# Linear Shape Deformation Models with Local Support using Graph-based Structured Matrix Factorisation

Florian Bernard[1,2] Peter Gemmar[2,3] Frank Hertel[1] Jorge Goncalves[2] Johan Thunberg[2]
[1]Centre Hospitalier de Luxembourg, Luxembourg
[2]Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Luxembourg
[3]Trier University of Applied Sciences, Trier, Germany

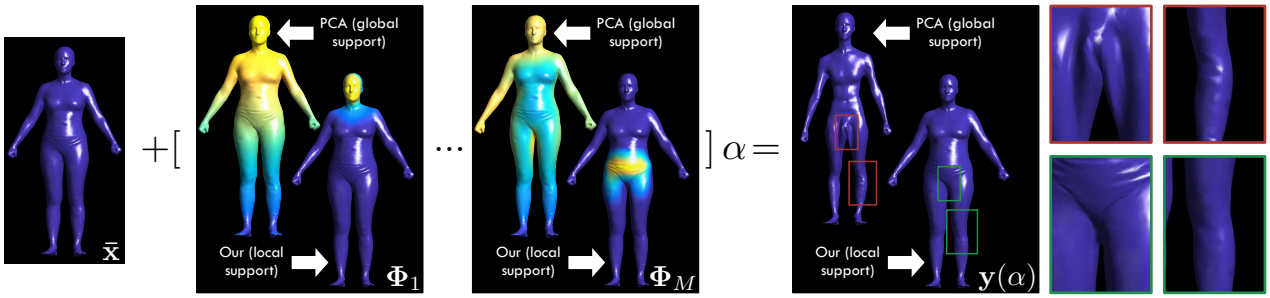


Figure 1. Global support factors of PCA lead to implausible body shapes, whereas the local support factors of our method give more realistic results. See our accompanying video for animated results.

## Abstract

*Representing 3D shape deformations by high-dimensional linear models has many applications in computer vision and medical imaging. Commonly, using Principal Components Analysis a low-dimensional subspace of the high-dimensional shape space is determined. However, the resulting factors (the most dominant eigenvectors of the covariance matrix) have global support, i.e. changing the coefficient of a single factor deforms the entire shape. Based on matrix factorisation with sparsity and graph-based regularisation terms, we present a method to obtain deformation factors with local support. The benefits include better flexibility and interpretability as well as the possibility of interactively deforming shapes locally. We demonstrate that for brain shapes our method outperforms the state of the art in local support models with respect to generalisation and sparse reconstruction, whereas for body shapes our method gives more realistic deformations.*



## 1. Introduction

Due to their simplicity, linear models in high-dimensional space are frequently used for modelling non-linear deformations of shapes in 2D or 3D space. For example, Active Shape Models (ASM) [13], based on a statistical shape model, are popular for image segmentation. Usually, surface meshes, comprising faces and vertices, are employed for representing the surfaces of shapes in 3D. Dimensionality reduction techniques are used to learn a low-dimensional representation of the *vertex coordinates* from training data. Frequently, an affine subspace close to the training shapes is used. To be more specific, mesh deformations are modelled by expressing the vertex coordinates as the sum of a mean shape $\bar{\mathbf{x}}$ and a linear combination of $M$ modes of variation $\mathbf{\Phi} = [\mathbf{\Phi}_1, \dots, \mathbf{\Phi}_M]$, i.e. the vertices deformed by the weight or coefficient vector $\boldsymbol{\alpha}$ are given by $\mathbf{y}(\boldsymbol{\alpha}) = \bar{\mathbf{x}} + \mathbf{\Phi}\boldsymbol{\alpha}$, see Fig. 1. Commonly, by using Principal Components Analysis (PCA), the modes of variation are set to the most dominant eigenvectors of the sample covariance matrix. PCA-based models are computationally convenient due to the orthogonality of the eigenvectors of the (real and symmetric) covariance matrix. Due to the diagonalisation of the covariance matrix, an axis-aligned Gaussian distribution of the weight vectors of the training data is obtained. A problem of PCA-based models is that eigenvectors have *global support*, i.e. adjusting the weight of a single factor affects *all* vertices of the shape (Fig. 1).

Thus, in this work, instead of eigenvectors, we consider more general *factors* as modes of variation that have *local support*, i.e. adjusting the weight of a single factor leads only to a spatially localised deformation of the shape (Fig. 1). The set of all factors can be seen as a dictionary for representing shapes by a linear combination of the factors. Benefits of factors with local support include more realistic deformations (cf. Fig. 1), better interpretability of the deformations (e.g. clinical interpretability in a medical context [32]), and the possibility of interactive local mesh deformations (e.g. editing animated mesh sequences in computer graphics [27], or enhanced flexibility for interactive 3D segmentation based on statistical shape models [3, 4]).

### 1.1. PCA Variants

PCA is a non-convex problem that admits the efficient computation of the global optimum, e.g. by Singular Value Decomposition (SVD). However, the downside is that the incorporation of arbitrary (convex) regularisation terms is not possible due to the SVD-based solution. Therefore, incorporating regularisation terms into PCA is an active field of research and several variants have been presented: Graph-Laplacian PCA [21] obtains factors with smoothly varying components according to a given graph. Robust PCA [10] formulates PCA as a convex low-rank matrix factorisation problem, where the nuclear norm is used as convex relaxation of the matrix rank. A combination of both Graph-Laplacian PCA and Robust PCA has been presented in [31]. The Sparse PCA (SPCA) method [18, 43] obtains sparse factors. Structured Sparse PCA (SSPCA) [20] additionally imposes structure on the sparsity of the factors using group sparsity norms, such as the mixed $\ell_1/\ell_2$ norm.

### 1.2. Deformation Model Variants

In [24], the flexibility of shape models has been increased by using PCA-based factors in combination with a *per-vertex* weight vector, in contrast to a single weight vector that is used for all vertices. In [14, 39], it is shown that additional elasticity in the PCA-based model can be obtained by manipulation of the sample covariance matrix. Whilst both approaches increase the flexibility of the shape model, they result in global support factors.

In [32], SPCA is used to model the anatomical shape variation of the 2D outlines of the corpus callosum. In [37], 2D images of the cardiac ventricle were used to train an Active Appearance Model based on Independent Component Analysis (ICA) [19]. Other applications of ICA for statistical shape models are presented in [34, 42]. The Orthomax method, where the PCA basis is determined first and then rotated such that it has a "simple" structure, is used in [33]. The major drawback of SPCA, ICA and Orthomax is that the spatial relation between vertices is completely ignored.

The Key Point Subspace Acceleration method based on Varimax, where a statistical subspace and key points are automatically identified from training data, is introduced in [25]. For mesh animation, in [36] the clusters of spatially close vertices are determined first by spectral clustering, and then PCA is applied for each vertex cluster, resulting in one sub-PCA model per cluster. This two-stage procedure has the problem, that, due to the independence of both stages, it is unclear whether the clustering is optimal with respect to the deformation model. Also, a blending procedure for the individual sub-PCA models is required. A similar approach of first manually segmenting body regions and then learning a PCA-based model has been presented in [41].

The *Sparse Localised Deformation Components* method (SPLOCS) obtains localised deformation modes from animated mesh sequences by using a matrix factorisation formulation with a weighted $\ell_1/\ell_2$ norm regulariser [27]. Local support factors are obtained by explicitly modelling local support regions, which are in turn used to update the weights of the $\ell_1/\ell_2$ norm *in each iteration*. This makes the non-convex optimisation problem even harder to solve and dismisses convergence guarantees. With that, a suboptimal initialisation of the support regions, as presented in the work, affects the quality of the found solution.

The *compressed manifold modes* method [23, 26] has the objective to obtain local support basis functions of the (discretised) Laplace-Beltrami operator of a *single* input mesh. In [22], the authors obtain smooth functional correspondences between shapes that are spatially localised by using an $\ell_1$ norm regulariser in combination with the *row* and *column Dirichlet energy*. The method proposed in [30] is able to localise *shape differences* based on functional maps between two shapes. Recently, the *Shape-from-Operator* approach has been presented [7], where shapes are reconstructed from more general intrinsic operators.

### 1.3. Aims and Main Contributions

The work presented in this paper has the objective of learning local support deformation factors from training data. The main application of the resulting shape model is recognition, segmentation and shape interpolation [3, 4]. Whilst our work remedies several of the mentioned shortcomings of existing methods, it can also be seen as complementary to SPLOCS, which is more tailored towards artistic editing and mesh animation. The most significant difference to SPLOCS is that we aim at letting the training shapes *automatically determine the location and size* of each local support region. This is achieved by formulating a matrix factorisation problem that incorporates regularisation terms which simultaneously account for *sparsity* and *smoothness* of the factors, where a graph-based smoothness regulariser accounts for smoothly varying neighbour vertices. In contrast to SPLOCS or sub-PCA, this results in an implicit clustering that is part of the optimisation and does *not require*

*an initialisation* of local support regions, which in turn simplifies the optimisation procedure. Moreover, by integrating a *smoothness prior* into our framework we can *handle small training datasets*, even if the desired number of factors exceeds the number of training shapes. Our optimisation problem is formulated in terms of the Structured Low-Rank Matrix Factorisation framework [16], which has *appealing theoretical properties*.

## 2. Methods

First, we introduce our notation and linear shape deformation models. Then, we state the objective and its formulation as optimisation problem, followed by the theoretical motivation. Finally, the block coordinate descent algorithm and the factor splitting method are presented.

### 2.1. Notation

$\mathbf{I}_p$ denotes the $p\times p$ identity matrix, $\mathbf{1}_p$ the $p$-dimensional vector containing ones, $\mathbf{0}_{p\times q}$ the $p \times q$ zero matrix, and $\mathbb{S}_p^+$ the set of $p \times p$ positive semi-definite matrices. Let $\mathbf{A} \in \mathbb{R}^{p\times q}$. We use the notation $\mathbf{A}_{\mathcal{A},\mathcal{B}}$ to denote the submatrix of $\mathbf{A}$ with the rows indexed by the (ordered) index set $\mathcal{A}$ and columns indexed by the (ordered) index set $\mathcal{B}$. The colon denotes the full (ordered) index set, e.g. $\mathbf{A}_{\mathcal{A},:}$ is the matrix containing all rows of $\mathbf{A}$ indexed by $\mathcal{A}$. For brevity, we write $\mathbf{A}_r$ to denote the $p$-dimensional vector formed by the $r$-th column of $\mathbf{A}$. The operator $\mathrm{vec}(\mathbf{A})$ creates a $pq$-dimensional column vector from $\mathbf{A}$ by stacking its columns, and $\otimes$ denotes the Kronecker product.

### 2.2. Linear Shape Deformation Models

Let $\mathbf{X}_k\in\mathbb{R}^{N\times 3}$ be the matrix representation of a shape comprising $N$ points (or vertices) in 3 dimensions, and let $\{\mathbf{X}_k : 1 \le k \le K\}$ be the set of $K$ training shapes. We assume that the rows in each $\mathbf{X}_k$ correspond to homologous points. Using the vectorisation $\mathbf{x}_k=\mathrm{vec}(\mathbf{X}_k)\in\mathbb{R}^{3N}$, all $\mathbf{x}_k$ are arranged in the matrix $\mathbf{X}=[\mathbf{x}_1,\dots,\mathbf{x}_K]\in\mathbb{R}^{3N\times K}$. We assume that all shapes have the same pose, are centred at the mean shape $\bar{\mathbf{X}}$, i.e. $\sum_k \mathbf{X}_k=\mathbf{0}_{N\times 3}$, and that the standard deviation of $\mathrm{vec}(\mathbf{X})$ is one.

Pairwise relations between vertices are modelled by a weighted undirected graph $\mathcal{G}=(\mathcal{V},\mathcal{E},\boldsymbol{\omega})$ that is shared by all shapes. The node set $\mathcal{V}=\{1,\dots,N\}$ enumerates all $N$ vertices, the edge set $\mathcal{E}\subseteq\{1,\dots,N\}^2$ represents the connectivity of the vertices, and $\boldsymbol{\omega}\in\mathbb{R}_+^{|\mathcal{E}|}$ is the weight vector. The (scalar) weight $\boldsymbol{\omega}_e$ of edge $e=(i,j) \in \mathcal{E}$ denotes the affinity between vertex $i$ and $j$, where "close" vertices have high value $\boldsymbol{\omega}_e$. We assume there are no self-edges, i.e. $(i,i)\notin\mathcal{E}$. The graph can either encode pairwise *spatial* connectivity, or affinities that are not of spatial nature (e.g. symmetries, or prior anatomical knowledge in medical applications).

For the standard PCA-based method [13], the modes of variation in the $M$ columns of the matrix $\boldsymbol{\Phi}\in\mathbb{R}^{3N\times M}$ are defined as the $M$ most dominant eigenvectors of the sample covariance matrix $\mathbf{C}=\frac{1}{K-1}\mathbf{X}\mathbf{X}^T$. However, we consider the generalisation where $\boldsymbol{\Phi}$ contains general $3N$-dimensional vectors, the *factors*, in its $M$ columns. In both cases, the (linear) deformation model (modulo the mean shape) is given by

$$\mathbf{y}(\boldsymbol{\alpha}) = \boldsymbol{\Phi}\boldsymbol{\alpha}\,, \tag{1}$$

with weight vector $\boldsymbol{\alpha} \in \mathbb{R}^M$. Due to vectorisation, the rows with indices $\{1,\dots,N\},\{N{+}1,\dots,2N\}$ and $\{2N{+}1,\dots,3N\}$ of $\mathbf{y}$ (or $\boldsymbol{\Phi}$), correspond to the $x$, $y$ and $z$ components of the $N$ vertices of the shape, respectively.

### 2.3. Objective and Optimisation Problem

The objective is to find $\boldsymbol{\Phi}=[\boldsymbol{\Phi}_1,\dots,\boldsymbol{\Phi}_M]$ and $\boldsymbol{A}=[\boldsymbol{\alpha}_1,\dots,\boldsymbol{\alpha}_K]\in\mathbb{R}^{M\times K}$ for a given $M < 3N$ such that, according to eq. (1), we can write

$$\mathbf{X} \approx \boldsymbol{\Phi}\boldsymbol{A}\,, \tag{2}$$

where the factors $\boldsymbol{\Phi}_m$ have *local support*. Local support means that $\boldsymbol{\Phi}_m$ is sparse and that all *active* vertices, i.e. vertices that correspond to the non-zero elements of $\boldsymbol{\Phi}_m$, are connected by (sequences of) edges in the graph $\mathcal{G}$.

Now we state our problem formally as an optimisation problem. The theoretical motivation thereof is based on [2, 9, 16] and is recapitulated in section 2.4, where it will also become clear that our chosen regularisation term is related to the *Projective Tensor Norm* [2, 16].

A general matrix factorisation problem with squared Frobenius norm loss is given by

$$\min_{\boldsymbol{\Phi},\mathbf{A}} \|\mathbf{X}-\boldsymbol{\Phi}\mathbf{A}\|_F^2 + \Omega(\boldsymbol{\Phi},\mathbf{A})\,, \tag{3}$$

where the regulariser $\Omega$ imposes certain properties upon $\boldsymbol{\Phi}$ and $\mathbf{A}$. The optimisation is performed over *some* compact set (which we assume implicitly from here on). An obvious property of local support factors is sparsity. Moreover, it is desirable that neighbour vertices vary smoothly. Both properties together seem to be promising candidates to obtain local support factors, which we reflect in our regulariser. Our optimisation problem is given by

$$\min_{\substack{\boldsymbol{\Phi}\in\mathbb{R}^{3N\times M}\\ \mathbf{A}\in\mathbb{R}^{M\times K}}} \|\mathbf{X}-\boldsymbol{\Phi}\mathbf{A}\|_F^2 + \lambda\sum_{m=1}^{M}\|\boldsymbol{\Phi}_m\|_\Phi\|(\mathbf{A}_{m,:})^T\|_A\,, \tag{4}$$

where $\|\cdot\|_\Phi$ and $\|\cdot\|_A$ denote vector norms. For $\mathbf{z}' \in \mathbb{R}^K, \mathbf{z}\in\mathbb{R}^{3N}$, we define

$$\|\mathbf{z}'\|_A = \lambda_A\|\mathbf{z}'\|_2\,, \text{ and} \tag{5}$$

$$\|\mathbf{z}\|_\Phi = \lambda_1\|\mathbf{z}\|_1+\lambda_2\|\mathbf{z}\|_2+\lambda_\infty\|\mathbf{z}\|_{1,\infty}^{\mathcal{H}}+\lambda_{\mathcal{G}}\|\mathbf{E}\mathbf{z}\|_2\,. \tag{6}$$

Both $\ell_2$ norm terms will be discussed in section 2.4. The $\ell_1$ norm is used to obtain sparsity in the factors. The (mixed)

$\ell_1/\ell_\infty$ norm is defined by

$$\|\mathbf{z}\|_{1,\infty}^{\mathcal{H}} = \sum_{g\in\mathcal{H}} \|\mathbf{z}_g\|_\infty \,, \tag{7}$$

where $\mathbf{z}_g$ denotes a subvector of $\mathbf{z}$ indexed by $g \in \mathcal{H}$. By using the collection $\mathcal{H} = \{\{i, i+N, i+2N\} : 1 \le i \le N\}$, a grouping of the x, y and z components per vertex is achieved, i.e. within a group $g$ only the component with largest magnitude is penalised and no extra cost is to be paid for the other components being non-zero.

The last term in eq. (6), the graph-based $\ell_2$ (semi-)norm, imposes smoothness upon each factor, such that neighbour elements according to the graph $\mathcal{G}$ vary smoothly. Based on the incidence matrix of $\mathcal{G}$, we choose $\mathbf{E}$ such that

$$\|\mathbf{E}\mathbf{z}\|_2 = \sqrt{\sum_{d\in\{0,N,2N\}} \sum_{(i,j)=e_p\in\mathcal{E}} \boldsymbol{\omega}_{e_p} (\mathbf{z}_{d+i} - \mathbf{z}_{d+j})^2} \,. \tag{8}$$

As such, $\mathbf{E}$ is a discrete (weighted) gradient operator and $\|\mathbf{E}\cdot\|_2^2$ corresponds to Graph-Laplacian regularisation [21]. $\mathbf{E}$ is specified in the supplementary material.

The structure of our problem formulation in eqs. (4), (5), (6) allows for various degrees-of-freedom in the form of the parameters. They allow to weigh the data term versus the regulariser ($\lambda$), control the rank of the solution ($\lambda_A$ and $\lambda_2$ together, cf. last paragraph in section 2.4), control the sparsity ($\lambda_1$), control the amount of grouping of the x, y and z components ($\lambda_\infty$) and control the smoothness $\lambda_\mathcal{G}$. The number of factors $M$ has an impact on the size of the support regions (for small $M$ the regions tend to be larger, whereas for large $M$ they tend to be smaller).

## 2.4. Theoretical Motivation

For a matrix $\mathbf{X} \in \mathbb{R}^{3N\times K}$ and vector norms $\|\cdot\|_\Phi$ and $\|\cdot\|_A$, let us define the function

$$\psi^M(\mathbf{X}) = \min_{\substack{\{(\boldsymbol{\Phi}\in\mathbb{R}^{3N\times M},\\ \mathbf{A}\in\mathbb{R}^{M\times K}):\\ \boldsymbol{\Phi}\mathbf{A}=\mathbf{X}\}}} \sum_{m=1}^{M} \|\boldsymbol{\Phi}_m\|_\Phi \|(\mathbf{A}_{m,:})^T\|_A \,. \tag{9}$$

The function $\psi(\cdot) = \lim_{M\to\infty} \psi^M(\cdot)$ defines a norm known as *Projective Tensor Norm* or *Decomposition Norm* [2, 16].

**Lemma 1.** *For any $\epsilon > 0$ there exists an $M(\epsilon) \in \mathbb{N}$ such that $\|\psi(\mathbf{X}) - \psi^{M(\epsilon)}(\mathbf{X})\| < \epsilon$.*

*Proof.* For $\psi(\mathbf{X})$ there are sequences $\{\boldsymbol{\Phi}_i\}_{i=1}^\infty$ and $\{\mathbf{A}_i^T\}_{i=1}^\infty$ such that $\psi(\mathbf{X}) = \sum_{i=1}^\infty \|\boldsymbol{\Phi}_i\|_\Phi \|\mathbf{A}_i^T\|_A$. Let $l_m = \sum_{i=1}^m \|\boldsymbol{\Phi}_i\|_\Phi \|\mathbf{A}_i^T\|_A$. The sequence $l_m$ is monotone, bounded from above and convergent. Let $l_\infty = \psi(\mathbf{X})$ denote its limit. Since the sequence is convergent, there is $M(\epsilon)$ such that $\|l_\infty - l_j\| < \epsilon$ for $j \ge M(\epsilon)$. □

We now proceed by introducing the optimisation problem

$$\min_{\mathbf{Z}} \|\mathbf{X} - \mathbf{Z}\|_F^2 + \lambda\psi^M(\mathbf{Z}) \,. \tag{10}$$

Next, we establish a connection between problem (10) and our problem (4). First, we assume that we are given a solution pair $(\boldsymbol{\Phi}, \mathbf{A})$ minimising problem (4). By defining $\mathbf{Z} = \boldsymbol{\Phi}\mathbf{A}$, $\mathbf{Z}$ is a solution to problem (10). Secondly, assume we are given a solution $\mathbf{Z}$ minimising problem (10). To find a solution solution pair $(\boldsymbol{\Phi}, \mathbf{A})$ minimising problem (4), one needs to compute the $(\boldsymbol{\Phi}, \mathbf{A})$ that achieves the minimum of the right-hand side of (9) for a given $\mathbf{Z}$.

This shows that given a solution to one of the problems, one can infer a solution to the other problem. Next we reformulate problem (10). Following [16], we define the matrices $\mathbf{Q} \in \mathbb{R}^{3N+K\times M}$, $\mathbf{Y} \in \mathbb{R}^{3N+K\times 3N+K}$ as

$$\mathbf{Q} = \begin{bmatrix} \boldsymbol{\Phi} \\ \mathbf{A}^T \end{bmatrix} \,, \quad \mathbf{Y} = \mathbf{Q}\mathbf{Q}^T = \begin{pmatrix} \boldsymbol{\Phi}\boldsymbol{\Phi}^T & \boldsymbol{\Phi}\mathbf{A} \\ \mathbf{A}^T\boldsymbol{\Phi}^T & \mathbf{A}^T\mathbf{A} \end{pmatrix} \,, \tag{11}$$

and the function $F : \mathbb{S}_{3N+K}^+ \to \mathbb{R}$ as

$$F(\mathbf{Y}) = F(\mathbf{Q}\mathbf{Q}^T) = \|\mathbf{X} - \boldsymbol{\Phi}\mathbf{A}\|_F^2 + \lambda\psi^M(\boldsymbol{\Phi}\mathbf{A}) \,. \tag{12}$$

$$\text{Let} \qquad \mathbf{Y}^* = \operatorname*{arg\,min}_{\mathbf{Y}\in\mathbb{S}_{3N+K}^+} F(\mathbf{Y}) \,. \tag{13}$$

For a given $\mathbf{Y}^*$, problem (10) is minimised by the upper-right block matrix of $\mathbf{Y}^*$. The difference between (10) and (13) is that the latter is over the set of positive semi-definite matrices, which, at first sight, does not present any gain. However, under certain conditions, the global solution for $\mathbf{Q}$, rather than the product $\mathbf{Y} = \mathbf{Q}\mathbf{Q}^T$, can be obtained directly [9]. In [2] it is shown that if $\mathbf{Q}$ is a *rank deficient* local minimum of $F(\mathbf{Q}\mathbf{Q}^T)$, then it is also a global minimum. Whilst these results only hold for twice differentiable functions $F$, Haeffele et al. have presented analogous results for the case of $F$ being a sum of a twice-differentiable term and a non-differentiable term [16], such as ours above.

As such, *any* (rank deficient) local optimum of problem (4) is also a global optimum. If in $\psi(\cdot)$, both $\|\cdot\|_\Phi$ and $\|\cdot\|_A$ are the $\ell_2$ norm, $\psi(\cdot)$ is equivalent to the nuclear norm, commonly used as convex relaxation of the matrix rank [16, 29]. In order to steer the solution towards being rank deficient, we include $\ell_2$ norm terms in $\|\cdot\|_\Phi$ and $\|\cdot\|_A$ (see (5) and (6)). With that, part of the regularisation term in (4) is the nuclear norm that accounts for low-rank solutions.

## 2.5. Block Coordinate Descent

A solution to problem (4) is found by block coordinate descent (BCD) [40] (algorithm 1). It employs alternating proximal steps, which can be seen as generalisation of gradient steps for non-differentiable functions. Since computing the proximal mapping is repeated in each iteration, an efficient computation is essential. The proximal mapping of $\|\cdot\|_A$ in eq. (5) has a closed-form solution by *block soft thresholding* [28]. The proximal mapping of $\|\cdot\|_\Phi$ in eq. (6) is solved by dual forward-backward splitting [11, 12] (see supplementary material). The benefit of BCD is that it scales well to large problems (cf. complexity analysis in

**repeat**

// update $\mathbf{\Phi}$

$\mathbf{\Phi}' \leftarrow \mathbf{\Phi} - \epsilon_\Phi \nabla_\Phi \|\mathbf{X} - \mathbf{\Phi}\mathbf{A}\|_F^2$ // gradient step (loss)

**for** $m = 1, \ldots, M$ **do** // proximal step $\mathbf{\Phi}$ (penalty)

$\mathbf{\Phi}_m \leftarrow \text{prox}_{\lambda\|\cdot\|_\Phi\|(\mathbf{A}_{m,:})^T\|_A}(\mathbf{\Phi}'_m)$

// update $\mathbf{A}$

$\mathbf{A}' \leftarrow \mathbf{A} - \epsilon_A \nabla_A \|\mathbf{X} - \mathbf{\Phi}\mathbf{A}\|_F^2$ // gradient step (loss)

**for** $m = 1, \ldots, M$ **do** // proximal step $\mathbf{A}$ (penalty)

$\mathbf{A}_{m,:} \leftarrow \text{prox}_{\lambda\|\mathbf{\Phi}_m\|_\Phi\|\cdot\|_A}((\mathbf{A}'_{m,:})^T)^T$

**until** *convergence*

**Algorithm 1:** Simplified view of block coordinate descent. For details see [16, 40].

the supplementary material). However, a downside is that by using the alternating updates one has only guaranteed convergence to a Nash equilibrium point [16, 40].

### 2.6. Factor Splitting

Whilst solving problem (4) leads to smooth and sparse factors, there is no guarantee that the factors have only a *single* local support region. In fact, as motivated in section 2.4, the solution of eq. (4) is steered towards being low-rank. However, the price to pay for a low-rank solution is that capturing multiple support regions in a *single* factor is preferred over having each support region in an individual factor (e.g. for $M = 2$ and any $\mathbf{a} \neq \mathbf{0},\ \mathbf{b} \neq \mathbf{0}$, the matrix $\mathbf{\Phi} = [\mathbf{\Phi}_1 \mathbf{\Phi}_2]$ has a lower rank when $\mathbf{\Phi}_1 = [\mathbf{a}^T\ \mathbf{b}^T]^T$ and $\mathbf{\Phi}_2 = \mathbf{0}$, compared to $\mathbf{\Phi}_1 = [\mathbf{a}^T\ \mathbf{0}^T]^T$ and $\mathbf{\Phi}_2 = [\mathbf{0}^T\ \mathbf{b}^T]^T$).

A simple yet effective way to deal with this issue is to split factors with multiple disconnected support regions into multiple (new) factors (see supplementary material). Since this is performed *after* the optimisation problem has been solved, it is preferable over pre- or intra-processing procedures [36, 27] since the optimisation remains unaffected and the data term in eq. (4) does not change due to the splitting.

## 3. Experimental Results

We compared the proposed method with *PCA* [13], *kernel PCA* (kPCA, cf. 3.2.2), *Varimax* [17], *ICA* [19], *SPCA* [20], *SSPCA* [20], and *SPLOCS* [27] on two real datasets, brain structures and human body shapes. Only our method and the SPLOCS method explicitly aim to obtain local support factors, whereas the SPCA and SSPCA methods obtain sparse factors (for the latter the $\ell_1/\ell_2$ norm with groups defined by $\mathcal{H}$, cf. eq. (7), is used). The methods kPCA, SPCA, SSPCA, SPLOCS and ours require to set various parameters, which we address by random sampling (see supplementary material).

For all evaluated methods a factorisation $\mathbf{\Phi}\mathbf{A}$ is obtained. W.l.o.g. we normalise the rows of $\mathbf{A}$ to have standard deviation one (since $\mathbf{\Phi}\mathbf{A} = (\frac{1}{s}\mathbf{\Phi})(s\mathbf{A})$ for $s \neq 0$). Then, the factors in $\mathbf{\Phi}$ are ordered descendingly according to their $\ell_2$ norms.

In our method, the number of factors may be changed due to factor splitting, thus, in order to allow a fair comparison, we only retain the first $M$ factors. Initially, the columns of $\mathbf{\Phi}$ are chosen to $M$ (unique) columns selected randomly from $\mathbf{I}_{3N}$. This is in accordance with [16], where empirically good results are obtained using trivial initialisations. Convergence plots for different initialisations can be found in the supplementary material.

### 3.1. Quantitative Measures

For $\mathbf{x} = \text{vec}(\mathbf{X})$ and $\tilde{\mathbf{x}} = \text{vec}(\tilde{\mathbf{X}})$, the *average error*

$$e_{\text{avg}}(\mathbf{x}, \tilde{\mathbf{x}}) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{X}_{i,:} - \tilde{\mathbf{X}}_{i,:}\|_2 \tag{14}$$

and the *maximum error*

$$e_{\max}(\mathbf{x}, \tilde{\mathbf{x}}) = \max_{i=1,\ldots,N} \|\mathbf{X}_{i,:} - \tilde{\mathbf{X}}_{i,:}\|_2 \tag{15}$$

measure the agreement between shape $\mathbf{X}$ and shape $\tilde{\mathbf{X}}$.

The *reconstruction error* for shape $\mathbf{x}_k$ is measured by solving the overdetermined system $\mathbf{\Phi}\boldsymbol{\alpha}_k = \mathbf{x}_k$ for $\boldsymbol{\alpha}_k$ in the least-squares sense, and then computing $e_{\text{avg}}(\mathbf{x}_k, \mathbf{\Phi}\boldsymbol{\alpha}_k)$ and $e_{\max}(\mathbf{x}_k, \mathbf{\Phi}\boldsymbol{\alpha}_k)$, respectively.

To measure the *specificity error*, $n_s$ shape samples are drawn randomly ($n_s$=1000 for the brain shapes and $n_s$=100 for the body shapes). For each drawn shape, the average and maximum errors between the closest shape in the training set are denoted by $s_{\text{avg}}$ and $s_{\max}$, respectively. For simplicity, we assumed that the parameter vector $\boldsymbol{\alpha}$ follows a zero mean Gaussian distribution, where the covariance matrix $\mathbf{C}_{\boldsymbol{\alpha}}$ is estimated from the parameter vectors $\boldsymbol{\alpha}_k$ of the $K$ training shapes. With that, a random shape sample $\mathbf{x}_r$ is generated by drawing a sample of the vector $\boldsymbol{\alpha}_r$ from its distribution and setting $\mathbf{x}_r = \mathbf{\Phi}\boldsymbol{\alpha}_r$. The specificity can be interpreted as a score for assessing how realistic synthesized shapes are on a coarse level of detail (the contribution of errors on fine details to the specificity score is marginal due to the dominance of the errors on coarse scales).

For evaluating the *generalisation error*, a collection of index sets $\mathcal{I} \subset 2^{\{1,\ldots,K\}}$ is used, where each set $\mathcal{J} \in \mathcal{I}$ denotes the set of indices of the *test shapes* for one run and $|\mathcal{I}|$ is the number of runs. We used five-fold cross-validation, i.e. $|\mathcal{I}| = 5$ and each set $\mathcal{J}$ contains $\text{round}(\frac{K}{5})$ random integers from $\{1, \ldots, K\}$. In each run, the deformation factors $\mathbf{\Phi}^{\bar{\mathcal{J}}}$ are computed using only the shapes with indices $\bar{\mathcal{J}} = \{1, \ldots, K\} \setminus \mathcal{J}$. For all test shapes $\mathbf{x}_j$, where $j \in \mathcal{J}$, the parameter vector $\boldsymbol{\alpha}_j$ is determined by solving $\mathbf{\Phi}^{\bar{\mathcal{J}}}\boldsymbol{\alpha}_j = \mathbf{x}_j$ in the least-squares sense. Eventually, the average reconstruction error $e_{\text{avg}}(\mathbf{x}_j, \mathbf{\Phi}^{\bar{\mathcal{J}}}\boldsymbol{\alpha}_j)$ and the maximum reconstruction error $e_{\max}(\mathbf{x}_j, \mathbf{\Phi}^{\bar{\mathcal{J}}}\boldsymbol{\alpha}_j)$ are computed for each test shape, which we denote as $g_{\text{avg}}$ and $g_{\max}$, respectively. Moreover, the *sparse reconstruction errors* $g_{\text{avg}}^{0.05}$ and $g_{\max}^{0.05}$ are computed in a similar manner, with the difference that $\boldsymbol{\alpha}_j$ is now determined by using only 5% of the rows (selected randomly) of $\mathbf{\Phi}^{\bar{\mathcal{J}}}$ and $\mathbf{x}_j$, denoted by $\tilde{\mathbf{\Phi}}^{\bar{\mathcal{J}}}$ and $\tilde{\mathbf{x}}_j$. For that, we minimise $\|\tilde{\mathbf{\Phi}}^{\bar{\mathcal{J}}}\boldsymbol{\alpha}_j - \tilde{\mathbf{x}}_j\|_2^2 + \|\Gamma\boldsymbol{\alpha}_j\|_2^2$

with respect to $\boldsymbol{\alpha}_j$, which is a least-squares problem with Tikhonov regularisation, where $\Gamma$ is obtained by Cholesky factorisation of $\mathbf{C}_{\boldsymbol{\alpha}} = \Gamma^T\Gamma$. The purpose of this measure is to evaluate how well a deformation model interpolates an unseen shape from a small subset of its points, which is relevant for shape model-based surface reconstruction [3, 4].

### 3.2. Brain Structures

The first evaluated dataset comprises 8 brain structure meshes of $K$=17 subjects [5]. All 8 meshes together have a total number of $N$=1792 vertices that are in correspondence among all subjects. Moreover, all meshes have the same topology, i.e. seen as graphs they are isomorphic. A single deformation model is used to model the deformation of all 8 meshes in order to capture the interrelation between the brain structures. We fix the number of desired factors to $M$=96 to account for a sufficient amount of local details in the factors. Whilst the meshes are smooth and comparably simple in shape (cf. Fig. 3), a particular challenge is that the training dataset comprises only $K$=17 shapes.

#### 3.2.1 Second-order Terms

Based on anatomical knowledge, we use the brain structure interrelation graph $\mathcal{G}_{\text{bs}} = (\mathcal{V}_{\text{bs}}, \mathcal{E}_{\text{bs}})$ shown in Fig. 2, where an edge between two structures denotes that a deformation of one structure may have a direct effect on the deformation of the other structure. Using $\mathcal{G}_{\text{bs}}$, we now build

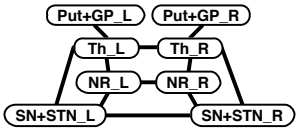


Figure 2. Brain structure interrelation graph.

a distance matrix that is then used in the SPLOCS method and our method. For $\mathfrak{o} \in \mathcal{V}_{\text{bs}}$, let $g_{\mathfrak{o}} \subset \{1, \dots, N\}$ denote the (ordered) indices of the vertices of brain structure $\mathfrak{o}$. Let $\mathbf{D}_{\text{euc}} \in \mathbb{R}^{N\times N}$ be the Euclidean distance matrix, where $(\mathbf{D}_{\text{euc}})_{ij} = \|\bar{\mathbf{X}}_{i,:} - \bar{\mathbf{X}}_{j,:}\|_2$ is the Euclidean distance between vertex $i$ and $j$ of the mean shape $\bar{\mathbf{X}}$. Moreover, let $\mathbf{D}_{\text{geo}} \in \mathbb{R}^{N\times N}$ be the geodesic graph distance matrix of the mean shape $\bar{\mathbf{X}}$ using the graph induced by the (shared) mesh topology. By $\mathbf{D}^{\mathfrak{o}}_{\text{euc}} \in \mathbb{R}^{|g_{\mathfrak{o}}|\times|g_{\mathfrak{o}}|}$ and $\mathbf{D}^{\mathfrak{o}}_{\text{geo}} \in \mathbb{R}^{|g_{\mathfrak{o}}|\times|g_{\mathfrak{o}}|}$ we denote the Euclidean distance matrix and the geodesic distance matrix of brain structure $\mathfrak{o}$, which are submatrices of $\mathbf{D}_{\text{euc}}$ and $\mathbf{D}_{\text{geo}}$, respectively. Let $\bar{d}^{\mathfrak{o}}$ denote the average vertex distance between neighbour vertices of brain structure $\mathfrak{o}$. We define the normalised geodesic graph distance matrix of brain structure $\mathfrak{o}$ by $\tilde{\mathbf{D}}^{\mathfrak{o}}_{\text{geo}} = \frac{1}{\bar{d}^{\mathfrak{o}}}\mathbf{D}^{\mathfrak{o}}_{\text{geo}}$ and the matrix $\tilde{\mathbf{D}}_{\text{geo}} \in \mathbb{R}^{N\times N}$ is composed by the individual blocks $\tilde{\mathbf{D}}^{\mathfrak{o}}_{\text{geo}}$.

The normalised distance matrix between structure $\mathfrak{o}$ and $\tilde{\mathfrak{o}}$ is given by $\tilde{\mathbf{D}}^{\mathfrak{o},\tilde{\mathfrak{o}}}_{\text{bs}} = \frac{2}{\bar{d}^{\mathfrak{o}}+\bar{d}^{\tilde{\mathfrak{o}}}}[(\mathbf{D}_{\text{euc}})_{g_{\mathfrak{o}},g_{\tilde{\mathfrak{o}}}} - \mathbf{1}_{|g_{\mathfrak{o}}|}\mathbf{1}^T_{|g_{\tilde{\mathfrak{o}}}|}d^{\mathfrak{o},\tilde{\mathfrak{o}}}_{\min}] \in \mathbb{R}^{|g_{\mathfrak{o}}|\times|g_{\tilde{\mathfrak{o}}}|}$, where $d^{\mathfrak{o},\tilde{\mathfrak{o}}}_{\min}$ is the smallest element of $(\mathbf{D}_{\text{euc}})_{g_{\mathfrak{o}},g_{\tilde{\mathfrak{o}}}}$. The (symmetric) distance matrix $\tilde{\mathbf{D}}_{\text{bs}} \in \mathbb{R}^{N\times N}$ between all structures is constructed by

$$(\tilde{\mathbf{D}}_{\text{bs}})_{g_{\mathfrak{o}},g_{\tilde{\mathfrak{o}}}} = \begin{cases} \tilde{\mathbf{D}}^{\mathfrak{o},\tilde{\mathfrak{o}}}_{\text{bs}} & \text{if} \quad (\mathfrak{o},\tilde{\mathfrak{o}}) \in \mathcal{E}_{\text{bs}} \\ \mathbf{0}_{|g_{\mathfrak{o}}|\times|g_{\tilde{\mathfrak{o}}}|} & \text{else} \end{cases} . \tag{16}$$

For the SPLOCS method we used the distance matrix $\mathbf{D} = \alpha_D\tilde{\mathbf{D}}_{\text{geo}} + (1-\alpha_D)\tilde{\mathbf{D}}_{\text{bs}}$. For our method, we construct the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{\omega})$ (cf. section 2.2) by having an edge $e = (i,j)$ in $\mathcal{E}$ for each $\boldsymbol{\omega}_e = \alpha_D \exp(-(\tilde{\mathbf{D}}_{\text{geo}})^2_{i,j}) + (1-\alpha_D)\exp(-(\tilde{\mathbf{D}}_{\text{bs}})^2_{ij})$ that is larger than the threshold $\theta = 0.1$. In both cases we set $\alpha_D = 0.5$.

#### 3.2.2 Dealing with the Small Training Set

For PCA, Varimax and ICA the number of factors cannot exceed the rank of $\mathbf{X}$, which is at most $K-1$ for $K<3N$. For the used matrix factorisation framework, setting $M$ to a value larger than the expected rank of the solution but smaller than full rank has empirically led to good results [16]. However, since our expected rank is $M = 96$ and the full rank is at most $K-1 = 16$, this is not possible.

We compensate the insufficient amount of training data by assuming smoothness of the deformations. Based on concepts introduced in [14, 38, 39], instead of the data matrix $\mathbf{X}$, we factorise the *kernelised covariance* matrix $\mathbf{K}$. The standard PCA method finds the $M$ most dominant eigenvectors of the covariance matrix $\mathbf{C}$ by the (exact) factorisation $\mathbf{C} = \boldsymbol{\Phi}\,\text{diag}(\lambda_1, \dots, \lambda_{K-1})\boldsymbol{\Phi}^T$. The kernelised covariance $\mathbf{K}$ allows to incorporate additional elasticity into the resulting deformation model. For example, $\mathbf{K}$=$\mathbf{I}_{3N}$ results in independent vertex movements [39]. A more interesting approach is to combine the (scaled) covariance matrix with a smooth vertex deformation. We define $\mathbf{K} = \frac{1}{\|\text{vec}(\mathbf{X}\mathbf{X}^T)\|_\infty}\mathbf{X}\mathbf{X}^T + \mathbf{K}_{\text{euc}}$, with $\mathbf{K}_{\text{euc}} = \mathbf{I}_3 \otimes \mathbf{K}'_{\text{euc}} \in \mathbb{R}^{3N\times 3N}$. Using the bandwidth $\beta$, $\mathbf{K}'_{\text{euc}}$ is given by

$$(\mathbf{K}'_{\text{euc}})_{ij} = \exp(-(\frac{(\mathbf{D}_{\text{euc}})_{ij}}{2\beta\|\,\text{vec}(\mathbf{D}_{\text{euc}})\|_\infty})^2) . \tag{17}$$

Setting $\boldsymbol{\Phi}$ to the $M$ most dominant eigenvectors of the symmetric and positive semi-definite matrix $\mathbf{K}$ gives the solution of kPCA [6]. In terms of our proposed matrix factorisation problem in eq. (4), we find a factorisation $\boldsymbol{\Phi}\tilde{\mathbf{A}}$ of $\mathbf{K}$, instead of factorising the data $\mathbf{X}$. Since the regularisation term remains unchanged, the factor matrix $\boldsymbol{\Phi}\in\mathbb{R}^{3N\times M}$ is still sparse and smooth (due to $\|\cdot\|_\Phi$). Moreover, due to the nuclear norm term being contained in the product $\|\cdot\|_\Phi\|\cdot\|_A$ (cf. section 2.4), the resulting factorisation $\boldsymbol{\Phi}\tilde{\mathbf{A}}$ is steered towards being low-rank, in favour of the elaborations in section 2.4. However, the resulting matrix $\tilde{\mathbf{A}}\in\mathbb{R}^{M\times 3N}$ now contains the weights for approximating $\mathbf{K}$ by a linear combination of the columns of $\boldsymbol{\Phi}$, rather than approximating the data matrix $\mathbf{X}$ by a linear combination of the columns of $\boldsymbol{\Phi}$. For the known $\boldsymbol{\Phi}$, the weights that best approximate the *data matrix* $\mathbf{X}$ are found by solving the linear system

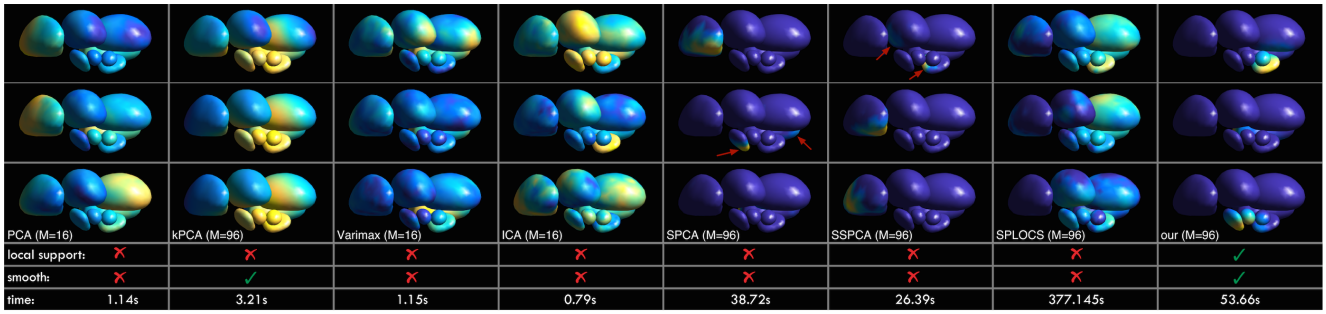


Figure 3. The colour-coded magnitude (blue corresponds to zero, yellow to the maximum deformation in each plot) for the three deformation factors with largest $\ell_2$ norm is shown in the rows. The factors obtained by SPCA and SSPCA are sparse but not spatially localised (see red arrows). Our method is the only one that obtains local support factors.

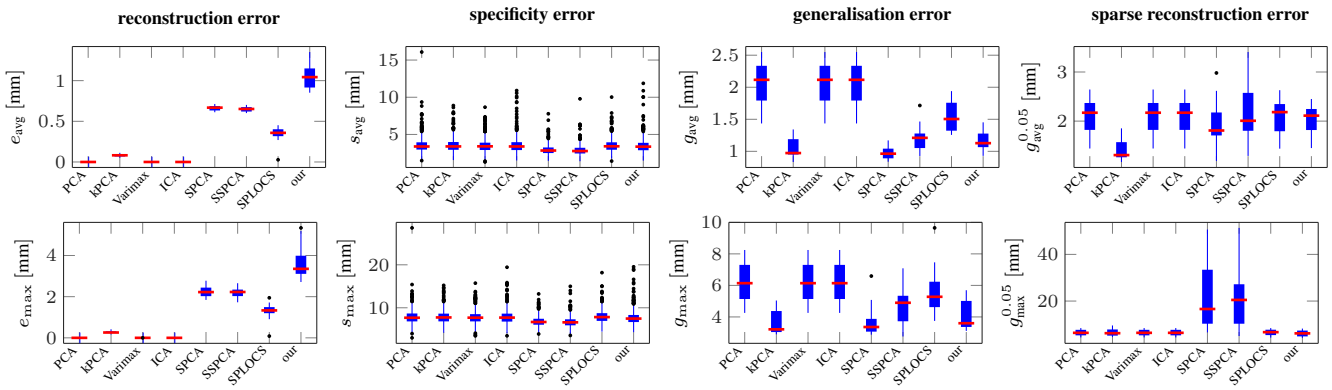


Figure 4. Boxplots of the quantitative measures for the brain shapes dataset. In each plot, the horizontal axis shows the individual methods and the vertical axis the error scores described in section 3.1. Compared to SPLOCS, which is the only other method explicitly striving for local support deformation factors, our method has a smaller generalisation error and sparse reconstruction error. The sparse but not spatially localised factors obtained by SPCA and SSPCA (cf. Fig. 3) result in a large maximum sparse reconstruction error (bottom right).

$\mathbf{\Phi A}{=}\mathbf{X}$ in the least-squares sense for $\mathbf{A}{\in}\mathbb{R}^{M\times K}$.

#### 3.2.3 Results

The magnitude of the deformation of the first three factors are shown in Fig. 3, where it can be seen that only SPCA, SSPCA, SPLOCS and our method obtain *sparse* deformation factors. The SPCA and SSPCA methods do not incorporate the spatial relation between vertices and as such the deformations are not spatially localised (see red arrows in Fig. 3, where more than a single region is active). The factors obtained by SPLOCS are non-smooth and do not exhibit local support, in contrast to our method, where smooth deformation factors with local support are obtained.

The quantitative results presented in Fig. 4 reveal that our method has a larger reconstruction error. This can be explained by the fact that due to the sparsity and smoothness of the deformation factors a very large number of basis vectors is required in order to exactly span the subspace of the training data. Instead, our method finds a simple (sparse and smooth) representation that explains the data approximately, in favour of Occam's razor. The average reconstruction error is around 1mm, which is low considering that the brain structures span approximately 6cm from left to right. Considering specificity, all methods are comparable. PCA, Varimax and ICA, which have the lowest reconstruction errors, have the highest generalisation errors, which underlines that these methods overfit the training data. The kPCA method is able to overcome this issue due to the smoothness assumption. SPCA and SSPCA have good generalisation scores but at the same time a very high maximum reconstruction error. Our method and SPLOCS are the only ones that explicitly strive for local support factors. Since our method outperforms SPLOCS with respect to generalisation and sparse reconstruction error, we claim that our method outperforms the state of the art.

### 3.3. Human Body Shapes

Our second experiment is based on 1531 female human body shapes [41], where each shape comprises 12500 vertices that are in correspondence among the training set. Due to the large number of training data and the high level of details in the meshes, we directly factorise the data matrix $\mathbf{X}$. The edge set $\mathcal{E}$ now contains the edges of the triangle mesh topology and the weights for edge $e = (i,j) \in \mathcal{E}$ are given by $\boldsymbol{\omega}_e = \exp(-(\frac{(\mathbf{D}_{\text{euc}})_{ij}}{\bar{d}})^2)$, where $\bar{d}$ denotes the average vertex distance between neighbour vertices. Edges with weights lower than $\theta{=}0.1$ are ignored.

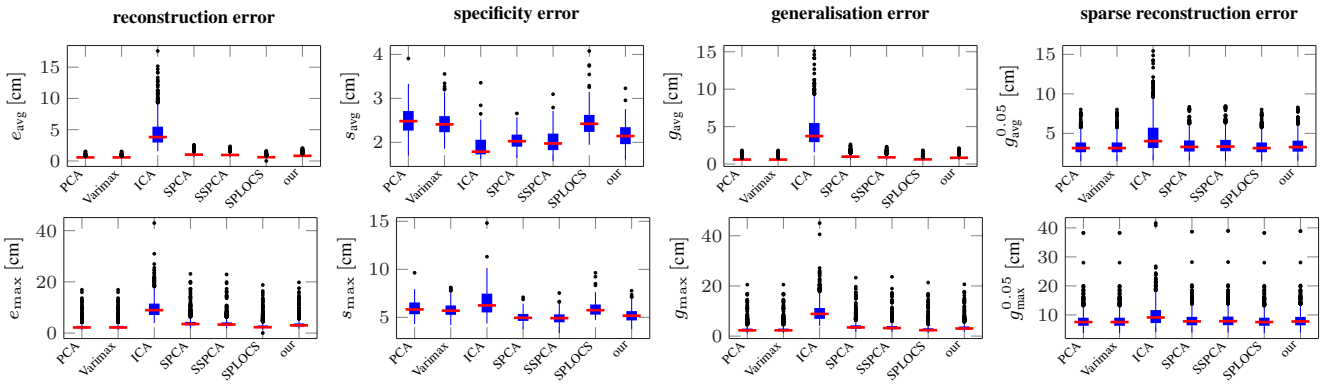


Figure 5. Boxplots of the quantitative measures in each column for the body shapes dataset. Quantitatively all methods have comparable performance, apart from ICA which performs worse.

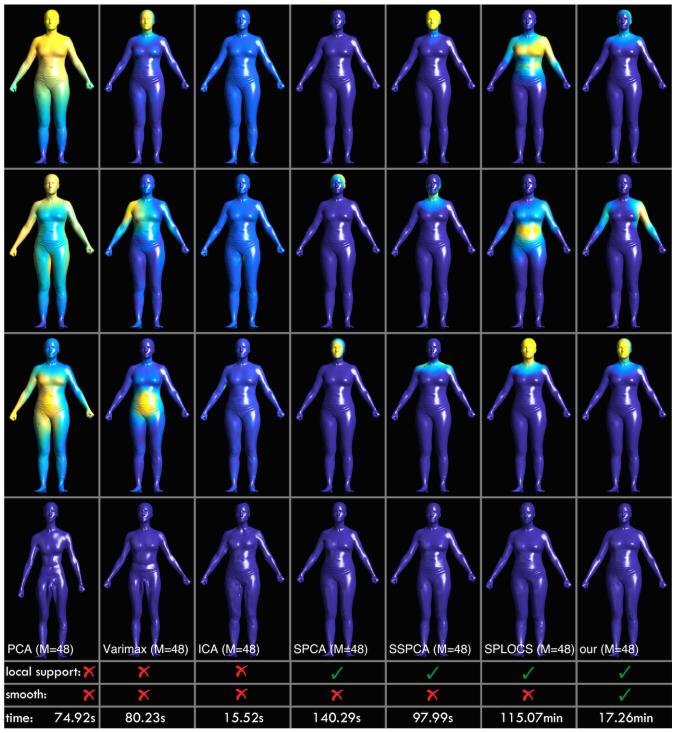


Figure 6. The deformation magnitudes reveal that SPCA, SSPCA, SPLOCS and our method obtain local support factors (in the second factor of our method the connection is at the back). The bottom row depicts randomly drawn shapes, where only the methods with local support deformation factors result in plausible shapes.

#### 3.3.1 Results

Quantitatively the evaluated methods have comparable performance, with the exception that ICA has worse overall performance (Fig. 5). The most noticeable difference between the methods is the specificity error, where SPCA, SSPCA and our method perform best. Fig. 6 reveals that SPCA, SSPCA, SPLOCS and our method obtain factors with local support. Apparently, for large datasets, sparsity alone, as used in SPCA and SSPCA, is sufficient to obtain local support factors. However, our method is the only one that explicitly aims for smoothness of the factors, which leads to more realistic deformations, as shown in Fig. 7.

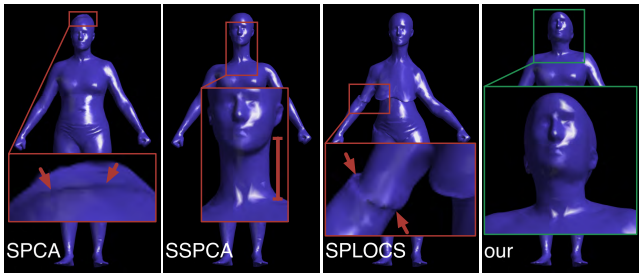


Figure 7. Shapes $\bar{\mathbf{x}}-1.5\mathbf{\Phi}_m$ for SPCA ($m$=1), SSPCA ($m$=3), SPLOCS ($m$=1) and our ($m$=1) method (cf. Fig. 6). Our method delivers the most realistic per-factor deformations.

## 4. Conclusion

We presented a novel approach for learning a linear deformation model from training shapes, where the resulting factors exhibit local support. By embedding sparsity and smoothness regularisers into a theoretically well-grounded matrix factorisation framework, we model local support regions *implicitly*, and thus get rid of the initialisation of the size and location of local support regions, which so far has been necessary in existing methods. On the small brain shapes dataset that contains relatively simple shapes, our method improves the state of the art with respect to generalisation and sparse reconstruction. For the large body shapes dataset containing more complex shapes, quantitatively our method is on par with existing methods, whilst it delivers more realistic per-factor deformations. Since articulated motions violate our smoothness assumption, our method cannot handle them. However, when smooth deformations are a reasonable assumption, our method offers a higher flexibility and better interpretability compared to existing methods, whilst at the same time delivering more realistic deformations.

## Acknowledgements

We thank Yipin Yang and colleagues for making the human body shapes dataset publicly available; Benjamin D.

Haeffele and René Vidal for providing their code; Thomas Bühler and Daniel Cremers for helpful comments on the manuscript; Luis Salamanca for helping to improve our figures; and Michel Antunes and Djamila Aouada for pointing out relevant literature. The authors gratefully acknowledge the financial support by the Fonds National de la Recherche, Luxembourg (6538106, 8864515).

## References

[1] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. *Optimization for Machine Learning*, pages 19–53, 2011. 11

[2] F. Bach, J. Mairal, and J. Ponce. Convex sparse matrix factorizations. *arXiv.org*, 2008. 3, 4

[3] F. Bernard, L. Salamanca, J. Thunberg, F. Hertel, J. Goncalves, and P. Gemmar. Shape-aware 3D Interpolation using Statistical Shape Models. In *Proceedings of Shape Symposium*, Delemont, Sept. 2015. 2, 6

[4] F. Bernard, L. Salamanca, J. Thunberg, A. Tack, D. Jentsch, H. Lamecker, S. Zachow, F. Hertel, J. Goncalves, and P. Gemmar. Shape-aware Surface Reconstruction from Sparse Data. *arXiv.org*, Feb. 2016. 2, 6

[5] F. Bernard, N. Vlassis, P. Gemmar, A. Husch, J. Thunberg, J. Goncalves, and F. Hertel. Fast correspondences for statistical shape models of brain structures. In *SPIE Medical Imaging*, San Diego, 2016. 6

[6] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 6

[7] D. Boscaini, D. Eynard, D. Kourounis, and M. M. Bronstein. Shape-from-Operator: Recovering Shapes from Intrinsic Operators. In *Computer Graphics Forum*, pages 265–274. Wiley Online Library, 2015. 2

[8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2009. 12

[9] S. Burer and R. D. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005. 3, 4

[10] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11:1–11:37, 2011. 2

[11] P. L. Combettes, D. Dũng, and B. C. Vũ. Dualization of signal recovery problems. *Set-Valued and Variational Analysis*, 18(3-4):373–404, 2010. 4, 11, 12

[12] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011. 4, 11, 12

[13] T. F. Cootes and C. J. Taylor. Active Shape Models - Smart Snakes. In *In British Machine Vision Conference*, pages 266–275. Springer-Verlag, 1992. 1, 3, 5

[14] T. F. Cootes and C. J. Taylor. Combining point distribution models with shape models based on finite element analysis. *Image and Vision Computing*, 13(5):403–409, 1995. 2, 6

[15] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008. 11

[16] B. Haeffele, E. Young, and R. Vidal. Structured low-rank matrix factorization: Optimality, algorithm, and applications to image processing. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 2007–2015, 2014. 3, 4, 5, 6, 11

[17] H. H. Harman. *Modern factor analysis*. University of Chicago Press, 1976. 5

[18] M. Hein and T. Bühler. An inverse power method for nonlinear eigenproblems with applications in 1-spectral clustering and sparse PCA. In *Advances in Neural Information Processing Systems 23*, pages 847–855, 2010. 2

[19] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, Apr. 2001. 2, 5

[20] R. Jenatton, G. Obozinski, and F. Bach. Structured sparse principal component analysis. *The Journal of Machine Learning Research*, (AISTATS Proceedings 9), 2010. 2, 5, 13

[21] B. Jiang, C. Ding, B. Luo, and J. Tang. Graph-Laplacian PCA: Closed-form solution and robustness. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3492–3498, 2013. 2, 4

[22] A. Kovnatsky, M. M. Bronstein, X. Bresson, and P. Vandergheynst. Functional correspondence by matrix completion. In *Computer Vision and Pattern Recognition (CVPR)*, pages 905–914, 2015. 2

[23] A. Kovnatsky, K. Glashoff, and M. M. Bronstein. MADMM: a generic algorithm for non-smooth optimization on manifolds. *arXiv.org*, May 2015. 2

[24] C. Last, S. Winkelbach, F. M. Wahl, K. W. Eichhorn, and F. Bootz. A locally deformable statistical shape model. In *Machine Learning in Medical Imaging*, pages 51–58. Springer, 2011. 2

[25] M. Meyer and J. Anderson. Key point subspace acceleration and soft caching. *ACM Transactions on Graphics (TOG)*, 26(3):74, 2007. 2

[26] T. Neumann, K. Varanasi, C. Theobalt, M. Magnor, and M. Wacker. Compressed manifold modes for mesh processing. In *Computer Graphics Forum*, pages 35–44. Wiley Online Library, 2014. 2

[27] T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. Magnor, and C. Theobalt. Sparse localized deformation components. *ACM Transactions on Graphics (TOG)*, 32(6):179, 2013. 2, 5, 13

[28] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 2013. 4, 11, 12

[29] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010. 4

[30] R. M. Rustamov, M. Ovsjanikov, O. Azencot, M. Ben-Chen, F. Chazal, and L. Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Transactions on Graphics (TOG)*, 32(4):72, 2013. 2

[31] N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. Robust Principal Component Analysis on Graphs. In *International Conference on Computer Vision (ICCV)*, 2015. 2

[32] K. Sjostrand, E. Rostrup, C. Ryberg, R. Larsen, C. Studholme, H. Baezner, J. Ferro, F. Fazekas, L. Pantoni, D. Inzitari, and G. Waldemar. Sparse Decomposition and Modeling of Anatomical Shape Variation. *IEEE Transactions on Medical Imaging*, 26(12):1625–1635, 2007. 2

[33] M. B. Stegmann, K. Sjöstrand, and R. Larsen. Sparse modeling of landmark and texture variability using the orthomax criterion. In *SPIE Medical Imaging*, pages 61441G–61441G. SPIE, 2006. 2

[34] A. Suinesiaputra, A. F. Frangi, M. Üzümcü, J. H. Reiber, and B. P. Lelieveldt. Extraction of myocardial contractility patterns from short-axes MR images using independent component analysis. In *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis*, pages 75–86. Springer, 2004. 2

[35] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972. 12

[36] J. R. Tena, F. De la Torre, and I. Matthews. Interactive region-based linear 3d face models. *ACM Transactions on Graphics (TOG)*, 30(4):76, July 2011. 2, 5

[37] M. Üzümcü, A. F. Frangi, M. Sonka, J. H. C. Reiber, and B. P. F. Lelieveldt. ICA vs. PCA Active Appearance Models: Application to Cardiac MR Segmentation. *MICCAI*, pages 451–458, 2003. 2

[38] Y. Wang and L. H. Staib. Boundary finding with correspondence using statistical shape models. In *Computer Vision and Pattern Recognition (CVPR)*, pages 338–345, 1998. 6

[39] Y. Wang and L. H. Staib. Boundary finding with prior shape and smoothness models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):738–743, 2000. 2, 6

[40] Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013. 4, 5

[41] Y. Yang, Y. Yu, Y. Zhou, S. Du, J. Davis, and R. Yang. Semantic Parametric Reshaping of Human Body Models. In *Proceedings of the 2014 Second International Conference on 3D Vision-Volume 02*, pages 41–48. IEEE Computer Society, 2014. 2, 7

[42] R. Zewail, A. Elsafi, and N. Durdle. Wavelet-Based Independent Component Analysis For Statistical Shape Modeling. In *Canadian Conference on Electrical and Computer Engineering*, pages 1325–1328. IEEE, 2007. 2

[43] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006. 2

# A. Linear Shape Deformation Models with Local Support using Graph-based Structured Matrix Factorisation – Supplementary Material

## A.1. The matrix $\mathbf{E}$ in eq. (8)

The matrix $\mathbf{E} \in \mathbb{R}^{3|\mathcal{E}|\times 3N}$ is defined as $\mathbf{E} = \mathbf{I}_3 \otimes \mathbf{E}'$, where $\mathbf{E}' \in \mathbb{R}^{|\mathcal{E}|\times N}$ is the weighted incidence matrix of the graph $\mathcal{G}{=}(\mathcal{V}, \mathcal{E}, \boldsymbol{\omega})$ with elements

$$\mathbf{E}'_{pq} = \begin{cases} \sqrt{\boldsymbol{\omega}_{e_p}} & \text{if } q = i \text{ for } e_p = (i,j) \\ -\sqrt{\boldsymbol{\omega}_{e_p}} & \text{if } q = j \text{ for } e_p = (i,j) \\ 0 & \text{otherwise}\,. \end{cases} \tag{18}$$

For $p = 1, \ldots, |\mathcal{E}|$, $e_p \in \mathcal{E}$ denotes the $p$-th edge.

## A.2. Proximal Operators

**Definition 1.** *The* proximal operator *(or* proximal mapping*)* $\operatorname{prox}_{s\theta}(y) : \mathbb{R}^n \to \mathbb{R}^n$ *of a lower semicontinuous function* $\theta : \mathbb{R}^n \to \mathbb{R}$ *scaled by* $s > 0$ *is defined by*

$$\operatorname{prox}_{s\theta}(\mathbf{y}) = \arg\min_{\mathbf{x}} \left\{ \frac{1}{2s}\|\mathbf{x}-\mathbf{y}\|_2^2 + \theta(\mathbf{x}) \right\}. \tag{19}$$

### A.2.1 Proximal Operator of $\|\cdot\|_A$

The proximal operator $\operatorname{prox}_{\|\cdot\|_A}(\mathbf{y})$ of

$$\|\cdot\|_A = \lambda_A \|\cdot\|_2\,,$$

cf. eq. (5), can be computed by the so-called *block soft thresholding* [28], i.e.

$$\operatorname{prox}_{\lambda_A\|\cdot\|_2}(\mathbf{y}) = (1 - \lambda_A/\|\mathbf{y}\|_2)_+ \mathbf{y}\,, \tag{20}$$

where for a vector $\mathbf{x} \in \mathbb{R}^p$, $(\mathbf{x})_+$ replaces each negative element in $\mathbf{x}$ with 0. As such, a very efficient way for computing the proximal mapping of $\|\cdot\|_A$ is available.

### A.2.2 Proximal Operator of $\|\cdot\|_\Phi$

The proximal mapping of

$$\begin{aligned} \|\cdot\|_\Phi = &\lambda_1\|\cdot\|_1 + \lambda_2\|\cdot\|_2 \\ &+ \lambda_\infty\|\cdot\|_{1,\infty}^{\mathcal{H}} + \lambda_{\mathcal{G}}\|\mathbf{E}\cdot\|_2\,, \end{aligned}$$

cf. eq. (6), is given by

$$\begin{aligned} &\operatorname{prox}_{\|\cdot\|_\Phi}(\mathbf{z}) = \\ &\arg\min_{\mathbf{x}} \{ \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|_2^2 + \lambda_1\|\mathbf{x}\|_1 + \lambda_2\|\mathbf{x}\|_2 \\ &\qquad + \lambda_\infty\|\mathbf{x}\|_{1,\infty}^{\mathcal{H}} + \lambda_{\mathcal{G}}\|\mathbf{E}\mathbf{x}\|_2 \} \,. \end{aligned} \tag{21}$$

Due to the non-separability caused by the linear mapping $\mathbf{E}$ inside the $\ell_2$ norm, this case is more difficult compared to $\operatorname{prox}_{\|\cdot\|_A}(\mathbf{y})$. However, by introducing

$$f(\mathbf{x}) = \lambda_1\|\mathbf{x}\|_1 + \lambda_2\|\mathbf{x}\|_2 + \lambda_\infty\|\mathbf{x}\|_{1,\infty}^{\mathcal{H}} \tag{22}$$

and

$$g(\mathbf{x}) = \lambda_{\mathcal{G}}\|\mathbf{x}\|_2\,, \tag{23}$$

eq. (21) can be rewritten as

$$\arg\min_{\mathbf{x}} \{ \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|_2^2 + f(\mathbf{x}) + g(\mathbf{E}\mathbf{x}) \}\,. \tag{24}$$

In this form, (24) can now be solved by a dual forward-backward splitting procedure [11, 12] as shown in algorithm 2, which we will explain in the rest of this section. The efficient computability hinges on the efficient computation of the (individual) proximal mappings of $f$ and $g$.

**Input**: $\mathbf{z} \in \mathbb{R}^{3N}$
**Output**: $\mathbf{x} = \operatorname{prox}_{\|\cdot\|_\Phi}(\mathbf{z})$
**Parameters**: $\lambda,\ \lambda_1,\ \lambda_\infty,\ \lambda_2,\ \lambda_{\mathcal{G}}$
**Initialise**: $\mathbf{v}, \epsilon \leftarrow 1e^{-4}, \gamma \leftarrow 1.999, \beta = \frac{1+\epsilon}{2}$
`// Normalise E (homogeneity of norm)`
$\lambda_{\mathcal{G}} \leftarrow \lambda_{\mathcal{G}}\|\mathbf{E}\|_F$; $\mathbf{E} \leftarrow \frac{\mathbf{E}}{\|\mathbf{E}\|_F}$
**repeat**
  $\mathbf{x} \leftarrow \operatorname{prox}_{\lambda_1\|\cdot\|_1}(\mathbf{z} - \mathbf{E}^T\mathbf{v})$ `// ℓ1 prox, (25)`
  $\mathbf{x} \leftarrow \operatorname{prox}_{\lambda_\infty\|\cdot\|_{1,\infty}^{\mathcal{H}}}(\mathbf{x})$ `// ℓ1/ℓ∞, [1, 15]`
  $\mathbf{x} \leftarrow \operatorname{prox}_{\lambda_2\|\cdot\|_2}(\mathbf{x})$ `// ℓ2 prox, (20)`
  $\mathbf{v} \leftarrow \mathbf{v} + \beta\gamma(\mathbf{E}\mathbf{x} - \operatorname{prox}_{\lambda_{\mathcal{G}}/\gamma\|\cdot\|_2}(\frac{\mathbf{v}+\gamma\mathbf{E}\mathbf{x}}{\gamma}))$ `//` †
**until** *convergence*

**Algorithm 2:** Dual forward-backward splitting algorithm to compute the proximal mapping of $\|\cdot\|_\Phi$.

Since $g$ is a (weighted) $\ell_2$ norm, its proximal mapping is given by *block soft thresholding* presented in eq. (20).

In $f$, the sum of weighted $\ell_1$ and $\ell_1/\ell_\infty$ norms is a term that appears in the *sparse group lasso* and can be computed by applying the *soft thresholding* [28]

$$\operatorname{prox}_{s\|\cdot\|_1}(\mathbf{y}) = (\mathbf{y} - s)_+ - (-\mathbf{y} - s)_+ \tag{25}$$

first, followed by *group soft thresholding* [1]. As shown in [16, Thm. 3], the $\ell_2$ term can be additionally incorporated by composition, i.e. by subsequently applying *block soft thresholding* as presented in (20).

Now, to solve the *group soft thresholding* for the proximal mapping of the $\ell_1/\ell_\infty$ norm, one can use the fact that for any norm $\omega$ with dual norm $\omega^*$, $\operatorname{prox}_{s\omega}(\mathbf{y}) = \mathbf{y} - \operatorname{proj}_{\omega^*\le s}(\mathbf{y})$. By $\operatorname{proj}_{\omega^*\le s}(\mathbf{y})$, we denote the projection of $\mathbf{y}$ onto the $\omega^*$ norm ball with radius $s$ [1]. The dual norm of the $\ell_1/\ell_\infty$ norm, eq. (7), is the $\ell_\infty/\ell_1$ norm

$$\|\mathbf{z}\|_{\infty,1}^{\mathcal{H}} = \max_{g\in\mathcal{H}} \|\mathbf{z}_g\|_1\,. \tag{26}$$

The orthogonal projection $\operatorname{proj}_{\|\cdot\|_{\infty,1}^{\mathcal{H}}\le s}$ onto the $\ell_\infty/\ell_1$ ball is obtained by projecting separately each subvector $\mathbf{z}_g$ onto the $\ell_1$ ball in $\mathbb{R}^{|g|}$ [1]. This demands an efficient projection onto the $\ell_1$ norm ball. Due to the special structure of our groups in $\mathcal{H}$, i.e. there are exactly $N$ non-overlapping groups, each of which consisting of three elements, a vectorised Matlab implementation of the method by Duchi et al. [15] can be employed.

**Definition 2.** *(Convex Conjugate)*

*Let* $\theta^\star(\mathbf{y}) = \sup_{\mathbf{x}}(\mathbf{y}^T\mathbf{x} - \theta(\mathbf{x}))$ *be the* convex conjugate *of* $\theta$.

The update of the dual variable $\mathbf{v}$ in algorithm 2 (see the line marked with $^\dagger$) is based on the update

$$\mathbf{v} \leftarrow \mathbf{v} + \beta(\mathrm{prox}_{\gamma(\lambda_\mathcal{G}\|\cdot\|_2)^\star}(\mathbf{v} + \gamma\mathbf{E}\mathbf{x}) - \mathbf{v})\,, \qquad (27)$$

presented in [12, 11], where $(\cdot)^\star$ denotes the convex conjugate.

Let us introduce some tools first.

**Lemma 2.** *(Extended Moreau Decomposition) [28] It holds that*

$$\forall \mathbf{y}:\ \mathbf{y} = \mathrm{prox}_{s\theta}(\mathbf{y}) + s\,\mathrm{prox}_{\theta^\star/s}(\mathbf{y}/s)\,. \qquad (28)$$

**Lemma 3.** *(Conjugate of conjugate) [8] For closed convex* $\theta$*, it holds that* $\theta^{\star\star} = \theta$.

**Corollary 1.** *For convex and closed* $\theta$*, it holds that*

$$\forall \mathbf{y}:\ \mathbf{y} = \mathrm{prox}_{s\theta^\star}(\mathbf{y}) + s\,\mathrm{prox}_{\theta/s}(\mathbf{y}/s)\,. \qquad (29)$$

*Proof.* Define $\theta' = \theta^\star$ and apply Lemma 2 and Lemma 3 with $\theta'$ in place of $\theta$. □

Since for our choice of $\theta = \lambda_\mathcal{G}\|\cdot\|_2$, $\theta$ is a closed convex function, by Corollary 1, one can write

$$\mathrm{prox}_{\gamma(\lambda_\mathcal{G}\|\cdot\|_2)^\star}(\mathbf{y}) = \mathbf{y} - \gamma\,\mathrm{prox}_{(\lambda_\mathcal{G}/\gamma)\|\cdot\|_2}(\mathbf{y}/\gamma)\,. \qquad (30)$$

With that, the right-hand side of eq. (27) can be written as

$$\mathbf{v} + \beta(\mathbf{v} + \gamma\mathbf{E}\mathbf{x} - \gamma\,\mathrm{prox}_{\lambda_\mathcal{G}/\gamma\|\cdot\|_2}(\frac{\mathbf{v} + \gamma\mathbf{E}\mathbf{x}}{\gamma}) - \mathbf{v}) =$$

$$\mathbf{v} + \beta\gamma(\mathbf{E}\mathbf{x} - \mathrm{prox}_{\lambda_\mathcal{G}/\gamma\|\cdot\|_2}(\frac{\mathbf{v} + \gamma\mathbf{E}\mathbf{x}}{\gamma}))\,. \qquad (31)$$

## A.3. Computational Complexity

In practice it holds that $N \gg \max(M, K)$. The gradient steps for the updates of $\mathbf{\Phi}$ and $\mathbf{A}$ have both time complexity $\mathcal{O}(N\max(M^2, K^2))$, the proximal step for $\mathbf{A}$ has complexity $\mathcal{O}(MK)$, and the proximal step for $\mathbf{\Phi}$ has complexity $\mathcal{O}(MN|\mathcal{E}|n_{\text{it}})$, where $n_{\text{it}}$ is the number of iterations for the dual forward-backward splitting procedure (we used a maximum of $n_{\text{it}}$=20). Thus, the total time complexity for one iteration in algorithm 1 is $\mathcal{O}(N\max(M^2, K^2) + MN|\mathcal{E}|n_{\text{it}})$. Since in practice the number of vertices $N$ and the number of edges in the graph $|\mathcal{E}|$ are larger than $M$ and $K$, the runtime complexity is dominated by the proximal step for $\mathbf{\Phi}$, which in our experiments takes around 60% of the time for the brain shapes dataset ($N$=1792, $M$=96, $K$=17 , $|\mathcal{E}|$=19182), and uses more than 90% of the time for the human body shapes dataset ($N$=12500, $M$=48, $K$=1531, $|\mathcal{E}|$=99894).

## A.4. Factor Splitting

The *factor splitting* procedure is presented in algorithm 3.

**Input**: factor $\boldsymbol{\phi} \in \mathbb{R}^{N\times 3}$ where $\mathrm{vec}(\boldsymbol{\phi}) = \mathbf{\Phi}_m$, graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
**Output**: $J$ factors $\mathbf{\Phi}' \in \mathbb{R}^{3N\times J}$ with local support
**Initialise**: $\mathcal{E}' = \emptyset$, $\mathbf{\Phi}' = [\,]$
`// build activation graph` $\mathcal{G}'$
**foreach** $(i, j) = e \in \mathcal{E}$ **do**
  **if** $\boldsymbol{\phi}_{i,:} \neq \mathbf{0} \vee \boldsymbol{\phi}_{j,:} \neq \mathbf{0}$ **then** `// vertex` $i$ `or` $j$ `is active`
    $\mathcal{E}' = \mathcal{E}' \cup \{e\}$ `// add edge`
$\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$
`// find connected components [35]`
$\mathcal{C}$ = connectedComponents($\mathcal{G}'$) `//` $\mathcal{C} \subset 2^\mathcal{V}$
**foreach** $c \in \mathcal{C}$ **do** `// add new factor for` $c \subset \mathcal{V}$
  $\boldsymbol{\phi}' = \mathbf{0}_{N\times 3}$
  $\boldsymbol{\phi}'_{c,:} = \boldsymbol{\phi}_{c,:}$
  $\mathbf{\Phi}' = [\mathbf{\Phi}',\ \mathrm{vec}(\boldsymbol{\phi}')]$

**Algorithm 3:** Factor splitting procedure.

## A.5. Convergence Plots

For 100 different initialisations (cf. section 3), the convergence plots are shown for both datasets in Fig. 8. It can be seen that the main convergence occurs after around 10 iterations. Moreover, for all 100 initialisations the objective value are near-congruent.

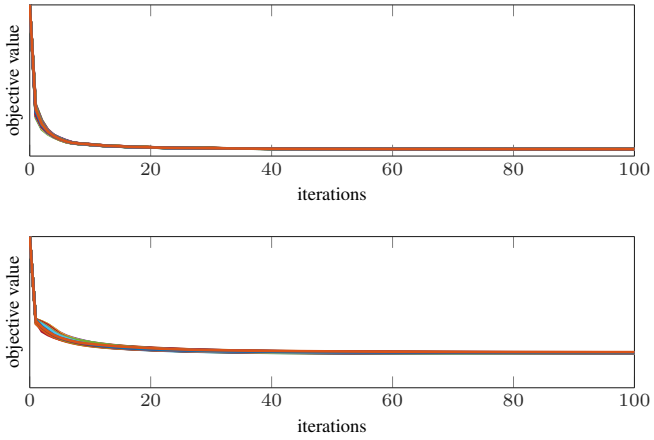


Figure 8. Convergence plots for the brain shapes dataset (top) and the human body shapes dataset (bottom). The iterations are shown on the horizontal axis and the (relative) objective value in eq. (4) is shown on the vertical axis. Note that in each subfigure all 100 lines are near-congruent.

## A.6. Parameter Random Sampling

The methods kPCA, SPCA, SSPCA, SPLOCS and our method require various parameters to be set. In order to find a good parametrisation we conducted random sampling over the parameter space, where we determined reasonable ranges for the parameters experimentally.

In Table 1 the distributions and default values of the parameters for each method are given. $\mathcal{U}(a, b)$ is the uniform distribution with the open interval $(a, b)$ as support, $10^{\mathcal{U}(\cdot,\cdot)}$ is a distribution of the random variable $y = 10^x$, where $x \sim \mathcal{U}(\cdot,\cdot)$, and $\bar{d}_{\max}$ is the largest distance between all pairs of vertices of the mean shape $\bar{X}$.

| Method | Parameter Distribution / Default Value |
|---|---|
| kPCA | $\beta \sim (\mathcal{U}(1,10))^{-1}$ |
| SPCA | $\lambda \sim \frac{1}{3N} 10^{\mathcal{U}(-4,-3)}$ (see eq. (2) in [20]) |
| SSPCA | $\lambda \sim \frac{1}{N} 10^{\mathcal{U}(-4,-3)}$ (see eq. (2) in [20]) |
| SPLOCS | $\lambda \sim (3K) \cdot 10^{\mathcal{U}(-4,-3)}$; $d_{\min} \sim c - w$; $d_{\max} \sim c + w$ (see eq. (6) in [27]), where $c \sim \mathcal{U}(0.1, \bar{d}_{\max} - 0.1)$ and $w \sim \mathcal{U}(0, \min(\lvert c - 0.1 \rvert, \lvert \bar{d}_{\max} - 0.1 - c \rvert))$ |
| our | $\beta \sim (\mathcal{U}(1,10))^{-1}$; $\lambda = 64 \cdot \frac{3NK}{M}$; $\lambda_{\mathcal{G}} = \frac{1}{\sqrt{3\lvert\mathcal{E}\rvert}}$; $\lambda_1 = \frac{1}{\sqrt{3N}}$; $\lambda_2 = \frac{1}{\sqrt{3N}}$; $\lambda_\infty = 2 \cdot \frac{1}{\sqrt{N}}$; $\lambda_A = 10^{-4} \cdot \frac{1}{\sqrt{K}}$ |

Table 1. Assumed distributions of the parameters interpreted as random variables.

For each of the $n_r$ random samples (we set $n_r = 500$ for the brain shapes dataset, and $n_r = 50$ for the human body shapes, due to the large size of the dataset) we compute the $8$ scores described in section 3.1 and store them in the score matrix $\mathbf{S} \in \mathbb{R}^{n_r \times 8}$. After (linearly) mapping the elements of each $n_r$-dimensional column vector in $\mathbf{S}$ onto the interval $[0, 1]$, the best parametrisation is determined by finding the index of the smallest value of the vector $\mathbf{S}\mathbf{1}_8 \in \mathbb{R}^{n_r}$.

For the lambda parameters of the proposed method we identified default values that we used for the evaluation of both datasets. Moreover, a normalisation of the parameters is conducted for all methods. For kPCA, SPCA, SSPCA and our method the random sampling was conducted only for a single parameter, whereas for the SPLOCS method three parameters had to be set, two of them related to the size of the local support region.