






## Construction of Almost Everywhere $C^2$ Continuous Shape Interpolating Spaces Via Progressive Interpolation of Loop Subdivision

Linghan Xu<sup>1</sup> , Biyuan Yao<sup>1</sup> , Yongwei Nie<sup>1</sup>, Jinfeng Jiang<sup>1</sup> , Shihao Wu<sup>2</sup>, Guiqing Li<sup>1</sup>

<sup>1</sup>South China University of Technology, [csxulinghan@mail.scut.edu.cn](mailto:csxulinghan@mail.scut.edu.cn), [yaobiyuanyy@163.com](mailto:yaobiyuanyy@163.com), [nieyongwei@scut.edu.cn](mailto:nieyongwei@scut.edu.cn), [201821033459@mail.scut.edu.cn](mailto:201821033459@mail.scut.edu.cn)

<sup>2</sup>Swiss Federal Institute of Technology Zurich, [shihao.wu312@gmail.com](mailto:shihao.wu312@gmail.com)

Corresponding author: Guiqing Li, [ligq@scut.edu.cn](mailto:ligq@scut.edu.cn)

**Abstract.** Shape interpolation is an important methodology for 3D modeling in digital geometry processing. It has been widely applied in computer animation, geometric design, and so on. This paper proposes a framework for producing smooth interpolation sequences on a high dimensional manifold, which is spanned by given 3D keyframes represented as meshes with the same connectivity. Specifically, we convert each keyframe to a high dimensional vector stacked with edge lengths and dihedral angles and regard it as a point in the corresponding shape space. We then project these points (keyframes) onto a planar space and further triangulate the 2D projections into a mesh. A 2D manifold embedded in the shape space can be obtained by using Loop subdivision surface to interpolate the high-dimensional mesh which is constructed by transferring the connectivity of the 2D mesh onto the corresponding keyframes. For the convenience of modeling, we design an interactive interface which supports sampling  $C^2$  continuous animation sequences on the manifold by drawing cubic B-splines on its projected plane. Thorough experiments show that our approach is efficient and capable of producing high quality animation sequences, compared to the state-of-the-art 3D shape interpolation methods.

**Keywords:** 3D shape interpolation, edge length and dihedral angle, shape subdivision manifold, progressive interpolation, animation sequence

**DOI:** <https://doi.org/10.14733/cadaps.2021.486-501>

## 1 INTRODUCTION

Shape interpolation is an important method for 3D modeling in digital geometry processing, which has widely been used in computer animation and geometric design. Though shape interpolation has been investigated thoroughly, most approaches belong to linear interpolation, which usually extracts geometric quantities encoding the given shapes [34] or transformation deforming one shape to the other [2], and then compute the corresponding quantities of the intermediate shape by linear interpolation.

However, both aforementioned mechanisms use only two keyframes to create intermediate shapes. If a set of keyframes are given, they usually generate a whole sequence by interpolating between successive frames pair by pair. Such a shape curve only achieves  $C^0$  continuity at the keyframes which usually exhibit unsmooth transition artifacts. To address the issue, Heeren et al. [11] directly extended their shapes interpolation methods [12, 10] to construct geodesic paths passing through all given shapes. The improvement could produce pleasing animations, but is time-consuming due to involving global optimization. Huber et al. [15] introduced discrete cardinal splines and interpolatory subdivision curves in the above Riemann shell space to achieve smooth interpolation. This approach appears a little difficult for engineering applications due to its low efficiency. Xia et al. [33] directly employed a cubic spline to interpolate the control polygon of keyframes in the shape space defined by linear rotation-invariants. The approach greatly reduces the computation time while maintaining the smooth transition property. Nevertheless, all these methods only construct a shape curve and therefore do not sufficiently explore the shape space blended by given keyframes.

This paper proposes a framework to produce smooth interpolation sequences on a high dimensional manifold, which is spanned by given 3D keyframes represented as meshes. And these meshes share the same connectivity. Specifically, as shown in Fig. 1, we convert each keyframe to a high dimensional vector stacked with edge lengths and dihedral angles, and regard it as a point in the corresponding shape space. We then project these high dimensional points onto a planar space via Local Linear Embedding (LLE) [22] and further triangulate the projections into a 2D mesh base on Delaunay triangular. The connectivity of the 2D mesh is finally transferred to the high dimensional points to obtain a shape mesh in the shape space. A 2D manifold embedded in the shape space can thus be obtained by using Loop subdivision surface to interpolate the shape mesh. The loop surface constructed in the shape space is  $C^1$  continuous in the general case, and  $C^2$  continuous for regular meshes [19, 30]. In order to obtain the shape sequence, we draw a cubic B-spline curve on the parametric space of the surface and map it to the surface for sampling. When the curve does not pass through the vertexes of shape mesh, the curve constructed on the loop surface is theoretically  $C^2$ . If there is a actual control point of curve passing through the vertexes of shape mesh, the curve is  $C^1$  continuous. For the convenience of modeling, we design an interactive interface which supports sampling animation sequences on the manifold by drawing cubic B-splines on the corresponding projected plane. Our contributions include:

- A shape subdivision manifold is constructed to pass through the given set of keyframes (animation meshes) in a high dimensional shape space defined by the edge lengths and dihedral angles, which greatly enriches the optional animation sequences;
- Nonlinear dimensionality reduction of locally linear embedding is elaborately selected to perform the 2D manifold embedding, such a projected layout can capture the similarity among keyframes well.
- An interactive interface is designed to help navigate the shape manifold. By drawing a cubic B-spline freely within the 2D projected region, users can obtain their desired  $C^2$  continuous smooth mesh sequence.

## 2 RELATED WORK

Closely relevant to the central theme, we now briefly review previous approaches and their related applications in three categories: linear shape interpolation, geodesic path generation for two shapes, and geodesic path interpolating multiple keyframes.

### 2.1 Linear Shape Interpolation

At present, most shape interpolation methods belong to linear interpolation. The essence of shape interpolation is to find geometric or motion quantities whose linear change can drive the shape smoothly and naturally

deforming. Various techniques have been proposed, such as ARAP (As Rigid As Possible) [2] shape interpolation reconstructs the intermediate shape by linearly blending the identity matrix and the transformations from source triangles to the corresponding target triangles. Since then, the idea of ARAP algorithm has been used for reference by many subsequent researchers, such as [4], [26] and [17]. MeshIK [28] can be regarded as a variant of the ARAP interpolation, which linearly interpolates deformation gradients from source triangles to the corresponding target triangles to obtain intermediate shapes.

Multi-scale interpolation by Winkler et al. [32] blends the edge lengths and dihedral angles of given meshes and then use the multi-scale strategy to recover the intermediate shapes. Scheme [8] also reconstructed the intermediate pose by specifying the length and dihedral angle of its edges. And [3] defined isometry-invariant intrinsic coordinates (IICs) for each mesh edge, which consists of the length of the current edge, the exterior angle of between adjacent triangles, and the exterior angle between two edges. All these methods can be viewed as extensions of the approach in [25] and [24].

In addition, there are methods based on extrinsic attributes based methods, such as [32], [14], [21] and [9].

However, all these methods only use two keyframes to create the intermediate shapes between them and therefore results in piecewise smooth interpolation sequences. Namely, the animation sequence generated by multiple keyframes usually exhibits the artifact of unsmooth shape transition near a internal keyframe.

## 2.2 Geodesic Paths Interpolating Two Shapes

The idea of using geodesic for navigating shape space is recent. Kilian et al. [16] accounted for 3D shapes as points of a Riemannian shape space equipped with an isometric inner-product. Shape interpolation is then cast to find a geodesic path in the shape space. Meshless modeling [1] represents a shape by blending a set of nodes sampled from the volume of the shape. Keyframe interpolation is then cast into a constrained minimization on the trajectories of the nodes. Heeren et al. [12] found a geodesic path by modeling time-discrete geodesic paths of shape sequences in the space of shells instead. Both methods result in a large-scale nonlinear optimization problem and cannot cope with the large-scale deformation problem due to linear initialization.

By viewing the interpolation as a motion problem, AIAP (As Isometric As Possible) [18] introduced an as-isometric-as-possible framework which is derived by minimizing the total variation of edge lengths of all frames. Zhang et al. [35] strive to find the solution using only one (edge) anchor for each interpolation example. Recently, NNwarp [20] proposed a neural network-based nonlinear deformation method.

Interpolating geodesic paths between two shapes still suffers from unsmooth shape transition. In addition, this kind of algorithms usually involves global optimization and therefore is computationally inefficient.

## 2.3 Geodesic Paths Passing Through Multiple Keyframes

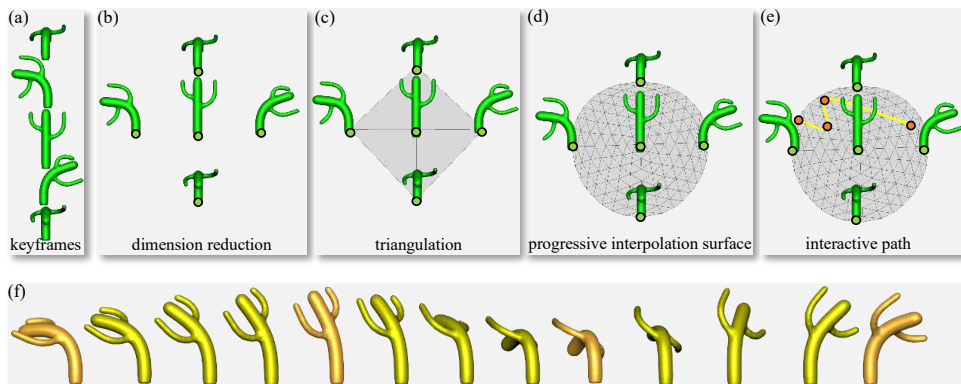
The shape curve of interpolating two keyframes mentioned above obviously only achieves  $C^0$  continuity at the keyframes. This will lead to an unsmooth transition. To address the above issue, Heeren et al. [11] directly extended their two shapes interpolation method to construct geodesic paths passing through all given shapes. The improvement may produce pleasing animations, but is time-consuming due to involving global optimization. Huber et al. [15] introduced discrete cardinal splines and interpolated subdivision curves in the above Riemann shell space to achieve smooth interpolation. This approach appears a little difficult for engineering applications and seems not efficient enough considering that shapes are represented with Loop subdivision surfaces. Xia et al. [33] directly employed a cubic spline to interpolate the control polygon of keyframes in the shape space defined by their linear rotation-invariants.

All these methods focus on constructing a smooth shape sequence interpolating multiple keyframes, and did not exploit the possible shape space spanned by the given keyframes. In view of this, our method constructs a shape subdivision surface defined by the keyframes to generate more animation sequences. By interactively

selecting keyframes on the shape subdivision surface, users can obtain a huge number of animation sequences. This undoubtedly helps users to realize their design intention more easily.

### 3 OVERVIEW

We first introduce the notations. Assume we are given  $n$  triangular meshes  $\{M_1, M_2, \dots, M_n\}$  as keyframes, where  $n$  is not less than three and all  $M_i = (V, E, F)$  share the same connectivity. Let  $|V|$ ,  $|E|$ , and  $|F|$  be the number of vertices, edges and faces respectively. Furthermore, let  $\Phi_i = (\Theta_i, L_i)$  be the  $2|E|$  dimension vector, where  $\Theta_i$  is a vector stacked with  $|E|$  dihedral angles of edges and  $L_i$  is a vector filled with  $|E|$  edge lengths of triangles. We then view each keyframe as a point in  $\mathbb{R}^{2|E|}$  and denote the set of key vectors by  $\mathcal{D} = \{\Phi_1, \Phi_2, \dots, \Phi_n\}$ . Our shape interpolation is based on subdivision surfaces and consists of four steps: dimension reduction, triangulation manifold, construction of interpolatory Loop subdivision surface in the shape space, and shape sequence reconstruction as shown in Fig. 1. A brief explanation for each step is sketched respectively in the following and more technical details will be discussed in Section 4.



**Figure 1:** Pipeline: (a) a set of keyframes  $\{M_1, M_2, \dots, M_n\}$ ; (b) dimension reduction of high dimensional points (keyframes) in the shape space; (c) triangulation of the 2D points using Delaunay algorithm; (d) progressive interpolatory Loop subdivision surface; (e) a cubic B-spline interactive path (yellow) generated by user-selected points (orange) on plane; (f) a shape sequence obtained by back-projection onto the shape manifold.

**Step 1: Dimensionality reduction.** Given a set of keyframes as shown in Fig. 1(a), each of which is viewed as a point in a high-dimensional shape space, we want to construct a shape space expanded by them. Our idea is to create a control mesh with the keyframes as vertices and then generate an shape interpolatory subdivision surface. In addition, adjacent vertices of the control mesh should have similar shape. Finally, once the shape space has been created, we hope that it is easy for users to navigate interactively. Based on these considerations, we project the keyframes onto a 2D space using LLE [22] which is able to effectively learn the structured features of the dataset and to preserve them in the embedded space. As an example, Fig. 1(b) illustrates the projection of the shapes in Fig. 1(a). For the sake of description, we denote the projected point set of  $\mathcal{D}$  by  $d = \{\phi_1, \phi_2, \dots, \phi_n\}$  with  $\phi_i \in \mathbb{R}^2$  be the projection of  $\Phi_i$ .

**Step 2: Triangulation in the shape space.** After obtaining the 2D projection  $d$  of the keyframes, we triangulate the projected points into a planar triangular mesh using Delaunay triangulation algorithm, which we denote by  $M_d$  as shown in Fig. 1(c). The connectivity of  $M_d$  is then mapped onto the corresponding high-dimensional shape points in  $\mathcal{D}$  to obtain a mesh  $M_{\mathcal{D}}$  in the shape space. The vertices of  $M_{\mathcal{D}}$  are the corresponding initial keyframes.

**Step 3: Progressive interpolation Loop subdivision surfaces in shape space.** This step constructs a shape subdivision manifold  $M_{\overline{\mathcal{D}}}$  interpolating the vertices (keyframes) of  $M_{\mathcal{D}}$  based on a variant of Loop subdivision described in [5], which is the main goal of this paper. The shape manifold is almost everywhere  $C^2$  according to [7]. As we investigate shape interpolation, therefore require the shape subdivision manifold passing through the keyframes. Noting that Loop subdivision surfaces are approximation, we employ the progressive interpolation algorithm [5, 6] to generate an interpolatory one. Fig. 1(d) depicts the projected interpolatory subdivision surface of the keyframe set in Fig. 1(a). We can simply regard the interpolatory subdivision surface of  $M_d$ , denoted by  $M_{\overline{d}}$ , as the projection of  $M_{\overline{\mathcal{D}}}$ .

**Step 4: Reconstruction of shape sequences.** Obviously, a curve on  $M_{\overline{\mathcal{D}}}$  will correspond to an animated sequence. For the convenience of modeling, we design an interactive interface to support sampling animation sequences on  $M_{\overline{\mathcal{D}}}$  by drawing curves on its projected subdivision surface  $M_{\overline{d}}$ . The user actually only needs to specify a few points sequentially (see Fig. 1(e) for example). Our system will generate a cubic B-spline passing through these points. Uniformly sampling a set of points on the curve and then mapping them onto  $M_{\overline{\mathcal{D}}}$  will lead to a sequence of vectors of edge lengths and dihedral angles (shape points). Converting the vectors into shape meshes finally yields a smooth sequence of 3D shapes as shown in Fig. 1(f).

## 4 METHODS

This section will describe some important details according to the pipeline mentioned in the overview. Firstly, evaluating the most suitable dimensionality reduction approach to project keyframes onto a 2D plane. Then, progressively interpolating the Loop surface is discussed. And finally, explain how to reconstruct animation sequence according to the spline path on the subdivision surfaces.

### 4.1 Dimension Reduction of Shape Space

A lot of methods have been proposed to reduce the dimensionality of high dimensional datasets. For example, Isometric feature mapping (Isomap) [29] sustains the geodesic distance between samples in the dataset. LLE [22] assumes that local regions of a high-dimensional manifold are nearly planar and therefore a sample can be linearly blended by its neighbors. The linear relationship is maintained in the process of dimension reduction. Local Tangent Space Arrangement (LTSA) [36] approximates the local geometry of the high-dimensional manifold with tangent spaces learned by fitting an affine subspace to the neighborhood of each sample in the dataset.

We choose one from the aforementioned three algorithms by using them to project some shape sequences onto a plane and observing whether the similarity of two shapes and the distance between their projections are compatible. The results, discussed in Section 5.1, show that LLE is most suitable for our setting, therefore we briefly describe here how to adapt the universal LLE to our scenarios. Specifically, for each sample  $\Phi_i \in \mathcal{D}$  and the set of its neighbors  $\{\Phi_{i_j}, j = 1, 2, \dots, k\}$ , LLE calculates the blending weights  $w_{ii_j}$  between  $\Phi_i$  and  $\Phi_{i_j}$  by minimizing the following reconstruction errors

$$\mathcal{E}(W) = \sum_{i=1}^n \left\| \Phi_i - \sum_{1 \leq j \leq k} w_{ii_j} \Phi_{i_j} \right\|_2^2, \text{ s.t. } \sum_{1 \leq j \leq k} w_{ii_j} = 1 \quad (1)$$

We call  $W = (w_{ij})_{n \times k}$  the reconstruction weight matrix in which entry  $w_{ij} = 0$  if  $j$  is not a neighbor of  $i$ .  $W$  reflects the local geometric properties between each sample and its neighbors. It is worth noting that these weights remain unchanged under affine transformations such as rotation, scaling and translation [22, 23]. With  $W$  known, we then find the projections by enforcing them to meet the same structure:

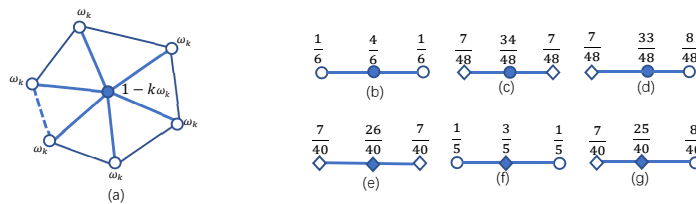
$$\arg \min_{\{\phi\}} \sum_{i=1}^n \left\| \phi_i - \sum_{1 \leq j \leq k} w_{ij} \phi_{i_j} \right\|_2^2 \quad (2)$$

### 4.2 Shape Interpolation Subdivision Surfaces

Given  $n$  discrete high dimensional mesh  $\mathcal{D}$  as keyframes, this section describes a pose manifold  $M_{\mathcal{D}}$  spanned by keyframes via progressively interpolate the triangulation of  $\mathcal{D}$  base on Loop subdivision [5, 6]. Specifically, we use the following formula to make the pose manifold  $M_{\mathcal{D}}$  iteratively approximate the triangulation of  $\mathcal{D}$ .

$$\Phi_i^{(t)} = \Phi_i^{(t-1)} + (\Phi_i - \Phi_i^{(t-1, \infty)}), t = 1, 2, \dots \tag{3}$$

where  $\Phi^{(t, \infty)} = (\Phi_1^{(t, \infty)}, \Phi_2^{(t, \infty)}, \dots, \Phi_n^{(t, \infty)})$  are the limit positions of the subdivision surface with control vertices  $\Phi^{(t)} = (\Phi_1^{(t)}, \Phi_2^{(t)}, \dots, \Phi_n^{(t)})$ , and  $\Phi_i^{(0)} = \Phi_i$ . Iterate  $k$  times to get a control mesh  $\Phi^{(k)}$  and the corresponding limit mesh  $\Phi^{(k, \infty)}$ . When  $k$  tends to infinity, the distance between  $\Phi^{(0)}$  and  $\Phi^{(k, \infty)}$  decreases to zero. It is easy to extend the proof in [5] to show the convergence of the iterative process in Equation 3. In all our experiments we set  $k = 5$ , since under such a condition, the difference between  $\Phi^{(0)}$  and  $\Phi^{(k, \infty)}$  becomes very small. Fig. 2 illustrates the masks of limit position [13] applied in the paper. Fig. 2 (a) depicts the limit mask of an interior vertex while Fig. 2 (b)~(g) illustrate limit masks of total six different boundary configurations. A boundary vertex is called a regular vertex if its valence is four. Otherwise, it is called an extraordinary vertex.



**Figure 2:** The masks of limit position mask of Loop subdivision for (a) interior vertices with  $k$  neighbors, where  $\omega_k = 3[11 - 8(\frac{3}{8} + (\frac{3}{8} + (\frac{1}{4}) \cos(\frac{2\pi}{k}))^2)]^{-1}$ , and (b)~(g) are different kinds of boundary vertices, where the solid shape indicates the vertex to be evaluated, circular points are regular vertices and rhombus points are irregular vertices.

### 4.3 Reconstruction of Shape Sequences

The shape sequence reconstruction mainly consists of two parts. First, we need to get a sequence of sample points on the 2D plane. For this purpose, we design an interactive interface for users to select points and generate a cubic B-spline curve through these points, and then sample the points on the curve. The second part finds the corresponding vector of edge lengths and dihedral angles in the shape space based on these 2D sampling points, and then converts the vector into the vertex coordinates to recover the mesh.

#### 4.3.1 Interactive Interface and Curve Creation

For the convenience of modeling, we design an interactive interface to support sampling animation sequences on the manifold. Firstly, the user is required to select a few points  $\{d_i\}_{i=0}^n$  on our 2D interface as actual control points. These actual control points have corresponding meshes in high dimensional space. We regard these corresponding meshes as user-selected meshes, which will be interpolated. On the 2D plane, after selecting points by users, the system then automatically generates a cubic B-spline curve. Specifically, the cubic B-spline

passing through  $n + 1$  user-selected points  $\{p_i, i = 0, 1, \dots, n\}$  is

$$L_i(t) = \frac{1}{6} (1 - 3t + 3t^2 - t^3) q_i + \frac{1}{6} (4 - 6t^2 + 3t^3) q_{i+1} + \frac{1}{6} (1 + 3t + 3t^2 - 3t^3) q_{i+2} + \frac{1}{6} t^3 q_{i+3} \quad (4)$$

Where  $q_i$  is the theoretical control point of cubic B-spline curve and  $p_i$  is the actual control point selected by the user on the subdivision surface. By introducing  $L_i(0) = p_i$  into Equation 4, we can get  $n - 2$  equations with  $n$  unknown quantity. In addition, there are two boundary conditions  $\dot{L}_0(0) = d_0, \dot{L}_{n-3}(0) = d_{n-3}$  that the default is the natural boundary condition  $d_0 = 0, d_{n-3} = 0$ . Then the theoretical control points of the curve can be obtained, and finally the cubic B-spline curve passing through the user-selected points can be obtained. At the same time, the corresponding cubic B-spline curve will also be generated in the shape space, which passes through all the user-selected meshes.

Our interface supports modifying the spline by adjusting control points. Here, adjusting the control points includes deleting or adding the last one of the currently selected points, or adding a new control point onto the generated curve. Our system supports two editing modes: points selection and spline sampling. Switching the mode from points selection to spline sampling, we can then uniformly discretize the curve to obtain a sequence of 2D points. Accordingly, a series of parameters, edge lengths and dihedral angles are obtained by cubic B-spline in the high dimensional space  $M_{\mathcal{S}}$ . These parameters are used to reconstruct the 3D mesh sequence, which passes through the user-selected meshes.

### 4.3.2 Mesh Reconstruction

Given a point  $m$  on the 2D spline, our shape reconstruction procedure first maps  $m$  back to a point on the shape subdivision surface  $M_{\mathcal{S}}$ , which is a high-dimensional vector of edge lengths and dihedral angles. Mesh  $M$  can then be reconstructed from this vector by using the algorithm described in [31]. Repeat the same operation on all 2D sampled points, we can obtain a smooth sequence of animated meshes.

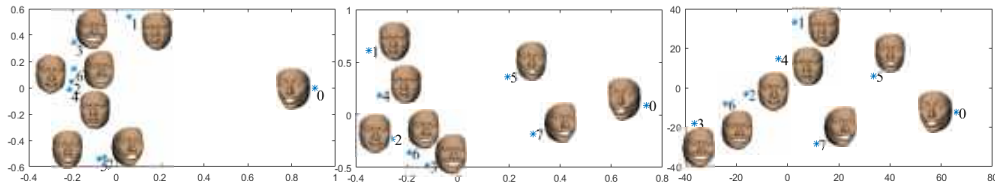
## 5 EXPERIMENTAL RESULTS

We implemented the proposed approach and an interactive interface using C/C++ on a PC with Intel (R) Core (TM) i7-2600 CPU @ 3.4GHz. This section will present a variety of experiments to verify the performance of the proposed framework. Before presenting shape interpolation results, we first conduct experiments to show why we choose LLE in Section 4.1 to project our shapes onto the plane (Section 5.1), and then briefly analyze the impact of the element number of KNN (K-nearest neighbors) using in the shape projection (Section 5.1). Next, a variety of examples are presented to show the ability of our approach in modeling animation sequences, and finally our method is compared with the two closely related approaches [11, 33] from three aspects: interpolation of keyframes with large deformation, smoothness of edge length variation, and time efficiency (Sections 5.3 and 5.4).

### 5.1 Selection of the Projection Approaches

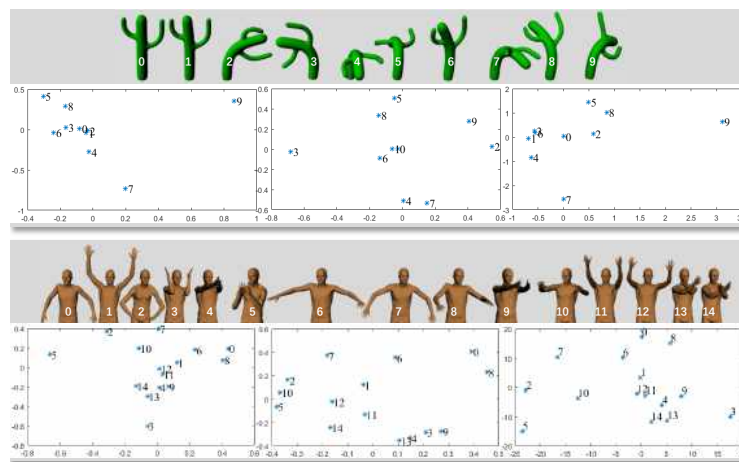
We first perform three reduction algorithms, LTSA, LLE, and Isomap, on a set of eight facial keyframes, as shown in Fig. 3. Intuitively, the result of LLE is most compatible with the similarity among frames because the more similar the expressions are, the closer their projections.

The top row of Fig. 4 is an even more telling example in which keyframes include rest (0 and 1), bending to right (2, 9), bending to left (3), bending backward (5, 8), and bending forward (6,7) poses. The layout of projections by LLE almost perfectly reflects the bending orientations. The bottom row of Fig. 4 presents the projections of 15 keyframes of human arm pose which also show the result of LLE is most reasonable among three.



**Figure 3:** Projections of eight facial expressions by LTSA (left), LLE (middle) and Isomap (right) respectively. Stars (\*) indicate the position of the projections and numbers index the keyframes.

Note that we do not need to care about coordinate scales in all these examples but only concern the relationship among keyframes. Taking these experiments into account, we finally use LLE to perform dimension reduction in our system.

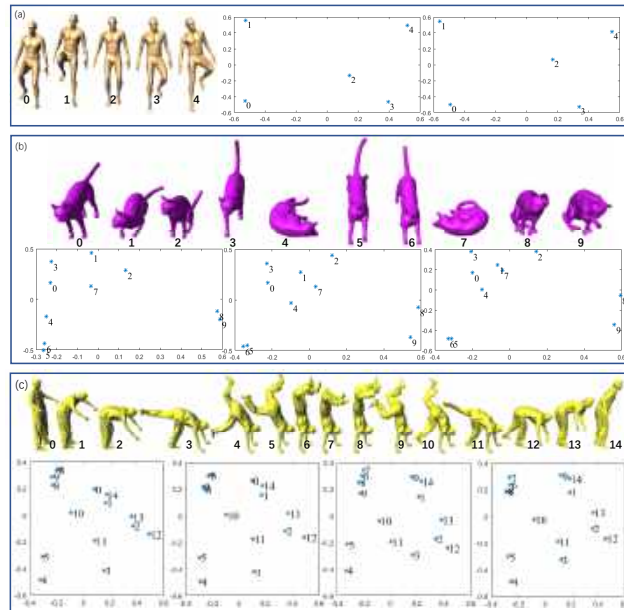


**Figure 4:** Projections of 10 key cactus poses (top row) and 15 key human poses (bottom row) by LTSA (left), LLE (middle) and Isomap (right) respectively. In each row, keyframes marked a serial number are first shown and their projections are depicted with stars (\*) within the corresponding projected space.

## 5.2 The Impact of KNN Size on the Projections

We use three examples to analyze the impact of the KNN size on the LLE projected layout. Let  $n$  be the number of given keyframes and  $k$  be the element number of KNN. Considering that  $n$  used for shape interpolation is usually small in practice, we alter  $k$  within an interval around  $\lceil \frac{n}{2} \rceil$ . Fig. 5 (a) depicts an example with  $n = 5$  in which  $k = 3$  and 4 are tested. Two rightmost rectangular images of this figure show that the layout of the two cases share a very similar structure. Two more examples are presented in Fig. 5 (b) ( $n = 10$ ,  $k = 3, 5, 7$ ) and (c) ( $n = 15$ ,  $k = 5, 6, 7, 8$ ), respectively. Sufficient experimental results manifest that when the size  $k$  of KNN neighborhood is near half of the number of keyframes, the similarity between keyframes can be captured well. This motivates us to set  $k = \lceil \frac{n}{2} \rceil$  during the dimensionality reduction stage in all experiments.





**Figure 5:** The impact of the KNN neighborhood size on the structure of LLE projections: (a) left:  $n = 5$  human key poses; right: the two LLE projections for  $k = 3$  and  $4$  respectively; (b) upper:  $n = 10$  cat poses; lower: projections for  $k = 3, 5$  and  $7$ , respectively; (c) upper:  $n = 15$  human poses; lower: projections for  $k = 5, 6, 7$ , and  $8$  respectively.

### 5.3 Visual Results

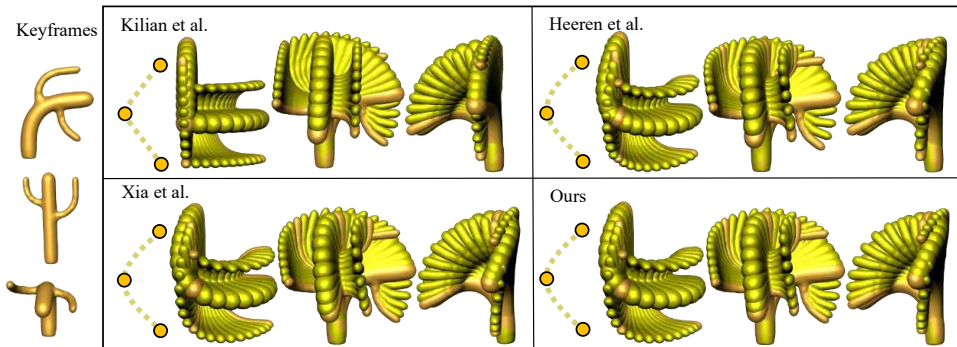
This section presents some examples to illustrate the visual effect of interpolating sequences by our approach. We first show some sequences by drawing different paths on the projection domains of two sets of keyframes and then present two examples with large deformation between adjacent keyframes.

#### 5.3.1 Smoothness of Sequences

Fig. 6 and 7 depict the frame transition smoothness of sequences by Kilian et al. [16], Heeren et al. [11], Xia et al. [33] and our approach via stacking up the frames together. Except for the result of the geodesic interpolation approach [16], the sequences of the latter three methods look visually smooth enough. We will further present quantitative analysis in the next subsection about this performance.

#### 5.3.2 Sampling Arbitrary Sequences on Subdivision Manifolds

Compared to previous methods that create only a clip of animation, our approach is able to generate infinite sequences theoretically. To generate a sequence, we first need to define the sequence path. The path is a cubic B-spline curve whose actual control points come from the frames selected by the user. Fig. 8 shows two examples on the same shape subdivision manifold interpolating five keyframes of an elephant, the orange points in shape subdivision manifold indicate the selected-frames by users, and the numbers near the points indicate the sequence of the selected-frames with 0 is the first selected-frame. The generated curve is shown by the yellow line. The results show that drawing different paths on the projected domain of the subdivision

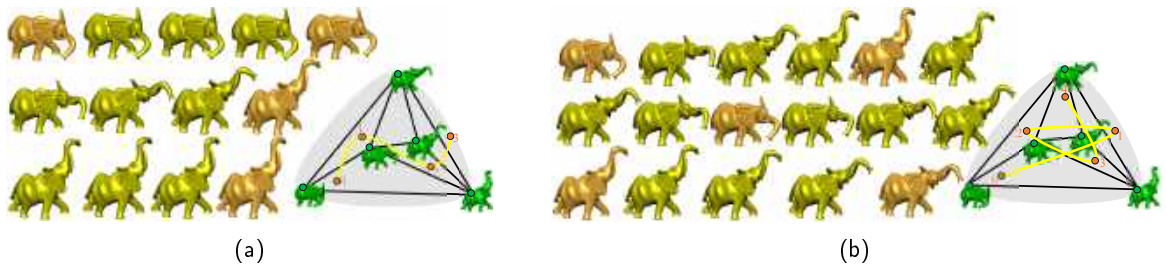


**Figure 6:** Comparison of the shape completion sequences by Kilian et al. [16], Heeren et al. [11], Xia et al. [33] and our approach, respectively. The input keyframes are shown in the leftmost column. For each approach, top, front and right views are illustrated. Key and intermediate frames are separately visualized in orange and yellow. Our method is smoother than [16], and comparable to [11] and [33] on the visual effect.

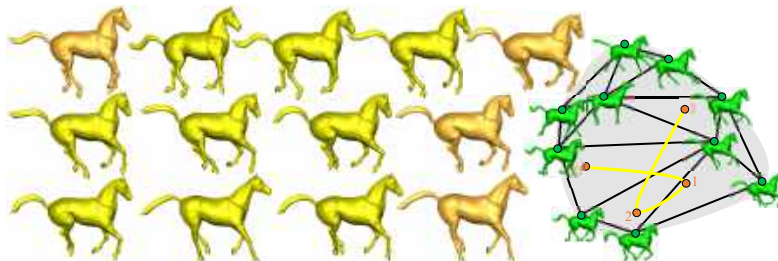


**Figure 7:** Comparison of the shape completion sequences by [16], [11], [33] and our approach, respectively show in the right four columns. The input keyframes are shown in the left three columns. Key and intermediate frames are separately visualized in orange and yellow.

manifold can yield completely different shape sequences. Users can use any number of frames with any pose to get the desired model sequence. Fig. 9 depicts another example with more (ten) keyframes.



**Figure 8:** Arbitrary shape interpolation sequence of elephant model. (a): Interpolating four selected frames (orange) on the shape manifold blended by five keyframes (right), where three intermediate frames (yellow) are generated between each two adjacent selected frames. (b): A more complex interpolation sequence path with several intersections.



**Figure 9:** Shape sequence by specifying four actual control points on the shape manifold blended by ten keyframes.

### 5.3.3 Sampling Keyframes with Large Deformation

In this section, we sample and display some interpolation sequences with large deformation between adjacent keyframes. Fig. 10 and 11 show the sequence of large deformation shapes generated from five helix shapes and three human dance postures respectively, some of which rotate more than  $180^\circ$  or even  $360^\circ$ .



**Figure 10:** Sequence of large deformation shape generated from five helix shapes. Two frames (yellow) are inserted between two successive keyframes (orange).

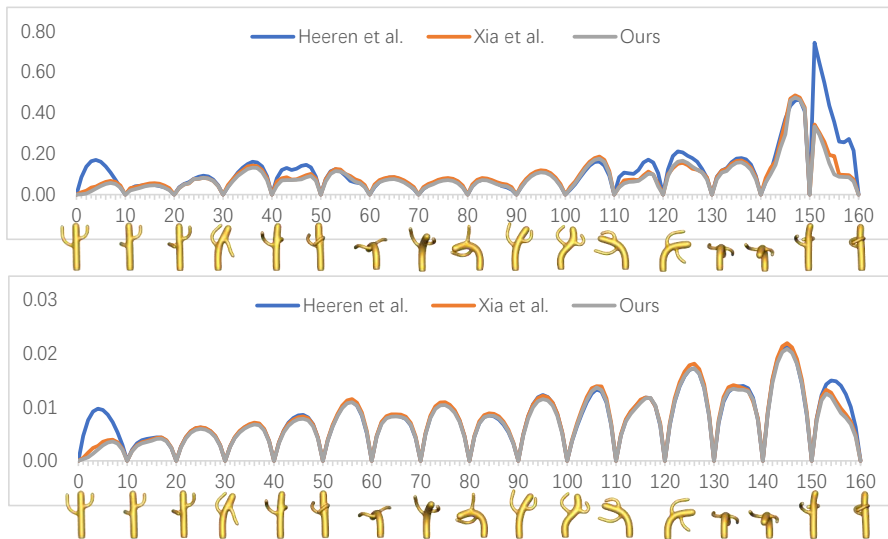


**Figure 11:** Sequence of large deformation shape generated from three human dancing poses. Four intermediate frames (yellow) are created between each pair of adjacent keyframes (orange).

## 5.4 Numerical Comparison

### 5.4.1 Variation of Edge Length

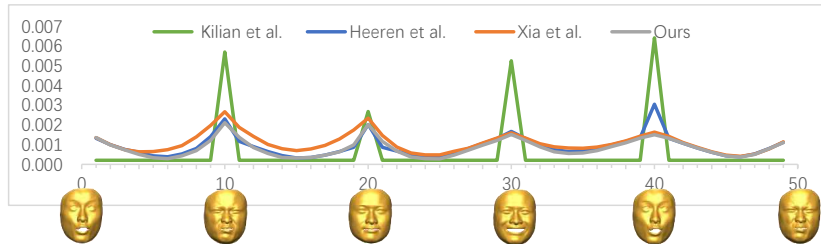
In shape interpolation, not only semantic parts of the motion subject are desirable to move smoothly but also geometric primitives such as vertices and edges in mesh frames are expected to change without extra fluctuation. Here we measure the variation of edge across the generated sequence with respect to edge length obtained by linearly interpolating edge lengths of adjacent keyframes. As a comparison, we also do the same analysis on the approaches in Heeren et al. [11] and Xia et al. [33]. The top row in Fig. 12 shows the maximal deviation in each generated frame while the bottom row illustrates the average deviation of edge lengths for each generated frame. Both images demonstrate that our approach is slightly better than the two previous methods in two cases.



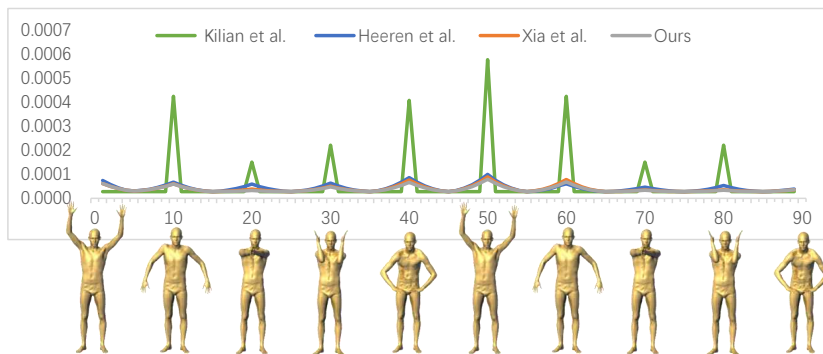
**Figure 12:** Variation of edge length across the generated sequence with respect to linear change: maximum (top) and average (bottom) are estimated for the approaches in Heeren et al. [11], Xia et al. [33] and this paper. A set of cactus models is used in this experiment and 17 fixed keyframe poses are shown in gray. The horizontal axis represents frame indices and the vertical axis stands for edge length deviation.

### 5.4.2 Smoothness of Shape Sequence

We examine the smoothness of sequences created by our method in Fig. 13 and 14. As a comparison, we also present the results of the isometric shape space [16], the splines in shell space [11] and cubic spline interpolation [33]. Here, the smoothness is measured by the second-order difference (curvature) on the vertex trajectories of the reconstructed sequence. The smaller the curvature, the smoother the trajectory. Both examples demonstrate that our method achieves the best performance among the four approaches.



**Figure 13:** Smoothness comparison on face expression sequences by the isometric shape space [16] (Kilian et al.), the splines in shell space [11] (Heeren et al.), cubic spline interpolation [33] (Xia et al.) and our methods.



**Figure 14:** Smoothness comparison on body pose sequences by the isometric shape space [16] (Kilian et al.), the splines in shell space [11] (Heeren et al.), cubic spline interpolation [33] (Xia et al.) and our methods.

### 5.4.3 Timings

Table 1 lists timings by performing three methods on four sets of keyframes. Generate an animation curve with  $K$  frames with each frame having  $|V|$  vertices, and  $\gamma$  indicates the vertex percent after decimation;  $t_{ir}$  (interpolation and reconstruction on decimated key frames),  $t_{ts}$  (detail transfer),  $t_t$  (the total runtime),  $t_o$  (interpolate one shape without using ghost). It can be seen from the three  $t_o$  columns that our approach is much faster than the two approaches from [11] and [33], where  $t_o$  indicates the time for yielding a frame on the original data. In order to improve the efficiency, both [11] and [33] conducted their shape interpolation on the simplified keyframes to obtain a sequence of coarse animated meshes and then use deformation transfer [8, 27] to yield the final detailed animated meshes. We also implemented our method with the simplifying strategy. In this case, our approach is slower than that of [33] but still faster than that of [11]. Nevertheless, the first three data rows of Table 1 demonstrate that the running time of the approach in [33] increases faster

than that of our approach with the number of mesh vertices growing, and exceeds that of our approach in some moment. So to sum up, our approach is quite efficient.

**Table 1:** Timings for three approaches (Unit: second).

Model ( $ V , K, \gamma$ )	Heeren et al.				Xia et al.				Ours			
	$t_{ir}$	$t_{ts}$	$t_t$	$t_o$	$t_{ir}$	$t_{ts}$	$t_t$	$t_o$	$t_{ir}$	$t_{ts}$	$t_t$	$t_o$
Cactus (5.2k, 20, 5%)	1.7	1.7	3.4	83	0.16	1	<b>1.16</b>	4.35	0.43	0.9	1.33	<b>0.72</b>
Cactus (5.2k, 20, 30%)	–	–	–	–	1.79	2.15	<b>3.94</b>	4.35	2.86	1.94	4.8	<b>0.72</b>
Cactus (5.2k, 20, 60%)	–	–	–	–	7.57	54.51	<b>62.08</b>	4.35	7.24	49.06	56.3	<b>0.72</b>
Cactus (5.2k, 170, 5%)	22	14	36	762	0.31	3.2	<b>3.51</b>	23.9	4.41	2.86	7.27	<b>0.58</b>
Horse (8.5k, 40, 10%)	22	5	27	238	0.35	2.2	<b>2.55</b>	11.4	1.73	1.93	3.66	<b>0.87</b>
Armadillo (166k, 50, 0.6%)	80	200	280	–	1.34	47.2	<b>48.54</b>	258.6	3.56	41.53	45.09	<b>1.56</b>

## 6 CONCLUSIONS

A shape subdivision manifold is proposed for shape sequence generation from a set of keyframes. It is suitable for triangular meshes with the same number of vertices and connection relations. By progressive interpolating the loop subdivision surface of keyframes, the shape subdivision manifold about the keyframe collection is constructed, and the cubic B-spline curve which throughs all the selected-frames pointed by users is generated by using the interactive system we designed, so as to realize user to obtain arbitrary  $C^2$  continuous and high-quality 3D mesh interpolation sequence quickly. Specifically, we view a keyframe as a point in the high dimensional shape space, which is composed of its edge lengths and dihedral angles, and constructs the control mesh with keyframes as vertices by projecting the keyframes onto a planar space using Delaunay triangulation. The triangulation is then embedded back to the high dimensional space and a subdivision manifold is therefore obtained by conducting the Loop progressive interpolation on embedded mesh. To navigate on the manifold, we design a concise interface to sample animation sequences by interactively drawing splines on the corresponding 2D projection.

Experiments are intended to evaluate the performance of the proposed approach. Compared to the state-of-the-art approaches, our method defines a 2D manifold interpolating given keyframes instead of a 1D curve, and therefore greatly extends shape space. The generated sequences by our method have better smoothness, higher mesh quality and higher efficiency for complicated models.

As a future work, the generated sequences can be optimized by the AIAP shape interpolation described in [35] in order to further reduce the variation scale of edge lengths and dihedral angles. It is also possible to enhance real-time performance based on GPU acceleration. In the current implementation, the triangulation of the LLE projections has not been optimized. It is also worthwhile to clarify whether we can gain more bonuses from interactively modifying the position of projections.

## ORCID

Linghan Xu <http://orcid.org/0000-0002-7708-8309>

Biyuan Yao <http://orcid.org/0000-0002-8862-1826>

Jinfeng Jiang <http://orcid.org/0000-0003-1341-1119>

## REFERENCES

- [1] Adams, B.; Ovsjanikov, M.; Wand, M.; Seidel, H.P.; Guibas, L.J.: Meshless modeling of deformable shapes and their motion. In Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 77–86. Eurographics Association, 2008.

- [2] Alexa, M.; Cohen-Or, D.; Levin, D.: As-rigid-as-possible shape interpolation. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 157–164. ACM Press/Addison-Wesley Publishing Co., 2000.
- [3] Baek, S.Y.; Lim, J.; Lee, K.: Isometric shape interpolation. *Computers & Graphics*, 46, 257–263, 2015.
- [4] Baxter, W.; Barla, P.; Anjyo, K.i.: Rigid shape interpolation using normal equations. In Proceedings of the 6th international symposium on Non-photorealistic animation and rendering, 59–64. ACM, 2008.
- [5] Cheng, F.H.F.; Fan, F.T.; Lai, S.H.; Huang, C.L.; Wang, J.X.; Yong, J.H.: Loop subdivision surface based progressive interpolation. *Journal of Computer Science and Technology*, 24(1), 39–46, 2009.
- [6] Deng, C.; Ma, W.: Weighted progressive interpolation of loop subdivision surfaces. *Computer-Aided Design*, 44(5), 424–431, 2012.
- [7] Dyn, N.; Gregory, J.A.; Levin, D.: Analysis of uniform binary subdivision schemes for curve design. *Constructive Approximation*, 7(1), 127–147, 1991.
- [8] Fröhlich, S.; Botsch, M.: Example-driven deformations based on discrete shells. In *Computer graphics forum*, vol. 30, 2246–2257. Wiley Online Library, 2011.
- [9] Gao, L.; Lai, Y.K.; Huang, Q.X.; Hu, S.M.: A data-driven approach to realistic shape morphing. In *Computer graphics forum*, vol. 32, 449–457. Wiley Online Library, 2013.
- [10] Heeren, B.; Rumpf, M.; Schröder, P.; Wardetzky, M.; Wirth, B.: Exploring the geometry of the space of shells. In *Computer Graphics Forum*, vol. 33, 247–256. Wiley Online Library, 2014.
- [11] Heeren, B.; Rumpf, M.; Schröder, P.; Wardetzky, M.; Wirth, B.: Splines in the space of shells. In *Computer Graphics Forum*, vol. 35, 111–120. Wiley Online Library, 2016.
- [12] Heeren, B.; Rumpf, M.; Wardetzky, M.; Wirth, B.: Time-discrete geodesics in the space of shells. In *Computer Graphics Forum*, vol. 31, 1755–1764. Wiley Online Library, 2012.
- [13] Hoppe, H.; DeRose, T.; Duchamp, T.; Halstead, M.; Jin, H.; McDonald, J.; Schweitzer, J.; Stuetzle, W.: Piecewise smooth surface reconstruction. In Proceedings of the 21st annual conference on Computer graphics and interactive techniques, 295–302. ACM, 1994.
- [14] Huang, J.; Tong, Y.; Zhou, K.; Bao, H.; Desbrun, M.: Interactive shape interpolation through controllable dynamic deformation. *IEEE Transactions on Visualization and Computer Graphics*, 17(7), 983–992, 2011.
- [15] Huber, P.; Perl, R.; Rumpf, M.: Smooth interpolation of key frames in a riemannian shell space. *Computer Aided Geometric Design*, 52, 313–328, 2017.
- [16] Kilian, M.; Mitra, N.J.; Pottmann, H.: Geometric modeling in shape space. In *ACM Transactions on Graphics (TOG)*, vol. 26, 64. ACM, 2007.
- [17] Levi, Z.; Gotsman, C.: Smooth rotation enhanced as-rigid-as-possible mesh animation. 21(2), 264–277, 2014.
- [18] Li, G.; Yang, L.; Wu, S.; Tan, W.; Chen, X.; Xian, C.: Planar shape interpolation using relative velocity fields. *Computers & Graphics*, 37(5), 364–375, 2013.
- [19] Loop, C.: Smooth subdivision surfaces based on triangles. Master's Thesis, University of Utah, Department of Mathematics, 1987.
- [20] Luo, R.; Shao, T.; Wang, H.; Xu, W.; Chen, X.; Zhou, K.; Yang, Y.: Nnwrap: Neural network-based nonlinear deformation. *IEEE Transactions on Visualization and Computer Graphics*, 26(4), 1745–1759, 2020.
- [21] Martin, S.; Thomaszewski, B.; Grinspun, E.; Gross, M.: Example-based elastic materials. In *ACM Transactions on Graphics (TOG)*, vol. 30, 72. ACM, 2011.
- [22] Roweis, S.T.; Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500), 2323–2326, 2000.

- [23] Saul, L.K.: Think globally, fit locally: Unsupervised learning of nonlinear manifolds. *Journal of Machine Learning Research*, 4, 2002.
- [24] Sederberg, T.W.; Greenwood, E.: A physically based approach to 2-d shape blending. In *ACM SIGGRAPH computer graphics*, vol. 26, 25–34. ACM, 1992.
- [25] Sederberg, T.W.; Parry, S.R.: Free-form deformation of solid geometric models. *ACM SIGGRAPH computer graphics*, 20(4), 151–160, 1986.
- [26] Sorkine, O.; Alexa, M.: As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, vol. 4, 109–116, 2007.
- [27] Sumner, R.W.; Popović, J.: Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3), 399–405, 2004.
- [28] Sumner, R.W.; Zwicker, M.; Gotsman, C.; Popović, J.: Mesh-based inverse kinematics. *ACM transactions on graphics (TOG)*, 24(3), 488–495, 2005.
- [29] Tenenbaum, J.B.; De Silva, V.; Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500), 2319–2323, 2000.
- [30] Umlauf, G.: Analyzing the characteristic map of triangular subdivision schemes. *Constructive Approximation*, 16(1), 145–155, 2000.
- [31] Wang, Y.; Li, G.; Zeng, Z.; He, H.: Articulated-motion-aware sparse localized decomposition. In *Computer Graphics Forum*, vol. 36, 247–259. Wiley Online Library, 2017.
- [32] Winkler, T.; Drieseberg, J.; Alexa, M.; Hormann, K.: Multi-scale geometry interpolation. In *Computer graphics forum*, vol. 29, 309–318. Wiley Online Library, 2010.
- [33] Xia, Q.; Chen, C.; Liu, J.; Li, S.; Hao, A.; Qin, H.: Efficient 4d shape completion from sparse samples via cubic spline fitting in linear rotation-invariant space. *Computers & Graphics*, 2019.
- [34] Xu, D.; Zhang, H.; Wang, Q.; Bao, H.: Poisson shape interpolation. *Graphical models*, 68(3), 268–281, 2006.
- [35] Zhang, Z.; Li, G.; Lu, H.; Ouyang, Y.; Yin, M.; Xian, C.: Fast as-isometric-as-possible shape interpolation. *Computers & Graphics*, 46, 244–256, 2015.
- [36] Zhang, Z.; Zha, H.: Principal manifolds and nonlinear dimension reduction via local tangent space alignment (dept of computer science, pennsylvania state univ, university park, pa). Tech. rep., Tech Rep CSE-02-019, 2002.