Wang C, Liu Y, Guo X *et al.* Spectral animation compression. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 30(3): 540–552 May 2015. DOI 10.1007/s11390-015-1544-z

Spectral Animation Compression

Chao Wang[1] (王 超), Yang Liu[2] (刘 旸), Xiaohu Guo[1,*] (郭小虎), *Member, ACM*, Zichun Zhong[1] (钟子春)
Binh Le[3], and Zhigang Deng[3] (邓志刚), *Senior Member, IEEE, Member, ACM*

[1]*Department of Computer Science, University of Texas at Dallas, Richardson, Texas 75080, U.S.A.*

[2]*Facebook, Menlo Park, California 94025, U.S.A.*

[3]*Department of Computer Science, University of Houston, Houston, Texas 77004, U.S.A.*

E-mail: chao.wang3@utdallas.edu; thomasyoung.liu@gmail.com; {xguo, zichunzhong}@utdallas.edu
bhle2@cs.uh.edu; zdeng4@uh.edu

Received December 3, 2014; revised March 12, 2015.

Abstract This paper presents a spectral approach to compress dynamic animation consisting of a sequence of homeomorphic manifold meshes. Our new approach directly compresses the field of deformation gradient defined on the surface mesh, by decomposing it into rigid-body motion (rotation) and non-rigid-body deformation (stretching) through polar decomposition. It is known that the rotation group has the algebraic topology of 3D ring, which is different from other operations like stretching. Thus we compress these two groups separately, by using Manifold Harmonics Transform to drop out their high-frequency details. Our experimental result shows that the proposed method achieves a good balance between the reconstruction quality and the compression ratio. We compare our results quantitatively with other existing approaches on animation compression, using standard measurement criteria.

Keywords dynamic animation, animation compression, deformation gradient, polar decomposition

1 Introduction

Dynamic mesh animations consist of a (time-series) sequence of 3D meshes sharing the same connectivity but having different shapes. For years, people have been looking for techniques to reduce memory and storage required for processing detailed mesh animation data.

In general, compression technologies for mesh animation could be roughly classified as lossless methods and lossy methods. Lossless methods preserve all the original information while lossy methods reduce the size by dropping less important details of the data. As for data compression, lossy methods play important roles due to the nature of large data and in-sensitivity to reconstruction accuracy. In the case of mesh animation compression, we believe that shape fidelity is more important than the accuracy of geometric coordinates. To achieve better visual results, preserving the shape and deformation of the surface is more vital.

As Sumner and Popović[1] suggested, the deformation of a surface could be represented as the field of *deformation gradient* over the surface. This field, however, unlike many other fields on surfaces, is defined per simplex, i.e., triangle rather than vertex. The deformation gradient of each simplex could be presented as a 3×3 transformation matrix. Deformation gradient contains enough information to describe the deformation of the whole surface. That is, the compression of mesh animation could be achieved by compressing its deformation gradient.

A 3×3 transformation matrix can represent any linear transform of a specific triangle. With polar decomposition[2], it can be further decomposed as the combination of two components: 1) 3-D rotation; and 2) planar stretching. 3-D rotation has a structure of 3-D torus[3], while the planar deformation has a structure of

Regular Paper

Special Section on Computational Visual Media

Chao Wang, Xiaohu Guo, and Zichun Zhong are partially supported by the National Science Foundation under Grant Nos. IIS-1149737 and CNS-1012975.

*Corresponding Author

infinite linear space. The field of deformation gradient can be handled separately for these two components in the lossy compression process, and thus better control of loss over both components could be achieved.

In this paper, we propose a spectral animation compression (SAC) approach to perform lossy compression of mesh animations. Specifically, we use the eigenfunctions of Laplace-Beltrami Operator (LBO) defined on the surface[4], also called Manifold Harmonics[5], to compress the two components of deformation gradient field, respectively. This paper shows the experiments of our new approach, compared with several existing animation compression methods, along with data supporting the various stages of our algorithm design decisions.

2 Related Work

In the past few years, a great number of algorithms have been proposed to compress dynamic mesh animations. Among them, principle component analysis (PCA) is often used for reducing the dimensions of the vertex displacement information[6-7]. The PCA bases need to be compressed because they are needed to restore the animation[8]. Moreover, PCA can also be combined with clustering and segmentation[9], Laplace deformation[10], or mesh simplification[11] for better performance in compressing dynamic animations.

Another common class of compression approaches is predictive methods which encode the residual (difference between the actual value and the predicted value) instead of raw data. With a carefully designed predictor, the residual tends to be small and fits the need for further quantization and/or entropy encoding. In terms of animation compression, the predictor could depend on spatial information, temporal information, or both of them, as in Dynapack[12] and the fine-granular scalable coding method[13]. Moreover, Khodakovsky *et al.*[14] proposed to apply predictive method and arithmetic encoding on wavelet transform coefficients. Amjoun and Straβer[15] introduced a new connectivity-guided predictive scheme to encode the residues in the local coordinate frames to improve the efficiency of animation compression. Stefanoski and Ostermann[16] presented a fast and efficient scalable predictive coding (SPC) scheme by employing predictive encoding in the space of rotation-invariant coordinates with respect to a spatially and temporally scalable decomposition of dynamic animation. Another series of common compression methods are to combine prediction encoding with PCA to design predictors more accurately, such as Connectivity-Driven Dynamic Animation Compression method (Coddyac)[17] and Geometry-Driven Local Neighbourhood Based Predictors[18]. Besides the animation compression methods, some state-of-the-art single triangle mesh compression methods are also based on predictive coding schemes[19-20].

A different animation compression approach is based on the idea of skinning mesh techniques. In animation, the movements of neighbouring primitives (vertices or triangles) tend to be related rather than independent. Parts of the mesh that have similar movements are referred to as near-rigid structure in skinning mesh animation[21]. For instance, Mamou *et al.*[22] introduced a novel method based on a piecewise affine predictor coupled with a skinning model and a representation of the residuals. Parts of this skinning-based method[22] are utilized by the frame-based animated mesh compression (FAMC)[23] as an MPEG-4 standard (MPEG-4 FAMC). Le and Deng[24] referred to a similar concept in Smooth Skinning Decomposition with Rigid Bones (SSDR). Specifying movements of these pieces as a whole could reduce redundancy greatly. Skinning decomposition technique is employed during decompression to provide better fidelity.

To determine the performance of compression methods, E-RMS[25] and KG-error metric[6] are commonly used to evaluate errors introduced by compression. Váša and Skala proposed a new error metric called spatio-temporal edge difference (STED) to better evaluate subjective opinions[26], and also a new framework by reconfiguring FAMC and Coddyac to optimize the STED error[27].

Researchers have been looking for spectral geometric processing methods[28] for surface smoothing[29], segmentation[30], compression[4], watermarking[31], quadrangulation[32], matching and retrieval[33], etc. Manifold Harmonics, defined as the eigen-functions of LBO, can be computed on triangle mesh surfaces based on the Discrete Exterior Calculus (DEC) framework[5] or the Finite Element Method (FEM) framework[33]. It can be also computed on point-sampled surfaces[34], which is based on the framework of heat diffusion kernel[35-36].

3 Spectral Animation Compression

A mesh animation consists of a sequence of triangular meshes, which are also referred to as frames. Each frame is slightly deformed from the previous one, while the connectivity is the same. The canonical way of storing mesh animation is keeping the vertex coordinates of each frame separately. The data would have high redundancy. A more efficient way is keeping the vertex coordinates for the first frame and incrementally deforming it to get all following frames.

Intuitively, the deformation of a surface could be presented as a displacement field over vertices. In this paper, we take a different approach: we represent deformation as the field of deformation gradient[1] defined over triangles. By compressing the incremental shape deformations represented as deformation gradients using the spectral compression techniques[5], the animation can be compressed to cost less space for storage and transfer.

In the following parts, a new animation compression method based on deformation gradient and manifold harmonics is proposed and explained in details.

3.1 Overall Scheme

Suppose the animation to be compressed consists of a sequence of homeomorphic manifold surfaces represented as triangular meshes $\{M_0, M_1, \cdots, M_t\}$, where t is the total number of frames. Meshes $\{M_i\}$ are referred to *frames*, and M_0 is usually the reference frame. Considering the deformation between adjacent frames M_i and M_{i+1}, the deformations of neighbouring primitives (vertices, edges and triangles in our case) are assumed to be closely related. That is, the deformation field representation has data redundancy, which makes compression possible.

As shown in Fig.1, the scheme of spectral animation compression consists of the following steps:

1) Reference frame compression: the first frame M_0 is assumed to be compressed using an existing triangular mesh compression method. Note that the decompressed frame M_0', rather than the original M_0, is used in the following steps of compression process. Therefore, the error will not be accumulated.

2) Deformation compression on $\{M_i\}$ $(1 \leqslant i \leqslant t)$: for each frame M_i, its deformation from the previous reconstructed frame M_{i-1}' is decomposed into components (Subsection 3.3) and analyzed using Manifold Harmonics (Subsection 3.4). Some detailed information is dropped to achieve a balance between compression ratio and reconstruction accuracy. Quantization and arithmetic coding are performed to further lower the data size (Subsection 3.5).

3) Decompression and reconstruction: using the previous (reconstructed) frame M_{i-1}' and deformation description information of the i-th frame from the above step 2, the animation $M_i{}'$ is recovered with error (Subsection 3.6).

The detailed SAC algorithm outline is introduced in Subsection 3.7. Note that in the proposed method, the deformation of each frame M_i (P-frame) is computed from the reconstructed previous frame, except for the reference frame M_0 (I-frame). That is, the decompression of each frame depends on that of the previous one. Such a strategy has the following advantages:

1) *No Error Accumulation.* Our method is designed as lossy compression for each frame. By choosing the decompressed previous frame M_{i-1}' as reference shape for computing the incremental deformation, we essentially eliminate the accumulation of error introduced in reconstructing each frame, since the error only comes from the decompression of deformation description.

2) *Smaller Chance to Suffer from Extreme Rotations.* Here an extreme rotation refers to a rotate angle larger than $\pi/4$ and close to $\pi/2$. In our method,

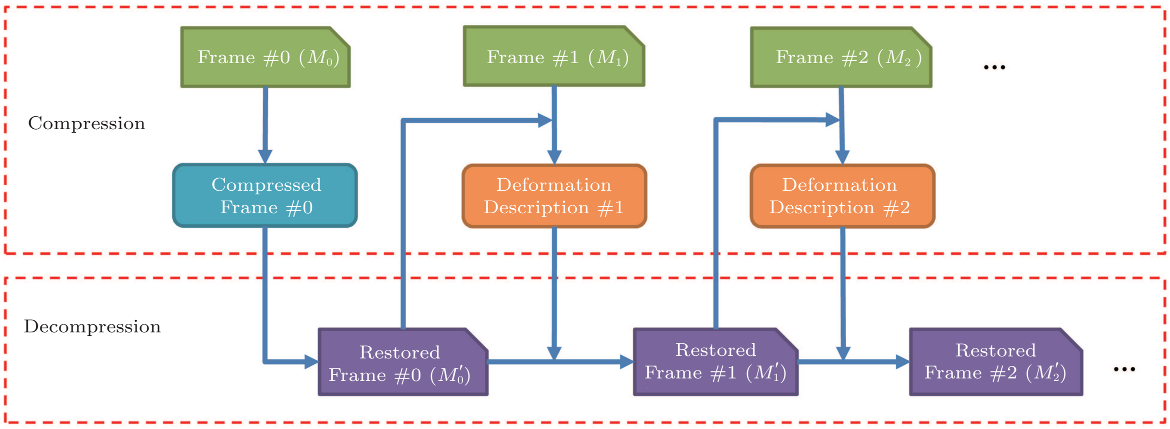

Compression
Frame #0 (M_0)
Frame #1 (M_1)
Frame #2 (M_2)
...
Compressed Frame #0
Deformation Description #1
Deformation Description #2
Decompression
Restored Frame #0 (M_0')
Restored Frame #1 (M_1')
Restored Frame #2 (M_2')
...

Fig.1. Scheme of compression and decompression.

the deformation to be compressed is decomposed as the combination of rotation and stretching. An extreme rotation may corrupt the continuity of the rotation field and cause more distortion.

A disadvantage of our method is that it cannot support random access to the compressed animation. That is, all frames must be decompressed one by one. Users cannot access a specified frame directly. To compensate for the disadvantage, a reference frame (I-frame) could be chosen after certain number of P-frames. In case of random access, the users only need to go through a small set of the reference frames rather than the whole animation.

3.2 Deformation Gradient

In our method, we choose *deformation gradient*[1] as the basic representation of deformations between frames.

Consider the deformation of a solid $\mathcal{M}$ to $\tilde{\mathcal{M}}$. Each point $\boldsymbol{p} \in \mathcal{M}$ is mapped to point $\tilde{\boldsymbol{p}}$ after deformation. Denote the coordinates of each point as a column vector $\boldsymbol{p} = (p_1, p_2, p_3)^{\mathrm{T}}$. Then the mapping $\boldsymbol{t} = (t_1, t_2, t_3)^{\mathrm{T}}$ can be defined as:

$$\tilde{\boldsymbol{p}} = \boldsymbol{t}(\boldsymbol{p}) = \begin{pmatrix} t_1(p_1, p_2, p_3) \\ t_2(p_1, p_2, p_3) \\ t_3(p_1, p_2, p_3) \end{pmatrix}.$$

The deformation gradient is defined as a second order tensor:

$$\boldsymbol{J} = \frac{\partial \boldsymbol{t}}{\partial \boldsymbol{p}} = \begin{pmatrix} \frac{\partial t_1}{\partial p_1} & \frac{\partial t_1}{\partial p_2} & \frac{\partial t_1}{\partial p_3} \\ \frac{\partial t_2}{\partial p_1} & \frac{\partial t_2}{\partial p_2} & \frac{\partial t_2}{\partial p_3} \\ \frac{\partial t_3}{\partial p_1} & \frac{\partial t_3}{\partial p_2} & \frac{\partial t_3}{\partial p_3} \end{pmatrix}.$$

At point $\boldsymbol{p}$, the infinitesimal vector after deformation is given by $d\tilde{\boldsymbol{p}} = \left(\frac{\partial \boldsymbol{t}}{\partial \boldsymbol{p}}\right) d\boldsymbol{p}$.

Here the deformation mapping $\boldsymbol{t} : \mathbb{R}^3 \to \mathbb{R}^3$ maps each point to a new position. The discrete counter part of $\boldsymbol{J}$, denoted as $\boldsymbol{J}_j$, is defined as a piece-wise constant function over simplex j (e.g., a tetrahedron). The tensor $\boldsymbol{J}_j$ is referred to as the deformation gradient. For triangular meshes that consist of triangles rather than tetrahedrons, the method is to construct a tetrahedron on each triangle by adding an extra vertex in the normal direction and compute the deformation gradient for each triangle[1].

Given the restored frame M'_{i-1} $(1 \leqslant i \leqslant t)$, and a desired field of deformation gradient $\{\boldsymbol{J}_j\}$ between restored M'_{i-1} and original M_i as input, we are trying to reconstruct a shape M'_i such that the deformation between M'_{i-1} and M'_i is as close to the deformation between M'_{i-1} and M_i as possible. This process is achieved by solving the following least-squares optimization problem:

$$\min_{\boldsymbol{v}_0, \cdots, \boldsymbol{v}_{n-1}} \sum_{j=0}^{m-1} \|\boldsymbol{S}_j - \boldsymbol{J}_j\|_F^2, \tag{1}$$

$$\text{subject to } \tilde{\boldsymbol{v}}_0 = \boldsymbol{v}_0, \tag{2}$$

where n is the number of vertices, m is the number of triangles, and $\boldsymbol{S}_j$ is the deformation gradient between the j-th pair of triangles in M'_{i-1} and that in M'_i. The vertices $\{\boldsymbol{v}_0, \cdots, \boldsymbol{v}_{n-1}\}$ in M'_i could be achieved by converting the above minimization problem into an equivalent linear system and then solving it efficiently[1]. Note that deformation gradients are invariant to translation[1]. Therefore, there are infinitely many solutions to the optimization problem in (1), which admit the same minimum: all translations of one optimal solution are also optimal. Then, the constraint in (2) is added to fix the position of one vertex and makes the solution unique[1].

Deformation Gradient vs Displacement. The displacement field of vertices is the canonical way to describe deformation. But in the case of animation compression, the displacement field is not an effective way compared with deformation gradient. Consider a triangle mesh animation with only rigid-body rotation and translation. It is easy to see that each triangle has the same deformation gradient which is the global rotation. Thus the deformation gradient is a constant field, while the displacement field may be complicated and hard to compress with acceptable errors. Thus we choose deformation gradient rather than displacement for representing the deformation field.

3.3 Analysis of Deformation Gradient Field

As presented in Subsection 3.2, the deformation gradient defines a 3-by-3 linear transformation matrix $\boldsymbol{J}_j$ for each triangle j. Any invertible linear transformation matrix like $\boldsymbol{J}_j$ could be decomposed as the combination of rotation and stretching through polar decomposition[2]:

$$\boldsymbol{J}_j = \boldsymbol{U}_j \boldsymbol{P}_j, \tag{3}$$

where $\boldsymbol{U}_j \in SO(3)$ (3D rotation group) is a rotation matrix and $\boldsymbol{P}_j = \boldsymbol{P}_j^{\mathrm{T}}$ is a stretching matrix that is symmetric.

It is known that the rotation matrix group $SO(3)$ is isomorphic to 3-D ring in 4-D Euclidean space[37]. Therefore, it resembles the topology of 3-D ring.

For the stretching matrix $\boldsymbol{P}_j$, since it is symmetric, we can define it as:

$$\boldsymbol{P}_j = \begin{pmatrix} a & d & e \\ d & b & f \\ e & f & c \end{pmatrix}. \tag{4}$$

Denote the set of all 3×3 symmetric matrices as $\mathcal{S}$. Define the maping $f_{\mathcal{S}} : \mathcal{S} \rightarrow \mathbb{R}^6$ as

$$f_{\mathcal{S}}(\boldsymbol{P}_j) = \begin{pmatrix} a & b & c & d & e & f \end{pmatrix}^{\mathrm{T}}.$$

Thus we know that $\mathcal{S}$ resembles the topology of 6-D Euclidean space $\mathbb{R}^6$, which is different from the topology of 3-D ring of $SO(3)$.

As described above, the rotation matrix group and the stretching matrix set resemble different topologies. Intuitively, rotation towards a fixed direction gets back periodically, while stretching extends infinitely. To achieve better control and lower distortion for our spectral compression, the two pieces of information are handled and compressed separately in the proposed method. Fig.2 shows the SAC process with polar decomposition.

Here is an intuitive explanation about the rationality of polar decomposition in SAC. Firstly, compressing displacement and/or deformation gradient directly do/does not provide good results, and we observe that the surface of these results would easily show artifact like "wrinkle" which suggests the rotation info is not preserved well all the time, even though the error metric does not look that bad. Then, we understand that rotation is different from stretching in that rotation in $\mathbb{R}^3$ has topology of 3-D torus, and thus it is bounded and never goes to infinity, but stretching may. If we handle them together, we may easily lose control of rotation information. When rotation gets too many errors here and there, "wrinkle" happens and hurts the result a lot. Therefore, if we handle rotation and stretching separately by utilizing polar decomposition, we have a "bound" for rotation errors. That is, we may not have extreme errors of rotation on some triangles that hurts the result. Experiments testify that it works, as shown in the comparative results between the spectral compression with and without using polar decomposition in Subsection 4.2.

3.4 Spectral Compression of Deformation Field

Several spectral transformation tools, like Manifold Harmonics[5], have been proposed for manifold surfaces represented as triangular meshes. By solving generalized eigen problem defined by the finite element method (FEM)-based discrete Laplacian operator, a set of bases called manifold harmonic bases (MHBs) are constructed. Because the discrete Laplacian operator retains the property of symmetricity as LBO does, it could be proved that MHBs are orthogonal to each other with inner product. Thus MHB could be used to analyze functions defined over the mesh.

Note that Manifold Harmonics is designed for scalar functions defined "per-vertex". All the bases are per-vertex functions, so are the functions to be analyzed. But in our scheme, the deformation gradients are piecewise constant, i.e., they are defined "per-triangle". Thus the original Manifold Harmonics functions cannot

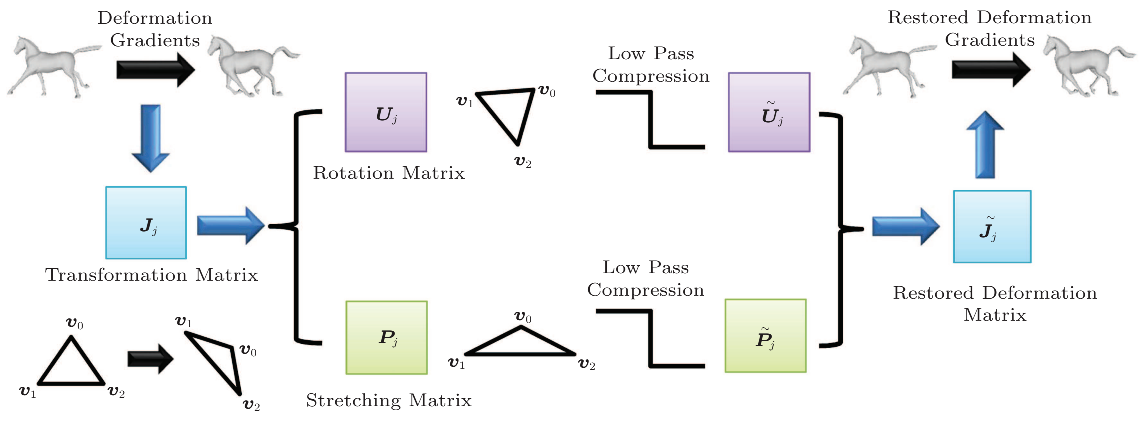


Fig.2. Spectral compression with polar decomposition.

be applied as-is, and should be converted to per-triangle functions at first.

Suppose we have a "per-vertex" function which is denoted as a column vector $\boldsymbol{f}_v$, one intuitive way to get its per-triangle form $\boldsymbol{f}_t$ is averaging the function values on the vertices. Thus we have :

$$\boldsymbol{f}_t = \boldsymbol{C}\boldsymbol{f}_v, \tag{5}$$
$$\boldsymbol{f}_v = (\boldsymbol{C}^{\mathrm{T}}\boldsymbol{C})^{-1}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{f}_t, \tag{6}$$

where $\boldsymbol{C} = \{c_{i,j}\}$ is the m-by-n interpolation matrix :

$$c_{i,j} = \begin{cases} \dfrac{1}{3}, \text{ if vertex } j \text{ is contained in triangle } i, \\ 0, \text{ otherwise.} \end{cases}$$

Here m and n are the number of triangles and vertices, respectively. Using (5) and (6), any per-vertex function defined over the mesh surface could be interpolated to its per-triangle form and vice versa. Thus per-triangle functions could also be analyzed using Manifold Harmonics which is designed for per-vertex functions.

For triangle $\forall j \in M_0$, consider its corresponding rotation matrix $\boldsymbol{U}_j \in SO(3)$. Denote its matrix logarithm[37] as $\boldsymbol{A}_j = \log \boldsymbol{U}_j$ where $\boldsymbol{A}_j \in SO(3)$ and

$$\boldsymbol{A}_j = a_1\boldsymbol{e}_1 + a_2\boldsymbol{e}_2 + a_3\boldsymbol{e}_3 \tag{7}$$

holds, where

$$\boldsymbol{e}_1 = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{e}_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix},$$
$$\boldsymbol{e}_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix},$$

and $a_1, a_2, a_3 \in \mathbb{R}$. Thus the rotation matrix field $\boldsymbol{U}_j$ could be treated as a 3-D vector field $(a_1, a_2, a_3)^{\mathrm{T}}$ after the logarithm mapping.

According to (4), the stretching matrix field $\boldsymbol{P}_j$ is guaranteed to be symmetric: $\boldsymbol{P}_j^{\mathrm{T}} = \boldsymbol{P}_j$. Similarly, $\boldsymbol{P}_j$ could be treated as a 6-D vector field $(a, b, c, d, e, f)^{\mathrm{T}}$ instead.

Altogether, $\boldsymbol{U}_j$ and $\boldsymbol{P}_j$ have nine degrees of freedom (DoFs) and could be processed as independent scalar fields. In the case of discrete triangular mesh, each scalar field is denoted as a column vector. Thus we have vectors $\boldsymbol{v}_{U,0} \cdots \boldsymbol{v}_{U,2}$ and $\boldsymbol{v}_{P,0} \cdots \boldsymbol{v}_{P,5}$, where each $\boldsymbol{v}_{U,p}$ $(0 \leqslant p \leqslant 2)$ and $\boldsymbol{v}_{P,q}$ $(0 \leqslant q \leqslant 5)$ are vectors of length m, with m being the number of triangles. With the MHB $\{\boldsymbol{H}_k\}$ and interpolation equation (6) presented earlier, we have their corresponding spectral descriptors:

$$v_{U,k,p} = \langle \boldsymbol{H}_k, (\boldsymbol{C}^{\mathrm{T}}\boldsymbol{C})^{-1}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{v}_{U,p} \rangle, \tag{8}$$
$$v_{P,k,q} = \langle \boldsymbol{H}_k, (\boldsymbol{C}^{\mathrm{T}}\boldsymbol{C})^{-1}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{v}_{P,q} \rangle, \tag{9}$$

where $0 \leqslant k < l$, and l is the number of spectral bases we use. If we use all the spectral bases, then $l = n$, where n is the number of vertices. By dropping high frequency descriptors (i.e., $l \ll n$), the compression is achieved.

3.5 Quantization and Arithmetic Coding

The spectral descriptors obtained from (8) and (9) are float numbers, which may cost 32 bits (single precision) or 64 bits (double precision). Quantization could further reduce the data size while introducing some errors.

The quantization process could reduce the size of a bunch of data by truncating the data to a desired accuracy and mapping them into integers that can be represented with a limited number of q bits[12]. After quantization, each spectral descriptor value will be quantized as an integer number in a range of $[0, 2^q]$. Usually the corresponding appearance numbers of a large amount of quantized integers are zero, then some entropy coding method such as arithmetic coding could be utilized to compress these integers to further reduce the bite rate losslessly. Note that the maximum and the minimum value of the original data also need to be compressed in order to restore the original data from the quantized integers during the decompression process. Fig.3 shows an example of quantization result. After quantization, there are 65 k possible numbers, but only a small portion of them would really appear.

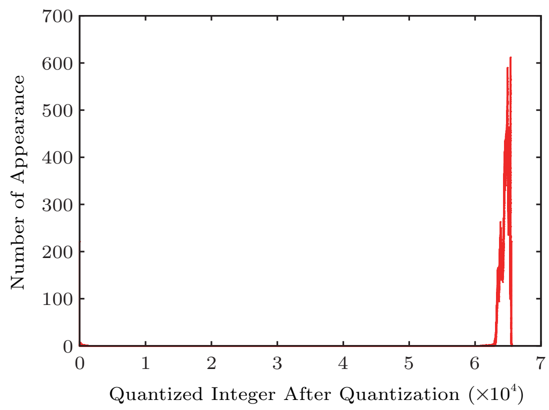


Fig.3. Distribution of signal numbers after 16-bit quantization on the model Jump with 300 spectral bases.

In summary, the compressed information in SAC includes the following parts:

1) the compressed geometry and connectivity of the first frame M_0;

2) the quantized integers of spectral descriptors with appearance number larger than 0, and the corresponding appearance number;

3) for each frame, the maximum and the minimum of spectral descriptors, along with bits produced by arithmetic coding for the quantized spectral descriptors $\tilde{v}_{U,k,p}$ and $\tilde{v}_{P,k,q}$.

3.6 Decompression and Reconstruction

Let $\tilde{v}_{U,k,p}$, $\tilde{v}_{P,k,q}$ be the restored spectral descriptors, and $\tilde{\boldsymbol{v}}_{U,p}$, $\tilde{\boldsymbol{v}}_{P,q}$ be the restored vector fields of $\{\tilde{\boldsymbol{U}}_j\}$ and $\{\tilde{\boldsymbol{P}}_j\}$, which denote the restored rotation and stretching matrices for the deformation gradients $\{\tilde{\boldsymbol{J}}_j\}$. During the decompression and reconstruction process, $\tilde{v}_{U,k,p}$, $\tilde{v}_{P,k,q}$ should be restored at first by decompressing their quantized data of $v_{U,k,p}$ and $v_{P,k,q}$ using the appropriate decoding method. The vector fields could be restored as

$$\tilde{\boldsymbol{v}}_{U,p} = \boldsymbol{C}\sum_{k=1}^{l}\tilde{v}_{U,k,p}\boldsymbol{H}_k, \tag{10}$$

$$\tilde{\boldsymbol{v}}_{P,q} = \boldsymbol{C}\sum_{k=1}^{l}\tilde{v}_{P,k,q}\boldsymbol{H}_k. \tag{11}$$

Then, the deformation gradient $\tilde{\boldsymbol{J}}_j$ can be easily achieved by $\tilde{\boldsymbol{J}}_j = \tilde{\boldsymbol{U}}_j\tilde{\boldsymbol{P}}_j$.

With restored deformation gradients $\{\tilde{\boldsymbol{J}}_j\}$, the optimization problem in (2) is like this now:

$$\min_{\boldsymbol{v}_0,\cdots,\boldsymbol{v}_{n-1}} \sum_{j=0}^{m-1}\left\|\boldsymbol{S}_j - \tilde{\boldsymbol{J}}_j\right\|_F^2,$$
$$\text{subject to } \tilde{\boldsymbol{v}}_0 = \boldsymbol{v}_0.$$

The vertices $\{\boldsymbol{v}_0,\cdots,\boldsymbol{v}_{n-1}\}$ could be calculated by converting the above minimization problem into the following linear system[1]:

$$\boldsymbol{AX} = \boldsymbol{f}, \tag{12}$$

where $\boldsymbol{X}$ is the column vector of vertex coordinates, matrix $\boldsymbol{A}$ and vector $\boldsymbol{f}$ are calculated from the mesh connectivity and deformation gradients.

3.7 Algorithm Outline

Given the original animation $\{M_0, M_1, \cdots, M_t\}$, the SAC outline can be summarized as:

1) compress M_0 using some lossless single mesh compression method, and then decompress it to get restored frame $M_0{}'$;

2) compute Manifold Harmonics bases $\{\boldsymbol{H}_k\}$ using the method in [5];

3) for the i-th frame ($1 \leqslant i \leqslant t$), given M'_{i-1} and M_i, we can compress M_i and restore M'_i like this:

a) compute the deformation gradient $\{\boldsymbol{J}_j\}$ using the method in [1], where $\boldsymbol{J}_j$ is the deformation gradient between the j-th pair of triangles in M_i and M'_{i-1};

b) run polar decomposition on each $\boldsymbol{J}_j$ via (3) to get rotation matrix $\boldsymbol{U}_j$ and stretching matrix $\boldsymbol{P}_j$;

c) run matrix logarithm on each $\boldsymbol{U}_j$ via (7) and pack the result as vectors $\boldsymbol{v}_{U,p}$, and then pack the six relative values in each $\boldsymbol{P}_q$ as vectors $\boldsymbol{v}_{P,q}$. Then, the spectral descriptors $v_{U,k,p}$ and $v_{P,k,q}$ for each base $\boldsymbol{H}_k$ ($0 \leqslant k < l$) can be obtained via (8) and (9) respectively. The high frequency descriptors with basis larger than threshold l will be dropped;

d) compress $v_{U,k,p}$ and $v_{P,k,q}$ using quantization and some entropy coding method such as arithmetic coding;

e) decompress the spectral descriptors to get $\tilde{v}_{U,k,p}$ and $\tilde{v}_{P,k,q}$, and then restore the vector fields $\tilde{\boldsymbol{v}}_{U,p}$ and $\tilde{\boldsymbol{v}}_{P,q}$ via (10) and (11), respectively. Then, the deformation gradient $\tilde{\boldsymbol{J}}_j$ can be restored by $\tilde{\boldsymbol{J}}_j = \tilde{\boldsymbol{U}}_j\tilde{\boldsymbol{P}}_j$;

f) reconstruct the vertex coordinates of M'_i by solving the linear system in (12) through the method in [1]; then the restored M'_i is achieved.

Besides the first step to compress and decompress M_0, step 2 and steps 3).a)$\sim$3).d) belong to compression process, while steps 3).e)$\sim$3).f) belong to decompression process.

4 Experiments

Experiments are conducted to evaluate SAC using models listed in Table 1 and Fig.4. The Cloth model showing a piece of falling cloth is selected as a representative bone-less animation where bone structure is difficult to be defined. The Horse-Collapse model from [24] presents a horse-shaped surface collapsing on itself that undergoes sophisticated soft tissue deformations. The Vase model contains a bouncing and rolling vase with only rigid-body rotations and translations. In our experiments, 18-bit quantization is used by default unless otherwise specified.

Most parts of our implementation are written in C++ and CUDA. Miscellaneous codes are implemented in Python and Shell Script. Table 1 lists the time cost of our implementation on a desktop computer with Intel® Xeon® E5645 CPU with 2.40 GHz, 16 GB RAM

Table 1. Experimental Datasets and Time Cost in Seconds

Animation	t	n	m	T_2	T_{3a}	T_{3b}	T_{3c}	T_{3d}	T_{3e}	T_{3f}	T_{total}
Cloth	199	5 525	10 752	0.140 1	0.002 3	0.042 0	0.143 6	0.008 9	0.066 1	0.079 1	0.482 1
Horse-Collapse	53	8 431	16 843	0.188 0	0.003 3	0.030 1	0.104 5	0.011 3	0.101 1	0.117 1	0.555 4
Dance	200	7 061	14 118	0.161 8	0.003 0	0.062 4	0.210 6	0.009 2	0.085 1	0.121 9	0.654 1
Humanoid	153	7 646	15 288	0.202 5	0.003 3	0.022 4	0.106 4	0.008 8	0.093 6	0.097 0	0.534 0
Dolphin	100	6 179	12 278	0.198 6	0.002 7	0.045 2	0.157 3	0.008 6	0.075 7	0.096 8	0.584 9
Jump	221	15 826	31 648	0.655 9	0.006 9	0.135 5	0.483 3	0.008 7	0.191 4	0.299 1	1.780 8
Vase	70	2 502	5 008	0.055 2	0.001 2	0.022 6	0.081 8	0.008 9	0.032 7	0.035 4	0.237 8

Note: t denotes the number of frames, n is the number of vertices, m is the number of triangles, T_2, T_{3a}, T_{3b}, T_{3c}, T_{3d}, T_{3e}, T_{3f} denote the average time cost of step 2 and steps 3).a) $\sim$ 3).f) for each frame in the SAC algorithm outline, respectively, and T_{total} denotes the total average time cost for each frame.

and NVIDIA GeForce GTX 770 GPU with 4 GB video memory. All the models are compressed using 100 spectral bases. The time cost of step 1 in the algorithm outline, i.e., compressing and decompressing the first frame, is not listed in Table 1, since it is very small and can be ignored compared to the cost in other steps. Note that the most time-consuming steps are step 2 and step 3).c), i.e., computing Manifold Harmonics bases and running matrix logarithm on rotation matrix.

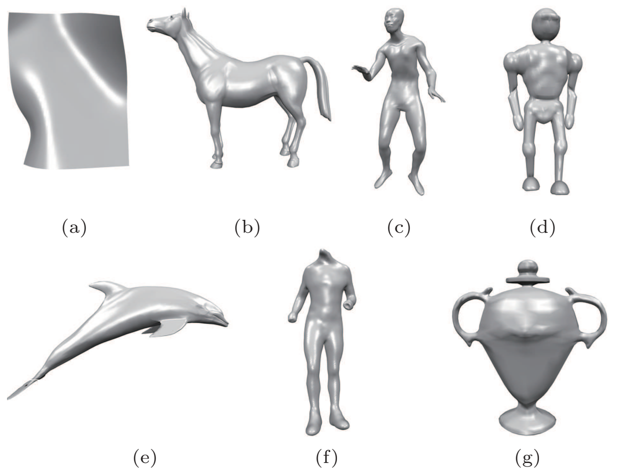


Fig.4. Models used in experiments. (a) Cloth. (b) Horse-Collapse. (c) Dance. (d) Humanoid. (e) Dolphin. (f) Jump. (g) Vase.

4.1 Metrics

The following metrics are chosen to evaluate the error introduced by compression:

1) The spatial-temporal edge difference (STED) introduced by Váša and Skala[26]:

$$STED = \sqrt{STED_s^2 + c^2 \times STED_t^2},$$

where $STED_s$ is the spatial error, $STED_t$ is the temporal error, and c is a relating constant. In our experiments, the values of all constants in (13) are the same as those used in [26].

2) The KG Error introduced by Karni and Gotsman[6]:

$$KG_{\text{error}} = 100\frac{\|A - \hat{A}\|}{\|A - \boldsymbol{E}(A)\|},$$

where $\|.\|$ is the Frobenius norm, A is a $3n \times t$ matrix containing the geometry of the original sequence, $\hat{A}$ is the same animation after compression, and $\boldsymbol{E}(A)$ is an average matrix in which the i-th column is defined by

$$(\bar{X}_i(1 \cdots 1), \bar{Y}_i(1 \cdots 1), \bar{Z}_i(1 \cdots 1))^{\text{T}},$$

with $\bar{X}_i$, $\bar{Y}_i$ and $\bar{Z}_i$ being the mean values of the coordinate sets of each frame i.

The size of data after compression is characterized by bit per vertex per frame (bpvf).

4.2 Justification of Polar Decomposition

In the proposed method, the deformation gradient $\boldsymbol{J}_j$ is decomposed into rotation matrix $\boldsymbol{U}_j$ and stretching matrix $\boldsymbol{P}_j$ that would be compressed separately. This is to gain better control on two components that have very different characteristics. Fig.5 shows the bpvf-error results of SAC with or without polar decomposition. We can see that the polar decomposition clearly reduces the KG-error, which measures the discrepancy between absolute coordinate values of two animations. This means polar decomposition restores the rotation and stretching features very well. However, the two curves under STED are very close when bpvf is large. The reason is that STED measures the difference between the corresponding edge lengths of animations, which is a local property independent of absolute position. Therefore, even though SAC without polar decomposition does not restore the rotation information well, the STED errors may still be very close to those with polar decomposition.

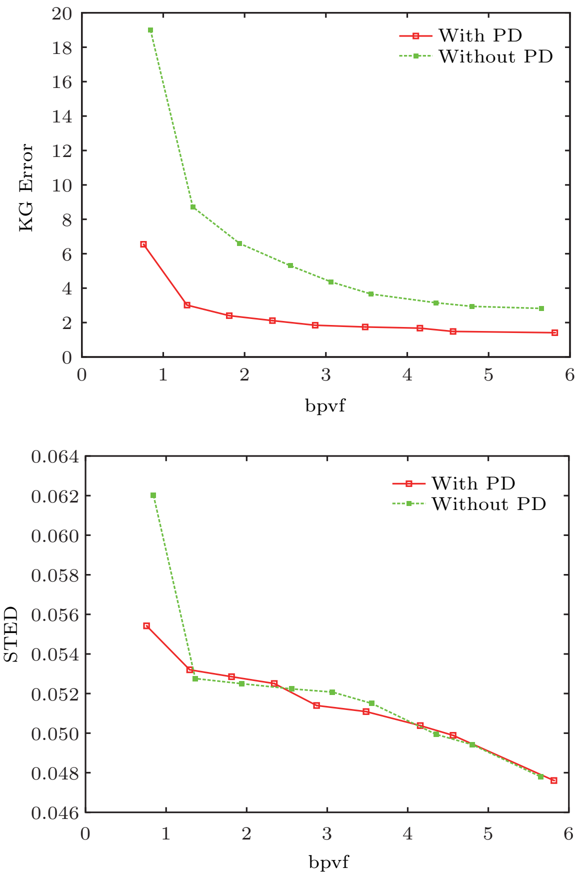


Fig.5. SAC results with and without polar decomposition on the model Horse-Collapse.

4.3 Performance for Different Animations

The results of SAC on all experimental models are shown in Fig.6. Please note that the errors on the model Vase are almost zero, since Vase contains only rigid-body rotation and translation, which means the deformation gradient fields are constant for each triangle in this model and can be restored by SAC very well. Meanwhile, the animations featuring smooth deformation like Cloth and Dolphin tend to have better performance than skeleton-driven animations such as Dance and Jump.

4.4 Comparison with CoDDyaC

We compare our method with CoDDyaC[17]. Figs.7 and 8 show the bpvf-error curves on three animations: Vase (only rigid-body rotation and translation) and Cloth (elastic rolling and bouncing transformation).

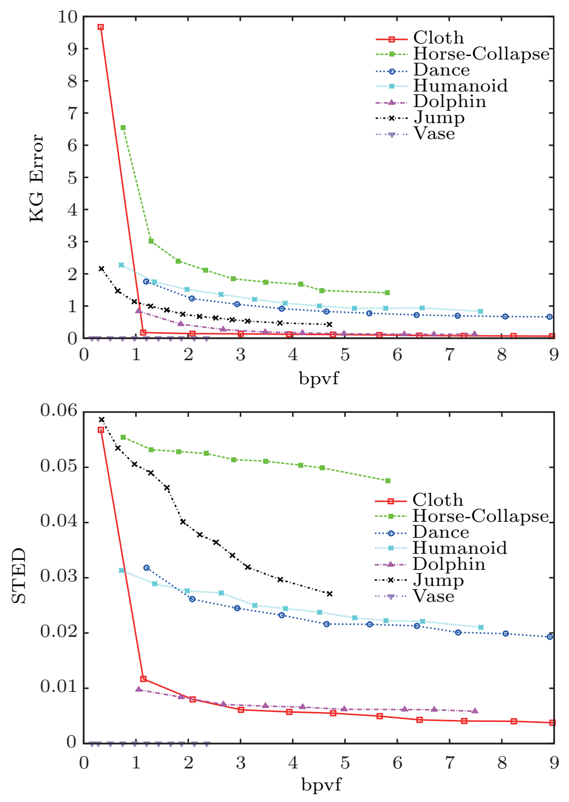


Fig.6. Performance of SAC on different animations.

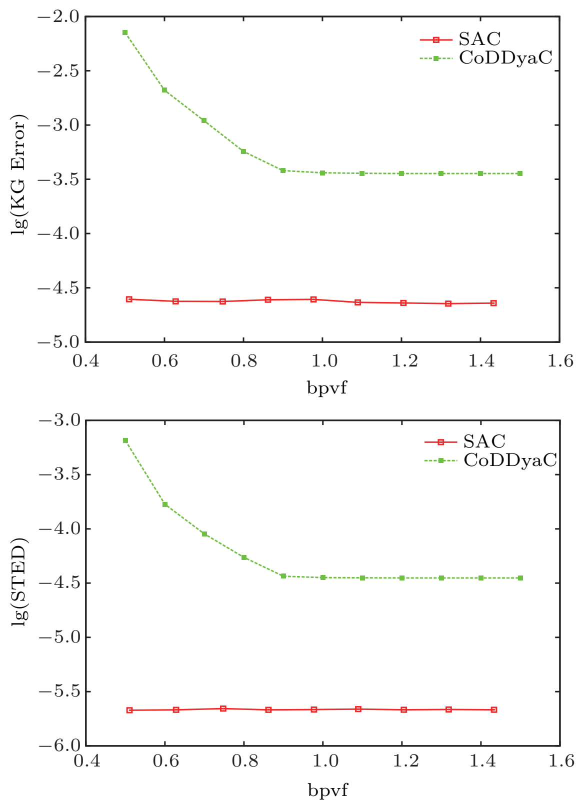
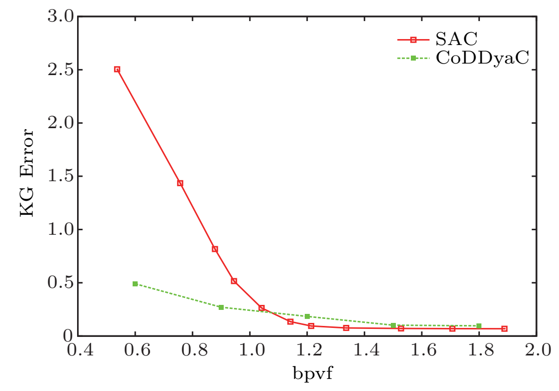


Fig.7. Comparison of SAC and CoDDyaC on the model Vase. Note that the errors are logarithm values with base 10.



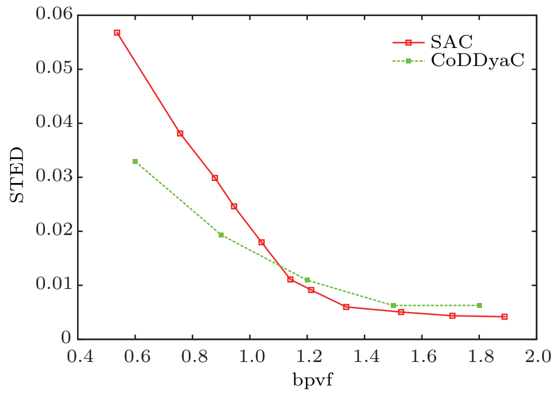


Fig.8. Comparison of SAC and CoDDyaC on model Cloth.

As shown in Fig.7, SAC performs better than CoDDyaC in that the errors of SAC are one order of magnitude smaller than those of CoDDyaC, which means SAC is better in restoring rigid-body transformations. Note that both KG and STED errors of SAC are almost constant under different bpvfs corresponding to different base numbers (actually these small errors are numerical errors during computation). The reason is that the deformation between frames in Vase is composed of only rotation and translation. Then, after polar decomposition, each rotation vector field for each frame is constant field, while each stretching matrix is identity matrix, theoretically. Therefore, only one basis is enough to present the rigid-body motion.

Fig.8 shows that SAC's errors are more than CoDDyaC's errors under low bit rate, but will be the same as or less than CoDDyac's errors with bpvf equal to 1.0 or more, since when bpvf is too small, the corresponding base number in SAC is not enough to describe the deformation between frames.

4.5 Effect of Arithmetic Coding

In this subsection, we will firstly analyze the effect of arithmetic coding on compression process. As shown in Fig.3, the distribution of numbers after quantization is uneven. Thus entropy coding like arithmetic coding could achieve further compression. As shown in Fig.9, arithmetic coding could reduce the bit rate for the Jump model by about a third, without introducing any error.

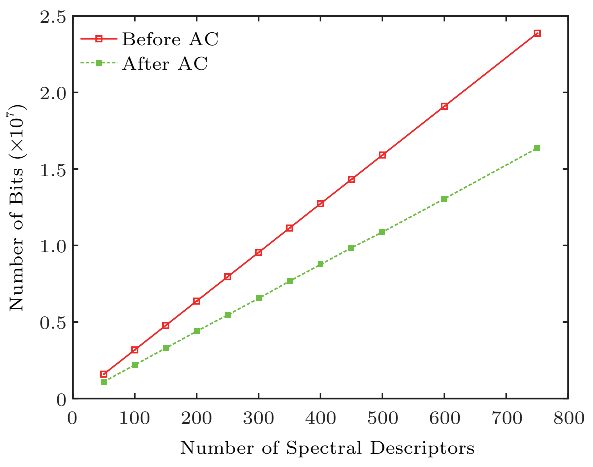


Fig.9. Number of compression bits under different numbers of spectral descriptors before and after arithmetic coding on model Jump.

During arithmetic coding, different quantization bit values should be selected at first. A small bit value will save a large storage amount, but will also bring in errors for restored spectral descriptors. Fig.10 shows the influence of different quantization bit values on the KG error result of SAC on the model Vase. It can be easily seen from the figure that the error under 18-bit quantization is enough for arithmetic coding in SAC.

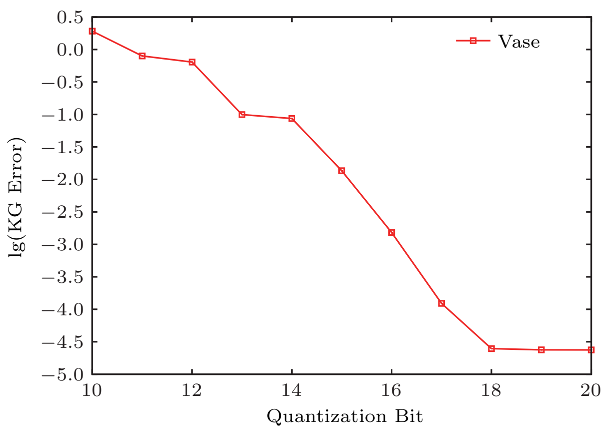


Fig.10. KG-errors of SAC on model Vase with respect to different quantization bits in arithmetic coding. Here base number is 5. Note that the errors are logarithm values with base 10.

5 Discussions

5.1 Excessive DoFs for Stretching

In our current work, we use six DoFs to represent the stretching information of each triangle. It is a bit awkward because the addition of the extra vertex in normal direction does not really introduce any stretching information, and the planar stretching has only three DoFs.

To see this fact, we can align the normal vector $\boldsymbol{v}_n - \boldsymbol{v}_0$ of a triangle to the z-axis and then decompose the stretching matrix as: $\boldsymbol{P} = \boldsymbol{R}_{nz}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{R}_{nz}$, where $\boldsymbol{R}_{nz}$ is a rotation matrix satisfying $\boldsymbol{R}_{nz}(\boldsymbol{v}_n - \boldsymbol{v}_0) = (0, 0, 1)^{\mathrm{T}}$. It is not difficult to show that:

$$\boldsymbol{S} = \begin{pmatrix} a & b & 0 \\ b & c & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{13}$$

which means that $\boldsymbol{S}$ is essentially a stretching matrix in the xy-plane, and has only three DoFs. It should be noted that the rotation matrix $\boldsymbol{R}_{nz}$ encodes the remaining three DoFs.

Therefore, the question is: can we store only the three DoFs information of $\boldsymbol{S}$, by figuring out an implicit way to compute $\boldsymbol{R}_{nz}$? Assuming that $\boldsymbol{P}$ is continuous over the surface, if we want to get a continuous field of $\boldsymbol{S}$, we need to implicitly define a continuous field of $\boldsymbol{R}_{nz}$ on the triangles of the surface. This amounts to the problem of defining a continuous field of orthogonal vector-pairs on the surface, i.e., defining the x- and y-directions of the local coordinate system for each triangle in its tangential space. Such smooth vector fields on the surface will inevitably have singularities. In our future work, we would like to investigate such an approach, and see how the singularities (discontinuities) of $\boldsymbol{R}_{nz}$ affect the performance of spectral compression.

5.2 Limitations and Future Work

Recall that SAC is proposed based on the assumption that deformation is continuous. But this may not hold true for some animations like the Origami deformation. In these cases, the topology of the mesh surface may remain intact, while the different sides of the folding edge may undergo very different transformations, which violates the continuity assumption. This may create large errors of the deformation gradients near the folding edge, and thus cause dramatic errors in surrounding vertices, e.g., the armpit of the running horse model undergoes folding/unfolding deformation in certain frames. The error of deformation gradient near it would accumulate through several straight unfolding/folding frames until getting large enough to cause significant reconstruction errors. In such a case, using more spectral bases would reduce such an error, at the cost of a higher bit rate.

Another limitation is that the proposed method is performing only spatial compression. In other words, it is only trying to eliminate spatial data redundancy inside each frame, rather than redundancy between frames. Methods to target both spatial and temporal redundancies would be our future research directions.

References

[1] Sumner R, Popović J. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 2004, 23(3): 399-405.

[2] Shoemake K, Duff T. Matrix animation and polar decomposition. In *Proc. the Conference on Graphics Interface*, May 1992, pp.258-264.

[3] Baker A. Matrix Groups: An Introduction to Lie Group Theory. Springer, 2002.

[4] Karni Z, Gotsman C. Spectral compression of mesh geometry. In *Proc. the 27th Annual Conference on Computer Graphics and Interactive Techniques*, July 2000, pp.279-286.

[5] Vallet B, Lévy B. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum*, 2008, 27(2): 251-260.

[6] Karni Z, Gotsman C. Compression of soft-body animation sequences. *Computers & Graphics*, 2004, 28(1): 25-34.

[7] Alexa M, Müller W. Representing animations by principal components. *Computer Graphics Forum*, 2000, 19(3): 411-418.

[8] Váša L, Skala V. COBRA: Compression of the basis for PCA represented animations. *Computer Graphics Forum*, 2009, 28(6): 1529-1540.

[9] Sattler M, Sarlette R, Klein R. Simple and efficient compression of animation sequences. In *Proc. the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, July 2005, pp.209-217.

[10] Tejera M, Hilton A. Compression techniques for 3D video mesh sequences. In *Articulated Motion and Deformable Objects*, Perales F J, Fisher R B, Moeslund T B (eds.), Springer Berlin Heidelberg, 2012, pp.12–25.

[11] Váša L, Skala V. Combined compression and simplification of dynamic 3D meshes. *Computer Animation and Virtual Worlds*, 2009, 20(4): 447-456.

[12] Ibarria L, Rossignac J. Dynapack: Space-time compression of the 3D animations of triangle meshes with fixed connectivity. In *Proc. the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, July 2003, pp.126-135.

[13] Ahn J, Koh Y, Kim C. Efficient fine-granular scalable coding of 3D mesh sequences. *IEEE Transactions on Multimedia*, 2013, 15(3): 485-497.

[14] Khodakovsky A, Schröder P, Sweldens W. Progressive geometry compression. In *Proc. the 27th Annual Conference on Computer Graphics and Interactive Techniques*, July 2000, pp.271-278.

[15] Amjoun R, Straßer W. Single-rate near lossless compression of animated geometry. *Computer-Aided Design*, 2009, 41(10): 711-718.

[16] Stefanoski N, Ostermann J. SPC: Fast and efficient scalable predictive coding of animated meshes. *Computer Graphics Forum*, 2010, 29(1): 101-116.

[17] Váša L, Skala V. CODDYAC: Connectivity driven dynamic mesh compression. In *Proc. 3DTV Conference*, May 2007.

[18] Váša L, Skala V. Geometry-driven local neighbourhood based predictors for dynamic mesh compression. *Computer Graphics Forum*, 2010, 29(6): 1921-1933.

[19] Lee D, Sull S, Kim C. Progressive 3D mesh compression using MOG-based Bayesian entropy coding and gradual prediction. *The Visual Computer*, 2014, 30(10): 1077-1091.

[20] Zhang J, Zheng C, Hu X. Triangle mesh compression along the Hamiltonian cycle. *The Visual Computer*, 2013, 29(6/7/8): 717-727.

[21] James D, Twigg C. Skinning mesh animations. *ACM Transactions on Graphics*, 2005, 24(3): 399-407.

[22] Mamou K, Zaharia T, Prêteux F. A skinning approach for dynamic 3D mesh compression. *Computer Animation and Virtual Worlds*, 2006, 17(3/4): 337-346.

[23] Mamou K, Zaharia T, Prêteux F. FAMC: The MPEG-4 standard for animated mesh compression. In *Proc. the 15th IEEE International Conference on Image Processing*, October 2008, pp.2676-2679.

[24] Le B, Deng Z. Smooth skinning decomposition with rigid bones. *ACM Transactions on Graphics*, 2012, 31(6): Article No. 199.

[25] Kavan L, Sloan P, O'Sullivan C. Fast and efficient skinning of animated meshes. *Computer Graphics Forum*, 2010, 29(2): 327-336.

[26] Váša L, Skala V. A perception correlated comparison method for dynamic meshes. *IEEE Transactions on Visualization and Computer Graphics*, 2011, 17(2): 220-230.

[27] Váša L, Petřík O. Optimising perceived distortion in lossy encoding of dynamic meshes. *Computer Graphics Forum*, 2011, 30(5): 1439-1449.

[28] Zhang H, Kaick O, Dyer R. Spectral methods for mesh processing and analysis. In *Proc. EUROGRAPHICS*, Sept. 2007.

[29] Taubin G. A signal processing approach to fair surface design. In *Proc. the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, September 1995, pp.351-358.

[30] Liu R, Zhang H. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum*, 2007, 26(3): 385-394.

[31] Cotting D, Weyrich T, Pauly M, Markus G. Robust watermarking of point-sampled geometry. In *Proc. Shape Modeling International*, June 2004, pp.233-242.

[32] Dong S, Bremer P, Garland M, Pascucci V, Hart J. Spectral surface quadrangulation. *ACM Transactions on Graphics*, 2006, 25(3): 1057-1066.

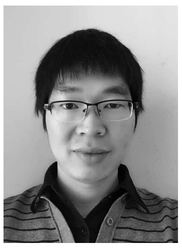
[33] Reuter M, Wolter F, Peinecke N. Laplace-Beltrami spectra as "Shape-DNA" of surfaces and solids. *Computer-Aided Design*, 2006, 38(4): 342-366.

[34] Liu Y, Prabhakaran B, Guo X. Point-based manifold harmonics. *IEEE Transaction on Visualization and Computer Graphics*, 2012, 18(10): 1693-1703.

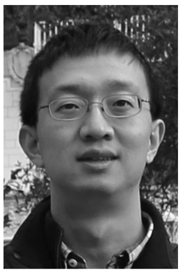
[35] Belkin M, Niyogi P. Towards a theoretical foundation for Laplacian-based manifold methods. In *Learning Theory*, Auer P, Meir R (eds.), Springer Berlin Heidelberg, 2005, pp.486-500.

[36] Belkin M, Sun J, Wang Y. Constructing Laplace operator from point clouds in R^d. In *Proc. the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 2009, pp.1031-1040.

[37] Tapp K. Matrix Groups for Undergraduates. American Mathematical Society, 2005.

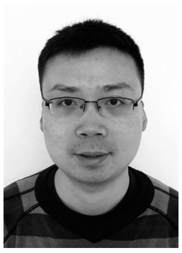


Chao Wang is a currently a Ph.D. candidate in the Department of Computer Science at University of Texas at Dallas. Before that, he received his M.S. degree in computer science in 2012, and B.S. degree in automation in 2009, both from Tsinghua University. His research interests include geometric modeling, spectral geometric analysis, and large-scale environment scanning and reconstruction.

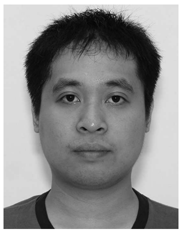


Yang Liu received his Ph.D. degree in computer science from University of Texas at Dallas in 2011. He is currently working at Facebook. His research interests are in computer graphics, spectral gemotric analysis and geometric modeling.

Xiaohu Guo received his Ph.D. degree in computer science from Stony Brook University in 2006. He is currently an associate professor of computer science at University of Texas at Dallas. His research interests include computer graphics and animation, with an emphasis on geometric modeling and processing, mesh generation, centroidal Voronoi tessellation, spectral geometric analysis, deformable models, 3D and 4D medical image analysis, etc. He received the prestigious National Science Foundation CAREER Award, United States, in 2012.



Zichun Zhong is a postdoctoral researcher in the Department of Radiation Oncology at UT Southwestern Medical Center at Dallas currently. He received his Ph.D. degree in computer science at University of Texas at Dallas in summer, 2014. Before that, He received his B.S. degree in computer science and technology (software engineering) and M.S. degree in computer science from University of Electronic Science and Technology of China (UESTC) in 2006 and 2009, respectively. His primary research interests include computer graphics, medical imaging processing, deformable image registration, image reconstruction, geometric modeling, computer animation, visualization, GPU algorithms, and high-performance computing.



Binh Le received his B.S. degree in computer science at the Vietnam National University in 2007, and Ph.D. degree in computer science at the University of Houston (UH) in 2014, where he worked under the supervision of Prof. Zhigang Deng at the UH Computer Graphics and Interactive Media Lab. His research interests include computer graphics, computer animation, virtual human modeling and animation, and computer vision. His work focuses on data-driven methods for character skinning and facial animation. He held a Vietnam Education Foundation's Fellowship from 2008 to 2010.

Zhigang Deng is currently an associate professor of computer science at the University of Houston, USA. His research interests include computer graphics, computer animation, virtual human modeling and animation, human computer inter action, and visual computing for biomedical applications. He received his Ph.D. degree in computer science at the University of Southern California in 2006, M.S. degree in computer science from Peking University in 2000, and B.S. degree in mathematics from Xiamen University in 1997. He is the recipient of a number of awards including ACM ICMI Ten Year Technical Impact Award, IEEE ICRA 2012 Best Medical Robotics Paper Award Runner-up, Google Faculty Research Award, Texas Norman Hackerman Advanced Research Award, University of Houston Teaching Excellence Award, NSFC Joint Research Fund for Overseas Chinese Scholars and Scholars in Hong Kong and Macao. His current research has been funded by the National Science Foundation of USA, National Institute of Health of USA, National Aeronautics and Space Administration (NASA), Texas NHARP Program, Google, Nokia, and other industry resources. Besides the CASA 2014 general co-chair and SCA 2015 general co-chair, currently he also serves as an associate editor of Computer Graphics Forum, and Computer Animation and Virtual Worlds Journal. He is a senior member of IEEE and a member of ACM.