Choose the midpoint between the two closest points of the splines as the intersection point $\boldsymbol{t}^{l,m} \in \mathbb{R}^d$.
  end for
  Pick a random intersection point as the origin $O \in \mathbb{R}^d$, and smoothing splines corresponding to the origin as axis splines.
  Find the closest intersection point $\tilde{\boldsymbol{z}}$ for $\boldsymbol{z}$ in terms of Euclidean distance by equation (6), and the tangents to the splines at this point are called the local spline directions $(\boldsymbol{u}_{\boldsymbol{z}}^1, \boldsymbol{u}_{\boldsymbol{z}}^2)$.
  Compute the distances $[\tilde{x}_1, \tilde{x}_2]^{\mathrm{T}}$ from the manifold origin O to $\tilde{\boldsymbol{z}}$ along axis splines by equation (7).
  Project the vector $\tilde{\boldsymbol{z}} - \boldsymbol{z}$ onto the local coordinate system created using the spline directions as equation (8) at the intersection point $\tilde{\boldsymbol{z}}$ and find the projections $[\delta x_1, \delta x_1]^{\mathrm{T}}$. The final embedding coordinates are given as $x_1 = \tilde{x}_1 + \delta x_1$, $x_2 = \tilde{x}_2 + \delta x_2$.
end procedure

Appendix B. Computational complexity

The three steps of the algorithm, namely, clustering, smoothing, and embedding have different computational complexities. In particular, for n points with dimensionality d, the clustering algorithm has a complexity

$$O\left(m^3\right); \quad \text{where} \quad m = \min(n, d), \tag{B.1}$$

which is dominated by the PCA [48, 51] step.

The computational complexity of the smoothing algorithm is dominated by the Dijkstra's algorithm [49] for calculating the longest geodesics. Here, we first assume that each cluster with respect to the first reference point contains an equal number of points. Thus, partitioning a data set of n point into n_C^1 clusters yields n/n_C^1 points in a cluster. The complexity of generating the longest geodesic in each such cluster is $O\left(n/n_C^1 \log\left(n/n_C^1\right)\right)$ [49, 50], which sums-up n_C^1 times and implies the total complexity $O\left(n \log\left(n/n_C^1\right)\right)$ of making all geodesics with respect to the first reference point. Similarly, for the second reference point, the same procedure is followed for all n_C^2 clusters, each containing n/n_C^2 points, to obtain a complexity of $O\left(n \log\left(n/n_C^2\right)\right)$. Altogether, geodesics from both reference points contributes a computational complexity of

$$O\left(n \log\left(\frac{n^2}{n_C^1 n_C^2}\right)\right) \tag{B.2}$$

for the smoothing algorithm.

In the embedding algorithm, discretizing splines and approximating the minimum distance are dominant in the total cost of that algorithm. Here, we first calculate the mean length of splines with respect to the first reference point, denoted as $\tilde{\boldsymbol{S}}^1 \in \mathbb{R}^d$, and the one for the second reference point, denoted as $\tilde{\boldsymbol{S}}^2 \in \mathbb{R}^d$. Then, the computational cost associated with approximating intersection of these two splines are computed. For a pair of d-dimensional splines, computation of the Euclidean distance between any two points, one from each spline, has complexity of $3d$ from the operations of subtraction, squaring, and summation along all d dimensions. Since each spline $\tilde{\boldsymbol{S}}^{1,2}$ is discretized at a mesh size h, while $\tilde{\boldsymbol{S}}^1$ has $\tilde{S}^1/h$ points on it, $\tilde{\boldsymbol{S}}^2$ has $\tilde{S}^2/h$ points. The computational cost of approximating closest distance between any two points one from each spline $\tilde{\boldsymbol{S}}^{1,2}$ is $3d\tilde{\boldsymbol{S}}^1\tilde{S}^2/h^2$. Here, for simplicity, we assume that the length of each spline with respect to the first and second reference points have same lengths as $\tilde{\boldsymbol{S}}^1$ and $\tilde{S}^2$ consecutively. Thus, the total complexity of the embedding algorithm, which has n_C^1 and n_C^2 number of splines with respect to the

first and second reference points, is

$$O\left(\frac{3d}{h^2} n_C^1 n_C^2 \tilde{\boldsymbol{S}}^1 \tilde{\boldsymbol{S}}^2\right) \tag{B.3}$$

Clustering, embedding algorithms and steps $1-6$ in the embedding algorithm are performed only once for a given data set, thus after they are completed, n new points are embedded with $O(n)$.

References

[1] B. L. Partridge, The structure and function of fish schools, Scientific American 246 (6) (1982) 114–123.
[2] R. Gerlai, High-throughput behavioral screens: the first step towards finding genes involved in vertebrate brain function using zebrafish., Molecules 15 (4) (2010) 2609–2622.
[3] M. Nagy, Z. Ákos, D. Biro, T. Vicsek, Hierarchical group dynamics in pigeon flocks, Nature 464 (7290) (2010) 890–893.
[4] M. Ballerini, N. Cabibbo, R. Candelier, A. Cavagna, E. Cisbani, I. Giardina, A. Orlandi, G. Parisi, A. Procaccini, M. Viale, Empirical investigation of starling flocks: a benchmark study in collective animal behaviour, Animal behaviour 76 (1) (2008) 201–215.
[5] K. Branson, A. A. Robie, J. Bender, P. Perona, M. H. Dickinson, High-throughput ethomics in large groups of Drosophila, Nature Methods 6 (2009) 451–457.
[6] H. P. Zhang, A. Be'Er, R. S. Smith, E. Florin, H. L. Swinney, Swarming dynamics in bacterial colonies, EPL (Europhysics Letters) 87 (4) (2009) 48011.
[7] A. Kolpas, J. Moehlis, I. G. Kevrekidis, Coarse-grained analysis of stochasticity-induced switching between collective motion states., Proceedings of the National Academy of Sciences of the United States of America 104 (14) (2007) 5931–5935.
[8] N. Y. Miller, R. Gerlai, Oscillations in shoal cohesion in zebrafish (Danio rerio)., Behavioural Brain Research 193 (1) (2008) 148–151.
[9] T. A. Frewen, I. D. Couzin, A. Kolpas, J. Moehlis, R. Coifman, I. G. Kevrekidis, Coarse collective dynamics of animal groups, in: Coping with Complexity: Model Reduction and Data Analysis, 2011, pp. 299–309.
[10] N. Miller, R. Gerlai, Redefining membership in animal groups., Behavior research methods 43 (4) (2011) 964–970.
[11] L. Vand der Maaten, E. Postma, H. Van den Herik, Dimensionality reduction: A comparative review, TiCC TR 2009–005, Tilburg University, The Netherlands (2009).
[12] M. Kirby, Geometric data analysis: an empirical approach to dimensionality reduction and the study of patterns, John Wiley & Sons, Inc., 2000.
[13] T. F. Cox, M. A. Cox, Multidimensional scaling, CRC Press, 2010.
[14] J. B. Tenenbaum, V. De Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[15] N. Abaid, E. Bollt, M. Porfiri, Topological analysis of complexity in multiagent systems, Physical Review E 85 (4) (2012) 041907.
[16] P. DeLellis, G. Polverino, G. Ustuner, N. Abaid, S. Macrì, E. M. Bollt, M. Porfiri, Collective behaviour across animal species, Scientific reports 4.
[17] M. Arroyo, L. Heltai, D. Millán, A. DeSimone, Reverse engineering the euglenoid movement, Proceedings of the National Academy of Sciences 109 (44) (2012) 17874–17879.
[18] O. Samko, A. D. Marshall, P. L. Rosin, Selection of the optimal parameter value for the isomap algorithm, Pattern Recognition Letters 27 (9) (2006) 968–979.
[19] M. E. Tipping, C. C. Nh, Sparse kernel principal component analysis.
[20] P. DeLellis, M. Porfiri, E. M. Bollt, Topological analysis of group fragmentation in multi-agent systems, Physical Review E 87 (2) (2013) 022818.
[21] H. Li, L. Teng, W. Chen, I.-F. Shen, Supervised learning on local tangent space, in: Advances in Neural Networks–ISNN 2005, Springer, 2005, pp. 546–551.
[22] R. R. Coifman, S. Lafon, Diffusion maps, Applied and computational harmonic analysis 21 (1) (2006) 5–30.
[23] M. Aureli, F. Fiorilli, M. Porfiri, Portraits of self-organization in fish schools interacting with robots, Physica D: Nonlinear Phenomena 241 (9) (2012) 908–920.
[24] T. A. Frewen, I. D. Couzin, A. Kolpas, J. Moehlis, R. Coifman, I. G. Kevrekidis, Coarse collective dynamics of animal groups, in: Coping with Complexity: Model Reduction and Data Analysis, Springer, 2011, pp. 299–309.
[25] J. Shawe-Taylor, N. Cristianini, Kernel methods for pattern analysis, Cambridge university press, 2004.
[26] D. L. Donoho, C. Grimes, Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data, Proceedings of the National Academy of Sciences 100 (10) (2003) 5591–5596.
[27] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering., in: NIPS, Vol. 14, 2001, pp. 585–591.
[28] M. Gashler, D. Ventura, T. R. Martinez, Iterative non-linear dimensionality reduction with manifold sculpting., in: NIPS, Vol. 8, 2007, pp. 513–520.
[29] P. Demartines, J. Hérault, Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets, Neural Networks, IEEE Transactions on 8 (1) (1997) 148–154.
[30] S. Xiang, F. Nie, C. Zhang, C. Zhang, Nonlinear dimensionality reduction with local spline embedding, Knowledge and Data Engineering, IEEE Transactions on 21 (9) (2009) 1285–1298.
[31] J. A. Lee, A. Lendasse, M. Verleysen, Nonlinear projection with curvilinear distances: Isomap versus curvilinear distance analysis, Neurocomputing 57 (null) (2004) 49–76.
[32] W. J. Dominey, Mobbing in colonially nesting fishes, especially the bluegill, Lepomis macrochirus, Copeia 1983 (4) (1983) 1086–1088.
[33] G. H. Ball, H. D. J., ISODATA, A novel method of data analysis and pattern clasification, Tech. rep., Stanford Research Institute (1965).
[34] T. Hastie, W. Stuetzle, Principal curves, Journal of the American Statistical Association 84 (406) (1989) 502–516.

[35] M. H. Protter, Calculus with Analytic Geometry, 4th Edition, Jones and Bartlett, 1988, Ch. 01, p. 29.
[36] S. A. Nene, S. K. Nayar, A simple algorithm for nearest neighbor search in high dimensions, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (9) (1997) 989–1003.
[37] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on Mathematics, Statistics and Probability, 1967, pp. 281–297.
[38] G. Biau, A. Fischer, Parameter Selection for Principal Curves, IEEE Transactions on Information Theory 58 (3) (2012) 1924–1939.
[39] E. M. Bollt, Attractor Modeling and Empirical Nonlinear Model Reduction of Dissipative Dynamical Systems, International Journal of Bifurcation and Chaos 17 (4) (2007) 1199–1219.
[40] L. van der Maaten, E. Postma, H. van den Herik, Matlab toolbox for dimensionality reduction, MICC, Maastricht University.
[41] J. B. Tenenbaum, V. de Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction., Science 290 (5500) (2000) 2319–2323.
[42] W. J. Dominey, Mobbing in colonially nesting fishes, especially the bluegill, Lepomis macrochirus, Copeia 1983 (4) (1983) 1086–1088.
[43] J. H. Friedman, J. L. Bentley, R. A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Transactions on Mathematical Software (TOMS) 3 (3) (1977) 209–226.
[44] R. Balakrishnan, K. Ranganathan, A textbook of graph theory, Springer, 2012.
[45] B. Shaw, T. Jebara, Structure preserving embedding, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 937–944.
[46] N. Mekuz, J. K. Tsotsos, Parameterless isomap with adaptive neighborhood selection, Pattern Recognition 4174 (2006) (2006) 364–373.
[47] G. H. Golub, C. F. Van Loan, Matrix computations, Vol. 3, JHU Press, 2012.
[48] I. Jolliffe, Principal component analysis, Wiley Online Library, 2005.
[49] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische mathematik 1 (1) (1959) 269–271.
[50] C. E. Leiserson, R. L. Rivest, C. Stein, T. H. Cormen, Introduction to algorithms, MIT press, 2001.
[51] J. E. Jackson, A user's guide to principal components, Vol. 587, John Wiley & Sons, 2005.