DOI: 10.1111/cgf.12492
Pacific Graphics 2014
J. Keyser, Y. J. Kim, and P. Wonka
(Guest Editors)

Volume 33 (2014), Number 7

Sparse Localized Decomposition of Deformation Gradients

Zhichao Huang[1], Junfeng Yao[1], Zichun Zhong[2], Yang Liu[3], Xiaohu Guo[1,2†]

[1]Xiamen University
[2]University of Texas at Dallas
[3]NVIDIA

Abstract

Sparse localized decomposition is a useful technique to extract meaningful deformation components out of a training set of mesh data. However, existing methods cannot capture large rotational motion in the given mesh dataset. In this paper we present a new decomposition technique based on deformation gradients. Given a mesh dataset, the deformation gradient field is extracted, and decomposed into two groups: rotation field and stretching field, through polar decomposition. These two groups of deformation information are further processed through the sparse localized decomposition into the desired components. These sparse localized components can be linearly combined to form a meaningful deformation gradient field, and can be used to reconstruct the mesh through a least squares optimization step. Our experiments show that the proposed method addresses the rotation problem associated with traditional deformation decomposition techniques, making it suitable to handle not only stretched deformations, but also articulated motions that involve large rotations.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

With the advancement of 3D scanning, performance capture, and geometric registration techniques, there has been a need for computing low-dimensional control parameterization of the captured animation sequences, so that artists can conveniently reuse, edit, and create new shapes or animations.

There are mainly two categories of methods to tackle this problem. The first category is to fit a specific parameterized model to the data, such as skeletal bone structure with linear regression, or pre-determined blend-shapes for facial modeling, etc. In this category the parametric model is typically pre-determined and it is hard to be automatically generalized to arbitrary input animation. The second category relies on dimensional reduction inspired from matrix factorization ideas, such as Principal Component Analysis (PCA). Note that the deformation components computed from traditional dimensional reduction methods like PCA capture the prominent deformation "trends" in the animation sequence, but are typically of global support and lack interpretable meaning, which makes it difficult for artist to control the shape and create new animations. Neumann et al. [NVW*13] proposed a method to compute *sparse localized deformation components* (SPLOCS). Their computed components are of local support and sparse (with zero values outside the local support regions), thus capture the prominent "semantic" deformation, and make it easier for the artists to intuitively recreate new shapes by manipulating these local handles.

Figure 1: *An animation of Humanoid walking up the stairs.*

There is one fundamental difficulty associated with the above mentioned dimensional reduction techniques, including Neumann et al.'s SPLOCS method [NVW*13]: they cannot handle large rotations in their components. This is due to the fact that they are representing the displacement fields as a linear combination of components, where each component

† Corresponding author

is in essense a pre-determined displacement field. However, it is well known that only deformations with small rotations can be approximated using linear combination of components, while large rotations cannot (see Appendix A). Figure 1 shows an example animation of Humanoid walking up the stairs. In this sequence, there are not only translations of its body, but also rotations of its limbs. Section 4.4 shows the problematic effects of the components generated by SPLOCS.

In this paper, we present a new method to compute the sparse localized decomposition of deformation gradients, given some example animation meshes. We decompose the deformation gradients into two groups: rotation and stretching, via polar decomposition. The rotation matrices are mapped to its logarithm space and represented in a vector-form. Together with the vector-form of the stretching matrices, we can perform SPLOCS on these vector fields. The linear combination of these decomposed components can be used to reconstruct a deformation gradient field, which can be further converted to a 3D animated mesh. These automatically-computed components represent the spatially-localized, semantically-meaningful, and user-friendly control handles for artists to easily create new shapes and animations from the "styles" given in the example meshes. It overcomes the difficulty of rotational artifacts associated with traditional dimension reduction approaches including the original SPLOCS, while preserving most of its advantages, such as superb localized control and biomechanically meaningful decomposition (for some captured real human datasets).

2. Related Work

2.1. Skeletal and Facial Animation Fitting

Given some example animation meshes, researchers have been looking into the problem of fitting some hierarchical skeletal models [MG03], based on linear blend skinning or its extensions [LCF00] [KJP02]. Weber et al. [WSLG07] presented a skeletal animation system that can both preserve the local details of the original shape (e.g. wrinkles) and also capture the characteristic shape (e.g. muscle bulge) given some examples of animated meshes. The arbitrarily given animated meshes can be also automatically converted to a skeletal model, with corresponding motion parameters and skinning weights, as shown in de Aguiar et al.'s method [dATTS08]. It was later extended to capture body shape variations [HTRS10]. In addition, methods based on other examples [KM04] [HZY*11] were proposed by researchers as well.

Another type of methods is to fit unorganized (non-hierarchical and un-ordered) collections of bone transformations and their weights to the given animation mesh sequence [JT05]. The bone transformation can be either rigid [LD12] or flexible [KSO10]. The difference between this type of methods with the sparse localized decomposition [NVW*13] is that their sparsity is pre-fixed, e.g. to be 4 bones per vertex, instead of determined from the given dataset.

For facial animation, researchers have proposed methods to interactively posing 3D facial expressions, based on priors from a large set of expression data [LCXS10]. Li et al. [LWP10] proposed an automatic approach to customize the blendshape model for fitting the given example poses of character dataset. Tena et al. [TlTM11] presented a linear facial modeling approach using region-based collections of PCA sub-models, which are independently trained in each region but share boundaries. A hybrid method based on WPSD [RLN06] was proposed by Bickel et al. [BLB*08] for real-time animation of highly-detailed facial expressions. It allowed both large-scale deformations and fine-scale wrinkles to be edited intuitively.

2.2. Dimension Reduction and Sparse Decomposition

Dimension reduction, such as Principal Component Analysis (PCA) or Independent Component Analysis (ICA) [HKO01], has been applied to the field of Computer Graphics and Animation, for learning a low-dimensional sapce from example poses, shapes, or motions. It allows the artists to easily control the model through the reduced space, and has been used for synthesizing new motions out of examples [LWH*12], and physically-based animation of virtual characters [TR12]. For shape deformations, appropriate deformation subspaces can be learned from a set of example shapes, and new deformations can be generated on-the-fly by manipulating a few control points. Feng et al. [FKY08] proposed a kernel Canonical Correlation Analysis (CCA) with Poisson-based technique for such deformation control. For animating the geometry of human body, Anguelov et al. [ASK*05] proposed to learn a pose deformation model and a body shape variation model, separately, through linear regression from the training set. Hasler et al. [HSS*10] combined the descriptions of human body pose and shape into a unified statistical model.

The dimensional reduction techniques mentioned above typically produce the deformation components that are of global support, i.e. each component will have non-zero values for all vertices (or triangles). This makes it difficult for artists to intuitively design a new shape or pose since each component is not spatially localized. Some alternatives, such as Sparce PCA [ZHT06, Mac09] introduces L_1 norm into the formulation as a sparsity regularization, but are mainly used for medical imaging applications. Neumann et al. [NVW*13] proposed a sparse localized decomposition of deformation components, which shows better performance than ICA in terms of localized control. However, a limitation is that they cannot handle large rotations in the animation sequences. In this paper we propose an extension of it based on deformation gradients to address this problem.

It is only in recent years that sparse modeling was introduced into the field of computer graphics. Pokrass et al. [PBB*13] used sparse coding to compute the intrinsic non-rigid correspondence between two shapes. Wang et al. [WYL*14] presented an L_1-analysis compressed sensing approach for denoising meshes while recovering their sharp features, based on the observation that sharp features are typically sparse in geometry.

2.3. Deformation Gradients for Mesh Animation

Deformation gradient is a fundamental concept in continuum mechanics for studying the deformation of solids. Sumner and Popović [SP04] extended it to triangle meshes and applied it for handling deformation transfer from one animation sequence to a target shape. Later on Sumner et al. [SZGP05] applied deformation gradients to compute meaningful mesh deformations from given example meshes. Der et al. [DSP06] extended the usage of deformation gradient, and proposed to learn a reduced deformable model from a given set of example shapes, based on their near-rigid deformations [JT05]. The idea of our paper is similar to Summer et al. and Der et al.'s works [SZGP05] [DSP06] in the sense that we use the blending of feature vectors computed from deformation gradients. The main difference is that we automatically compute a sparse localized decomposition of these feature vectors from example meshes, so that each individual component of feature vectors will have a spatially-localized meaning, making it easier for the artists to control and design new shapes and animations.

3. The Method

In our method, we use deformation gradients to represent deformations of the mesh sequence and decompose them to produce the deformation components that are sparse and localized. In the following parts, basic concepts of deformation gradient are introduced in Section 3.1 firstly; representation of deformation gradients in our algorithms is described in Section 3.2; we show the decomposition method based on SPLOCS [NVW*13] in Section 3.3 and give the final algorithms in Section 3.4.

3.1. Deformation Gradient

Suppose a solid $\mathcal{M}$ is deformed to $\mathcal{M}'$. Every point $\mathbf{x} \in \mathcal{M}$ is mapped to its image $\mathbf{x}' = \mathbf{f}(\mathbf{x}) \in \mathcal{M}'$. The deformation gradient is defined as the second-order tensor $\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, so that the infinitesimal vectors before and after deformation can be related by: $d\mathbf{x}' = \mathbf{J}d\mathbf{x}$.

Sumner and Popović [SP04] extended the usage of deformation gradients to triangle meshes. For a triangle $\mathcal{T} = \{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2\}$ and its deformed version $\mathcal{T}' = \{\mathbf{v}'_0, \mathbf{v}'_1, \mathbf{v}'_2\}$, we can represent their edge vectors using these two 3×2 matrices:

$$\mathbf{V} = [\mathbf{v}_1 - \mathbf{v}_0, \mathbf{v}_2 - \mathbf{v}_0], \quad (1)$$

$$\mathbf{V}' = [\mathbf{v}'_1 - \mathbf{v}'_0, \mathbf{v}'_2 - \mathbf{v}'_0]. \quad (2)$$

They are related by the deformation gradient $\mathbf{J}$:

$$\mathbf{V}' = \mathbf{J}\mathbf{V}. \quad (3)$$

With QR decomposition of $\mathbf{V}$ matrix:

$$\mathbf{V} = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix} = [\mathbf{Q}_2\mathbf{Q}_1]\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix} = \mathbf{Q}_2\mathbf{R}, \quad (4)$$

where $\mathbf{R}$ is a 2×2 upper-triangular matrix, and $\mathbf{Q}$ is a 3×3 orthogonal matrix. $\mathbf{Q}_2$ and $\mathbf{Q}_1$ are the 3×2 and 3×1 sub-matrices of $\mathbf{Q}$, respectively. Then the deformation gradient $\mathbf{J}$ with minimum norm [Sum05] is given by:

$$\mathbf{J} = \mathbf{V}'\mathbf{R}^{-1}\mathbf{Q}_2^\top. \quad (5)$$

To reconstruct a mesh from a given field of deformation gradients $\mathbf{J}_j$ defined on each triangle j, we can try to minimize the following energy:

$$E = \sum_j ||\mathbf{V}'_j\mathbf{R}_j^{-1}\mathbf{Q}_{2,j}^\top - \mathbf{J}_j||_F^2, \quad (6)$$

where $||\cdot||_F$ is the Frobenius norm of a matrix, $\mathbf{V}'_j$, $\mathbf{R}_j$, and $\mathbf{Q}_{2,j}$ are the corresponding $\mathbf{V}'$, $\mathbf{R}$, and $\mathbf{Q}_2$ matrices for the reconstructed triangle j, respectively. Since the deformation gradient is translation-invariant, the solution of the above energy minimization is only unique up to a translation. We can fix a vertex, e.g. by setting $\mathbf{v}'_0 = \mathbf{v}_0$ to remove such ambiguity. Then the above energy minimization is simply a least squares problem and can be solved by a linear system.

3.2. Decomposition of Deformation Gradients

The deformation gradient $\mathbf{J}$ can be decomposed as a combination of rotation and stretching through polar decomposition [SD92]:

$$\mathbf{J} = \mathbf{U}\mathbf{P}, \quad (7)$$

where $\mathbf{U}$ is a 3×3 rotation matrix, and $\mathbf{P}$ is a 3×3 symmetric matrix representing the stretching along three orthogonal directions.

Rotation: The rotation matrix $\mathbf{U} \in SO(3)$ can be mapped to its counterpart $\tilde{\mathbf{U}} \in so(3)$ via the logarithm map:

$$\tilde{\mathbf{U}} = \log \mathbf{U} = u_1\mathbf{e}_1 + u_2\mathbf{e}_2 + u_3\mathbf{e}_3, \quad (8)$$

where:

$$\mathbf{e}_1 = \begin{bmatrix}0 & 1 & 0\\ -1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}, \mathbf{e}_2 = \begin{bmatrix}0 & 0 & 1\\ 0 & 0 & 0\\ -1 & 0 & 0\end{bmatrix}, \mathbf{e}_3 = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 1\\ 0 & -1 & 0\end{bmatrix}, \quad (9)$$

and $u_1, u_2, u_3 \in \mathbb{R}$. If we denote the vector:

$$\mathbf{u} = [u_1, u_2, u_3]^\top, \quad (10)$$

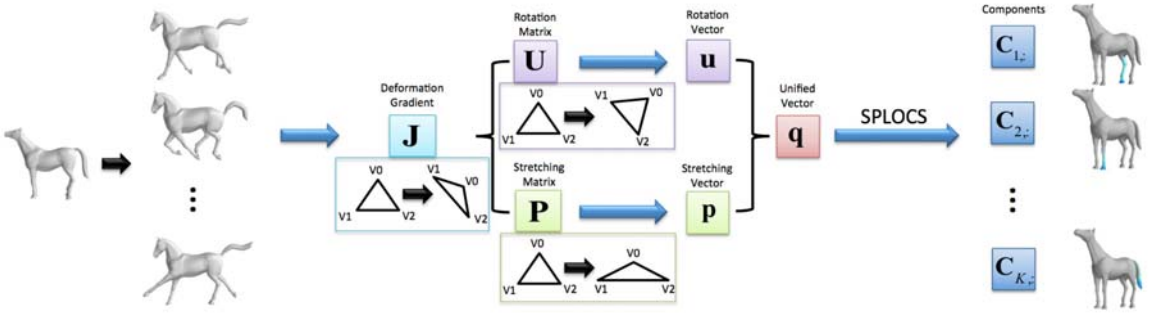

Figure 2: *The algorithmic framework to decompose the sparse localized components of deformation gradients.*

then its direction $\mathbf{u}/||\mathbf{u}||$ corresponds to the axis of rotation and its magnitude $||\mathbf{u}||$ is the rotation angle. Thus if $\mathbf{u} = \vec{0}$, then the corresponding rotation matrix $\mathbf{U} = \mathbf{I}$ (3×3 identity matrix), meaning there is no rotation.

Stretching: The stretching matrix $\mathbf{P}$ is symmetric, and can be represented as:

$$\mathbf{P} = \begin{bmatrix} a & d & e \\ d & b & f \\ e & f & c \end{bmatrix}, \tag{11}$$

where $a,b,c,d,e,f \in \mathbb{R}$. We denote the set of all 3×3 symmetric matrices as $\mathcal{S}(3)$, and define the mapping $\mathbf{g} : \mathcal{S}(3) \rightarrow \mathbb{R}^6$ as:

$$\mathbf{g}(\mathbf{P}) = [a,b,c,d,e,f]^\top. \tag{12}$$

We denote the vector:

$$\mathbf{p} = \mathbf{g}(\mathbf{P} - \mathbf{I}) = [a-1,b-1,c-1,d,e,f]^\top. \tag{13}$$

When $\mathbf{p} = \vec{0}$, the corresponding stretching matrix $\mathbf{P} = \mathbf{I}$, meaning there is no stretching.

The vector representations $\mathbf{u}$ and $\mathbf{p}$ have the same important characteristic: the smaller their magnitudes $||\mathbf{u}||$ and $||\mathbf{p}||$ are, the smaller are the corresponding rotation and stretching, respectively, and vice versa. This property is not seen in either the rotation matrices $\mathbf{U}$ (since they are always orthogonal and have unit determinants) or the deformation gradients $\mathbf{J}$. Thus in the following sparse localized decomposition algorithm, we use their vector representations $\mathbf{u}$ and $\mathbf{p}$ to perform the decomposition.

3.3. Sparse Localized Decomposition

Neumann et al. [NVW*13] proposed a sparse and localized deformation decomposition for mesh sequences using a sparse matrix decomposition method. In this paper, we adopt their method to perform our decomposition of deformation gradients. To make the paper self-contained, in this sub-section we introduce the basic idea of their decomposition method, along with our improvement inside. More detailed derivations can be found in their paper [NVW*13].

Suppose we are given a mesh animation with F frames (over time) and N triangles in each mesh, with consistent mesh connectivity over frames. We select the first frame as a reference frame, and compute the deformation gradients of all other frames w.r.t. this reference frame, and obtain their $\mathbf{u}$ and $\mathbf{p}$ vector representations.

We define a unified vector $\mathbf{q}$ as:

$$\mathbf{q} = [\mathbf{u}^\top, \mathbf{p}^\top]^\top. \tag{14}$$

The matrix that we are going to decompose contains the vector representations $\mathbf{q}$ of all triangles in all frames:

$$\mathbf{X} = \begin{bmatrix} (\mathbf{q}_1^{(1)})^\top & (\mathbf{q}_2^{(1)})^\top & \dots & (\mathbf{q}_N^{(1)})^\top \\ (\mathbf{q}_1^{(2)})^\top & (\mathbf{q}_2^{(2)})^\top & \dots & (\mathbf{q}_N^{(2)})^\top \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{q}_1^{(F)})^\top & (\mathbf{q}_2^{(F)})^\top & \dots & (\mathbf{q}_N^{(F)})^\top \end{bmatrix}, \tag{15}$$

where $\mathbf{q}_i^{(j)}$ represents the $\mathbf{q}$-vector of triangle i at frame j ($i = 1 \dots N$, $j = 1 \dots F$).

The problem is to factorize $\mathbf{X}$ into K components $\mathbf{C}$ and their associated weights $\mathbf{W}$:

$$\mathbf{X} = \mathbf{W}\mathbf{C}, \tag{16}$$

where $\mathbf{C}$ is a matrix with each row representing a component, $\mathbf{W}$ is a matrix with each row representing the weights (corresponding to various components in $\mathbf{C}$) for each frame. Matrix $\mathbf{C}$ has $K \times N$ blocks of row vectors $\mathbf{c}_i^{(k)}$ ($i = 1 \dots N$, $k = 1 \dots K$). The dimension of each row vector $\mathbf{c}_i^{(k)}$ is 9: 3 for vector $\mathbf{u}$ and 6 for vector $\mathbf{p}$.

Neumann et al. [NVW*13] formulated the matrix factorization problem as a joint regularized minimization problem:

$$\arg\min_{\mathbf{W},\mathbf{C}} ||\mathbf{X} - \mathbf{W}\mathbf{C}||_F^2 + \Omega(\mathbf{C}), \tag{17}$$

subject to the following constraints:

$$\begin{array}{lll} & \max(|\mathbf{W}_{:,k}|) = 1, & \forall k, \\ or & \max(\mathbf{W}_{:,k}) = 1, \quad \mathbf{W} \geq 0, & \forall k, \end{array} \tag{18}$$

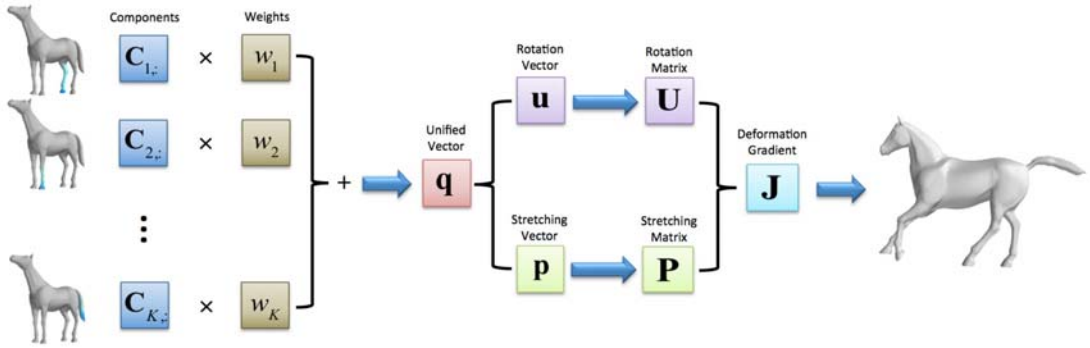

Figure 3: *The algorithmic framework to reconstruct a mesh from the linear combination of components.*

where $\mathbf{W}_{:,k}$ is the k-th column of $\mathbf{W}$, and $\Omega(\mathbf{C})$ is a sparsity regularizer:

$$\Omega(\mathbf{C}) = \sum_{k=1}^{K}\sum_{i=1}^{N} \Lambda_{ki}||\mathbf{c}_i^{(k)}||_2. \tag{19}$$

Here $||\cdot||_2$ denotes the unsquared length of a row vector, and Λ_{ki} are the spatially-varying regularization parameters that are used to enforce local support of the components.

Specifically, the non-zero items of the k-th row of $\mathbf{C}$ (which corresponds to component k) are centered around a set of triangles $\mathcal{J}_k$. This is based on the geodesic distance d_{ki} from each triangle i to $\mathcal{J}_k$, computed in the reference mesh. Λ_{ki} is defined to linearly map the range of geodesic distances $[d_{min}, d_{max}]$ to $[0,1]$, while clamping the values to either 0 or 1 when they are out of this range.

To compute the geodesic distance, we represent each triangle using their centroids, and subdivide each triangle into three sub-triangles by connecting each vertex to the centroid. Thus the triangle-to-triangle geodesic distance becomes the centroid-to-centroid geodesic distance in the subdivided mesh. Then we can use Crane et al.'s heat flow method [CWW13] to compute the geodesic distances.

The optimization of Eq. (17) follows the iterations interleaving the three steps: updating the support map of components based on geodesic distances; solving for best weights $\mathbf{W}$ while fixing the components $\mathbf{C}$; optimizing for sparse components $\mathbf{C}$ with given fixed $\mathbf{W}$ using the Alternating Direction Method of Multipliers (ADMM) [BPC*10].

3.4. Decomposition and Reconstruction Algorithms

Given an animated mesh sequence and a user-specified number of output components, the detailed algorithm to compute our sparse localized components is given in Algorithm 1 and shown in Figure 2. Note that step 1 and step 9 in the algorithm are modified from SPLOCS while the others are new. After the components are computed, users can use weights to linearly combine the components and reconstruct a mesh out of the composed deformation gradients. The detailed algorithm to reconstruct such a mesh is given in Algorithm 2 and shown in Figure 3.

Algorithm 1: Decomposition of Components

Input: An animation mesh sequence with frame indices $0,\ldots,F$
Input: K, the number of output components
Output: The decomposed components $\mathbf{C}$

1 Compute the geodesic distances on the reference mesh;
2 **for** *each frame with index in* $\{1,\ldots,F\}$ **do**
3 **for** *each triangle in the current frame* **do**
4 Compute the deformation gradient w.r.t. the same triangle in frame #0, using Eq. (5);
5 Compute polar decomposition, i.e. Eq. (7);
6 Convert rotation to vector form with Eq. (10);
7 Convert stretching to vector form with Eq. (13);
8 Compose the unified vector $\mathbf{q}$ with Eq. (14);
9 Run SPLOCS algorithm [NVW*13] on $\mathbf{X}$, and obtain the K decomposed components $\mathbf{C}$.

4. Results

The implementation of SPLOCS algorithm was provided by Neumann et al. [NVW*13] on their website. The implementation of the rest of decomposition and reconstruction algorithms are written in Microsoft Visual C++ 2010. The experiments are run on a desktop computer with Intel(R) Xeon E5645 CPU (2.40GHz) and 34GB DDR3 RAM.

To evaluate our method, experiments are conducted by using different models (Face, Horse, Arm, Humanoid). Some results are compared with SPLOCS to show our advantage in Section 4.2 and Section 4.4. We also show the potential applications on shape editing and analysis in Section 4.1, Section 4.2 and Section 4.4. Table 1 gives the statistics of

Animation Datasets			Parameters			Decomposition Time (s) - All Frames						Reconstruction / frame (s)		
Datasets	N	F	K	d_{min}	d_{max}	Geod.	DG	PD	LOG	SPLOCS / #Iter	Total	EXP	Rec.	Total
Face	46,853	384	50	0.1	0.7	8.419	59.163	85.546	245.081	16,812.205/50	17,211.203	0.066	0.135	0.201
Horse	16,843	49	100	0.1	0.5	1.364	2.679	3.701	11.284	2,020.804/100	2,039.867	0.026	0.055	0.081
Arm	10,164	186	100	1	400	0.805	6.158	8.634	25.928	1,059.976/30	1,101.588	0.014	0.032	0.046
Humanoid	15,281	154	100	0.1	0.7	1.221	7.616	10.239	31.896	1,307.590/30	1,358.693	0.020	0.053	0.073

Table 1: *Statistics of the datasets, parameters, and running time. "Geod.": computing the geodesic distance on the reference mesh; "DG": computing deformation gradients for all triangles; "PD": computing the polar decomposition for all deformation gradients; "LOG": computing the matrix logarithm; "SPLOCS / #Iter": the timing of running SPLOCS algorithm with the number of iterations; "EXP": computing the matrix exponential; "Rec.": reconstructing the mesh from deformation gradients. All timing information are shown in seconds.*

Algorithm 2: Reconstruction of a Shape

Input: The decomposed components $\mathbf{C}$
Input: The weights $w_1, \ldots, w_K$ associated with components
Output: The reconstructed triangle mesh

for *each triangle i in the mesh* **do**
  $\mathbf{q} = \vec{0}$;
  for *each component with index* $k \in \{1, \ldots, K\}$ **do**
    $\mathbf{q} = \mathbf{q} + w_k \cdot \mathbf{c}_i^{(k)}$;
  Convert the **u**-part to rotation matrix **U** by matrix exponential;
  Convert the **p**-part to stretching matrix **P**;
  Get the deformation gradient by $\mathbf{J} = \mathbf{UP}$;
Reconstruct the mesh from deformation gradients by minimizing the energy in Eq. (6).

the datasets presented in this paper and the running time of each individual step. Note that the steps of computing the geodesic distance (column of "Geod." in Table 1) and SPLOCS optimization are both implemented in Python by Neumann et al., which are less efficient in performance. The other steps are implemented in C++. Thus we can notice that the SPLOCS optimization is dominating the computational time throughout the decomposition of components. The reconstruction times shown in the table are for reconstructing each frame in the datasets.

In the following figures, we use the color-coding from grey (zero) to blue (max) to show the magnitude of **q** vectors in each computed component. Please refer to the accompanied video for more detailed visual comparison between SPLOCS and our method.

4.1. Face

Figure 5 shows an example of one component computed from the Face animation dataset [ZSCS04], which roughly corresponds to the left eye region. Changing the weight of this component generates interesting expressions of eye-openning ($w = 3$) and eye-closing ($w = -3$). By combining different components of the Face dataset, we can generate arbitrary interesting facial expressions, as shown in Figure 4.

Figure 4: *Combining three components to produce a new facial expression.*

Figure 5: *One component of the Face animation data, multiplied with different weights ($w = 3$ and $w = -3$).*

4.2. Horse

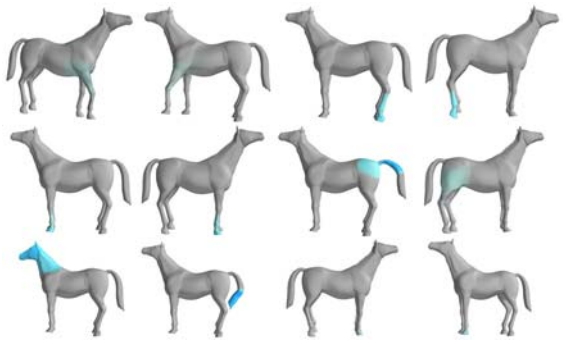

Figure 6: *First 12 components computed from Horse animation sequence with our method.*

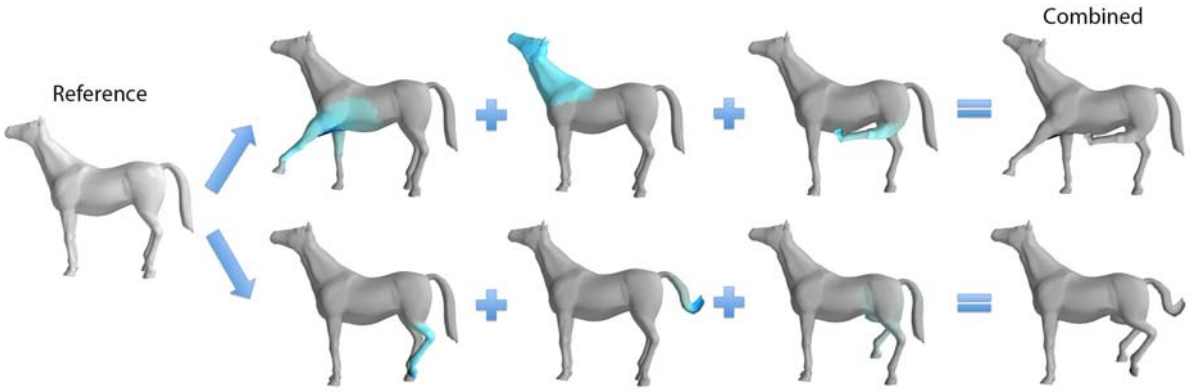

Figure 8: *Combining different components to produce new shapes of Horse.*

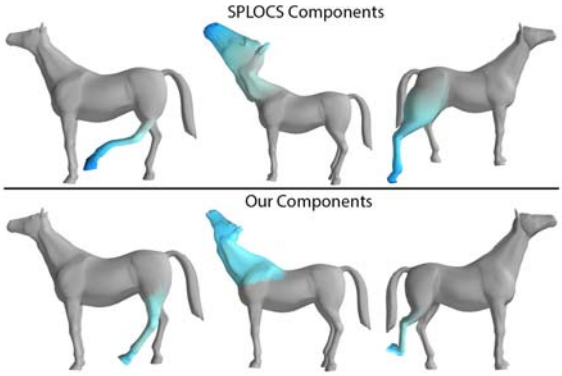

Figure 7: *Comparison between the corresponding components of SPLOCS [NVW*13] and our method.*

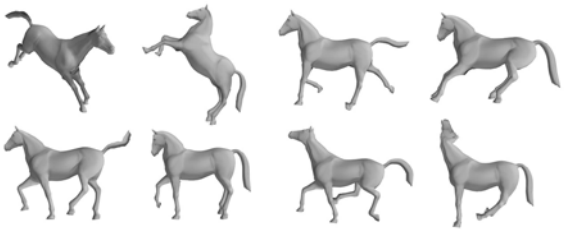

Figure 9: *More new shapes of Horse by combining different components.*

In Figure 6, we show the first 12 sparse localized components computed from Horse animation sequence (horse galloping) with our method. These components demonstrate the distribution of the localized bases and most have an intuitive meaning. We compare the corresponding components produced by SPLOCS and our method. As can be seen from Figure 7, the components computed by SPLOCS tend to produce stretching, shrinking, enlarging effects, and cannot really capture the rotational motion of the horse legs and head. In contrast, our method can faithfully capture these rotational motions from the given animation data. Using the computed sparse localized components, we can combine them to generate new poses of the horse, as shown in Figure 8 and Figure 9.

4.3. Arm

The Arm dataset captured by Neumann et al. [NVH*13] provides not only the rotation of arm, but also the deformation of muscles. We select a subset of the captured dataset consisting of two parts: the first part has 79 frames showing the arm motion without holding any weight at hand; the second part has 107 frames with the hand holding 14kg weight.

As shown in Figure 10, the first component computed from our method represents the rotational motion of the lower arm, while the second component captures the muscle deformation of the upper arm and chest. It can be seen from the curve of weights corresponding to this second component that it captures the muscles's activation without and with the hand holding external weights.

4.4. Humanoid

The motion of Humanoid walking up the stairs has both global translation and joint rotations. If we directly compute its components with SPLOCS, the components will have translational artifacts (top row of Figure 11). If we deliberately remove translation from the original motion before computing with SPLOCS, the components still have rotational artifacts (middle row of Figure 11). In contrast, the components computed from our method do not have these artifacts (bottom row of Figure 11).

We split the Humanoid animation sequence into two sets - training set and testing set. We select every one out of ten

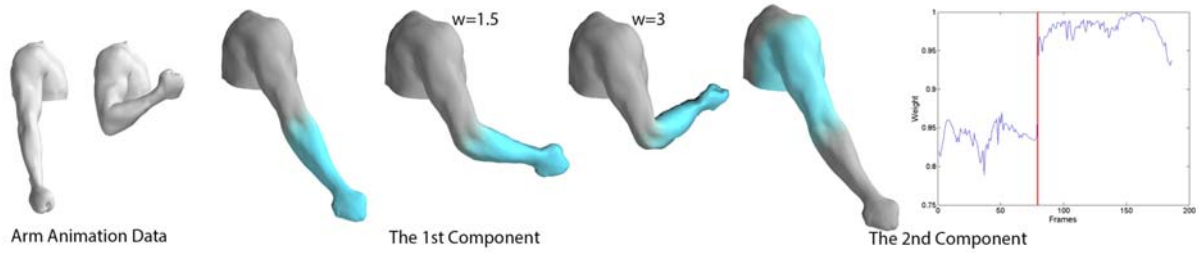

Figure 10: *The first two components of the Arm dataset.*

frames from the original animation, into the training set. The remaining 90% of the frames are used as testing set. The training set is used to compute the components, using either SPLOCS or our method. We use the components to reconstruct every frame in the testing set, and compare their reconstruction errors. Figure 12 shows the visual comparison of reconstructed results between SPLOCS and our components. It is easy to notice the rotational artifacts in the frames reconstructed by SPLOCS components, while our results look much better without these artifacts.

We quantitatively compare the reconstruction errors using two metrics: (1) E_{rms} metric (root mean squared error multiplied by 1000) used by Kavan et al. [KSO10] and Neumann et al. [NVW*13], which characterizes the average geometric error per vertex coordinate; (2) Spatiotemporal Edge Difference (STED) metric suggested by Váša and Skala [VS11] for computing the "perceptual" difference between two animation sequences. Figure 13 shows these comparison results between SPLOCS and our method. For E_{rms} metric, our method performs better when the number of components K is smaller than 20; as K continues to increase, SPLOCS can give smaller E_{rms} errors. However, when using the "perceptual" STED metric, it shows that SPLOCS gives very poor results. It is interesting to note that the STED error of SPLOCS keeps raising with the increase of K. This is because the more components they use, the more wiggling local shapes it will generate (due to the rotational artifacts of its components), even though its E_{rms} is reduced.

5. Limitation and Future Work

It should be noted that the number of triangles in a mesh is typically larger than the number of vertices, and the degree of freedom for each feature vector (i.e. 9) is also larger than that of the displacement vector (i.e. 3). Thus the SPLOCS optimization used in our method is several times slower than its original version [NVW*13]. We would like to explore more efficient GPU-based parallel implementation, in order to make it more accessible for interactively obtaining local activation bases. On the other hand, the less reconstruction time per frame shown in Table 1 indicates the probability to develop iterative applications like pose editing system. By adopting the technique similar to [SZGP05], we can implement an Ik-like interactive interface to provide users a more intuitive and convenient way to control the bases produced by our method.

As mentioned in Section 4.4, our method is translation-invariant. However, we cannot handle the models with global rotation very well, e.g. a horse walking around in circles. The deformation gradients used in our method cannot identify the local deformations (e.g. leg-lifting) and the global deformations (e.g. rotation of the horse) in this case. The components computed from these models are with artifacts. As the animation sequences with global rotation are not unusual, it is important for us to handle this issue in the future.

Acknowledgements

The authors would like to thank the anonymouse reviewers for their valuable comments. This research work was partially supported by Cancer Prevention and Research Institue of Texas (CPRIT) under Grant No. RP110329, and National Science Foundation (NSF) under Grant Nos. IIS-1149737 and CNS-1012975.

The authors would also acknowledge the support of the open funding project of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (No. BUAA-VR-13KF-06), Grant 61174161 from the Natural Science Foundation of China, Grant of Scientific and Technology Emphasis Project of Fujian Province (2011H0031 and 2011H0040), the Fundamental Research Funds for the Central Universities of Xiamen University (No. 0680-ZK10122013121030), the Special and Major Subject Project of the Industrial Science and Technology in Fujian Province 2013 (No. 2013HZ0004-1), 2014 Key Project of Anhui Science and Technology Bureau (No. 1301021018).

References

[ASK*05] ANGUELOV D., SRINIVASAN P., KOLLER D., THRUN S., RODGERS J., DAVIS J.: SCAPE: Shape completion and animation of people. *ACM Transactions on Graphics 24*, 3 (2005). 2

[BLB*08] BICKEL B., LANG M., BOTSCH M., OTADUY M. A.,

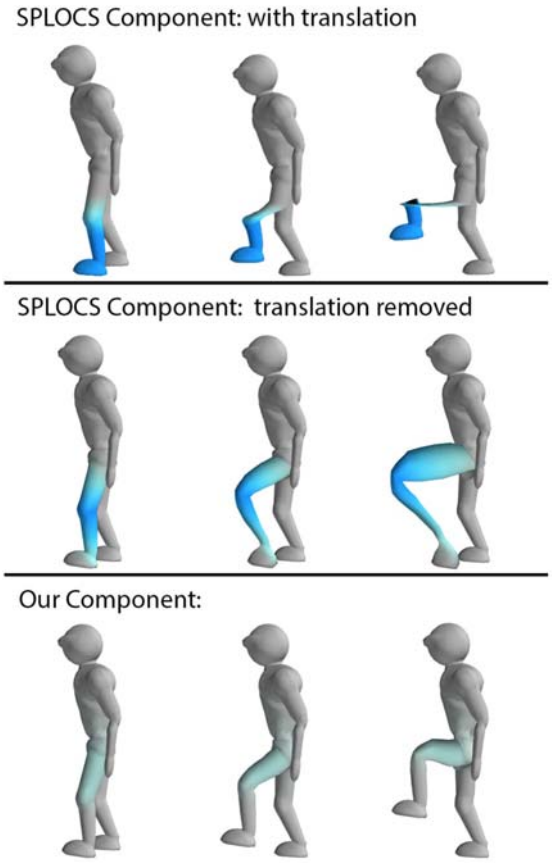

Figure 11: *The Humanoid walking motion has global translation and joint rotations. When computed directly with SPLOCS, the components will have translational artifacts (top row). If we remove translation from the motion before computing with SPLOCS, the components still have rotational artifacts (middle row). The components computed from our method do not have these artifacts (bottom row).*

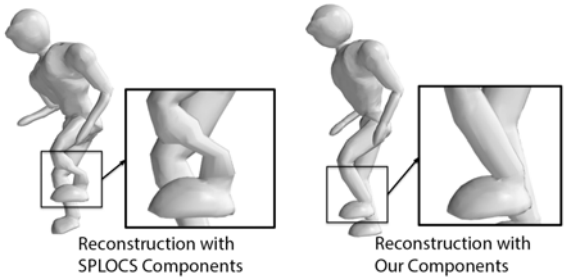

Figure 12: *Reconstruction of the Humanoid motion in the testing set using the components computed from the training set using SPLOCS (left) and our method (right), respectively.*

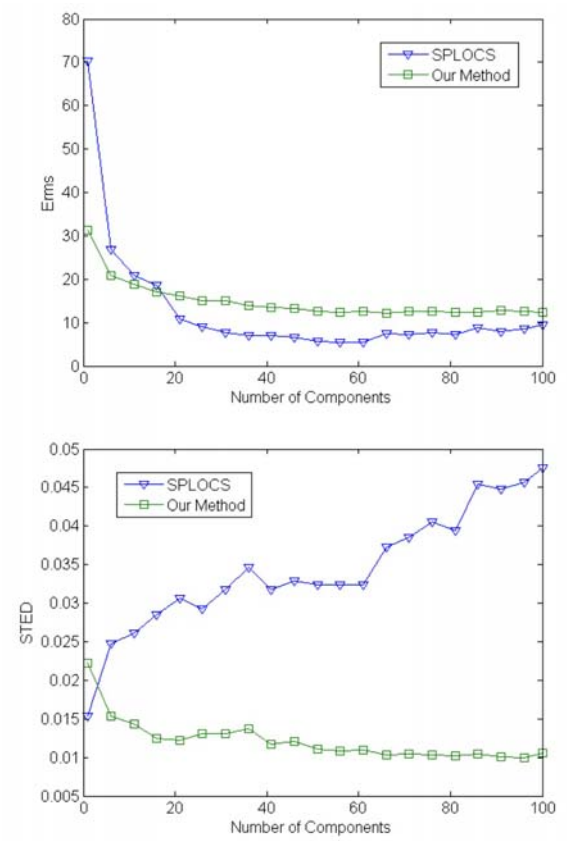

Figure 13: *Comparison of reconstruction errors using E_{rms} (top image) and STED (bottom image) metrics, on the Humanoid testing set.*

GROSS M.: Pose-space animation and transfer of facial details. In *Proceedings of Symposium on Computer Animation* (2008), pp. 57–66. 2

[BPC*10] BOYD S., PARIKH N., CHU E., PELEATO B., ECKSTEIN J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning 3* (2010), 1–122. 5

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics 32* (2013), 152:1–152:11. 5

[dATTS08] DE AGUIAR E., THEOBALT C., THRUN S., SEIDEL H.-P.: Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum 27*, 2 (2008). 2

[DSP06] DER K. G., SUMNER R. W., POPOVIĆ J.: Inverse kinematics for reduced deformable models. *ACM Transactions on Graphics 25*, 3 (2006), 1174–1179. 3

[FKY08] FENG W.-W., KIM B.-U., YU Y.: Real-time data driven deformation using kernel canonical correlation analysis. *ACM Transactions on Graphics 27*, 3 (2008). 2

[HKO01] HYVÄRINEN A., KARHUNEN J., OJA E.: *Independent Component Analysis*. Wiley-Interscience, 2001. 2

[HSS*10] HASLER N., STOLL C., SUNKEL M., ROSENHAHN

B., SEIDEL H.-P.: A statistical model of human pose and body shape. *Computer Graphics Forum 28*, 2 (2010), 337–346. 2

[HTRS10] HASLER N., THORMÄHLEN T., ROSENHAHN B., SEIDEL H.-P.: Learning skeletons for shape and pose. In *Proceedings of the 2010 symposium on Interactive 3D Graphics and Games* (2010), pp. 23–30. 2

[HZY*11] HUANG H., ZHAO L., YIN K., QI Y., YU Y., TONG X.: Controllable hand deformation from sparse examples with rich details. In *Proceedings of Symposium on Computer Animation* (2011), pp. 73–82. 2

[JT05] JAMES D. L., TWIGG C. D.: Skinning mesh animations. *ACM Transactions on Graphics 24*, 3 (2005). 2, 3

[KJP02] KRY P. G., JAMES D. L., PAI D. K.: Eigenskin: Real time large deformation character skinning in hardware. In *Proceedings of Symposium on Computer Animation* (2002), pp. 153–159. 2

[KM04] KURIHARA T., MIYATA N.: Modeling deformable human hands from medical images. In *Proceedings of Symposium on Computer Animation* (2004), pp. 355–363. 2

[KSO10] KAVAN L., SLOAN P.-P., O'SULLIVAN C.: Fast and efficient skinning of animated meshes. *Computer Graphics Forum 29*, 2 (2010). 2, 8

[LCF00] LEWIS J. P., CORDNER M., FONG N.: Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques* (2000), pp. 165–172. 2

[LCXS10] LAU M., CHAI J., XU Y.-Q., SHUM H.-Y.: Face poser: Interactive modeling of 3D facial expressions using facial priors. *ACM Transactions on Graphics 29*, 1 (2010). 2

[LD12] LE B., DENG Z.: Smooth skinning decomposition with rigid bones. *ACM Transactions on Graphics 31*, 6 (2012). 2

[LWH*12] LEVINE S., WANG J. M., HARAUX A., POPOVIĆ Z., KOLTUN V.: Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics 31*, 4 (2012). 2

[LWP10] LI H., WEISE T., PAULY M.: Example-based facial rigging. *ACM Transactions on Graphics 29*, 4 (2010). 2

[Mac09] MACKEY L.: Deflation methods for sparse PCA. In *Advances in Neural Information Processing Systems* (2009), pp. 1017–1024. 2

[MG03] MOHR A., GLEICHER M.: Building efficient, accurate character skins from examples. *ACM Transactions on Graphics 21*, 3 (2003). 2

[NVH*13] NEUMANN T., VARANASI K., HASLER N., WACKER M., MAGNOR M., THEOBALT C.: Capture and statistical modeling of arm-muscle deformations. *Computer Graphics Forum 32*, 2 (2013). 7

[NVW*13] NEUMANN T., VARANASI K., WENGER S., WACKER M., MAGNOR M., THEOBALT C.: Sparse localized deformation components. *ACM Transactions on Graphics 32*, 6 (2013), 179:1–179:10. 1, 2, 3, 4, 5, 7, 8

[PBB*13] POKRASS J., BRONSTEIN A. M., BRONSTEIN M. M., SPRECHMANN P., SAPIRO G.: Sparse modeling of intrinsic correspondences. *Computer Graphics Forum 32*, 2pt4 (2013), 459–468. 2

[RLN06] RHEE T., LEWIS J. P., NEUMANN U.: Real-time weighted pose-space deformation on the GPU. *Computer Graphics Forum 25*, 3 (2006), 439–448. 2

[SD92] SHOEMAKE K., DUFF T.: Matrix animation and polar decomposition. In *Proceedings of the Conference on Graphics Interface'92* (1992), pp. 258–264. 3

[SP04] SUMNER R. W., POPOVIĆ J.: Deformation transfer for triangle meshes. *ACM Transactions on Graphics 23*, 3 (2004), 399–405. 3

[Sum05] SUMNER R. W.: *Mesh modification using Deformation Gradients*. PhD thesis, Massachusetts Institude of Technology, 2005. 3

[SZGP05] SUMNER R. W., ZWICKER M., GOTSMAN C., POPOVIĆ J.: Mesh-based inverse kinematics. *ACM Transactions on Graphics 24*, 3 (2005), 488–495. 3, 8

[TITM11] TENA J. R., LA TORRE F. D., MATTHEWS I.: Interactive region-based linear 3D face models. *ACM Transactions on Graphics 30*, 4 (2011). 2

[TR12] TOURNIER M., REVERET L.: Principal geodesic dynamics. In *Proceedings of Symposium on Computer Animation* (2012), pp. 235–244. 2

[VS11] VÁŠA L., SKALA V.: A perception correlated comparison method for dynamic meshes. *IEEE Transactions on Visualization and Computer Graphics 17*, 2 (2011), 220–230. 8

[WSLG07] WEBER O., SORKINE O., LIPMAN Y., GOTSMAN C.: Context-aware skeletal shape deformation. *Computer Graphics Forum 26*, 3 (2007). 2

[WYL*14] WANG R., YANG Z., LIU L., DENG J., CHEN F.: Decoupling noises and features via weighted L_1-analysis compressed sensing. *ACM Transactions on Graphics 33*, 2 (2014). 3

[ZHT06] ZOU H., HASTIE T., TIBSHIRANI R.: Sparse principal component analysis. *Journal of Computational and Graphical Statistics 15*, 2 (2006). 2

[ZSCS04] ZHANG L., SNAVELY N., CURLESS B., SEITZ S. M.: Spacetime faces: High-resolution capture for modeling and animation. In *ACM SIGGRAPH* (2004), pp. 548–558. 6

Appendix A: Representing Rotation with Components

Without loss of generality, let us consider a model $\mathcal{M}$ being rotated by angle θ around axis $\mathbf{a}$. The displacement field is simply:

$$\mathbf{d}(\mathbf{x}) = (\mathbf{R}(\theta, \mathbf{a}) - \mathbf{I})\mathbf{x}, \tag{20}$$

where $\mathbf{R}(\theta, \mathbf{a})$ is the rotation matrix corresponding to rotation angle θ and axis $\mathbf{a}$, and $\mathbf{x}$ is an arbitrary point on $\mathcal{M}$.

From the Rodrigues' rotation formula:

$$\mathbf{R}(\theta, \mathbf{a}) = \mathbf{I} + \sin\theta[\mathbf{a}]_\times + (1 - \cos\theta)[\mathbf{a}]_\times^2, \tag{21}$$

where $[\mathbf{a}]_\times$ is the "cross-product matrix" for $\mathbf{a}$. Thus we can rewrite the displacement $\mathbf{d}(\mathbf{x})$ as:

$$\mathbf{d}(\mathbf{x}) = (\sin\theta[\mathbf{a}]_\times + (1 - \cos\theta)[\mathbf{a}]_\times^2)\mathbf{x}. \tag{22}$$

When θ is close to zero, we can expand $\sin\theta$ and $\cos\theta$ using Taylor series and omit $O(\theta^2)$ terms. Thus it gives us:

$$\mathbf{d}(\mathbf{x}) \approx \theta[\mathbf{a}]_\times\mathbf{x}, \quad \theta \to 0. \tag{23}$$

That is to say, when θ is close to zero, we can use the deformation component $\mathbf{c}(\mathbf{x}) = [\mathbf{a}]_\times\mathbf{x}$ with its weight θ to approximate the displacement: $\mathbf{d}(\mathbf{x}) \approx \theta\mathbf{c}(\mathbf{x})$. However, when θ is large enough, such approximation is no longer accurate.