Signal Processing 104 (2014) 232–247

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Tangent-based manifold approximation with locally linear models

Sofia Karygianni*, Pascal Frossard

Ecole Polytechnique Fédérale de Lausanne (EPFL), Signal Processing Laboratory (LTS4), CH-1015 Lausanne, Switzerland

ARTICLE INFO

Article history:
Received 27 August 2013
Received in revised form
26 February 2014
Accepted 28 March 2014
Available online 8 April 2014

Keywords:
Manifold approximation
Tangent space
Affine subspaces
Flats

ABSTRACT

In this paper, we consider the problem of manifold approximation with affine subspaces. Our objective is to discover a set of low dimensional affine subspaces that represent manifold data accurately while preserving the manifold's structure. For this purpose, we employ a greedy technique that partitions manifold samples into groups, which are approximated by low dimensional subspaces. We start by considering each manifold sample as a different group and we use the difference of local tangents to determine appropriate group mergings. We repeat this procedure until we reach the desired number of sample groups. The best low dimensional affine subspaces corresponding to the final groups constitute our approximate manifold representation. Our experiments verify the effectiveness of the proposed scheme and show its superior performance compared to state-of-the-art methods for manifold approximation.

1. Introduction

The curse of dimensionality is one of the most fundamental issues that researchers have to face across various data processing disciplines. High dimensional data is often difficult to manipulate: it might belong to huge parametric spaces that are challenging to exploit while the corresponding models can be complex enough to make learning challenging and prone to over-fitting. However, it is not rare that the data follows some underlying structure, which can lead to more efficient data representation and analysis if modeled properly.

The underlying structure of signals of a given family can often be described adequately by a manifold model that has a smaller dimensionality than the signal space. Prominent examples are signals that are related by transformations, like images captured under different viewpoints in a 3D scene, or signals that represent different observations of the same physical phenomenon like EEG and ECG data. Manifold models have been successfully used in many different applications like transformation-invariant classification, recognition and dimensionality reduction [1–3].

In general, manifolds are topological spaces that locally resemble a Euclidean space. Therefore, although they might be extremely complicated structures, they have locally, i.e., in the neighborhood of a point, the same characteristics as the usual Euclidean space. In this work, we are going to consider d-dimensional, differentiable manifolds that are embedded into a higher dimensional Euclidean space, $\mathbb{R}^N, N \gg d$. Intuitively, one can think of a d-dimensional manifold embedded into $\mathbb{R}^N$ as the generalization of a surface in N dimensions: it is a set of points that locally seem to live in $\mathbb{R}^d$ but that macroscopically synthesize a structure living into $\mathbb{R}^N$. For example, a sphere in $\mathbb{R}^3$ and a circle in $\mathbb{R}^2$ are both manifolds of dimensions 2 and 1 respectively. Although manifolds are appealing for effective data representation, their unknown and usually strongly non-linear structure makes their

* Corresponding author.
E-mail addresses: sofia.karygianni@epfl.ch (S. Karygianni), pascal.frossard@epfl.ch (P. Frossard).

http://dx.doi.org/10.1016/j.sigpro.2014.03.047
0165-1684/© 2014 Elsevier B.V. All rights reserved.

manipulation quite challenging. There are cases where an analytical model can represent the manifold, like a model built on linear combinations of atoms coming from a predefined dictionary [4]. However, an analytical model is unfortunately not always available. A workaround consists in trying to infer a global, data-driven parametrization scheme for the manifold by mapping the manifold data from the original space to a low-dimensional parametric space. The problem of unveiling such a parametrization is called manifold learning [1,2].

However, it is in general hard to compute a universal manifold representation that is accurate for all data in the datasets. In general, it is not possible to represent all the non-linearities of the manifold by one single mapping function. Therefore, instead of using just one global scheme, it is often preferable to employ a set of simpler structures to approximate the manifold's geometry. This can be done in the original space of the manifold. The objective of the approximation is to create a manifold model that is as simple as possible while preserving the most crucial characteristic of a manifold, namely its geometrical shape. An example of such an approximation for a 1D manifold is shown in Fig. 1a, where a set of lines approximates the spiral shape.

In this paper, we approximate generic manifolds with simple models that are affine subspaces (flats). Such a choice is motivated by the locally linear character of manifolds as well as the simplicity and efficiency of flats for performing local computations like projections. Our objective is to compute a set of low dimensional flats that represent the data as accurately as possible, and at the same time preserves the geometry of the underlying manifold. We formulate the manifold approximation problem as a constrained clustering problem for manifold samples. The constraints are related to the underlying geometry of the manifold, which is represented by the neighborhood graph of the data samples. We borrow elements of the constrained clustering theory to motivate the use of a greedy scheme for manifold approximation. We first propose to relate the capability of a set of points to be represented by a flat, with the variance of the tangents at these points. Then, we use the difference of tangents to uncover groups of points that comply with the low dimensionality of flats. The partitioning is done in a bottom-up manner where each manifold sample is considered as a different group at the beginning. Groups are then iteratively merged until their number reduces to the desired value. We have tested our algorithm on both synthetic and real data where it gives a superior performance compared to state-of-the-art manifold approximation techniques.

The rest of the paper is organized as follows. In Section 2, we discuss the related work in manifold approximation and other relevant fields like manifold learning and hybrid linear modeling. In Section 3, we give some mathematical definitions related to manifolds and tangent spaces, which are essential for the work presented in this paper. In Section 4, we motivate the use of a greedy strategy with concepts from constrained clustering theory and we present our novel problem formulation for the manifold approximation. We present our approximation algorithm in detail in Section 5. In Section 6, we describe the experimental setup and the results of our experiments. Finally, in Section 7, we provide concluding remarks.

2. Related work

Data representation with affine models has received quite some attention lately. Relative approaches usually fall under the name of either subspace clustering or hybrid linear modeling. Their objective is to find a set of affine models explaining the different data sources, i.e., to cluster the data into groups so that each group can be well represented by a low-dimensional affine space. A common approach is to use an iterative scheme to alternate between steps of data segmentation and subspace estimation aiming at either

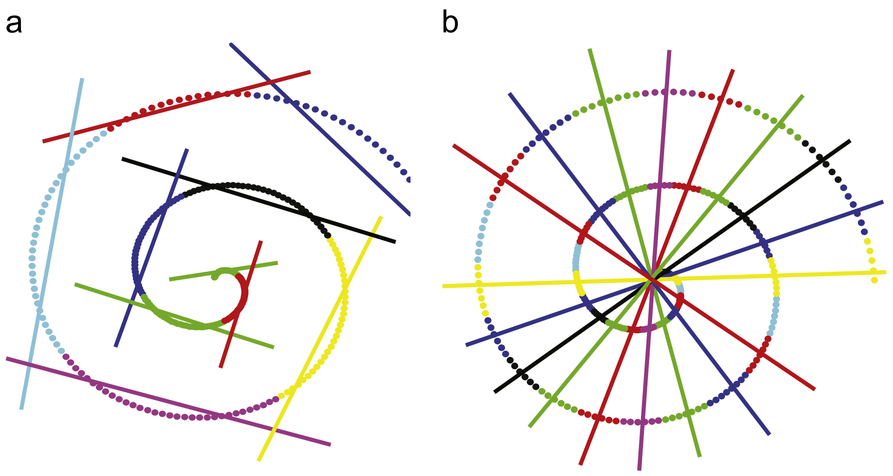


Fig. 1. Manifold approximation illustration. On the left, we have an example of a valid approximation by lines of a 1D manifold embedded into $\mathbb{R}^2$. The different colors represent the different groups of samples, each approximated by a line. On the right, we have an example where the approximation does not align well with the manifold structure, as a result of the median k-flats algorithm [6]. (a) Good manifold approximation example. (b) Bad manifold approximation example. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

minimizing the sum of reconstruction errors [5,6] or maximizing the likelihood of the data under a probabilistic model, like probabilistic PCA [7]. Alternatively, different kinds of algebro-geometric approaches have also been proposed. An interesting formulation has been presented in [8], where the problem of subspace clustering is transformed into a problem of fitting and manipulating polynomials. Moreover, in [9,10], the spectral analysis of an appropriately defined similarity matrix over the data is used to uncover the underlying low dimensional structures as well as the partition that favors them. Recently, in [11], the use of spectral analysis is combined with a multiscale analysis of the growth rate of the local neighborhoods' eigenvalues, so that the appropriate clustering as well as the model parameters, number and dimensionality of the subspaces, are simultaneously recovered from the data. While they are quite successful at times, the above methods apply mainly to cases where data is generated from different low dimensional subspaces that do not necessarily form a manifold. And as such, they uncover a set of linear spaces that do not necessarily comply with the manifold structure, such as the set of lines shown in Fig. 1b.

As far as manifold-driven data is concerned, there is a great variety of works in the so-called fields of manifold learning and dimensionality reduction. The goal of manifold learning is to devise a low dimensional and global parametrization for datasets that lie on high dimensional non-linear manifolds, while preserving some properties of the underlying manifold. Two pioneer works in that field are the Isomap [1] and the LLE algorithm [12]. In Isomap, the parametrization is uncovered in a way that preserves the geodesic distances between the points while in LLE the focus is on preserving the local linear properties of neighborhoods. Other well known approaches that aim at preserving local properties of the points' neighborhoods are provided by the Laplacian Eigenmaps (LE) [13] and the Hessian Eigenmaps (HLLE) [14]. Recently, these methods have been extended to points lying on Riemmanian manifolds as opposed to Euclidean spaces [2]. This opens the range of possible applications for manifold-based representations. A detailed list of the most popular algorithms for manifold learning can be found in [15,16], along with interesting comments on their relative strengths and weaknesses.

In manifold approximation, the goal is to represent the manifold structure in the original space. The ultimate target is not a global parametrization, but rather the definition of a set of local affine subspaces that accurately approximate the original geometry accurately. Although the locally linear nature of manifolds has been used as a tool for learning a global parametrization by aligning or combining local probabilistic data models (e.g., [17,18]), only a few works so far have tried to create a model of the manifold in the original space while preserving its structural properties. Two such examples are the works of Wang and Chen [3] and Fan and Yeung [19]. In [3], the authors introduce the Hierarchical Divisive Clustering (HDC) algorithm, which is a method for hierarchically partitioning the data by dividing highly non-linear clusters. As a linearity measure, it uses the deviation between the Euclidean and geodesic distances. In [19], the clustering is performed in a bottom-up manner, named Hierarchical Agglomerative Clustering (HAC), where again the geodesic distances are used to express the underlying manifold structure.

In our work, we have chosen a new bottom-up approach that uses a different linearity measure, namely the variance of the tangent spaces. As it will be shown in the next section, this measure emerges naturally from the definition of the local properties of a manifold while both linearity measures in [19,3] are more simplistic. The importance of the tangent spaces for manifold related tasks has been lately recognized by many researchers. In [20], the authors use the tangent spaces to infer valid parametrizations of a manifold. Moreover, in [21], the tangent computed by a face image and its perturbed versions is used to capture the local geometry of the corresponding data manifold; faces are then classified based on the distances between their tangents. In our work, however, tangents are computed based on a set of neighboring data. Similarly, the authors in [22] focus on the reliable estimation of the tangent spaces from the data. They also incorporate the tangent distance into a variation of a *k*-means algorithm to classify samples into linear groups, which is another piece of evidence that tangent distances can be used for identifying linear regions on manifolds. Our approach, however, specifically addresses the problem of linear manifold approximation as it effectively combines the tangent distances with the theory of constrained clustering towards the design of an accurate manifold model.

3. Preliminaries

The manifold approximation method proposed in this paper uses *d*-dimensional linear subspaces to approximate the distribution of the data, which is modeled as a differential manifold. At the same time, these linear subspaces form a Riemannian manifold as well, called the Grassmann manifold. The Grassmann manifold is often used when signals are modeled with linear low-dimensional models such as in [23] and in [3]. While many metrics exist for computing distances between linear subspaces [24], the most natural one is the geodesic distance on the Grassmann manifold that is computed based on angles between subspaces. In the rest of this section, we will provide some more basic definitions necessary for understanding our method, along with the description of proper metrics.

First of all, a set $\mathcal{M} \subseteq \mathbb{R}^N$ is a *d-dimensional differentiable manifold* [25] iff $\forall x \in \mathcal{M}$ there exist open sets $V \in \mathbb{R}^N$ with $x \in V$ and $W \in \mathbb{R}^d$ as well as a one-to-one, differentiable function $f: W \rightarrow \mathbb{R}^N$ with continuous inverse such that

1. $f(W) = \mathcal{M} \cap V$, and
2. $f'(y) = Df(y)$, the Jacobian matrix of *f*, has rank *d*, $\forall y \in W$.

The function *f* is called a coordinate system at *x*. Assuming that $f(a) = x$, the *d*-rank Jacobian matrix $Df(x)$ and the corresponding linear transformation $f_*: \mathbb{R}^d_a \rightarrow \mathbb{R}^N_x$ define a

d-dimensional subspace of $\mathbb{R}^N_x$, which is the *tangent space of $\mathcal{M}$ at x* denoted as M_x. Instead of working with a set of d-dimensional subspaces that are positioned at point x, it is more convenient to translate all of them to the origin of $\mathbb{R}^N$. For the rest of the paper, M_x refers to the tangent space of x translated to the origin of $\mathbb{R}^N$.

After the shifting, the tangent spaces of $\mathcal{M}$ belong to the space of all d-dimensional linear subspaces of $\mathbb{R}^N$; this space is called the Grassmann manifold and it is denoted as $G_{N,d}$ [26]. In $G_{N,d}$, the geodesic distance (arc length) between two subspaces is computed based on their principal angles [27]. In particular, we can define the *distance between two tangents M_x and M_y* as

$$D_T(M_x, M_y) = \left(\sum_{i=1}^{d} \theta_i^2\right)^2 = \|\theta\|_2 \tag{1}$$

where $\theta = \{\theta_1, \dots, \theta_d\}$ is the vector of the principal angles of M_x and M_y.

Finally, we describe the notion of the mean tangent of a set of samples C_i. To define such a quantity, we can use the generalization of the arithmetic mean to manifolds. To be more specific, the mean or center of a set C of points in the metric space S (with respect to a distance $\mathcal{D}$) has been given by Karcher in [28] as the element $m_C \in S$ that minimizes the sum of square distances $\mathcal{D}$'s to the points x in the set, i.e.,

$$m_C = \underset{s \in S}{\arg\min} \sum_{x \in C} \mathcal{D}^2(x, s) \tag{2}$$

For a set C_i, where each sample $x \in C_i$ has a tangent space M_x. The mean tangent M_{C_i} can be computed using the geodesic distance introduced in Eq. (1). Hence, Eq. (2) translates into

$$M_{C_i} = \underset{M \in G_{N,d}}{\arg\min} \sum_{x \in C_i} D_T^2(M_x, M_{C_i}) \tag{3}$$

There are several methods that can be used to solve for M_{C_i} in Eq. (3). In this work, we have used the algorithm based on the singular value decomposition [29].

4. Manifold approximation problem

4.1. General framework

Equipped with the above definitions, we can now present our problem formulation. We consider the problem of approximating a d-dimensional manifold $\mathcal{M}$, embedded into $\mathbb{R}^N$, with a set of d-dimensional affine subspaces, which we call flats. The dimension d is an external parameter in our problem; in practice, it is either specific to the application at hand or estimated a priori from the data. The manifold is represented by the set of samples $\mathcal{X} = \{x_k \in \mathbb{R}^N, k \in [1, m]\}$ and the undirected and symmetric neighborhood graph $G_\mathcal{X} = G(\mathcal{X}, E)$, which represents the manifold's geometry by connecting neighbor samples on the manifold. Our objective is to uncover a partition of $\mathcal{X}$ into $\mathcal{L}$ clusters, $\mathbf{C}_\mathcal{L}(\mathcal{X}) = \{C_i, i \in [1, \mathcal{L}]\}$, so that each cluster can be well represented by a d-dimensional flat that respects the underlying geometry of the manifold. The number of clusters $\mathcal{L}$ is also specific to the target application; it could also be inferred from the data through an iterative procedure that stops when the approximation error reaches a pre-defined threshold. In this paper, we simply consider that the number of clusters is given as an external parameter to the algorithm.

4.2. Feasible partitions

In order for $\mathbf{C}_\mathcal{L}(\mathcal{X})$ to be a partition of $\mathcal{X}$, the involved clusters should not overlap and they should cover the whole set $\mathcal{X}$, i.e., $C_j \cap C_i = \varnothing, \ \forall\ i \neq j$ and $\bigcup_{i=1}^{\mathcal{L}} C_i = \mathcal{X}$. There are many different ways to partition a set into L clusters. However, not all possible partitions of $\mathcal{X}$ are valid in our case since we are interested only in partitions that respect the underlying geometry of the manifold. In particular, we would like to ignore the partitions whose clusters spread over different regions of the manifold as the resulting flats do not comply with the local manifold structure, although these clusters can be approximated well with flats. An example of such a bad partitioning is illustrated in Fig. 1b.

In order to check the compliance of a partition $\mathbf{C}_\mathcal{L}(\mathcal{X})$ with the manifold's shape we can use the graph $G_\mathcal{X}$. Based on the above description, a sufficient condition for a partition to be valid is to have clusters with connected subgraphs. To be more specific, each cluster's subgraph is defined as $G_{C_i} = G_\mathcal{X}(C_i, E_i)$, where $E_i = \{a_{ij} \in E: x_i, x_j \in C_i\}$ is the set of edges in E with both endpoints in C_i. Then, the subgraph G_{C_i} is connected if every pair of nodes in C_i is connected with a path in E_i.

The set of all partitions that fulfill this condition, i.e., the 'good' partitions, is called the *feasible set of order $\mathcal{L}$* and denoted by $\mathbf{\Phi}_\mathcal{L}(\mathcal{X})$. The corresponding feasibility predicate,[1] $\Phi_\mathcal{X}(\mathbf{C}_\mathcal{L}) \equiv \mathbf{C}_\mathcal{L} \in \mathbf{\Phi}_\mathcal{L}(\mathcal{X})$, is then defined as

$$\Phi_\mathcal{X}(\mathbf{C}_\mathcal{L}) = \bigwedge_{C_i \in \mathbf{C}_\mathcal{L}} \phi(C_i)$$

$$\text{where } \phi(C_i) = \begin{cases} \text{true} & \text{if } G_{C_i} \text{ is connected} \\ \text{false} & \text{if } G_{C_i} \text{ is not connected,} \end{cases} \tag{4}$$

where the symbol $\wedge$ stands for logical addition.

In what follows, we are proposing a bottom-up approach we therefore need a rule that permits us to merge clusters while preserving the feasibility of the resulting partition. To this end, we define the fusibility predicate $\psi(C_i, C_j)$ that expresses whether two clusters C_i and C_j are 'related', i.e., they could be merged. It is closely related with the feasibility predicate ϕ of Eq. (4) by the following property of binary heredity:

$$\text{if } C_i, C_j \neq \emptyset, \quad C_i \cap C_j = \emptyset, \quad \phi(C_i) \wedge \phi(C_j) \quad \text{and} \\ \psi(C_i, C_j), \quad \text{then } \phi(C_i \cup C_j) \tag{5}$$

This property means that the fusion of two 'good' and 'related' clusters should give a 'good' cluster. In our case, the 'goodness' of a cluster is defined in (4) and is related to the connectivity of the clusters' graph G_C. Therefore, an appropriate choice for the 'related' predicate is to make sure that the graph corresponding to the union of the two

[1] The term 'predicate' is used to refer to boolean valued functions.

clusters is connected. A sufficient condition consists in the presence of an edge between any sample in C_i and any sample in C_j. Therefore, the fusibility predicate becomes

$$\psi(C_i, C_j) = \begin{cases} \text{true} & \text{if } C_i, C_j \text{ have an edge connecting them} \\ \text{false} & \text{otherwise.} \end{cases} \tag{6}$$

4.3. Evaluation of feasible partitions

Equipped with the definition of feasible partitions and with a method to create new feasible partitions from existing ones through merging fusible clusters, we now define a way to evaluate the effectiveness of a feasible partition in capturing the manifold's local geometry. We first need a criterion function P that is non-negative, distributive over the clusters in $\mathbf{C}$ and zero for the case of single-element clusters, i.e.,

$$P(\mathbf{C}) = \sum_{C_i \in \mathbf{C}} p(C_i) \quad \text{with } p(C_i) \geq 0 \quad \text{and} \quad p(\{x\}) = 0, \ \forall x \in \mathcal{X}. \tag{7}$$

The function $p(C_i)$, which represents the distribution of P over the clusters in a partition, has to be non-negative for all clusters and zero for single-element clusters. In our case, the goal is to uncover clusters that can be well-represented by d-dimensional flats; therefore, the function p should be measuring how well the points in the corresponding cluster can be represented by a linear d-dimensional space.

From the definition of manifolds in Section 3, we can observe that the regions of the manifold that can be well represented by linear d-dimensional spaces are the ones for which the function f is linear. In such a case, we have the Jacobian matrices $Df(a) = Df(b), \ \forall a, b \in W$, which means that the tangent spaces of all points $x \in \mathcal{M} \cap V$ coincide when they are seen as subspaces in $\mathbb{R}_N$. Therefore, an appropriate measure of the linearity of a manifold region is the variability of the tangent spaces in it. Hence, we introduce a variance-based criterion function $p(C_i)$ that measures the variance of the tangents of the samples in a cluster C_i, i.e.,

$$p(C_i) = \sum_{x \in C_i} D_T^2(M_{C_i}, M_x) \tag{8}$$

where M_{C_i} is the mean tangent over the tangents of the samples in C_i and D_T is the geodesic distance on the Grassman manifold given in Eq. (1).

4.4. Problem formulation

We now formalize our manifold approximation objective as the problem of finding the feasible partition $\mathbf{C}^*_{\mathcal{L}}(\mathcal{X})$ that minimizes P, i.e.,

$$\mathbf{C}^*_{\mathcal{L}}(\mathcal{X}) = \underset{\mathbf{C} \in \mathbf{\Phi}_{\mathcal{L}}(\mathcal{X})}{\operatorname{argmin}} P(C) = \underset{\mathbf{C} \in \mathbf{\Phi}_{\mathcal{L}}(\mathcal{X})}{\operatorname{argmin}} \sum_{C_i \in \mathbf{C}} p(C_i) \tag{9}$$

By substituting the exact form of the criterion function (8) in (9) we get the following constrained clustering problem:

$$\mathbf{C}^*_{\mathcal{L}}(\mathcal{X}) = \underset{\mathbf{C} \in \mathbf{\Phi}_{\mathcal{L}}}{\operatorname{argmin}} \sum_{C_i \in \mathbf{C}} \sum_{x \in C_i} D_T^2(M_{C_i}, M_x) \tag{10}$$

where $\mathbf{\Phi}_{\mathcal{L}}$ is defined in (4), D_T is the geodesic distance on the Grassman manifold and M_{C_i} is the mean tangent of cluster C_i.

The problem of Eq. (9) can be solved with dynamic programming, i.e. by incrementally creating the optimal partitions of different sizes starting with size 1 and exploring all possible ways to scale up. To be more specific, from [30], the constrained clustering problem of Eq. (9) can be expressed with the generalized Jensen equality [31]:

$$\mathbf{C}^*_{\mathcal{L}}(\mathcal{X}) = \begin{cases} \{\mathcal{X}\}, & \mathcal{L} = 1 \\ \mathbf{C}^*_{\mathcal{L}-1}(\mathcal{X} \backslash C^*) \cup \{C^*\}, & \mathcal{L} > 1 \end{cases} \tag{11}$$

where

$$C^* = \underset{\substack{\emptyset \subset C \subset \mathcal{X} \\ \exists \mathbf{C} \in \mathbf{\Phi}_{\mathcal{L}-1}(\mathcal{X} \backslash C) : \mathbf{C} \cup \{C\} \in \mathbf{\Phi}_{\mathcal{L}}(\mathcal{X})}}{\operatorname{argmin}} (P(\mathcal{X} \backslash C) + p(C)) \tag{12}$$

The symbol $\backslash$ stands for set subtraction and $\cup$ for set addition. This is a dynamic programming equation that may lead to polynomial time solutions under certain constraints, depending on the characteristics of the clustering problem [32]. However, in the general case, this approach gives rise to algorithms that have exponential time complexity.

An alternative way of solving problems of the form of Eq. (9) is presented in [33]. It allows for more efficient, but less accurate, algorithms as it proposes the use of a greedy framework instead of the dynamic programming one. We opt for such an alternative approach for solving the problem in Eq. (9).

In order to get to our greedy framework, we need a measure for comparing clusters and deciding on proper merging choices. Thus, we define the dissimilarity measure $d : (C_i, C_j) \to \mathbb{R}_0^+$ as the difference in the criterion function before and after the merging of two clusters, i.e.,

$$d(C_i, C_j) = p(C_i \cup C_j) - p(C_i) - p(C_j), \tag{13}$$

assuming that the merging of any two fusible clusters always gives rise to a cluster with a higher score in terms of the criterion function. Under some mild assumptions on the relations between P, d and Φ [33], we can now rewrite Eq. (9) as

$$\mathbf{C}^*_{\mathcal{L}}(\mathcal{X}) = (\mathbf{C}'_{\mathcal{L}+1}(\mathcal{X}) \backslash \{C'_i, C'_j\}) \cup \{C'_i \cup C'_j\} \tag{14}$$

where

$$\left(\mathbf{C}'_{\mathcal{L}+1}(\mathcal{X}), C'_i, C'_j\right) = \underset{\substack{C_i, C_j \in \mathbf{C} \\ \mathbf{C} \in \mathbf{\Phi}_{\mathcal{L}+1} \\ \psi(C_i, C_j) \text{ is true}}}{\operatorname{argmin}} \left(P(\mathbf{C}) + d\left(C_i, C_j\right)\right)$$

This equation still suggests a dynamic programming solution. The difference with Eq. (11) is that in Eq. (14) we move from higher values of L to lower ones, i.e., in order to find the best partition of size $\mathcal{L}$, we check all partitions of size $\mathcal{L}+1$ for the pair of fusible clusters that can be merged with the minimum cost.

From Eq. (14), it is now straightforward to derive a greedy approximation strategy for the clustering problem by eliminating the search over the set $\Phi_{\mathcal{L}+1}$, i.e.,

$$\hat{\mathbf{C}}_{\mathcal{L}}(\mathcal{X}) = (\hat{\mathbf{C}}_{\mathcal{L}+1}(\mathcal{X}) \backslash \{C_i', C_j'\}) \cup \{C_i' \cup C_j'\} \quad (15)$$

where

$$(C_i', C_j') = \underset{\substack{C_i, C_j \in \hat{\mathbf{C}}_{\mathcal{L}+1}(\mathcal{X}) \\ \psi(C_i, C_j) \text{ is true}}}{\operatorname{argmin}} d(C_i, C_j)$$

With this approach, we reduce significantly the computational complexity of the scheme, as we do not perform an exhaustive search over all possible partitions of size $\mathcal{L}+1$ anymore. Instead, we rely on one partition of size $\mathcal{L}+1$, the $\hat{\mathbf{C}}_{\mathcal{L}+1}(\mathcal{X})$, and perform a search over all fusible pairs of clusters in this one. However, as it is often the case with greedy strategies, we cannot guarantee the optimality of the resulting partitions $\hat{\mathbf{C}}_{\mathcal{L}}(\mathcal{X})$ anymore.

5. Greedy cluster merging for locally linear approximation

Following the greedy strategy that is introduced in the previous section, our manifold approximation algorithm is based on grouping the manifold samples $\mathcal{X}$ according to local tangent spaces, in order to minimize the cost function in Eq. (10) and to preserve the manifold geometry. Our method is divided in two main steps. First, we perform the necessary preprocessing steps on the samples in order to compute the graph $G_{\mathcal{X}}$ and the tangent spaces M_x. Second, we use the graph $G_{\mathcal{X}}$ and the tangent spaces M_x's to greedily merge the samples into clusters according to Eq. (15) until we reach a feasible partition with $\mathcal{L}$ components. The block diagram of the method is presented in Fig. 2.

5.1. Tangent space

In the first step of the algorithm, our objective is to compute the neighborhood graph $G_{\mathcal{X}}$ and the local tangent space M_x for each sample $x \in \mathcal{X}$. There exist various ways to construct $G_{\mathcal{X}}$. We have chosen to use the simplest one, namely the k-nearest neighbor approach, i.e., we connect each sample in $\mathcal{X}$ with its k-nearest neighbors. The resulting graph $G_{\mathcal{X}}$ is assumed to be undirected and symmetric. For each sample x we can then define a neighborhood $N_x = \{y \in \mathcal{X} : (x,y) \in E\}$ as the set of samples that are connected to x by an edge in $G_{\mathcal{X}}$. Then, we can approximate the tangent space at x by the d-dimensional subspace of $\mathbb{R}^N$ that best approximates the data in N_x. Equivalently, we compute M_x as the d-dimensional subspace of $\mathbb{R}^N$ that best approximates the neighborhood N_x^0 i.e., N_x shifted to the origin.[2] In other words, we need to compute the best d-rank approximation of the data matrix corresponding to N_x^0, denoted as $[N_x^0]$. Based on Eckart–Young theorem [34], this approximation is equal to the d-rank SVD of $[N_x^0]$. Therefore, the tangent space M_x corresponds to the subspace spanned by the left eigenvectors of the d dominant singular values of $[N_x^0]$.

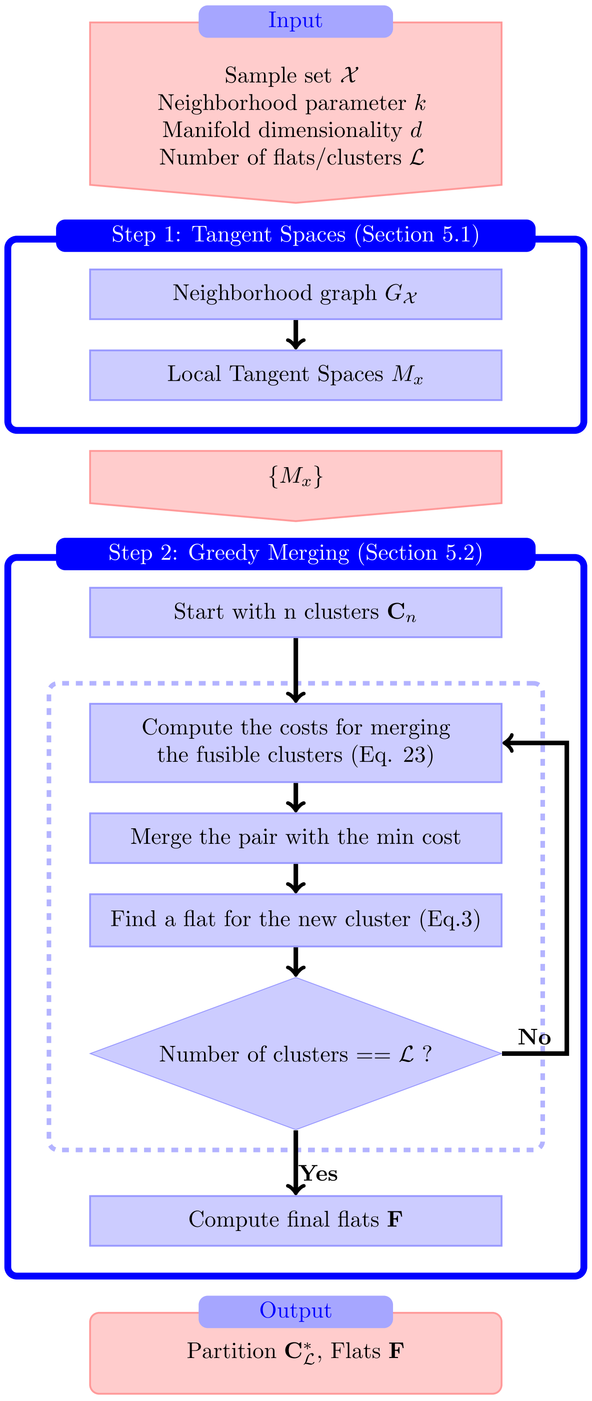

Input
Sample set $\mathcal{X}$
Neighborhood parameter k
Manifold dimensionality d
Number of flats/clusters $\mathcal{L}$
Step 1: Tangent Spaces (Section 5.1)
Neighborhood graph $G_{\mathcal{X}}$
Local Tangent Spaces M_x
$\{M_x\}$
Step 2: Greedy Merging (Section 5.2)
Start with n clusters $\mathbf{C}_n$
Compute the costs for merging the fusible clusters (Eq. 23)
Merge the pair with the min cost
Find a flat for the new cluster (Eq.3)
Number of clusters == $\mathcal{L}$?
No
Yes
Compute final flats $\mathbf{F}$
Output
Partition $\mathbf{C}^*_{\mathcal{L}}$, Flats $\mathbf{F}$

Fig. 2. The block diagram of the system.

The first step of our scheme is not the main focus of our work. Its purpose is to infer the local geometry of the manifold and as such can be replaced by any other algorithm that achieves the same objective. We have made simple choices, namely a k-nearest neighbor algorithm for representing the local manifold geometry and SVD decomposition for tangent computation. Our goal in this paper is to show that the tangent space information can be

[2] We apply a shift operator $T_{\vec{x}}$ to the whole neighborhood N_x, where $\vec{x}$ is the vector corresponding to the sample x in $\mathbb{R}^N$. This operator moves x to the origin and brings along all its neighborhood, while preserving all distances in it.

effective in manifold approximation even when tangents are computed with naive techniques. More sophisticated techniques can be used for tangent computation e.g. [35,22], as any further improvement of this step can only benefit the overall algorithm. For example, one can account for the noise in the data using the method shown in Fig. 6. After computing the tangents with the process described above, an additional improvement step is performed by averaging neighboring tangents. In this way, the final tangents are smoothed and the effect of noise is almost canceled. However, such alternative solutions for tangent space computation go beyond the scope of this work.

5.2. *Greedy merging*

Once the graph and the tangent spaces have been computed, we proceed with solving the optimization problem presented in Eq. (10). In order to minimize the cost function, we follow the method presented in Eq. (15). We start with $n=|\mathcal{X}|$ separate clusters, one for each sample. This is the optimal partition for n clusters, i.e., $\mathbf{C}_n^* = \{\{x\}, x\in\mathcal{X}\}$. Then, we reduce the number of clusters iteratively, by merging the clusters C_i and C_j with the minimum dissimilarity, until we reach the desired number of clusters $\mathcal{L}$.

At each iteration, there exists a set of possible mergings between the clusters in $\mathbf{C}$. The fusibility predicate, given in Eq. (5), defines the sufficient condition for a pair of clusters to be fusible: any cluster C_i can be merged with any of its neighbors, i.e., the set $NG_{C_i} = \{C_j\colon \exists x\in C_i,\ \exists y \in C_j \text{ s.t } (x,y)\in E\}$. The dissimilarity between C_i and $C_j \in NG_{C_i}$ is given by Eq. (13) and Eq. (8) as

$$\begin{aligned} d(C_i,C_j) &= \sum_{x\in C_i\cup C_j} D_T^2(M_x, M_{C_i\cup C_j}) - \sum_{x\in C_i} D_T^2(M_x,M_{C_i}) \\ &\quad - \sum_{x\in C_j} D_T^2(M_x,M_{C_j}) \\ &= \sum_{x\in C_i} D_T^2(M_x, M_{C_i\cup C_j}) - \sum_{x\in C_i} D_T^2(M_x,M_{C_i}) \\ &\quad + \sum_{x\in C_j} D_T^2(M_x, M_{C_i\cup C_j}) - \sum_{x\in C_j} D_T^2(M_x,M_{C_j}) \end{aligned} \tag{16}$$

Note that, since M_{C_i} and M_{C_j} are respectively the mean tangents of C_i and C_j, each of them is the subspace that minimizes the sum of the square distances from the tangents in each cluster (see Eq. (2)). As a result, $M_{C_i\cup C_j}$ can only produce the same or a higher value than M_{C_i} when measuring the sum of square distances from the tangents in C_i. In other words, $\sum_{x\in C_i} D_T^2(M_x, M_{C_i\cup C_j})$ is greater or equal to $\sum_{x\in C_i} D_T^2(M_x,M_{C_i})$. The same holds for the cluster C_j. Therefore, $d(C_i,C_j)$ is always non-negative.

However, it is costly to compute Eq. (16) for all feasible mergings as it requires the computation of the mean tangent for all possible merged clusters. We would rather use a measure that depends only on the information that is already available to the algorithm, i.e., the means of the clusters that we have computed so far and their distances to the tangents in their clusters. Moreover, since we are using a greedy bottom-up approach with an initial cost equal to zero, we have to ensure that, at each iteration of the algorithm, the chosen merging does only marginally increase the overall cost. Therefore, an upper bound for $d(C_i,C_j)$ that depends only on the means of the existing clusters would be a suitable approximate dissimilarity measure $\tilde{d}(C_i,C_j)$ for our algorithm. It would contribute in reducing the complexity of the algorithm by limiting the amount of necessary computations at each iteration.

In order to compute our approximate measure $\tilde{d}(C_i,C_j)$, we need to perform a series of steps. First, we observe that

$$\sum_{x\in C_i} D_T^2(M_x, M_{C_i\cup C_j}) \le \sum_{x\in C_i} D_T^2(M_x, M_{C_j}), \tag{17}$$

which means that the mean tangent of $C_i\cup C_j$ is closer to the mean tangent of C_i than the mean tangent of C_j. This statement, which also holds if we interchange the clusters C_i and C_j, is inevitably true. Indeed, by contradiction, if $\sum_{x\in C_i} D_T^2(M_x, M_{C_i\cup C_j})$ is larger than $\sum_{x\in C_i} D_T^2(M_x, M_{C_j})$, then $\sum_{x\in C_i\cup C_j} D_T^2(M_x, M_{C_i\cup C_j})$ is also strictly larger than $\sum_{x\in C_i\cup C_j} D_T^2(M_x, M_{C_j})$. But, this contradicts the optimal character of $M_{C_i\cup C_j}$ for representing $C_i\cup C_j$ in terms of the projection distance.

Then, by substituting Eq. (17), and its equivalent form for C_j in Eq. (16), we have

$$\begin{aligned} d(C_i,C_j) \le & \sum_{x\in C_i} [D_T^2(M_x,M_{C_j}) - D_T^2(M_x,M_{C_i})] \\ & + \sum_{x\in C_j} [D_T^2(M_x,M_{C_i}) - D_T^2(M_x,M_{C_j})] \end{aligned} \tag{18}$$

Moreover, by the triangle inequality:

$$D_T(M_x,M_{C_i}) \le D_T(M_x,M_{C_j}) + D_T(M_{C_i},M_{C_j}), \quad \forall x\in\mathcal{X} \tag{19}$$

$$D_T(M_x,M_{C_j}) \le D_T(M_x,M_{C_i}) + D_T(M_{C_i},M_{C_j}), \quad \forall x\in\mathcal{X} \tag{20}$$

Taking the square of these inequalities and summing over C_j and C_i respectively we get

$$\begin{aligned} &\sum_{x\in C_j} [D_T^2(M_x,M_{C_i}) - D_T^2(M_x,M_{C_j})] \\ &\qquad \le 2D_T(M_{C_i},M_{C_j}) \sum_{x\in C_j} D_T(M_x,M_{C_j}) + |C_j| D_T^2(M_{C_i},M_{C_j}) \\ &\sum_{x\in C_i} [D_T^2(M_x,M_{C_j}) - D_T^2(M_x,M_{C_i})] \\ &\qquad \le 2D_T(M_{C_i},M_{C_j}) \sum_{x\in C_i} D_T(M_x,M_{C_i}) + |C_i| D_T^2(M_{C_i},M_{C_j}) \end{aligned} \tag{21}$$

Substituting Eq. (21) into Eq. (18) we finally have the following upper bound for the dissimilarity measure:

$$\begin{aligned} d(C_i,C_j) \le & (|C_i|+|C_j|) D_T^2(M_{C_i},M_{C_j}) \\ & + 2D_T(M_{C_i},M_{C_j}) \left[\sum_{x\in C_i} D_T(M_x,M_{C_i}) + \sum_{x\in C_j} D_T(M_x,M_{C_j}) \right], \end{aligned} \tag{22}$$

which depends only on the pre-computed information. Therefore, we can define our approximate dissimilarity measure $\tilde{d}(C_i,C_j)$ as

$$\begin{aligned} \tilde{d}(C_i,C_j) = & (|C_i|+|C_j|) D_T^2(M_{C_i},M_{C_j}) \\ & + 2D_T(M_{C_i},M_{C_j}) \left[\sum_{x\in C_i} D_T(M_x,M_{C_i}) + \sum_{x\in C_j} D_T(M_x,M_{C_j}) \right], \end{aligned} \tag{23}$$

By comparing Eq. (23) with Eq. (16), we can observe that Eq. (23) is indeed more computationally efficient as it involves only the means of the existing clusters and not those of the clusters after merging the fusible pairs. In our algorithm, the costs for all possible mergings at each iteration are thus computed according to Eq. (23). The clusters with the minimum estimated merging cost are then combined and the mean of the newly formed cluster is computed as shown in Section 4.3. The procedure is then repeated until we reach the desired number of clusters $\mathcal{L}$.

At the end, each cluster represents a group of samples that can be well approximated by a d-dimensional flat. We compute the final flats for each cluster and we use the subspace spanned by the left eigenvectors corresponding to the d dominant singular values of each cluster's data matrix as a representative subspace. The overall manifold approximation algorithm is summarized in Algorithm 1.

Algorithm 1. Agglomerative clustering based on differences of tangents (ACDT), Part 1.

```
Input: $\mathcal{X}, k, \mathcal{L}, d$
      Step 1    *Preprocessing, Section 5.1*
1:    Construct $G(\mathcal{X}, E)$ by connecting each element in $\mathcal{X}$ with its
      $k$-nearest neighbors.
2:    for all $x \in \mathcal{X}$ do
3:        $N_x = \{y \in \mathcal{X} : (x,y) \in E\}$    *Compute neighborhoods*
4:        $[N_x^0] = USV^T$
      where $[N_x^0]$ is the data matrix formed by the elements in $N_x$
      shifted to the origin of $\mathbb{R}^N$ and $U, S, V$ are the results of its
      d-rank SVD.
5:        $M_x = U$    *Compute tangent spaces*
6:    end for
      Step 2    *Greedy computation of partition $\mathbf{C}_{\mathcal{L}}^*$*
7:    $n = |\mathcal{X}|$, $\lambda = 0$, $\mathbf{C}_n^* = \{\{x\} : x \in \mathcal{X}\}$    *Initialization*
8:    for $\lambda < n - \mathcal{L}$ do    *Greedy merging, Section 5.2*
9         $(C_i', C_j') = \underset{\substack{C_i, C_j \in \mathbf{C}_{n-\lambda}^* \\ \psi(C_i, C_j) \text{ is true}}}{\operatorname{argmin}} \tilde{d}(C_i, C_j)$    *Eq. (23)*
10:       $\mathbf{C}_{n-\lambda+1}^* = (\mathbf{C}_{n-\lambda}^* \setminus \{C_i', C_j'\}) \cup \{C_i' \cup C_j'\}$
11:       Compute $M_{C_i' \cup C_j'}$    *Eq. (3)*
12:       $\lambda = \lambda + 1$
13:   end for
14:   for $C_i \in \mathbf{C}_{\mathcal{L}}^*$ do    * Compute the final flats $F_i$*
15:       $[C_{m_i}^0] = USV^T$
      where $m_i$ is the sample mean of $C_i$, $[C_{m_i}^0]$ is the data matrix
      formed by the samples in $C_i$ shifted by $m_i$ and $U, S, V$ are the
      results of its d-rank SVD.
16:       $F_i = U$
17:   end for
Output: $\mathbf{C}_{\mathcal{L}}^*$, $\mathbf{F}$
```

5.3. Computational complexity

We now analyze briefly the complexity of both versions of the manifold approximation algorithm, that respectively use the exact dissimilarity measure of Eq. (16) and the approximate measure of Eq. (23). The preprocessing step is the same for both schemes and it is skipped in the following analysis. Then, the operations that are time consuming in our scheme are the tangent distance computations and the computation of mean tangents. In the following we will consider that both have similar computational costs.

Computing the cost of a possible merging with Eq. (23) requires only the computation of one additional tangent distance at each step. Denoting by $K_{n-\lambda}$ the number of possible mergings in the clustering $\mathbf{C}_{n-\lambda}$ at step λ, the complexity of one step of the greedy merging (line 9 in Algorithm 1) requires $K_{n-\lambda}$ computations of tangent distances. Then, the operation at line 11 also requires one mean tangent computation plus $|C_i' \cup C_j'|$ tangent distance computations for the newly formed cluster. Therefore, the greedy merging (lines 8–12 in Algorithm 1) will be performed with a time complexity of $T_{approx}(n) = O(\sum_{\lambda=1}^{n-\mathcal{L}}(1 + |C_i' \cup C_j'| + K_{n-\lambda}))$ where n is the number of data samples.

On the other hand, if the exact dissimilarity measure was used, Eq. (16) would require one mean tangent computation plus $|C_i' \cup C_j'|$ tangent distance computations for every possible merging. Then, after picking the winning merging, no further actions would be required. In total, the scheme would have a time complexity of $T_{exact}(n) = O\left(\sum_{\lambda=1}^{n-\mathcal{L}}(1 + |C_i' \cup C_j'|)K_{n-\lambda}\right)$.

To complete our analysis, we need to estimate the number of possible mergings $K_{n-\lambda}$. Since, at each step of the algorithm, we perform one merging operation, we will have exactly $n-\lambda$ clusters at step λ. Moreover, each $C_i \in \mathbf{C}_{n-\lambda}$ will have on average a size equal to $|\tilde{C}_i| = n/|\mathbf{C}_{n-\lambda}| = n/(n-\lambda)$ and therefore we have that C_i has at most $kn/(n-\lambda)$ different neighbors. Thus, the number of possible mergings is at most $K_{n-\lambda} \le \frac{1}{2}|\mathbf{C}_{n-\lambda}| kn/(n-\lambda) = \frac{1}{2}kn$.

By substituting $K_{n-\lambda}$ from above and $|C_i' \cup C_j'|$ with its average in $T_{approx}(n)$ we get

$$T_{approx}(n) = O\left(\sum_{\lambda=1}^{n-\mathcal{L}}\left(1 + 2\frac{n}{n-\lambda} + \frac{1}{2}kn\right)\right)$$

$$= O\left(\sum_{\lambda=1}^{n-\mathcal{L}}\frac{1}{2}kn\right) = O(n^2) \tag{24}$$

For the exact dissimilarity measure we have

$$T_{exact}(n) = O\left(\sum_{\lambda=1}^{n-\mathcal{L}}\left(1 + \frac{n}{n-\lambda}kn\right)\right)$$

$$= O(kn^2(H_{n-1} - H_{\mathcal{L}-1})) = O(n^2 \ln n) \tag{25}$$

which is higher than the average running time of our approximate algorithm.[3] Therefore, we can clearly see that the use of the approximate dissimilarity measure is beneficial for the computational complexity of the algorithm.

6. Experimental results

We have conducted two different sets of experiments to study the performance of our manifold approximation scheme. In the first one, we have tested the performance in approximating the manifold data for both synthetic and real datasets. In the second one, we have studied the use of the flats as projective spaces in the place of manifolds and evaluated their discriminative power on the MNIST dataset of handwritten digits [36].

[3] H_n and $H_{\mathcal{L}}$ are the harmonic numbers of order n and $\mathcal{L}$ respectively.

6.1. Manifold Data approximation

We compare our scheme (ACDT) with two other manifold approximation approaches from the literature, namely the Hierarchical Divisive Clustering (HDC) [3] and the Hierarchical Agglomerative Clustering (HAC) [19]. The HDC algorithm starts with considering all the data as one cluster and then hierarchically partitions them by dividing highly non-linear clusters. As a linearity measure, it uses the deviation between the Euclidean and geodesic distances, i.e., each cluster gets a nonlinearity score that is equal to the average ratio of the geodesic distance over the Euclidean one for all the pairs of samples in the cluster. The process continues until all existing clusters have a nonlinearity score that is lower than a given threshold. On the other hand, HAC is a bottom-up algorithm, i.e., each sample is considered at the beginning as a separate cluster and then clusters are merged iteratively until their number reduces to the desired target. At each iteration of the algorithm, the pair of clusters with the minimum distance is merged. The distance between two clusters is measured as the average geodesic distance between the samples of the one cluster and the samples of the other. Our scheme follows also a bottom-up strategy; however, our distance measure is completely different than the one in [19]. Instead of relating our merging decisions to the average geodesic distances, we use the variance of tangents to decide on proper mergings. This choice has been motivated by the definition of tangents as the best locally linear approximations of manifolds and has been proven very effective in practice.

In order to quantify the performance of the compared schemes, we use the mean squared reconstruction error (MSRE). The MSRE is defined as

$$MSRE = \frac{1}{N}\sum_{i=1}^{N} \|x_i - \hat{x}_i\|^2$$

where x_i and $\hat{x}_i$ are respectively a data sample and its projection on the corresponding approximating flat, while N is the total number of signals. For the HDC and HAC algorithms, whose output is a set of clusters and not a set of representative flats, we compute the corresponding flats by principal component analysis on the data of each cluster. The results of our experiments for all three algorithms are given below for three different datasets.

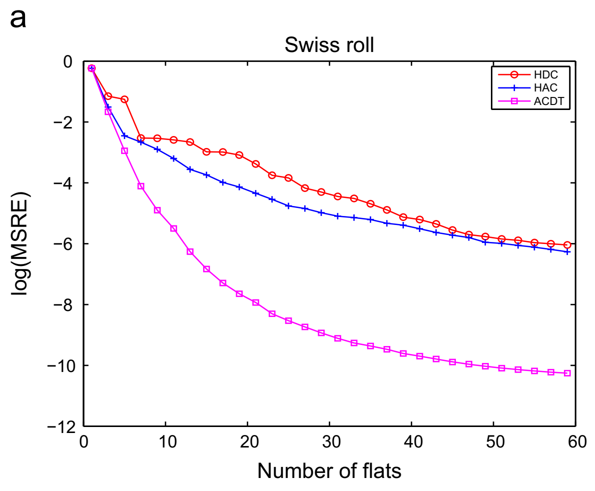
6.1.1. Synthetic data

Firstly, we test the performance of our scheme in approximating synthetic manifolds. We use the Swiss roll and the S-curve dataset. The training set for both cases consists of 5000 points, randomly sampled from the manifolds. The neighborhood size k is set equal to 15 in the experiments. We have observed that it is preferable to use low values for k, varying from 0.5% to 2% of the total number of samples, in order to avoid “short-circuit” effects that distort the manifold structure.

The MSRE versus the number of flats, for our synthetic manifolds, is presented in Fig. 3. The results are averaged over 10 randomly chosen training sets. From Fig. 3, we can see that our scheme approximates better the manifold structure than the other approaches. The approximation performance is better even for a small number of flats but the differences are more evident in the mid-range cases where the number of flats is between 15 and 30. For higher number of flats, the difference stabilizes around 50–60 flats when the MSREs of the algorithms converge. The effectiveness of our method is mainly due to the use of the difference of tangents for measuring the linearity of sample sets instead of the geodesic-based criteria used by other algorithms [3,19]. For the sake of completeness, we also give in Table 1 the running times for the three algorithms in

Table 1
Running times for the three algorithms in case of the Swiss role data and 60 flats. The results were obtained on an Intel Core Duo 2.66 GHz, MacBook Pro.

	ACDT	HDC	HAC
Running time (s)	389	15	288



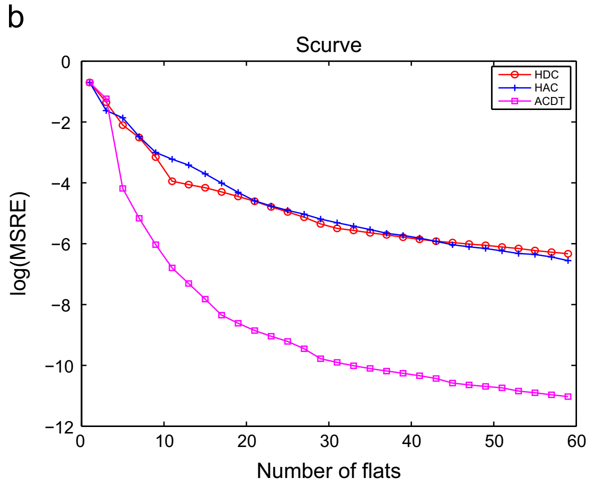


Fig. 3. Mean squared reconstruction error (MSRE) versus the number of flats. The error on the *y*-axis is shown in a logarithmic scale. (a) Swiss roll. (b) *S*-curve.

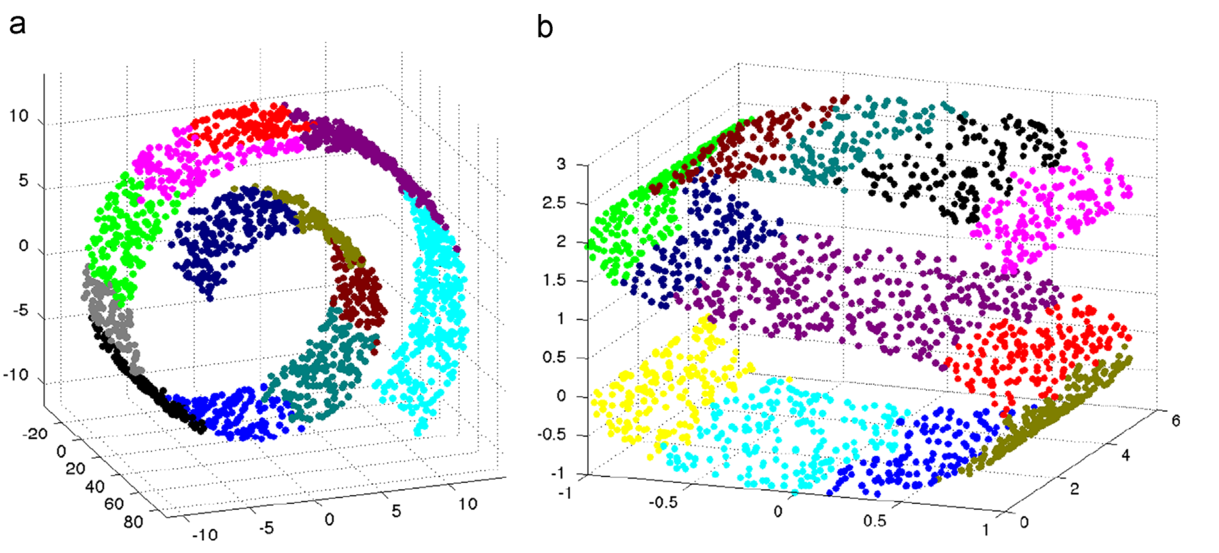


Fig. 4. The final groups formed by the proposed approximation algorithm with 12 flats. Each color represents a different cluster of points. (a) Swiss roll. (b) *S*-curve. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

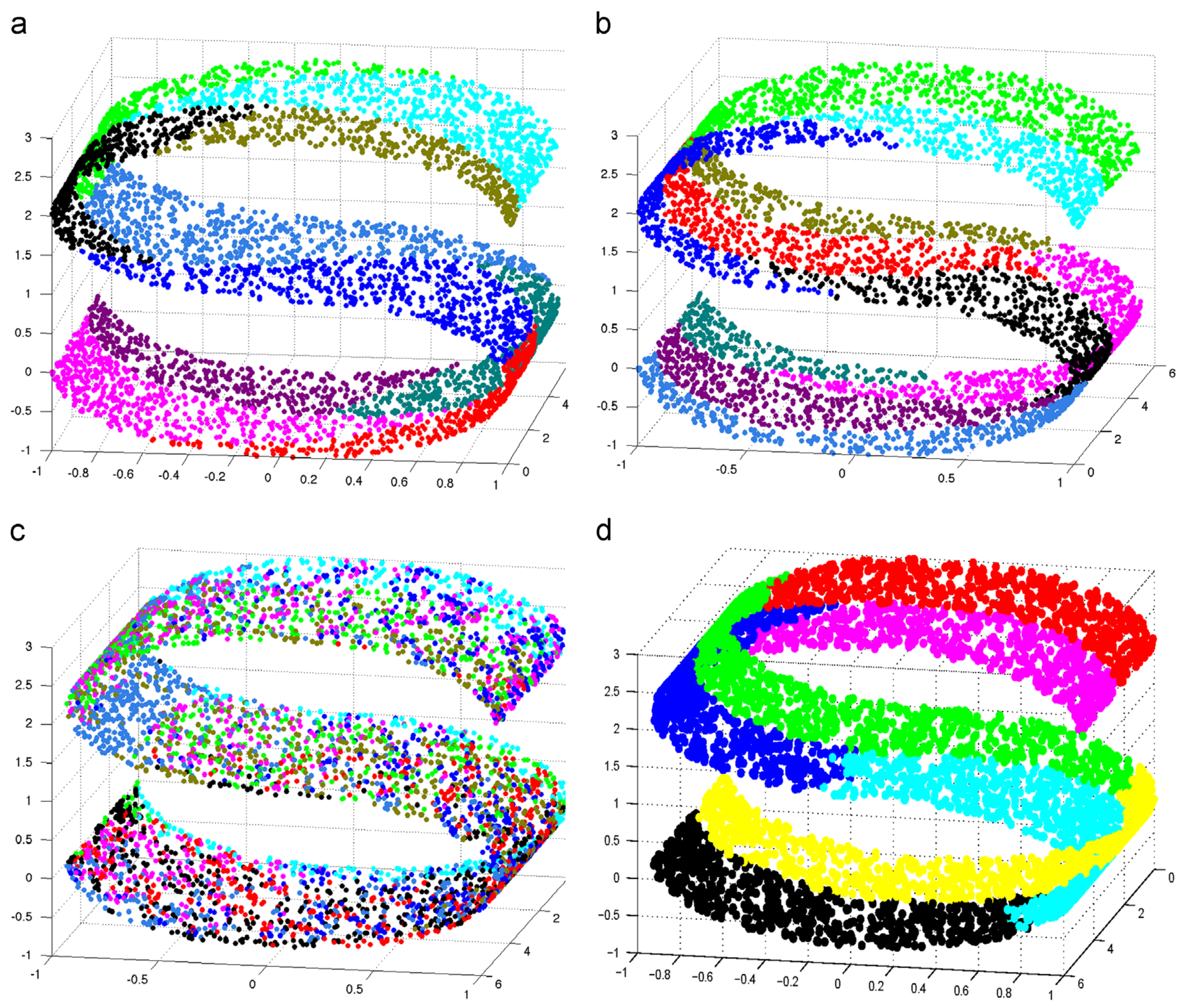


Fig. 5. The final groups formed by the HDC (a) HDC, (b) HAC, (c) LSA, (d) spectral clustering algorithms with 10 flats. Each color represents a different cluster of points.

the case of 60 flats. We can see that the two bottom-up schemes are a bit penalized in terms of complexity as they start with a high number of clusters (equal to the number of points) and proceed with mergings until they reach the desired number of clusters, which is significantly smaller in this experiment. On the other side, HDC has to perform fewer splittings, as it starts with considering all points as one cluster. As far as ACDT is concerned, we would like to note that there is still room for improvement as the code used is far from optimized. For example, a significant gain could be achieved by optimizing the SVD computations but this is beyond the scope of our paper.

Finally, an example of the final groups computed by our algorithm is shown in Fig. 4 for the case of 12 flats. In this figure, we see that the structure of the manifold is correctly preserved by the proposed manifold approximation algorithm. The final groups for the HDC and the HAC algorithm for the S-curve data are shown in Fig. 5. Moreover, to strengthen our argument on the inappropriateness of general subspace clustering methods for manifold data, we also

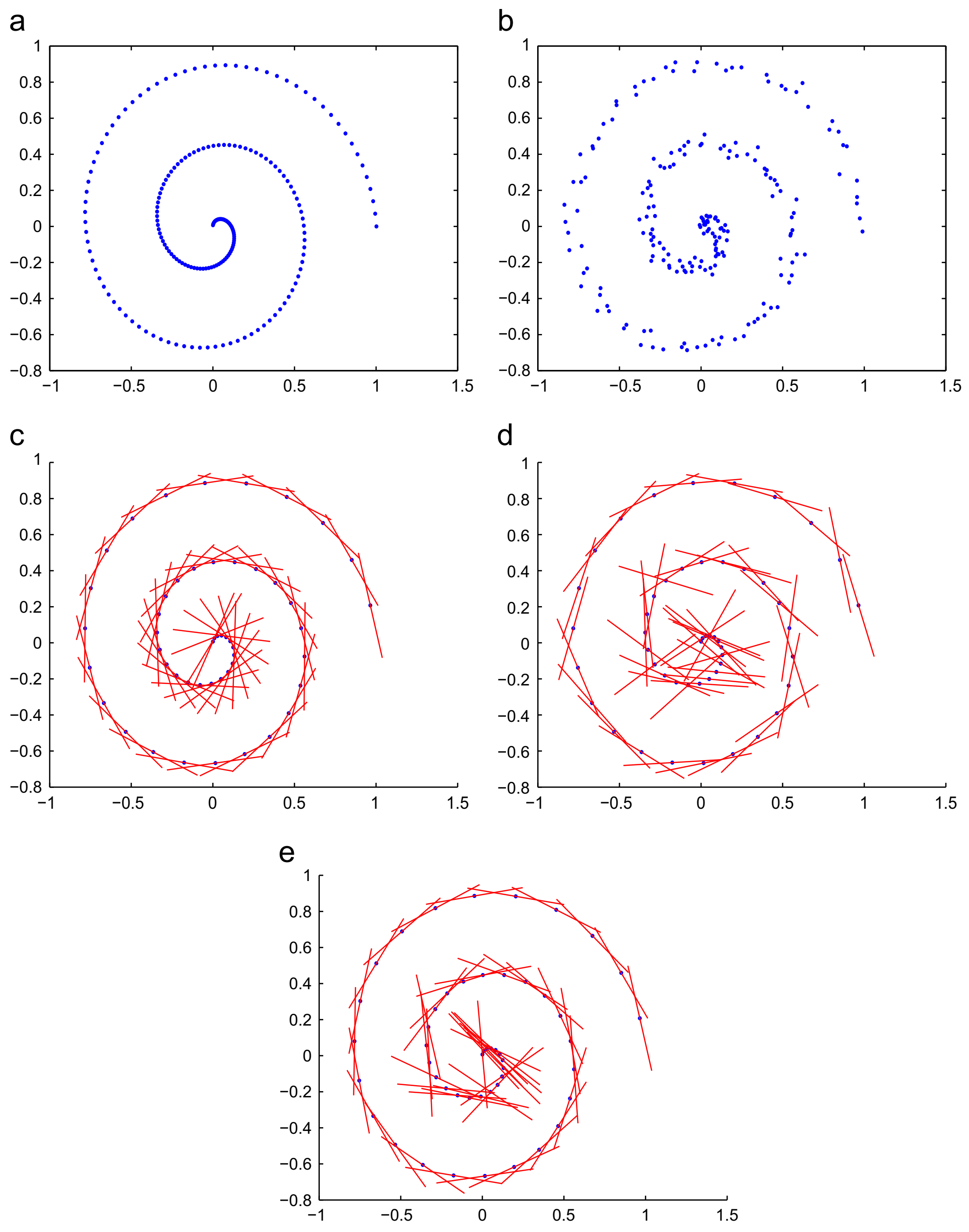


Fig. 6. An example of smoothing of the tangents in case of noisy data. As we can observe, the smoothing process (averaging in this case) improves significantly the appearance the computed tangents, resulting in almost removing the side effects of noise. (a) Non-noisy data. (b) Noisy data. (c) Tangents based on non-noisy data. (d) Tangents based on noisy data. (e) Noisy tangents after smoothing.

provide the results of Local Subspace Affinity (LSA) method [37] and spectral clustering in the same figure. For spectral clustering we used the same k-nearest neighbor graph as for our own scheme, weighted with tangent distances by the formula $w_{ij} = 1 - D_T(M_{x_i}, M_{x_j})/\max_{D_T}$, where $\max_{D_T}$ is the maximum tangent distance over the whole dataset. As we can see clearly from the plots, all algorithms fail to uncover clusters that comply with the manifold geometry. The spectral clustering, HDC and HAC achieve better results than LSA but when compared to ACDT it is obvious that they orient their clusters in the wrong way.

6.1.2. Natural patches

We have also tested the performance of our scheme in approximating natural image patches since they are often assumed to form a lower dimensional manifold, e.g. [38]. The manifold samples are taken from the training set of the Berkeley Segmentation Dataset (BSDS) [39]. Each patch is of size 8×8 and captures a square region of a natural image. Before approximating the manifold, we preprocess the patches so that they have zero mean and unit variance. For constructing the manifold we use 10,000 patches and k is set equal to 100.

The approximation performance (in terms of the MSRE) versus the number of flats is presented in Fig. 7. We have plotted the approximation error for three different choices

Fig. 8. Example faces from the VidTIMIT database after face detection and downsampling. The size of the images is 26×26.

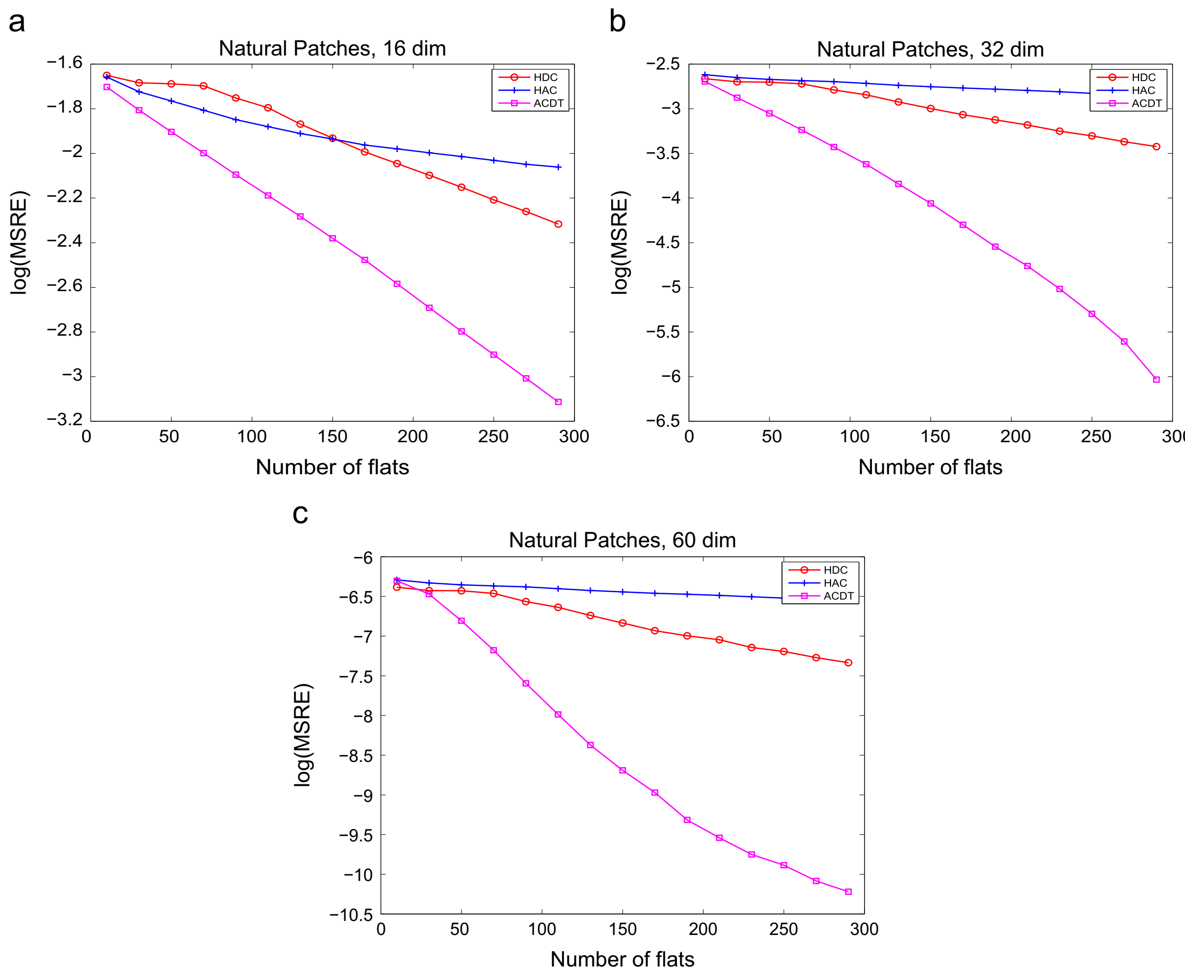


Fig. 7. MSRE for natural patches for different choices of the flats' dimensionality. The error on the y-axis is shown in a logarithmic scale. (a) 16-Dimensional flats. (b) 32-Dimensional flats. (c) 60-Dimensional flats.

of the flats' dimensionality, i.e., $d = 16$, 32 and 60 respectively. As we can observe from the plots, in all cases, our scheme approximates significantly better the manifold structure than the other approaches and the differences increase as the dimensionality of flats increases. The performance of the HDC and the HAC scheme is quite similar with the HDC usually outperforming the HAC. These results suggest that our approximation algorithm is very promising even in cases where the underlying structure of the data cannot be easily identified.

The effectiveness of our method is mainly due to the use of the difference of tangents for measuring the linearity of sample sets instead of the geodesic-based criteria used by other algorithms

6.1.3. *VidTIMITT faces*

In a last set of experiments, we have also tested the approximation power of ACDT on faces taken from the VidTIMIT database [40]. This face database contains three different video sequences for 43 subjects. In each video sequence, the person performs a head rotation starting from the frontal position and moving sequentially to the right, left, center, up and down. For our experiments, we have first isolated the faces with the P. Viola's face detector

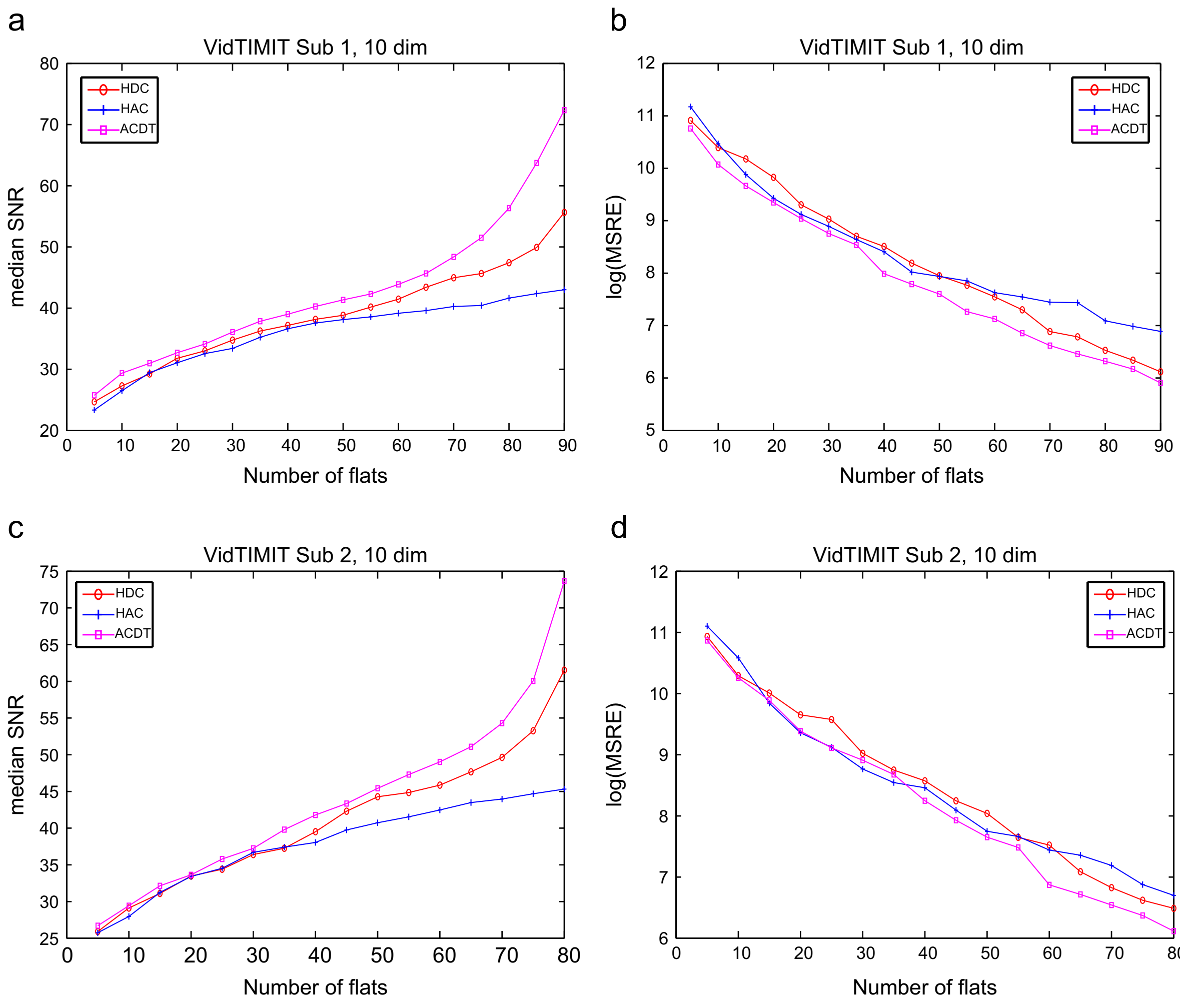


Fig. 9. (a) Median SNR for Subject 1. (b) MSRE for Subject 1. (c) Median SNR for Subject 2. (d) MSRE for Subject 2.

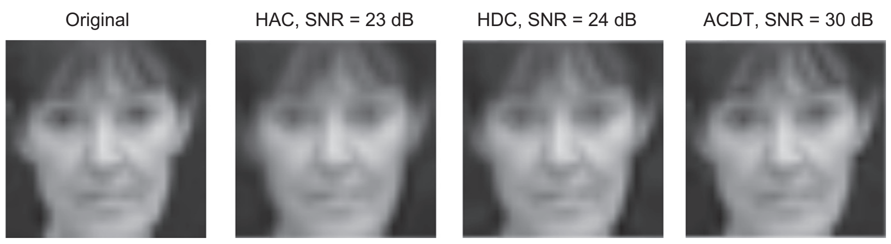


Fig. 10. The reconstruction of a sample face based on the approximating flats.

[41] from all the video sequences and then downsampled the images to size 26×26. Some resulting example faces are shown in Fig. 8.

Based on the assumption that all face images belonging to the same subject form a low dimensional manifold, we have used the previous algorithms to approximate this

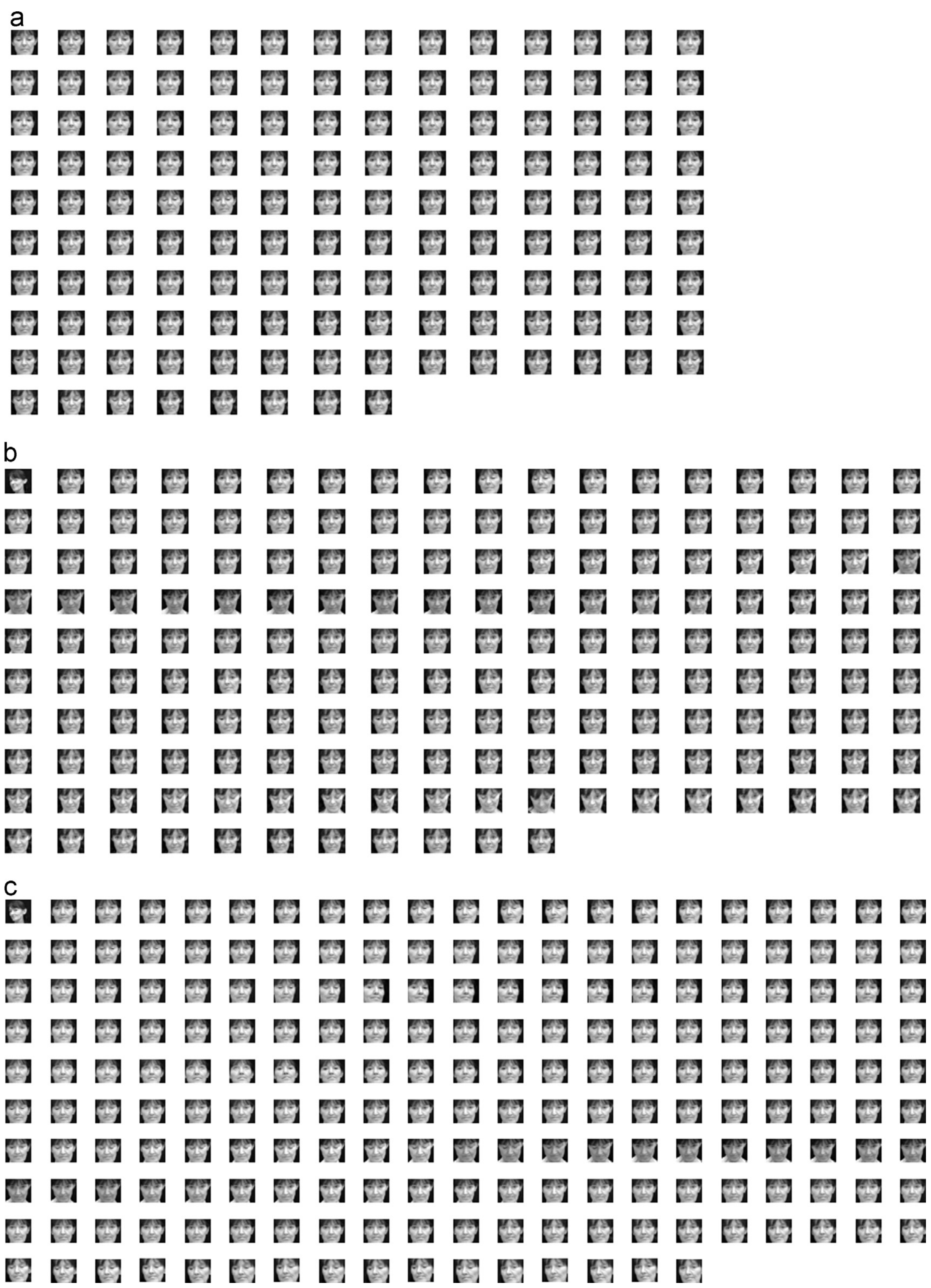


Fig. 11. (a) The corresponding group of sample image in Fig. 10 according to ACDT. (b) The corresponding group of sample image in Fig. 10 according to HDC. (c) The corresponding group of sample image in Fig. 10 according to HAC.

manifold with a different number of flats. The dimension of the manifold was set to 10 and the number of neighbors $k = 15$. The results of the approximation for two of the subjects are shown in Fig. 9. For this experiment, in addition to the MSRE, we also provide results in terms of the median SNR in the image reconstruction. As we can see from these plots ACDT generally outperforms the other two algorithms, although the differences are not extremely big. However, there are sample cases where the performance of the schemes is significantly different. Such an example is shown in Fig. 10, where we can see that ACDT achieves a significantly better approximation than the other schemes. The reason is that the group that the sample belongs to with ACDT is more uniform than the corresponding group uncovered by HDC and HAC. These groups are shown in Fig. 11 where it is obvious that the group of ACDT contains mainly frontal poses with open eyes, while the same group in the other algorithms also includes closed eyes and downwards or slightly profile poses.

6.2. *Geometric separability of the flat-based distance features*

We finally check the discriminative power of the flats approximating the manifolds when used as projective spaces in the place of the manifolds themselves. With this experiment we would like to check how the spatial distribution of the data coming from different classes changes in terms of separability when the data are represented by their relative position on the manifolds' approximating flats. For this purpose, we employ the notion of geometric separability of data which is described in [42] as an extension to linear separability; the *Geometric Separability Index* (GSI) of a multiclass dataset is defined as the ratio of the nearest neighbor pairs that share the same class label.

To be more specific, assuming that we have m datasets, each belonging to a different class, we run at first our scheme for approximating the underlying manifold of each class with $\mathcal{L}$ flats. We denote the set of resulting flats by S. Then, for each sample, we create an $m \times \mathcal{L}$ dimensional vector of features, where each entry corresponds to the distance of a data sample to a flat in S. After computing the new flat-based distance features, we can compute the separability index of this dataset.

In Table 2 we can see the separability index computed for a part of the MNIST dataset in different spaces. In the first column, we present the geometric separability in the original, 784 dimensional space, as this is measured by GSI when we use 2000 random samples for each digit. Then, we have used $\mathcal{L} = 10$ flats to approximate each digit's manifold. The GSI of the dataset in the space of the flat-based distance features is shown in the second column. As we can see it is almost the same as that in the original space despite the significant reduction in the dimensionality (original space: 784, reduced space: 100). Finally, in order to have a measure of comparison, we have also checked the separability index of the data after embedding them in lower dimensions with some well-known manifold learning algorithms namely the LLE [12], the Isomap [1] and the Laplacian Eigenmaps [13]. The resulting GSIs show that although there is room of improvement as the Laplacian Eigenmap scheme outperforms our flat-based distances embedding, the performance of our features is still promising as it is better than that of LLE and Isomap. Moreover, our flat-based algorithm can easily used to embed out of sample points; this is unfortunately impossible to do with Laplacian Eigenmaps, which has to recompute the whole manifold to include new training points.

To sum up, based on the relatively good GSI of the features, we can presume that the flats uncovered by our manifold approximation scheme manage to capture and preserve the crucial characteristics of the manifolds, which might prove to be very useful in classification applications. Finally, this simple embedding scheme can represent a method of projecting new samples to manifolds: instead of finding the principal directions of the data and project to them, we compute the principal flats of the manifold and use them to project new samples.

Table 2
Separability index for the MNIST dataset and different embedding algorithms. The original space is of dimension 784. The embedding space has a dimension equal to 100 in all cases.

Original space	Flats	LLE	Isomap	Laplacian
0.969	0.961	0.895	0.856	0.991

7. Conclusion

We have presented a new greedy algorithm for approximating a manifold with low dimensional flats based on the difference of tangent spaces. Our method is shown to be quite powerful for manifold approximation where it outperforms state-of-the-art manifold approximation approaches both in terms of preserving the manifold structure and reducing the approximation error. The final low-dimensional representation of signals from the manifold can be used for data compression or signal classification. In the future, we will explore ways to uncover manifold approximations that are especially useful for classification. We will also extend our method to other problems like image denoising and restoration, manifold to manifold distance computations as well as geodesic distance computations on manifolds.

References

[1] J.B. Tenenbaum, V. Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.

[2] A. Goh, R. Vidal, Clustering and dimensionality reduction on Riemannian manifolds, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008, IEEE, 2008, pp. 1–7.

[3] R. Wang, X. Chen, Manifold discriminant analysis, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, IEEE, 2009, pp. 429–436.

[4] E. Kokiopoulou, P. Frossard, Minimum distance between pattern transformation manifolds: algorithm and applications, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2009) 1225–1238.

[5] R. Cappelli, D. Maio, D. Maltoni, Multispace KL for pattern representation and classification, IEEE Trans. Pattern Anal. Mach. Intell. 23 (2001) 977–996.
[6] T. Zhang, A. Szlam, G. Lerman, Median k-flats for hybrid linear modeling with many outliers, in: IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), IEEE, 2009, pp. 234–241.
[7] C.M. Bishop, Mixtures of probabilistic principal component analyzers, Neural Comput. 11 (1999) 443–482.
[8] R. Vidal, Y. Ma, S. Sastry, Generalized principal component analysis (GPCA), IEEE Trans. Pattern Anal. Mach. Intell. 27 (2005) 1945–1959.
[9] G. Chen, G. Lerman, Spectral curvature clustering (SCC), Int. J. Comput. Vis. 81 (2009) 317–330.
[10] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, D. Kriegman, Clustering appearances of objects under varying illumination conditions, in: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, IEEE, 2003, pp. I–11.
[11] G. Chen, M. Maggioni, Multiscale geometric and spectral analysis of plane arrangements, in: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2011, pp. 2825–2832.
[12] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.
[13] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (2003) 1373–1396.
[14] D.L. Donoho, C. Grimes, Hessian eigenmaps: locally linear embedding techniques for high-dimensional data, Proc. Natl. Acad. Sci. 100 (2003) 5591–5596.
[15] L.J.P. Van der Maaten, E.O. Postma, H.J. van den Herik, Dimensionality Reduction: A Comparative Review, 2007.
[16] R. Pless, R. Souvenir, A survey of manifold learning for images, IPSJ Trans. Comput. Vis. Appl. 1 (2009) 83–94.
[17] S. Roweis, L.K. Saul, G.E. Hinton, Global coordination of local linear models, Advances in Neural Information Processing Systems, vol. 2, 2002, pp. 889–896.
[18] M. Brand, M. Brand, Charting a manifold, in: Advances in Neural Information Processing Systems, pp. 961–968.
[19] W. Fan, D.-Y. Yeung, Locally linear models on face appearance manifolds with application to dual-subspace based classification, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, IEEE, 2006, pp. 1384–1390.
[20] Z. Zhang, H. Zha, Principal manifolds and nonlinear dimension reduction via local tangent space alignment, SIAM J. Sci. Comput. 26 (2002) 313–338.
[21] Y.M. Lui, J.R. Beveridge, Grassmann registration manifolds for face recognition, in: Computer Vision–ECCV 2008, Springer, 2008, pp. 44–57.
[22] H. Xiaofei, L. Binbin, Tangent space learning and generalization, Front. Electr. Electron. Eng. China 6 (2011) 27–42.
[23] M.T. Harandi, C. Sanderson, S. Shirazi, B.C. Lovell, Kernel analysis on Grassmann manifolds for action recognition, Pattern Recognit. Lett. 34 (15) (2013) 1906–1915.
[24] J. Hamm, D. Lee, Grassmann discriminant analysis: a unifying view on subspace-based learning, in: Proceedings of the 25th International Conference on Machine Learning, ACM, pp. 376–383.
[25] M. Spivak, Calculus on Manifolds: A Modern Approach to Classical Theorems of Advanced Calculus, Addison-Wesley, 1965.
[26] A. Edelman, T.A. Arias, Steven T. Smith, The geometry of algorithms with orthogonality constraints, SIAM J. Matrix Anal. Appl. 20 (1998) 303–353.
[27] Y.C. Wong, Differential geometry of Grassmann manifolds, Proc. Natl. Acad. Sci. U.S.A. 57 (1967) 589.
[28] H. Karcher, Riemannian center of mass and mollifier smoothing, Commun. Pure Appl. Math. 30 (1977) 509–541.
[29] J.-M. Chang, Classification on the Grassmannians: theory and applications (Ph.D. thesis), 2008.
[30] V. Batagelj, Constrained clustering problems, in: IFCS, 1998.
[31] R.E. Jensen, A dynamic programming algorithm for cluster analysis, Oper. Res. (1969) 1034–1057.
[32] V. Batagelj, S. Korenjak-Cerne, S. Klavzar, Dynamic programming and convex clustering, Algorithmica 11 (1994) 93–103.
[33] V. Batagelj, Agglomerative methods in clustering with constraints, Preprint Series Dept. Math. Univ. Ljubljana, 1984.
[34] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (1936) 211–218.
[35] E. Elhamifar, R. Vidal, Sparse manifold clustering and embedding, in: Advances in Neural Information Processing Systems, 2011, pp. 55–63.
[36] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, in: Proceedings of the IEEE, vol. 86, IEEE, 1998, pp. 2278–2324.
[37] J. Yan, M. Pollefeys, A general framework for motion segmentation: independent, articulated, rigid, non-rigid, degenerate and non-degenerate, in: ECCV, pp. 94–106.
[38] K.N. Ramamurthy, J.J. Thiagarajan, A. Spanias, Improved sparse coding using manifold projections, in: The 18th IEEE International Conference on Image Processing (ICIP), IEEE, 2011, pp. 1237–1240.
[39] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: The Eighth IEEE International Conference on Computer Vision, ICCV 2001, vol. 2, IEEE, 2001, pp. 416–423.
[40] C. Sanderson, Biometric Person Recognition: Face Speech and Fusion, VDM Publishing, 2008.
[41] P. Viola, M.J. Jones, Robust real-time face detection, Int. J. Comput. Vis. 57 (2004) 137–154.
[42] C. Thornton, Separability is a learner's best friend, in: The Fourth Neural Computation and Psychology Workshop, London, Springer, 9–11 April 1997, pp. 40–46.