Sparse Localized Deformation Components

Thomas Neumann[1,2], Kiran Varanasi[3,4], Stephan Wenger[2], Markus Wacker[1], Marcus Magnor[2], Christian Theobalt[3]

[1]HTW Dresden, Germany, [2]Computer Graphics Lab, TU Braunschweig, Germany,
[3]Max-Planck-Institut Informatik, Saarbrücken, Germany, [4]Technicolor Research, France

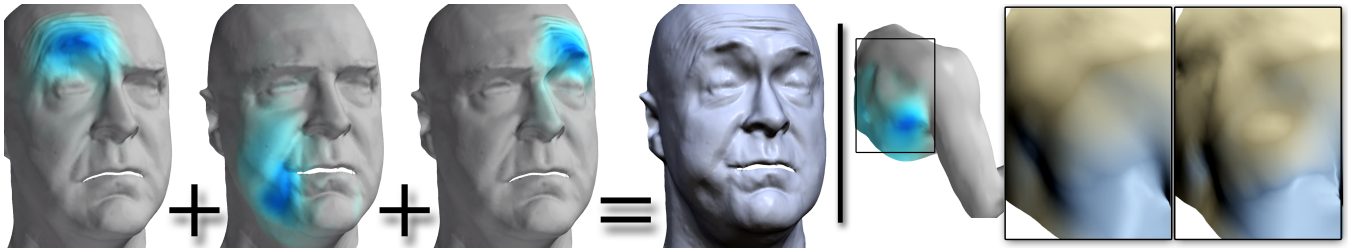


Figure 1: *Our method automatically decomposes any mesh animations like performance captured faces (left) or muscle deformations (right) into sparse and localized deformation modes (shown in blue).* Left: *a new facial expression is generated by summing deformation components. Our method automatically separates spatially confined effects like separate eyebrow motions from the data.* Right: *Our algorithm extracts individual muscle and bone deformations. The deformation components can then be used for convenient editing of the captured animation. Here, the deformation component of the clavicle is over-exaggerated to achieve an artistically desired look.*

Abstract

We propose a method that extracts sparse and spatially localized deformation modes from an animated mesh sequence. To this end, we propose a new way to extend the theory of sparse matrix decompositions to 3D mesh sequence processing, and further contribute with an automatic way to ensure spatial locality of the decomposition in a new optimization framework. The extracted dimensions often have an intuitive and clear interpretable meaning. Our method optionally accepts user-constraints to guide the process of discovering the underlying latent deformation space. The capabilities of our efficient, versatile, and easy-to-implement method are extensively demonstrated on a variety of data sets and application contexts. We demonstrate its power for user friendly intuitive editing of captured mesh animations, such as faces, full body motion, cloth animations, and muscle deformations. We further show its benefit for statistical geometry processing and biomechanically meaningful animation editing. It is further shown qualitatively and quantitatively that our method outperforms other unsupervised decomposition methods and other animation parameterization approaches in the above use cases.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: mesh deformation, editing captured animations, data-driven animation, dimensionality reduction

Links: DL PDF

ACM Reference Format
Neumann, T., Varanasi, K., Wenger, S., Wacker, M., Magnor, M., Theobalt, C. 2013. Sparse Localized Deformation Components. ACM Trans. Graph. 32, 6, Article 179 (November 2013), 10 pages.
DOI = 10.1145/2508363.2508417 http://doi.acm.org/10.1145/2508363.2508417.

0730-0301/13/11-ART179 $15.00.
DOI: http://dx.doi.org/10.1145/2508363.2508417

1 Introduction

Nowadays, time-varying dynamic geometry with very fine dynamic shape detail can be generated and rendered at very high visual fidelity. When creating such content, artists usually rely on a low-dimensional control parametrization, for example a kinematic skeleton rig to control surface motion via joint angles, a muscle system simulating face motion and tissue deformation, or a physics-based simulation model generating realistically deforming cloth or other materials. Despite increasing expressive power of such parametrizations and simulations, producing such realistic animations from scratch is a labor-intensive process, in particular since it is highly non-trivial to design or customize a specific parameterization to a new object to be animated.

Performance capture techniques were thus developed that measure detailed time-varying surface models of the real world in motion from sensor data [Theobalt et al. 2010]. These methods capture highly detailed animation models, mostly as space-time surface mesh sequences. However, their applicability in animation production so far has been strongly limited because a low-dimensional control parametrization for the captured data is missing. Conveniently re-using, editing or otherwise analysing this input data is therefore not possible.

To overcome this problem, some recent work suggested to fit specific parametric models to such input data: simple linear regression models, skeletons or bone transformations, or even pre-modeled facial blendshapes, see Sect. 2. Such methods require additional prior knowledge about the underlying physical process of the animation (e.g. a template shape or a parametric physics model), or additional data needs to be recorded alongside the capture. They are thus object-type specific and not very easy to use on general data. Dimensionality reduction techniques for animation parameterization rely less on specific prior information, and are thus more generally applicable to a broader range of animations. Many of them are, in essence, matrix decomposition techniques that explain observed deformations as a linear combination of factors, or deformation components, that can be controlled by an animator. Many previously used dimensionality reduction techniques, like Principal Component Analysis (PCA), reproduce input data faithfully and maintain certain compression guarantees, but the individual com-

puted dimensions usually lack interpretable meaning and are of global support, i.e. their modification acts on the whole mesh. In contrast, in order to generate professional quality animation, artists require crisp controls on plausible local effects which can be edited in an intuitive manner [Havaldar 2006].

In this paper, we present a new efficient, easy-to-implement, and versatile data-driven approach to decompose a mesh sequence into intuitively meaningful vocabulary of sparse deformation components, irrespective of what the true physical process underlying an animation is. It takes inspiration from matrix decomposition methods such as Non-Negative Matrix Factorization (NMF), Robust PCA, and Sparse PCA, that have recently been very popular in machine learning, image processing and computer vision. We contribute with the first theory for general sparse matrix decomposition for mesh sequence processing in computer graphics. The new algorithm decomposes deformations (here, vertex displacements) into a linear combination of weights and sparse deformation components (Sect. 3). Our method relies on a sparsity-inducing regularizer that is designed for the setting of mesh animations. We further include a mechanism to automatically find components that are not only *sparse* but also *localized* on the mesh (Sect. 3.2). We also provide means to easily let input provided by the user guide the sparse decomposition (Sect. 3.3), and contribute with a new efficient optimization and initialization scheme to compute the decomposition (Sect. 3.4).

Our experiments show that the *sparse localized deformation components* computed by our method are a powerful new tool to elegantly solve a variety of mesh processing and editing tasks. We show our method's strength in automatically building a localized and intuitively meaningful set of deformation components for a variety of captured mesh sequences, such as face animations (Sect. 4.2), muscle deformations (Sect. 4.3), and captured cloth deformations (Sect. 4.4). It is also shown that the automatically inferred components are similar to traditional blendshape models used in face animation, and thus enable much more intuitive animation editing, with meaningful local support, than previous dimensionality reduction techniques and other parameterization learning techniques (e.g. Fig. 1, left). Our decompositions also often permit semantically meaningful analysis and modification of animations, e.g. by finding biomechanically relevant deformation modes corresponding to individual muscles (e.g. Fig. 1, right). They also offer a powerful new way to approach the learning of statistical shape models from scanned examples, where they often enable convenient automatic separation of desired and undesired dimensions of shape variation (Sect. 4.5). In addition to qualitative results, we also quantitatively show that our method outperforms previous approaches in the literature (Sect. 4.1).

A reference implementation of the algorithm is provided at
`http://github.com/tneumann/splocs`

2 Related Work

Skeletal Rigging Artists typically resort to parametric models for synthesizing mesh animations. The de-facto standard for articulated motion is linear blend skinning (and its extensions), often combined with hierarchical kinematic skeletons. The high difficulty of fitting such a model to example mesh animations given by an artist is evaluated in [Mohr and Gleicher 2003; Weber et al. 2007]. Simple hierarchical skeletons can now be fit to articulated mesh sequences [de Aguiar et al. 2008] and explain body shape variations [Hasler et al. 2010]. Such skeletons often miss detail that can be re-introduced in pose space using the Eigenskin method [Kry et al. 2000], which essentially comes down to clustered PCA (with overlapping regions) and linear regression in pose space, and only works given sufficient training data. More general skinning decomposition methods fit unordered non-hierarchical collections of arbitrary [Kavan et al. 2010] or rigid [Le and Deng 2012] bone transformations and blend weights to mesh animations. While useful for quick rendering using GPU's, this is not so useful for editing - the bones offer many degrees of freedom that quickly produce deformations far outside the input range. Above cited approaches differ from our approach in the deformation representation (linear blend skinning vs. vertex displacements). They prescribe a fixed sparsity (*e.g.,* 4 bones per vertex) while our method finds the suitable sparsity for every single vertex. Our method can benefit from skinning decomposition methods as a preprocessing step to explain rough articulations. As we show in this paper, deformation on top of that, such as detailed folds or smaller pose changes, can be better explained using our method.

Facial Animation and Editing Blendshapes are widely used for facial animation. They are another highly important animation parameterization, and customizing them to a target face is a labor intensive process. Adopting given facial blendshape template model to a set of examples was demonstrated by Li et al. [2010]. To enable more local modification, Tena et al. [2011] cluster the human face into regions and learn a clustered PCA model based on marker based motion capture data. Such direct and local manipulation is also possible with our automatically found parameterization, and can be used with direct manipulation tools [Lewis and Anjyo 2010; Seo et al. 2011] without an explicit blending step. An elaborate facial editing system with many interesting user interaction concepts (that are compatible to our deformation components) was presented by Lau et al. [2009]. It relies on a facial prior based on a huge expression database. In contrast, we can automatically find an intuitive-to-control parameterization of arbitrary mesh sequences, not only faces.

Simulating Deformations Physical simulation is often used as parameterization for complex deformations based on material parameters and collisions. Fitting such models to captured mesh data is a very challenging task [Sifakis et al. 2005; Miguel et al. 2012]. Simulation can also be constrained to lie in an artist-designed subspace [Schumacher et al. 2012]. Learned simulation models are object-specific, where as our approach is general and explicitly exposes the learnt subspace for manipulation. On the other hand, pure data driven simulation methods such as [Guan et al. 2012; Kavan et al. 2011; de Aguiar et al. 2009] circumvent the need for physical simulation (almost) completely by learning from real world data. As just recently shown by Kim et al. [2013], successfully learning complex deformations from data is possible but requires a dauntingly huge amount of well structured training data which is often not available, in particular for captured data. A decomposition method that is able to build a latent deformation space that generalizes beyond the training data and allows for user input, as offered by our method, can help to overcome this limitation in the future.

Mesh Deformation Apart from parametrization based deformation techniques, directly editing mesh animations, with ideas similar to static mesh deformation, have emerged that overcome some of the limitations of parametric models [Sumner et al. 2007; Kircher and Garland 2009; Fröhlich and Botsch 2011]. These are powerful methods for editing, but solve a different problem compared to this paper which studies exploration of an underlying lower dimensioinal deformation space.

Low Dimensional Deformation Spaces The problem of finding and navigating a latent space that is able to explain deformations given by (captured) data is a more general problem, and ours is not the first approach to tackle it. The motivation comes from recent success in capturing performances directly from the real world, of human body movements or of detailed facial expressions [Theobalt et al. 2010]. Editing such geometry at each surface point or at each

frame in a sequence is impractical. Many methods transfer ideas from machine learning to identify a lower dimensional space of deformations based on linear (PCA) or non-linear dimensionality reduction techniques [Levine et al. 2012; Cashman and Hormann 2012; Tournier and Reveret 2012]. Feng et al. [2008] deform a mesh and reproduce fine-scale details from given input data and control points by using kernel canonical correlational analysis on top of low-dimensional skeletal deformation. Some methods resort to Independent Component Analysis (ICA) [Hyvärinen et al. 2001], for example for motion editing [Cao et al. 2007]. Shape and pose variations of human bodies can also be learned and modified using linear regression models [Anguelov et al. 2005; Hasler et al. 2009], e.g., by modifying height, weight, gender, etc. Multi-linear models can be used to represent variation in human facial expressions and identity [Vlasic et al. 2005]. A key problem with those approaches is that the extracted components show global correlation of deformations on the surface, whereas artists often require specific deformation control, e.g., of specific muscle groups, which are directly discovered by our method. To boost computation speed, Meyer et al. [2007] use the Varimax rotation of the PCA components as a basis that is localized. Experimentally, we found that Varimax yields unsatisfactory components that show global artifacts in the presence of noise and limited (captured) data.

Sparse Decompositions A major shortcoming of PCA is that it finds components which involve all original variables - in the context of mesh deformations every vertex deforms in each component. Several alternatives have been explored in the recent decade, such as non-negative matrix factorization [Lee and Seung 1999], which finds positive components, and Sparse PCA [Zou et al. 2006; Jolliffe et al. 2003], which introduces a sparsity inducing norm such as ℓ_1 and often drop the orthogonality constraint. The fact that such decomposition methods happen to retrieve a localized set of variables, such as face or brain regions in (fMRI) images, made them popular in computer vision, signal processing and medical imaging. But they have not yet been used in animation processing. Jenatton [2011] mentions two different formulations of sparse PCA: deflation methods [Mackey 2009] consider a single component at each step that constructs a subspace one by one, a strategy we use in our paper for *initialization*. Matrix factorization methods [Mairal et al. 2009] have a non-convex formulation to simultaneously optimize for all the principal components, which we adopt for *global optimization* of our components. Sparse decomposition methods are also related to independent component analysis (ICA) [Hyvärinen et al. 2001], it is known that both methods solve the same problem in the case of no noise and square system where the input and output dimensionalities are equal (see [Olshausen and Field 1997]). In our setting, these conditions are not satisfied and our decomposition outperforms the previously used ICA decomposition in terms of localized control. Not least because of the success of compressed sensing and sparse linear models, minimization of sparsity-inducing norms can be achieved efficiently [Boyd et al. 2011]. In computer graphics, such methods can help to find robust correspondences between meshes [Pokrass et al. 2013]. Deng et al. [2013] recently showed that sparsity inducing norms are very suitable to obtain *local* modifications on constrained static meshes which are popular in architecture. It is the only other study known to us that imposes sparsity on deformations themselves. Such intuitive editing paradigms can be achieved for any captured mesh sequence using the versatile approach presented in this paper.

3 Method

Our proposed method aims to decompose captured or animated mesh sequences into sparse, localized, and intuitive-to-control deformation components. The input data to the algorithm consists of a mesh animation with F frames. Each frame f consists of N vertex positions $\mathbf{v}_i^{(f)}$. The mesh topology is equal for all frames, and the vertices are in correspondence over time. We assemble a single "animation matrix" $\mathbf{X} \in \mathbb{R}^{F\times 3N}$ by stacking the vertices of all frames in a row-wise fashion

$$\mathbf{X} = \begin{bmatrix} (\mathbf{v}_1^{(1)})^\top & (\mathbf{v}_2^{(1)})^\top & \dots & (\mathbf{v}_N^{(1)})^\top \\ (\mathbf{v}_1^{(2)})^\top & (\mathbf{v}_2^{(2)})^\top & \dots & (\mathbf{v}_N^{(2)})^\top \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{v}_1^{(F)})^\top & (\mathbf{v}_2^{(F)})^\top & \dots & (\mathbf{v}_N^{(F)})^\top \end{bmatrix}$$

For convenience, we assume the vertex coordinates in the animation matrix are expressed as residual displacements to a "mean shape" $\hat{\mathbf{x}}$ of the mesh (e.g. the first frame or the average of all frames). We also assume that rigid alignment through global translation and rotation is performed before assembling the animation matrix.

Formally, from now on, we assume $\mathbf{X} \leftarrow \mathbf{X} - \hat{\mathbf{x}}$ and accordingly $\mathbf{v}_i^{(f)} \leftarrow \mathbf{v}_i^f - \hat{\mathbf{v}}_i$. We also normalize $\mathbf{X}$ by its standard variation (across frames and vertices) after subtracting the mean to make the algorithm invariant to the amount of deformation of different datasets.

3.1 Sparse PCA for Mesh Sequences

We are looking for an appropriate matrix factorization of $\mathbf{X}$ into K deformation components $\mathbf{C} \in \mathbb{R}^{K\times 3N}$ with weights $\mathbf{W} \in \mathbb{R}^{F\times K}$

$$\mathbf{X} = \mathbf{W}\,\mathbf{C} \quad . \tag{1}$$

The above model is comparable to the widely used blendshapes [Osipa 2003], where the weights $\mathbf{W}$ are usually key-framed by an animator and the components $\mathbf{C}$ (blendshapes minus mean shape) are prepared by a modeller.

The space of solutions to Eq. (1) has to be regularized by additional constraints on $\mathbf{W}$ and $\mathbf{C}$. For example, PCA (Principal Component Analysis) constrains the components to be orthogonal, $\mathbf{C}^\top\,\mathbf{C} = \mathbf{I}$. However, the principal components usually have global support on the whole mesh and are therefore unintuitive for artistically meaningful localized editing [Lewis and Anjyo 2010], see Fig. 6(a).

Sparse components can be discovered by imposing sparsity on the solution of Eq. (1). To this end, a sparsity inducing norm, such as the ℓ_1 norm can be employed as a regularizer $\Omega(\,\mathbf{C})$ on the components to yield *Sparse PCA* [Zou et al. 2006; Jenatton 2011]. Following their framework, the matrix factorization can be formulated as a joint regularized minimization problem,

$$\arg\min_{\mathbf{W},\mathbf{C}} \|\mathbf{X} - \mathbf{W}\cdot\mathbf{C}\|_{\mathrm{F}}^2 + \Omega(\,\mathbf{C}) \quad \text{s.t. } \mathcal{V}(\,\mathbf{W}) \tag{2}$$

where the constraint $\mathcal{V}$ can be one of the following

$$\max(|\,\mathbf{W}_{:,k}|) = 1, \qquad \forall k \tag{3}$$

$$\text{or} \quad \max(\,\mathbf{W}_{:,k}) = 1,\ \mathbf{W} \geq 0, \forall k \tag{4}$$

$\mathbf{W}_{:,k}$ denotes the k^{th} column, e.g. the weights of component k over all frames. The constraints on the weights $\mathbf{W}$ are essential to prevent the weights from getting too large (and components getting arbitrarily small). Depending on the use case (see Sect. 4), we use (4) which is inspired by blendshapes and works well for faces, or (3) that also allows negative weights which works better for data that can show two directions of deformation from the rest shape, like muscle bulges and dimples on body shapes.

To find an appropriate regularizer for animation data, let us also observe that each triplet in the rows of $\mathbf{C}$ forms a three-dimensional

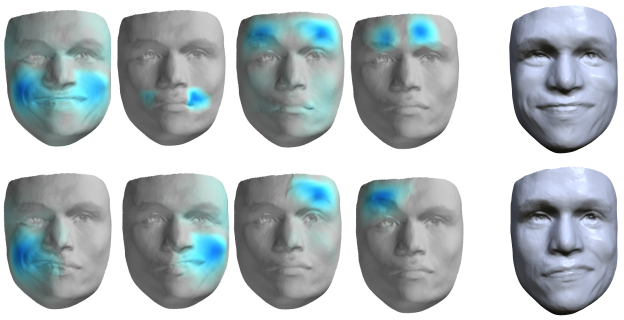


Figure 2: *Effect of automatic local support demonstrated on face dataset [Valgaerts et al. 2012]. Without local support(top), spatially distant regions are co-activated. By imposed local support (bottom), left and right side brows and cheek are separated, even though this separate deformation is never visible in the input. This allows generating novel facial expressions (bottom right).*

vector $\mathbf{c}_k^{(i)} = [x, y, z]_k^{(i)}$. Every such triplet corresponds to the x, y, and z displacement of vertex i in component k. While regularizing $\mathbf{C}$ with the element-wise ℓ_1 norm would induce sparsity, it would ignore this inherent group structure and would vanish each dimension of the displacement vector separately. To make the dimensions vanish simultanously, the ℓ_1 norm has to act on the (unsquared) *lengths* of the displacement vectors,

$$\Omega(\mathbf{C}) = \sum_{k=1}^{K} \sum_{i=1}^{N} \Lambda_{ki} \left\| \mathbf{c}_k^{(i)} \right\|_2 . \tag{5}$$

This norm, called the ℓ_1/ℓ_2 norm, is a form of group sparsity, see e.g. [Wright et al. 2009; Bach et al. 2012].

The spatially-varying regularization parameters Λ_{ki} (denoted in matrix form as $\mathbf{\Lambda}$) are an important innovation that we exploit to enforce local support for the deformation components. This extension is important for our mesh animation setting.

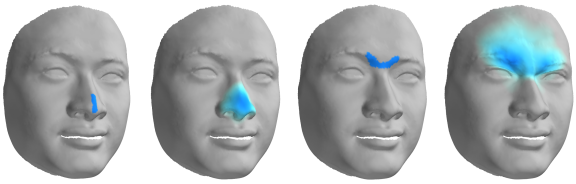
3.2 Local Support

To derive a deformation basis where the edits are spatially confined, we stipulate that each deformation component $\mathbf{C}_k$ should be centred around the set of vertices $\{j \in \mathcal{J}_k\}$ showing largest displacements in that component. We define a fuzzy support region that is centred around these vertices as follows. For every component k, let $\mathbf{d}_k \in \mathbb{R}^N$ be the geodesic distance of each vertex in the rest-pose mesh to the vertices in $\mathcal{J}_k$. To allow the control of the size of the support regions, we define a range $[d_{\text{min}}, d_{\text{max}}]$. We map geodesic distances from this range to $[0, 1]$ (with clipping when out of this range) and relate them to the regularization strength

$$\Lambda_{ki} = \lambda \cdot \begin{cases} 0 & \text{if } d_{ki} < d_{\text{min}} \\ 1 & \text{if } d_{ki} > d_{\text{max}} \\ \left(\frac{d_{ki} - d_{\text{min}}}{d_{\text{max}} - d_{\text{min}}} \right) & \text{otherwise} \end{cases} \tag{6}$$

This simply maps the geodesic distance linearly to the regularization strength. It thus changes the regularization strength of each component locally at each vertex.

This support region, and thus the regularization Eq. (5), are iteratively updated during the optimization of Eq. (2) by re-computing distances $\mathbf{d}_k$ at every step for each component. To do this quickly, we use the heat method presented by Crane et al. [2013], which re-computes geodesics on the mesh at extremely high speed.



user-stroke #1 component #1 user-stroke #2 component #2

Figure 3: *Incorporating user constraints: Given a rough scribble in form of a binary mask, a deformation component found in this region.*

In practice, we construct the set $\mathcal{J}_k$ of vertices in the center of each support region greedily, by picking vertices showing maximal activations for each component k, $\mathcal{J}_k = \{\arg\max_i \| \mathbf{c}_k^{(i)} \|_2\}$. But this set can also be obtained through user-input as we detail in the following. Fig. 2 demonstrates the effect of automatic local support.

3.3 User Constraints

In most cases, as we show in our experiments, the automatically found components already correspond to intuitively meaningful dimensions. But sometimes, artists wants to explore a different deformation space than the one found automatically. We therefore provide a simple user interface in which an artist can paint the center of a region for which a component shall be found, as shown in Fig. 3. This fits perfectly into the support region concept explained above, that we can conveniently set to the mask of vertices stroked by the artist. Due to diffusion by geodesic distances, this stroke mask does not need to be very precise.

3.4 Optimization

Due to the non-convexity of Eq. (2) and our added objective of Eq. (6), the problem is difficult to optimize directly. However, when fixing either $\mathbf{W}$ or $\mathbf{C}$ the problem is convex and can be solved easily, and therefore we use an iterative refinement method that alternates between the two optimization tasks [Mairal et al. 2009]. Usually, Sparse PCA is initialized by PCA, but this conflicts with our proposed extension for local support (in Sec. 3.2) and does not reach a good solution in practice. We therefore propose a new initialization strategy.

Initialization We use a deflation algorithm to greedily and iteratively find the component $\mathbf{C_k}$ (and corresponding weights $\mathbf{W}_{:,k}$) at each step that explain maximal variance in the data and subtract each of them from the animation matrix $\mathbf{X}$ to compute the residual $\mathbf{R}$. We assume that the next component can be detected by checking for the vertex $\mathbf{v}_j$ with the highest residual at the current step. From the chosen vertex, we build a support region around it using Eq. (6) and factorise the motion of vertices in this region into the product of a single component $\mathbf{C}_k$ and weight vector $\mathbf{W}_{:,k}$. We iterate until we find the required number of K components (that is set by the user). This solution is not optimal with respect to Eq. (2), but gives a good initialization that is already spatially localized and works very well in practice.

Optimization of weights Given the initial components $\mathbf{C}$, optimization of Eq. (2) with respect to the weights $\mathbf{W}$ is a constrained linear least squares problem. The problem is separable - the constraints act on the weight vector $\mathbf{W}_{:,k}$ of each component separately. We can thus use the block-coordinate descent algorithm, which in our case optimizes weights for each component successively, similar to [Mairal et al. 2009]. In practice, we re-use the

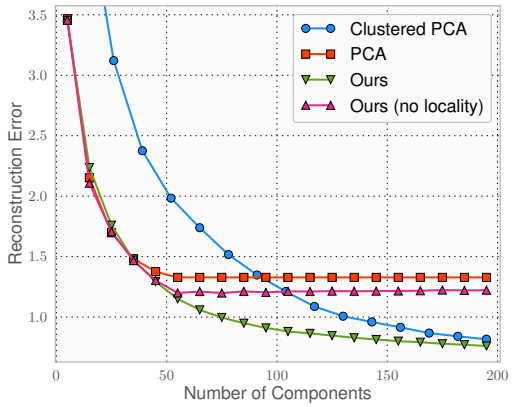


Figure 4: *Generalization to unseen data: we plot the reconstruction error (y-axis) with respect to the number of components used (x-axis) on a set of body scans from [Hasler et al. 2009] that were not part of the training data. Our method outperforms other related approaches in generalizing to unseen data from small training sets.*

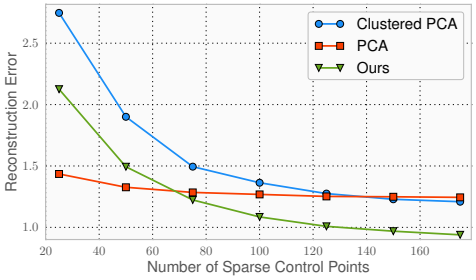


Figure 5: *Reconstruction Error with respect to sparse control points: we plot the reconstruction error (y-axis) of different approaches when used for regularizing a sparse number of control points (x-axis), in test poses which are not part of a limited training dataset [Zhang et al. 2004]. Our sparse localized deformation components require more control points than PCA, but show much lower reconstruction error from then on, as they are able to learn a wider variety of deformations from extremely limited training data.*

weights from the previous iteration ("warm restart") and for each component k solve

$$\arg\min_{\mathbf{W}_{:,k}\in\mathcal{V}} \|\mathbf{X} - \mathbf{W}\cdot\mathbf{C}\|_F^2 = \mathcal{P}_{\mathcal{V}}\left(\frac{(\mathbf{R}+\mathbf{W}_{:,k}\cdot\mathbf{C}_k)\cdot\mathbf{C}_k}{(\mathbf{C}_k)^\top\mathbf{C}_k}\right),$$

where $\mathbf{R} = \mathbf{X} - \mathbf{W}\,\mathbf{C}$ is the current residual. $\mathcal{P}_{\mathcal{V}}$ performs a projection of the weights onto the constraint set, depending on which constraint is used. For example, for Eq. (3), this comes down to a normalization, $\mathcal{P}_{\mathcal{V}}(\mathbf{w}) = \mathbf{w}/\max|\mathbf{w}|$. We perform a single iteration (for each k) of weight optimization before optimizing the sparse components in the next step.

Optimization of sparse components The optimization of $\mathbf{C}$ given fixed $\mathbf{W}$ can be tackled using convex optimization. Many algorithms exist that are able to optimize the involved ℓ_1/ℓ_2 norm regularizer. We chose to use the Alternating Direction Method of Multipliers (ADMM) [Boyd et al. 2011] as we observed quick convergence for our problems (also see the supplementary document). The basics of ADMM are extensively covered in [Boyd et al. 2011]. Here, we only give details on how to apply this algorithm in our setting: First, Eq. (2) needs to be modified by introducing a dual variable $\mathbf{Z} \in \mathbb{R}^{K\times 3N}$. With that, the optimization objective (with fixed $\mathbf{W}$) is rewritten in a form compatible to ADMM as follows

$$\begin{aligned}\arg\min_{\mathbf{C},\,\mathbf{Z}} \quad & \|\mathbf{X} - \mathbf{W}\cdot\mathbf{C}\|_F^2 + \Omega(\mathbf{Z}) \\ \text{s.t.} \quad & \mathbf{C} - \mathbf{Z} = \mathbf{0}\end{aligned}$$

The ADMM algorithm initializes $\mathbf{U} \in \mathbb{R}^{K\times 3N}$ to zero and then iterates the following steps

$$\begin{aligned}\mathbf{C} &\leftarrow \arg\min_{\mathbf{C}} \|\mathbf{X} - \mathbf{W}\cdot\mathbf{C}\|_F^2 + \frac{\rho}{2}\|\mathbf{C} - \mathbf{Z} + \mathbf{U}\|_F^2 \\ \mathbf{Z} &\leftarrow \arg\min_{\mathbf{Z}} \left(\Omega(\mathbf{Z}) + \frac{\rho}{2}\|\mathbf{C} - \mathbf{Z} + \mathbf{U}\|_F^2\right) \\ \mathbf{U} &\leftarrow \mathbf{U} + \mathbf{C} - \mathbf{Z}\,.\end{aligned}$$

The first step, updating $\mathbf{C}$, is a linear least squares problem. The second step, updating the dual variable $\mathbf{Z}$, corresponds to the proximal operator [Bach et al. 2012] of the ℓ_1/ℓ_2-norm Ω. A closed form solution is (see Ch. 3.3 in [Bach et al. 2012]),

$$\mathbf{z}_{ik} \leftarrow \mathbf{prox}_{\Omega}(\mathbf{p}_{ik}) = \left(0,\ 1 - \frac{\Lambda_{ki}}{\rho\|\mathbf{p}_{ik}\|_2}\right)_+ \mathbf{p}_{ik}$$

where $\mathbf{p}_{ik} = \mathbf{c}_{ik} + \mathbf{u}_{ik}$ and the notation $(\cdot)_+ = \max(\cdot\,, 0)$ are introduced just for brevity. We found that fixing the penalty parameter $\rho = 10$ works well in our situation for all the datasets considered; strategies to adjust it are discussed in [Boyd et al. 2011]. We run 10 iterations of ADMM to optimize the components and keep $\mathbf{U}$ across the whole procedure. In summary, the whole optimization procedure interleaves the following steps: recomputation of support maps based on geodesic distances; optimization of weights using block-coordinate-descent; optimization of components using ADMM.

Convergence Because of the changing regularization by updating the support maps (cf. Sect. 3.2), we cannot guarantee that the objective function is decreasing at each step. We monitor convergence by recording the change of objective function Eq. (2) at each step, and terminate if the average change of the last 5 iterations is below a threshold ϵ.

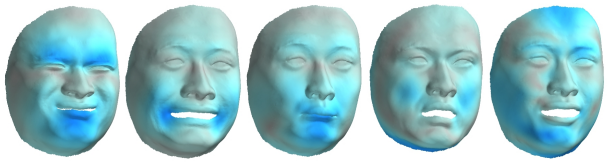
Tunable Parameters In summary, our method takes the following user-specified parameters: the required number K of deformation components, the minimal and maximal geodesic distance for the support maps d_{min} and d_{max} as well as λ in Eq. (6) that sets the importance of local support term. Users of our algorithm also have to decide whether the weights are allowed to be negative (Eq. (3)) or not (Eq. (4)) and if they want to use a specific frame (e.g. the first one) or the average of all frames as the rest shape.

4 Results

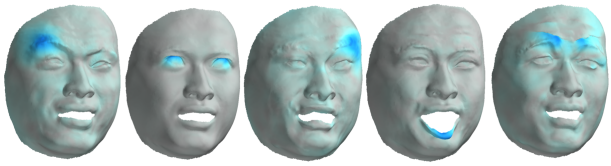
To evaluate our method, we perform quantitative evaluation to asses the generalizability and reconstruction quality on unseen data. We demonstrate the versatility of our method on a variety of captured animation data sets. As potential applications we show intuitive editing and shape analysis on faces (Sect. 4.2), full-body models (Sect. 4.5), muscle motion (Sect. 4.3) and cloth motion (Sect. 4.4). The processed datasets and the method parameters are listed in Table 1, where we also record computation time and reconstruction accuracy. Please see also accompanying video.

4.1 Quantitative Evaluation

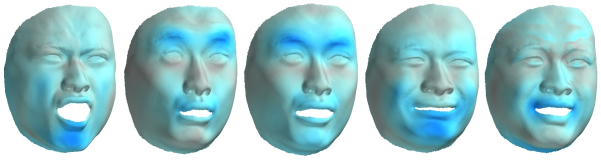
Since our method can be viewed as an unsupervised dimensionality reduction technique, it is interesting to assess the generalizeability of the extracted dimensions to unseen data. For this, we randomly split 111 scans of people roughly standing in the same pose that we



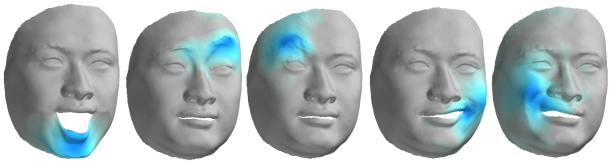
(a) Principal Component Analysis (PCA), Top 5 PC's



(b) PCA components rotated using Varimax

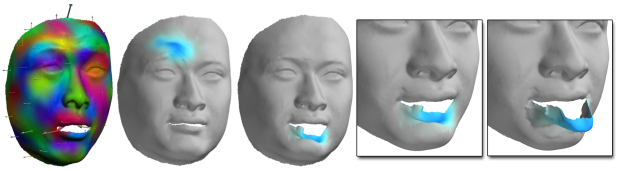


(c) Independent Component Analysis (ICA) [Hyvärinen et al. 2001]

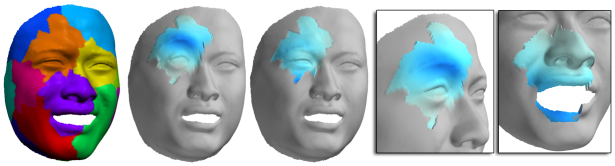


(d) Proposed Sparse Localized Deformation Components

Figure 6: *Comparison of general decomposition methods on a captured face dataset [Zhang et al. 2004]. Color coding shows magnitude of vertex displacements inside the components, from grey (zero) to blue (maximum). The deformation components of PCA and ICA act globally on each vertex in the mesh, which prohibits local modification. Varimax shows certain locality, but this cannot be controlled and shows artifacts for captured data. Our components show sparse and local deformations on confined regions, which is important for example for artistic editing.*



(a) Skinning Decomposition [Kavan et al. 2010], from left to right: Vertex influences; Influence and motion of 2 bones in isolation; silencing the motions of all but one bone gives artifacts; bones offer too many degrees of freedom, e.g. arbitrary translation of the lip which results in artifacts.



(b) Clustered PCA according to [Tena et al. 2011], from left to right: Underlying segmentation; first and second principal component in region 1; close up views clearly show artifacts that have to be removed with an explicit blending step, which our method does not require.

Figure 7: *Comparison of related decomposition methods.*

obtained from [Hasler et al. 2009] into training (55 scans) and test set (56 scans). We train Clustered PCA [Tena et al. 2011], PCA, and our method (without and with automatic locality) with varying number of components K. For Clustered PCA, we fixed the number of regions to 13, and set K so that it counts all the components and not the number of components per region as in [Tena et al. 2011]. We also run our algorithm without locality in Sec. 3.2 (constant $\Lambda_{ki} = 0.5$), which is closer to classical Sparse PCA. To reconstruct the training set, we invert Eq. (1) and measure the difference in vertex coordinates with the error metric introduced in [Kavan et al. 2010] (root mean squared error multiplied by 1000 for convenience). The averaged reconstruction errors over 3 different random splits are reported in Fig. 4. The results indicate that our method with localization generalizes better to new body shapes outside the training set than PCA and plain sparse PCA (our method without localization). To assess the the suitability of sparse localized deformation components to providing editing beyond limited capture data, we performed an experiment on the facial performance dataset provided by [Zhang et al. 2004]. We train PCA, Clustered PCA and our method on only 19 randomly selected frames from this 365 frame dataset. For each of the remaining frames, we randomly selected sparse control points on the mesh and reconstructed the weights for the components to fit only to the control points. We compare the resulting complete mesh to the ground truth and again use the error metric from [Kavan et al. 2010]. The experiment was repeated in 5 trials differing in training set, each trial was further repeated 10 times using different random control points. The average reconstruction error in relation to number of control points is shown in Fig. 5 Due to the locality of our components, our method requires more control points until it outperforms PCA, here with 50 components this happens at around 70 control points. However, from this point on, it can generate more accurate deformations compared to PCA. It also consistently outperforms Clustered PCA.

To understand why our method generalizes so well to unseen data, consider a facial geometry sequence in which both eyebrows are always raised together. PCA cannot reproduce a mesh showing only one eyebrow raising. In contrast, our approach finds localized components for every eyebrow, and can reproduce their motion separately, even though this was not seen in the training data.

4.2 Facial performances

Our method can be used to learn deformation components of captured facial animations that mimic the control properties of blendshapes that are widely used by animation artists to create facial animations, and which are normally created in a tedious manual process. Our automatically detected components are spatially localized and thus limit the influence of modifications to confined regions that correspond to individual intuitively meaningful effects (e.g., twitching of an eyebrow or a nostrill).

Comparison to Important Related Work Such spatially confined and intuitive edits are hard to perform with components from global decomposition methods such as PCA or ICA. Fig. 6 illustrates this on facial performance data [Zhang et al. 2004]. Even without facial prior or user input, our deformation components are spatially confined to local regions and already resemble artistically modeled blendshapes for controlling a facial animation [Osipa 2003].

Fig. 6(b) visually compares to PCA components after varimax rotation, a technique used by [Meyer and Anderson 2007] for extraction of key points. We observed that for our data of limited length, the Varimax components show severe artifacts which makes them unusable for direct editing. Notice that Varimax and PCA components only differ in a rotation, so Varimax will perform exactly the same

Figure 8: *Results of direct manipulation of the underlying basis. From left: user-constraints, PCA basis, Varimax Basis, Our basis. PCA effects unintuitive global deformation even if the edits are local. Varimax basis shows artifacts. Ours provides artistically meaningful local edits.*

as PCA with respect to measured reconstruction error and generalization.

In Clustered PCA, where we use spectral clustering as in [Tena et al. 2011] to provide the region segmentation, components are confined to regions but within these regions, they exhibit the same limitations as PCA components: they don't necessary show confined and independent effects - all PCA components have to be activated to achieve a specific deformation effect. Additionally, blending at region boundaries is needed to remove artifacts [Tena et al. 2011] (see also Fig. 7(b)).

We also compare to skinning decomposition methods that learn explicit bone transformations, especially to the method of [Kavan et al. 2010]. We observed two key problems which interfere with the target applications that our method is designed for. Firstly, the bone transformation only work when modifying several bones in concert, e.g. they don't allow individual control. For example, eyebrow-wrinkles are produced by setting transformations of at least 2 specific bones in a specific, hardly intuitive way. Secondly, bone transformations themselves do not provide intuitive control parameters at all. For example, editing the translation of a bone quickly results in deformation artifacts because parts of the mesh can easily be "pulled" outside of the mesh (see Fig. 7(a)). Notice that this is inherent to the underlying skinning representation to achieve translation and rotation invariance.

Automatic Local Support In Fig. 2, we show the benefit of the spatial locality term (Eq. (6)) by decomposing the dataset provided by [Valgaerts et al. 2012]. This example illustrates how imposing local support regions helps our method to separate motions of distant regions which are co-activated in the original actor's performance. One could say it helps us to model regions influenced by individual muscle-groups separately, such as the left eyebrow motion which is distinct from the right eyebrow motion.

Artistic Control In Fig. 9, we show the results of artistic edits using our sparse components on another face animation captured by [Beeler et al. 2011] that we downsampled to 40k vertices. For the editing, we can resort to the direct manipulation method of [Lewis and Anjyo 2010], where positional edits on mesh vertices are used with our sparse components for modifying the shape within the linear deformation space. Based on our sparse components, captured facial animations can be conveniently edited beyond the captured motion using only a few input constraints for crisp direct manipulation (see video). Also co-articulation effects that are never observed in the input, such as separate eye-brow movement, can be plausibly reproduced. We also tested this editing paradigm on the decomposed dataset of [Zhang et al. 2004] and just a single control point, see Fig. 8 We used the same set of user edits on PCA, Varimax, and our components. It can be seen (also refer to video) that our sparse components allow for localized edits constrained in a plausible space of face motion, a characteristic that artists often require [Havaldar 2006]. In contrast, in case of PCA, single vertex edits have unintuitive global effects on the entire face.

Figure 9: *Our method allows completely new exaggerated facial expressions far beyond the input range, here based on the dataset of [Beeler et al. 2011].*

4.3 Muscle deformations

To show its use in biomechanics, we applied our algorithm to captured moving arm geometry exhibiting muscle-induced surface deformation, that is provided by [Neumann et al. 2013]. This dataset shows deformation of the arm and shoulder of a curl motion done once without weight and twice with 14kg in the hand (we only selected a subset of the full captured dataset). These meshes are already temporally aligned and registered to a simple 3-piece bone skeleton. Prior to sparse component extraction, we subtract the skeleton pose induced deformation by aligning all meshes to the template skeleton pose. The extracted deformation components from our method (Fig. 1, left; supplementary video), without any user-constraints, already visually appear to correspond to local bulges due to individual muscle groups, such as the deltoid muscle, the biceps and triceps muscles.

In Fig. 10, we show the activations of some of these components in the input sequence. Fig. 10(d) shows that one component explains how the tendon at the elbow joint twitches during this sequence. Fig. 10(f) shows the activation of the triceps muscle during the same motion. The triceps muscle is summoned to counteract the external force on the hand, whereas the tendon at the elbow twitches due to change in the pose of the arm. In other words our method is able to provide such biomechanically informative muscle activation graphs, without requiring the user to explicitly fit a physiological muscle template to the data. Surface noise in the data due to limitations of the acquisition method can also be isolated by our method, which the user can easily remove by deleting the corresponding components and projecting the data into the remaining subspace. Animators can also selectively emphasise the effect of individual muscle groups in an animation, i.e, create a biologically inaccurate but artistically desirable appearance of an animation which is often needed in VFX productions (see video).

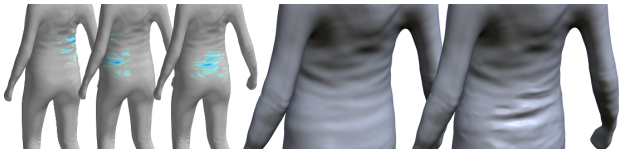


Figure 11: *Left: 3 sparse localized deformation components on a performance capture sequence showing cloth folds, Right: effect of exaggerating one component.*

Dataset			Algorithm Parameters						Results		
Source	N	F	K	$\hat{\mathbf{x}}$	$\mathcal{V}$	λ	$d_{\min}$	$d_{\max}$	iters	time	E_{RMS}
Face [Beeler et al. 2011]	40 000	322	80	first	Eq. (4)	2	0.1	0.6	49	6m57s	0.39
Face [Zhang et al. 2004]	23 725	384	50	first	Eq. (4)	1	0.1	0.7	82	3m58s	0.76
Body-scans [Hasler et al. 2009]	6 449	111	100	avg.	Eq. (3)	1	0.1	1.0	34	33s	0.76
Cloth folds [Wu et al. 2011]	10 684	459	200	avg.	Eq. (3)	25	0.01	0.3	18	4m39s	0.29
Face [Valgaerts et al. 2012]	44 153	566	50	first	Eq. (4)	2	0.1	0.7	113	14m09s	0.67
Muscles [Neumann et al. 2013]	3 467	242	80	avg.	Eq. (3)	5	0.1	0.4	77	1m36s	1.08

Table 1: *Overview of processed datasets. From left to right, columns show data set source, number of vertices N, number of frames F. The columns grouped under "Algorithm Parameters" show: number of components K, which frame to use as rest shape (first or average), type of constraint for weights (allow non-negativity or not), λ to control sparsity regularization, d_{min} and d_{max} to control size of support regions. Numbers below "Results" are number of iterations for global optimization until convergence, computation time in minutes (i7 3.40GHz, 16GB RAM, Python implementation), and reconstruction error E_{RMS} using the measure from [Kavan et al. 2010].*

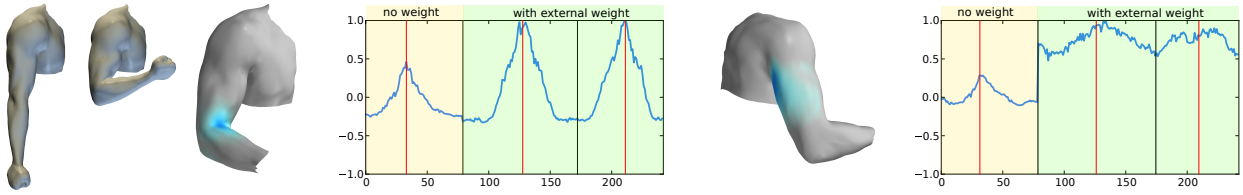


Figure 10: *Automatically found activations of components corresponding to specific muscles. The subject flexes the elbow pose (a) to pose (b). Our method happens to identify bio-mechanically meaningful deformation components: (c) Twitching of the tendon at the elbow joint - support area. (d) activation graph . (e) Triceps muscle - support area. (f) activation graph. In the graphs, the black columns correspond to pose-a and red columns to pose-b. The subject performs one motion cycle without any weight in the hand, and then two cycles holding an external load in the hand.*

4.4 Cloth deformations

Our method also enables visualization of cloth deformation patterns in captured mesh animations, such as the detailed full-body motion data including cloth wrinkles captured by [Wu et al. 2011] from multi-view video. Our objective is to derive a set of features that describe high-frequency detail of finer-scale folds. So we separate out the lower-frequency motion coming from the limbs and register all the meshes onto a single template pose. The moving cloth folds appear as floating residuals on this template mesh (see video). Our sparse decomposition identifies localized fold patterns on the cloth surface that can be visualized and controlled individually, Fig. 11. This hints at potential applications of sparse localized deformation components in data-driven cloth simulation and upsampling.

4.5 Statistical shape modeling, processing and visualisation

In statistical shape processing, researchers are often faced with the problem of aligning large data sets of shape models or scans that show variants of a class of shapes. These shapes may exhibit certain local shape variations, but also differ in proportions or pose. Certain statistical properties or correlations of the shapes shall be learned, while certain other effects shall be factored out. Our decomposition method can be of great help in such a setting. We exemplify it on the problem of building a static statistical shape model of human shape variations from a large corpus of laser scans of real human subjects, which are previously done with PCA and bilinear models [Anguelov et al. 2005; Hasler et al. 2009]. For instance, Hasler et al. [2009] provide a data set of 111 humans of different shape and gender that were asked to stand in the same pose during a scan. From this set one may want to capture shape variations and neglect variations due to pose. Hasler and other researchers resorted to deformation-based scan alignment relative to a template to get dense correspondences of the surfaces. However, pose variations due to different joint angles/hand poses still exist after that correspondence estimation as people can never stand in the exact same pose. Applying PCA to that set thus leads to components which intertwine pose and shape variations, i.e. dimensions that one wants to be separated. Experimentally, we showed that by applying our method to the above scan data, we are able to separate effects of pose and shape of specific body parts (Fig 12). Deformation related to pose and shape variations are gathered into different components. We can meaningfully pose normalise all the scans by manually removing the pose-specific deformation components and projecting the scans into the remaining subspace. This would be difficult with other means. At the same time we can automatically extract intuitive localized dimensions of shape variation, even without having to pose normalise the scans *before* the decomposition. These dimensions correspond to more meaningful dimensions than found in previous approaches, for instance, the waist girth, belly size, or the hip size. These can can now be individually explored and changed irrespective of global body pose. We foresee that this general idea could be used on scan sets with much stronger pose variation, enable more robust shape-retrieval under pose variation, and be used in the context of co-segmentation learning from shape sets.

4.6 Discussion

Sparse Localized Deformation components present a versatile decomposition method for space-time mesh animation data that is applicable to many settings: editing, control, scan alignment, construction of static and parametric shape models etc. The major advantage of our approach is that it is very general yet simple to implement. Quantitative experiments show that our method provides excellent generalizeability compared to other methods like

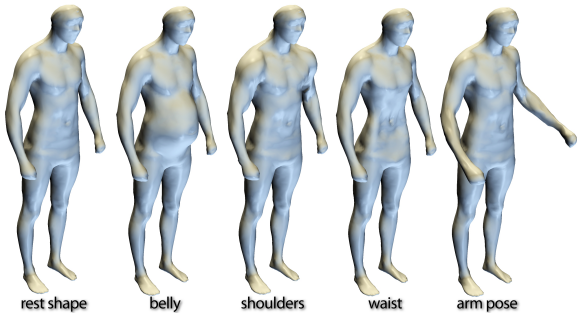


Figure 12: *Automatically extracted deformation components on a database of human body scans from [Hasler et al. 2009]. Exaggerating certain components produces spatially localized and semantically meaningful effects. Our method even isolates pose variations into separate components (bottom row, right).*

PCA, Clustered PCA, Varimax, and non-localized Sparse PCA. Our current model learns deformation components that show complex effects due to vertex displacements from a rest shape; articulated motions showing rotations are better decomposed using previous methods [Kavan et al. 2010; de Aguiar et al. 2008]. As it is, our method can explain local high-detail deformations such as muscle bulges, cloth folds and slight pose changes after using such skeletal decomposition methods for rough pose alignment. In the future, we plan to investigate the use of rotation invariant deformation encodings or hierarchical schemes in Eq. (2).

Similar to other data-driven deformation methods, ours can only learn deformation effects visible in the input animation, but separate them into meaningful individual dimensions. The option for user-constraints allows for the introduction of semantic information into the very process of the decomposition. Our algorithm automatically extracts components which are symmetric (e.g. left and right side of the face), but only if this symmetry is present in the input data. Asymmetry is a very important property of individual faces, but in the future, it would be interesting to extend our data-term to incorporate user-given knowledge about symmetry.

Controlling and exploring the latent deformation space, as offered by our method, has wider implications in the domain of statistical shape processing. This is a direction that we would like to explore in the future. Another promising research direction is to extend this theory of sparse deformation components to account for dynamics or physical material properties, which can potentially improve the realism for interactive applications such as computer games. We publish our source-code along with this paper to stimulate future research in these areas.

5 Conclusion

In this paper, we develop a sparse matrix decompositions method for processing 3D mesh animations. We learn deformation components from the input data that are spatially localized and identify local features of shape deformation. Unlike previous data-driven approaches, our method allows controlling the dimensions of the underlying space of deformations. In the results and in the accompanying video, we demonstrate how our method provides novel capabilities for intuitive and localized artistic edits on captured performances and on shape models. Our method automatically builds a rich space of deformations that generalizes better to unseen data. We have shown how this delivers a powerful tool for artistic editing, as well as for data exploration and statistical shape processing, and are eager to see interesting applications of our method in the future.

References

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. SCAPE: shape completion and animation of people. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3.

BACH, F. R., JENATTON, R., MAIRAL, J., AND OBOZINSKI, G. 2012. Optimization with sparsity-inducing penalties. *Found. Trends Mach. Learn. 4*, 1.

BEELER, T., HAHN, F., BRADLEY, D., BICKEL, B., BEARDSLEY, P., GOTSMAN, C., SUMNER, R. W., AND GROSS, M. 2011. High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*.

BOYD, S., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn. 3*, 1.

CAO, Y., SHAPIRO, A., FALOUTSOS, P., AND PIGHIN, F. 2007. Motion editing with independent component analysis. *Visual Computer*.

CASHMAN, T. J., AND HORMANN, K. 2012. A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape. *Comp. Graph. Forum (Proc. EG) 31*, 2.

CRANE, K., WEISCHEDEL, C., AND WARDETZKY, M. 2013. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graph.*, to appear.

DE AGUIAR, E., THEOBALT, C., THRUN, S., AND SEIDEL, H.-P. 2008. Automatic conversion of mesh animations into skeleton-based animations. *Comp. Graph. Forum (Proc. EG) 27*, 2.

DE AGUIAR, E., SIGAL, L., TREUILLE, A., AND HODGINS, J. K. 2009. Stable spaces for real-time clothing. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 3.

DENG, B., BOUAZIZ, S., DEUSS, M., ZHANG, J., SCHWARTZBURG, Y., AND PAULY, M. 2013. Exploring Local Modifications for Constrained Meshes. *Comp. Graph. Forum (Proc. EG) 32*, 2.

FENG, W.-W., KIM, B.-U., AND YU, Y. 2008. Real-time data driven deformation using kernel canonical correlation analysis. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3.

FRÖHLICH, S., AND BOTSCH, M. 2011. Example-driven deformations based on discrete shells. *Comp. Graph. Forum 30*, 8.

GUAN, P., REISS, L., HIRSHBERG, D., WEISS, A., AND BLACK, M. J. 2012. DRAPE: DRessing Any PErson. *ACM Trans. on Graphics (Proc. SIGGRAPH) 31*, 4.

HASLER, N., STOLL, C., SUNKEL, M., ROSENHAHN, B., AND SEIDEL, H.-P. 2009. A statistical model of human pose and body shape. *Comp. Graph. Forum (Proc. EG) 2*, 28.

HASLER, N., THORMÄHLEN, T., ROSENHAHN, B., AND SEIDEL, H.-P. 2010. Learning skeletons for shape and pose. In *Proc. of I3D*.

HAVALDAR, P. 2006. Performance driven facial animation. In *ACM SIGGRAPH 2006 Course 30 Notes*.

HYVÄRINEN, A., KARHUNEN, J., AND OJA, E. 2001. *Independent component analysis*. John Wiley & Sons.

JENATTON, R. 2011. *Structured Sparsity-Inducing Norms: Statistical and Algorithmic Properties with Applications to Neuroimaging*. PhD thesis, École Normale Supérieure Cachan.

JOLLIFFE, I. T., TRENDAFILOV, N. T., AND UDDIN, M. 2003. A modified principal component technique based on the LASSO. *J. Comp. Graph. Stat. 12*, 3.

KAVAN, L., SLOAN, P.-P., AND O'SULLIVAN, C. 2010. Fast and efficient skinning of animated meshes. *Comp. Graph. Forum (Proc. EG) 29*, 2.

KAVAN, L., GERSZEWSKI, D., BARGTEIL, A., AND SLOAN, P.-P. 2011. Physics-inspired upsampling for cloth simulation in games. *ACM Trans. Graph. 30*, 4.

KIM, D., KOH, W., NARAIN, R., FATAHALIAN, K., TREUILLE, A., AND O'BRIEN, J. F. 2013. Near-exhaustive precomputation of secondary cloth effects. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 4.

KIRCHER, S., AND GARLAND, M. 2009. Free-form motion processing. *ACM Trans. Graph. 27*, 2.

KRY, P. G., JAMES, D. L., AND PAI, D. K. 2000. EigenSkin: Real time large deformation character skinning in hardware. In *Proc. SCA*.

LAU, M., CHAI, J., XU, Y.-Q., AND SHUM, H.-Y. 2009. Face poser: Interactive modeling of 3D facial expressions using facial priors. *ACM Trans. Graph. 29*, 1.

LE, B., AND DENG, Z. 2012. Smooth skinning decomposition with rigid bones. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6.

LEE, D. D., AND SEUNG, H. S. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature 401*.

LEVINE, S., WANG, J. M., HARAUX, A., POPOVIĆ, Z., AND KOLTUN, V. 2012. Continuous character control with low-dimensional embeddings. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4.

LEWIS, J. P., AND ANJYO, K. 2010. Direct-manipulation blendshapes. *IEEE CGAA 30*, 4.

LI, H., WEISE, T., AND PAULY, M. 2010. Example-based facial rigging. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4.

MACKEY, L. 2009. Deflation methods for sparse pca. In *Adv. NIPS*.

MAIRAL, J., BACH, F., PONCE, J., AND SAPIRO, G. 2009. Online dictionary learning for sparse coding. In *Proc. ICML*.

MEYER, M., AND ANDERSON, J. 2007. Key point subspace acceleration and soft caching. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3.

MIGUEL, E., BRADLEY, D., THOMASZEWSKI, B., BICKEL, B., MATUSIK, W., OTADUY, M. A., AND MARSCHNER, S. 2012. Data-driven estimation of cloth simulation models. *Comp. Graph. Forum (Proc. EG) 31*, 2.

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *ACM Trans. Graph. (Proc. SIGGRAPH)*.

NEUMANN, T., VARANASI, K., HASLER, N., WACKER, M., MAGNOR, M., AND THEOBALT, C. 2013. Capture and Statistical Modeling of Arm-Muscle Deformations. *Comp. Graph. Forum (Proc. EG) 32*, 2.

OLSHAUSEN, B., AND FIELD, D. J. 1997. Sparse coding with an overcomplete basis set: a strategy employed by v1? *Vision Research 37*, 23.

OSIPA, J. 2003. *Stop Staring: Facial modeling and animation done right*, second ed. Sybex.

POKRASS, J., BRONSTEIN, A. M., BRONSTEIN, M. M., SPRECHMANN, P., AND SAPIRO, G. 2013. Sparse Modeling of Intrinsic Correspondences. *Comp. Graph. Forum (Proc. EG) 32*, 2.

SCHUMACHER, C., THOMASZEWSKI, B., COROS, S., MARTIN, S., SUMNER, R., AND GROSS, M. 2012. Efficient simulation of example-based materials. In *Proc. SCA*.

SEO, J., IRVING, G., LEWIS, J. P., AND NOH, J. 2011. Compression and direct manipulation of complex blendshape models. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 30*, 6.

SIFAKIS, E., NEVEROV, I., AND FEDKIW, R. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3.

SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3.

TENA, J. R., DE LA TORRE, F., AND MATTHEWS, I. 2011. Interactive region-based linear 3D face models. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4.

THEOBALT, C., DE AGUIAR, E., STOLL, C., SEIDEL, H.-P., AND THRUN, S. 2010. *Image and geometry processing for 3D Cinematography*. Springer, ch. Performance capture from multi-view video.

TOURNIER, M., AND REVERET, L. 2012. Principal Geodesic Dynamics. In *Proc. SCA*.

VALGAERTS, L., WU, C., BRUHN, A., SEIDEL, H.-P., AND THEOBALT, C. 2012. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 31*, 6.

VLASIC, D., BRAND, M., PFISTER, H., AND POPOVIĆ, J. 2005. Face transfer with multilinear models. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3.

WEBER, O., SORKINE, O., LIPMAN, Y., AND GOTSMAN, C. 2007. Context-aware skeletal shape deformation. *Comp. Graph. Forum (Proc. EG) 26*, 3.

WRIGHT, S., NOWAK, R., AND FIGUEIREDO, M. 2009. Sparse reconstruction by separable approximation. *Signal Processing, IEEE Transactions on 57*, 7.

WU, C., VARANASI, K., LIU, Y., SEIDEL, H.-P., AND THEOBALT, C. 2011. Shading-based dynamic shape refinement from multi-view video under general illumination. In *Proc. ICCV*.

ZHANG, L., SNAVELY, N., CURLESS, B., AND SEITZ, S. 2004. Spacetime faces: High-resolution capture for modeling and animation. *ACM Trans. Graph. (Proc. SIGGRAPH)*.

ZOU, H., HASTIE, T., AND TIBSHIRANI, R. 2006. Sparse principal component analysis. *J. Comp. Graph. Stat. 15*, 2.