

## 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  $\mathcal{G}=(\mathcal{V}, \mathcal{E}, \omega)$  with elements

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

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

### A.2. Proximal Operators

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

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

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

The proximal operator  $\text{prox}_{\|\cdot\|_A}(\mathbf{y})$  of  $\|\cdot\|_A = \lambda_A \|\cdot\|_2$ ,

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

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

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

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

cf. eq. (6), is given by

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

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

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

and

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

eq. (21) can be rewritten as

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

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:  $\mathbf{x} = \text{prox}_{\|\cdot\|_\Phi}(\mathbf{z})$ 
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  $\mathbf{E}$  (homogeneity of norm)
 $\lambda_\mathcal{G} \leftarrow \lambda_\mathcal{G} \|\mathbf{E}\|_F; \quad \mathbf{E} \leftarrow \frac{\mathbf{E}}{\|\mathbf{E}\|_F}$ 
repeat
   $\mathbf{x} \leftarrow \text{prox}_{\lambda_1 \|\cdot\|_1}(\mathbf{z} - \mathbf{E}^T \mathbf{v})$  //  $\ell_1$  prox, (25)
   $\mathbf{x} \leftarrow \text{prox}_{\lambda_\infty \|\cdot\|_{1,\infty}^\mathcal{H}}(\mathbf{x})$  //  $\ell_1/\ell_\infty$ , [1, 15]
   $\mathbf{x} \leftarrow \text{prox}_{\lambda_2 \|\cdot\|_2}(\mathbf{x})$  //  $\ell_2$  prox, (20)
   $\mathbf{v} \leftarrow \mathbf{v} + \beta \gamma (\mathbf{E}\mathbf{x} - \text{prox}_{\lambda_\mathcal{G}/\gamma \|\cdot\|_2}(\frac{\mathbf{v} + \gamma \mathbf{E}\mathbf{x}}{\gamma}))$  //  $\dagger$ 
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]

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

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  $\omega^*$ ,  $\text{prox}_{s\omega}(\mathbf{y}) = \mathbf{y} - \text{proj}_{\omega^* \leq s}(\mathbf{y})$ . By  $\text{proj}_{\omega^* \leq s}(\mathbf{y})$ , 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

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

The orthogonal projection  $\text{proj}_{\|\cdot\|_{\infty,1}^\mathcal{H} \leq s}$  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  $\theta^*(\mathbf{y}) = \sup_{\mathbf{x}}(\mathbf{y}^T \mathbf{x} - \theta(\mathbf{x}))$  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

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

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

Let us introduce some tools first.

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

It holds that

$$\forall \mathbf{y} : \mathbf{y} = \text{prox}_{s\theta}(\mathbf{y}) + s \text{prox}_{\theta^*/s}(\mathbf{y}/s). \quad (28)$$

**Lemma 3.** (Conjugate of conjugate) [8]

For closed convex  $\theta$ , it holds that  $\theta^{**} = \theta$ .

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

$$\forall \mathbf{y} : \mathbf{y} = \text{prox}_{s\theta^*}(\mathbf{y}) + s \text{prox}_{\theta/s}(\mathbf{y}/s). \quad (29)$$

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

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

$$\text{prox}_{\gamma(\lambda_{\mathcal{G}}\|\cdot\|_2)^*}(\mathbf{y}) = \mathbf{y} - \gamma \text{prox}_{(\lambda_{\mathcal{G}}/\gamma)\|\cdot\|_2}(\mathbf{y}/\gamma). \quad (30)$$

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

$$\begin{aligned} \mathbf{v} + \beta(\mathbf{v} + \gamma \mathbf{E} \mathbf{x} - \gamma \text{prox}_{\lambda_{\mathcal{G}}/\gamma\|\cdot\|_2}(\frac{\mathbf{v} + \gamma \mathbf{E} \mathbf{x}}{\gamma}) - \mathbf{v}) = \\ \mathbf{v} + \beta\gamma(\mathbf{E} \mathbf{x} - \text{prox}_{\lambda_{\mathcal{G}}/\gamma\|\cdot\|_2}(\frac{\mathbf{v} + \gamma \mathbf{E} \mathbf{x}}{\gamma})). \end{aligned} \quad (31)$$

### A.3. Computational Complexity

In practice it holds that  $N \gg \max(M, K)$ . The gradient steps for the updates of  $\Phi$  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  $\Phi$  has complexity  $\mathcal{O}(MN|\mathcal{E}|n_{it})$ , where  $n_{it}$  is the number of iterations for the dual forward-backward splitting procedure (we used a maximum of  $n_{it}=20$ ). Thus, the total time complexity for one iteration in algorithm 1 is  $\mathcal{O}(N \max(M^2, K^2) + MN|\mathcal{E}|n_{it})$ . 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  $\Phi$ , 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  $\phi \in \mathbb{R}^{N \times 3}$  where  $\text{vec}(\phi) = \Phi_m$ , graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ 
Output:  $J$  factors  $\Phi' \in \mathbb{R}^{3N \times J}$  with local support
Initialise:  $\mathcal{E}' = \emptyset$ ,  $\Phi' = []$ 
// build activation graph  $\mathcal{G}'$ 
foreach  $(i, j) = e \in \mathcal{E}$  do
    if  $\phi_{i,:} \neq \mathbf{0} \vee \phi_{j,:} \neq \mathbf{0}$  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]
 $\mathcal{C} = \text{connectedComponents}(\mathcal{G}')$  //  $\mathcal{C} \subset 2^{\mathcal{V}}$ 
foreach  $c \in \mathcal{C}$  do // add new factor for  $c \subset \mathcal{V}$ 
     $\phi'_c = \mathbf{0}_{N \times 3}$ 
     $\phi'_{c,:} = \phi_{c,:}$ 
     $\Phi' = [\Phi', \text{vec}(\phi'_c)]$ 

```

**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.

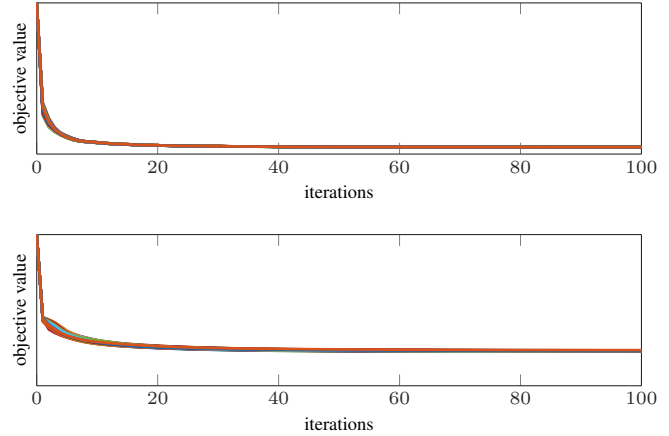


Figure 8. Convergence plots for the brain shapes dataset (top) and the human body shapes dataset (bottom). The iterations are shown on the horizontal axis and the (relative) objective value in eq. (4) is shown on the vertical axis. Note that in each subfigure all 100 lines are near-congruent.

### 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 $w \sim \mathcal{U}(0, \min( c - 0.1 ,  \bar{d}_{\max} - 0.1 - c ))$
our	$\beta \sim (\mathcal{U}(1, 10))^{-1}$ ; $\lambda = 64 \cdot \frac{3NK}{M}$ ; $\lambda_G = \frac{1}{\sqrt{3 \mathcal{E} }}$ ; $\lambda_1 = \frac{1}{\sqrt{3N}}$ ; $\lambda_2 = \frac{1}{\sqrt{3N}}$ ; $\lambda_\infty = 2 \cdot \frac{1}{\sqrt{N}}$ ; $\lambda_A = 10^{-4} \cdot \frac{1}{\sqrt{K}}$

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.