GPU-Friendly Shape Interpolation Based on Trajectory Warping

Lu Chen Jin Huang Hongxin Zhang
Wei Hua*
State Key Lab of CAD&CG, Zhejiang University

Abstract

In this paper, we propose a GPU-friendly shape interpolation method. In contrast with state-of-the-art interpolation algorithms, our method computes the trajectory of each vertex independently instead of solving large linear systems in every interpolation step. Given two poses being interpolated, we find trajectory parameters for each vertex by optimization with the consideration of the key pose reconstruction and as-rigid-as-possible deformation in the pre-computing stage. During run-time, the vertices coordinates on the intermediate shape can be computed in parallel according to a close form formulation. In the results we demonstrate that our method achieves extremely high performance on modern GPU and can be extended easily to multi-pose interpolation.

Keywords: Shape Interpolation, Vertex Path Problem, Trajectory Warping

1 Introduction

Shape interpolation is a widely utilized technique in computer animation, which interpolates between given shapes to create attractive effects. In general, shape interpolation includes two major issues: how to build a one-to-one mapping between key shapes [1, 2], and how to compute trajectory between corresponding vertices during the interpolation. In this paper, we focus on the latter issue, i.e. the trajectory issue.

*Corresponding author: Wei Hua

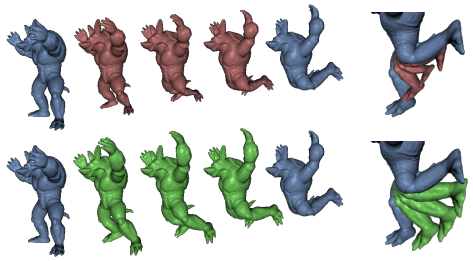

Figure 1: The pose interpolation results of armadillo model using linear interpolation (top) and our method (bottom) respectively. The blue models in each row are the input poses.

It is well-known that the *explicit linear interpolation* of vertex coordinates suffers shrinkage problem, as illustrated in Figure 1. To generate visual pleasant results, most of the state-of-the-art methods [3, 4, 5] interpolate gradient information or Laplacian coordinates instead of vertex coordinates, and then *implicitly* reconstruct vertex coordinates by optimizing least squares errors. Due to the as-rigid-as-possible preserving nature of gradient domain methods, such approaches can produce high quality shapes. However, these methods need to solve large linear systems in each interpolation step. The large amount memory consumption of matrices makes them hard to interpolate large models in realtime even with pre-computing step. Moreover they are not GPU-friendly as vertex coordinates cannot be reconstructed in a parallel way. Therefore gradient domain methods are not suitable for performance demanding applications such as games and virtual reality.

In this paper, we propose an *explicit nonlinear interpolation* method which can reconstruct the intermediate shape on GPU without shrinkage problem. The key idea is to bring in a close form formulation of the vertex trajectory for describing the interpolation procedure of each vertex. Inspired by the modal-warping technique [6], we regulate the linear interpolation trajectory with additional angular velocity component for tracing the local rotations. In this paper we call this technique *trajectory warping*, which can handle shape interpolation properly even under large deformation cases.

The major contributions of our work include:

- a new close form interpolation trajectory which can reconstruct the intermediate shape on GPU without solving any equations;
- an optimization method to find the parameters of the vertices trajectories which can interpolate given shapes with as-rigid-as-possible energy constraints to avoid producing interpolation artifacts.

2 Related Work

As [7] provided a good survey of earlier work, we only review several recent work related to 3D shape interpolation with feature preservation in this section.

Shape interpolation is related to feature preserving mesh deformation [8, 9, 10]. These methods reconstruct deformed shape according to given deformation gradient or Laplacian coordinates by minimizing the changing of local details. Most of these methods can be naturally extended to shape interpolation [3, 5, 11] as the rotation aspect of such local presentation can be easily separated from the stretch and scale. By applying proper rotation interpolation technique, intermediate shape can be reconstructed according to interpolated deformation gradient or Laplacian coordinates. All of these gradient domain methods produce high quality results but with the cost of solving large linear systems.

Alternatively, literature [12] formulates the shape interpolation problem as computing geodesic curves in the shape space. Although their method can provide high quality results, it is hard to generate intermediate shapes on the fly unless by storing all the vertex positions for the intermediate shapes.

To reduce the computational cost of gradient domain methods, some other approaches adopt subspace technique. For example, [13] uses the methods proposed in [14] to construct a generalized skeletal subspace. [15] can compute the transformations for each bone independently from the control points. To ensure the deformation quality, they apply a Poisson-based solver. With the help of the skeletal subspace, the dimensionality of the linear system for reconstruction is greatly reduced. It is worth mentioning that the recent work [16] also adopts generalized skeletal structure for improving the interpolation quality with significant pose variation. However, their method still requires solving a large linear system for shape reconstruction.

Same as gradient domain methods, our method needs to optimize shapes with the constraints of as-rigid-as-possible deformation. However, this optimization is only occurred in the pre-compute stage. During run-time, the trajectory of each vertex is evaluated independently on the fly according to trajectory warping. This is inspired by the modal warping technique [6], which greatly improves the traditional modal analysis to handle large rotation. In this paper, we extend such a trajectory representation from elastic object simulation into shape interpolation.

3 Algorithm

Given two consistent triangle meshes $\mathcal{M}^0$ and $\mathcal{M}^1$ with n vertices, the goal of shape interpolation is to find a displacement trajectory function $\mathbf{d}_i(t)$ $(t \in [0,1])$ for each vertex x_i. Let $\mathbf{p}_i^0$ and $\mathbf{p}_i^1$ be positions of vertex x_i on $\mathcal{M}^0$ and $\mathcal{M}^1$, respectively. The interpolation trajectory of this vertex can be expressed as:

$$\mathbf{p}_i(t) = \mathbf{p}_i^0 + \mathbf{d}_i(t), \tag{1}$$

which satisfies position interpolation constraints:

$$\mathbf{d}_i(0) = 0 \text{ and } \mathbf{p}_i^1 = \mathbf{p}_i^0 + \mathbf{d}_i(1). \tag{2}$$

In the rest of section, we will present a new close form of trajectory $\mathbf{d}_i(t)$ for handling large rota-

tion and introduce an optimization strategy for finding the best trajectory of each vertex.

3.1 Trajectory Warping

The most efficient trajectory calculation method is to use linear interpolation of the vertices positions:

$$\mathbf{d}_i(t) = t(\mathbf{p}_i^1 - \mathbf{p}_i^0). \tag{3}$$

Unfortunately, it cannot provide satisfied results under large rotation cases which introduce large non-linear conditions [7]. Based on this observation, the main idea of trajectory warping is to introduce non-linear correcting terms into the trajectory formulation for compensating the rotation of local frames.

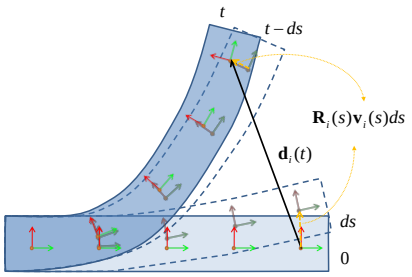

Figure 2: Local coordinate frames of rotational deformation.

As shown in Figure 2, the rotation can be achieved by integrating the displacement velocity $\mathbf{v}(t)$ under rotational local frame as infinitesimal deformation. The displacement trajectory of a rotational vertex can be described as below:

$$\mathbf{d}_i(t) = \int_0^t \mathbf{R}_i(s)\mathbf{v}_i(s)ds, \tag{4}$$

where $\mathbf{R}_i(t)$ is the rotation matrix representing the orientation of the local frame. We employ Rodrigues formula to express the rotation matrix in terms of the angle $\theta_i(t)$ and the unit axis of rotation $\mathbf{r}_i(t)$:

$$\begin{aligned}\mathbf{R}_i(t) = \mathbf{I} &+ [\mathbf{r}_i(t)]_\times \sin(\theta_i(t)) \\ &+ [\mathbf{r}_i(t)]_\times^2 (1 - \cos(\theta_i(t))),\end{aligned} \tag{5}$$

where $[\mathbf{r}_i]_\times$ denotes the standard skew symmetric matrix of unit axis vector $\mathbf{r}_i$.

During an interpolation procedure, steady and smooth deformation of shapes are desired. So it is reasonable to assume that the displacement velocity $\mathbf{v}_i(t) \equiv \mathbf{v}_i$ and $\mathbf{r}_i(t) \equiv \mathbf{r}_i$ are constant, while $\theta_i(t) = t\theta_i$. Denote $\mathbf{w}_i(t) = \theta_i(t)\mathbf{r}_i(t)$, then the displacement trajectory function can be simplified:

$$\mathbf{d}_i(t) = t\tilde{\mathbf{R}}(t; \mathbf{w}_i)\mathbf{v}_i, \tag{6}$$

where

$$\begin{aligned}\tilde{\mathbf{R}}(t; \mathbf{w}_i) = \mathbf{I} &+ [\mathbf{r}_i]_\times \frac{1 - \cos(t\theta_i)}{t\theta_i} \\ &+ [\mathbf{r}_i]_\times^2 \left(1 - \frac{\sin(t\theta_i)}{t\theta_i}\right).\end{aligned} \tag{7}$$

In this simplification, $\mathbf{w}_i$ and $\mathbf{v}_i$ can be considered as the per-vertex angular and displacement velocities, respectively. Compare to the linear interpolation trajectory, $\mathbf{w}_i$ introduces a nonlinear warping which is the essential part for handling large rotation.

3.2 Velocity Optimization

With trajectory warping, our interpolation problem is converted to finding out the optimal angular and displacement velocities for all the vertices. Let $\mathbf{v}$ and $\mathbf{w}$ be both $3n \times 1$ vectors which pack the $\mathbf{v}_i$ and $\mathbf{w}_i$ of all the vertex positions into column vectors. In our approach, to make the trajectory start from the $\mathcal{M}^0$ and end at $\mathcal{M}^1$ as well as keep the intermediate shapes as rigid as possible, velocities $\mathbf{w}$ and $\mathbf{v}$ are selected by minimizing following energy function:

$$E(\mathbf{v}, \mathbf{w}) = \alpha^2 E_{pose} + \beta^2 E_{rot} + \gamma^2 E_{grad} \tag{8}$$

where α, β and γ are weights. In the objective function, E_{pose} and E_{rot} are key pose and rotation energy terms, respectively. For further controlling the shape qualities during interpolation, we also introduce a gradient energy term E_{grad}.

Note that once we get the parameters of the vertices trajectories, the intermediate shape vertices positions can be calculated in parallel according to equation (1) and (6) for any interpolation parameter t.

Key pose energy As the requirement of shape interpolation, the trajectory must pass the target shape $\mathcal{M}^1$ at time $t = 1$. It implies the following pose constraint energy must be minimized

$$E_{pose} = \|\tilde{\mathbf{R}}(\mathbf{w})\mathbf{v} - (\mathbf{P}^1 - \mathbf{P}^0)\|^2, \tag{9}$$

where $\mathbf{P}^0$ and $\mathbf{P}^1$ are both $3n \times 1$ vectors which stand for the vertex coordinates of the interpolation shapes, and$\tilde{\mathbf{R}}(\mathbf{w}) = \mathrm{diag}(\tilde{\mathbf{R}}(1;\mathbf{w}_1),\ldots,\tilde{\mathbf{R}}(1;\mathbf{w}_n))$ is a block-diagonal matrix.

Note that $\tilde{\mathbf{R}}(t;\mathbf{w}_i) \equiv \tilde{\mathbf{R}}(1;t\mathbf{w}_i)$, then we have$\tilde{\mathbf{R}}(t\mathbf{w}) = \mathrm{diag}(\tilde{\mathbf{R}}(t;\mathbf{w}_1),\ldots,\tilde{\mathbf{R}}(t;\mathbf{w}_n))$, which will be used in the following sections.

Rotation energy In our method, angular velocities $\mathbf{w}$ is used to represent the rotation variety of the local frame. For keeping interpolation results as rigid as possible, $\mathbf{w}$ should be close to the rotation variety caused by the local displacement velocity $\mathbf{v}$ for tracking the local frame well. Let $\hat{\mathbf{v}}$ be per-vertex defined velocity field by $\mathbf{v}_i$ $(i = 1,\ldots,n)$. According to the kinematics of infinitesimal deformation analysis [6], it shows that the following least square on each vertex x_i should be minimized:

$$\|\mathbf{w}_i - \frac{1}{2}\nabla \times \hat{\mathbf{v}}(x_i)\|^2 \tag{10}$$

In Equation (10), symbol $\nabla\times$ stands for curl operation. $\frac{1}{2}\nabla \times \hat{\mathbf{v}}(x_i)$ is viewed as the rotational component of velocity field $\hat{\mathbf{v}}$ on vertex x_i. Note that here we assume that the curl operation is constant during all infinitesimal steps of the deformation.

However, it is not trivial to calculate $\frac{1}{2}\nabla \times \hat{\mathbf{v}}$ on a triangular mesh. Inspired by [17], we adopt following approximation method to calculate per-vertex rotational component of the velocity field. Let A be a triangle which undergoes a deformation and $(\mathbf{p}_i^0, \mathbf{v}_i)$ $(i = 1,2,3)$ represents its three vertex positions in static state and local displacement velocities, respectively. Then rotational variety of triangle A can be approximated by a linear matrix production $\mathbf{W}_A\mathbf{v}_A$ with

$$\begin{aligned} \mathbf{W}_A &= -\mathbf{K}^{-1}\left([\mathbf{q}_1]_\times \,|\, [\mathbf{q}_2]_\times \,|\, [\mathbf{q}_3]_\times\right), \\ \mathbf{v}_A^\top &= \left(\mathbf{v}_1^\top \,|\, \mathbf{v}_2^\top \,|\, \mathbf{v}_3^\top\right), \end{aligned} \tag{11}$$

where $\mathbf{q}_i = \mathbf{p}_i^0 - \frac{1}{3}\sum_i \mathbf{p}_i^0$, and $\mathbf{K} = \sum_i^3 [\mathbf{q}_i]_\times^2$.

For the rotational component on a vertex, we use the average of the rotation varieties of all the triangles sharing the vertex. Thus, we could assemble the matrices $\mathbf{W}_A$ of all the triangles to form the global curl matrix $\mathbf{W}$, which is in dimension $3n \times 3n$. Finally we obtain the rotation energy on the total mesh as following:

$$E_{rot} = \|\mathbf{w} - \mathbf{W}\mathbf{v}\|^2. \tag{12}$$

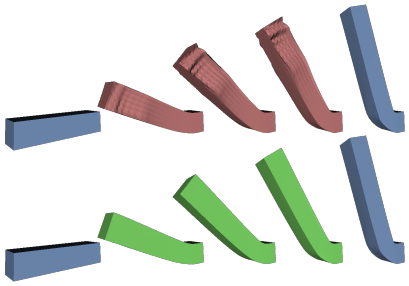

Figure 3: The comparison of without (top) / with (bottom) the intermediate gradient energy control.

Intermediate gradient energy Notice that the global curl matrix $\mathbf{W}$ only restricts the rotation component of the deformation. In many cases, the shape interpolation may involve a large degree of stretching or compression. In such cases, the rotation component is not dominant. If we only consider key pose and rotation energy, the resultant interpolation may give rise to a sort of distortions. Figure (3) illustrates one such example. Hence we need to introduce additional terms to control scaling and shearing during interpolation.

To overcome such problem, our solution is partly inspired by [18] and [4]. We bring in an as-rigid-as-possible deformation gradient to guide the shape interpolation. Consider a triangle A on shape $\mathcal{M}^0$ who is under deformation and its vertex positions in time t are $\mathbf{p}_i^t$ $(i = 1,2,3)$. For calculating the deformation gradient of this deforming triangle, we employ the surface-based deformation gradient proposed in [19], which is equivalent to the 'volumetric' one [3] on the tetrahedron mesh but does not involve a fourth vertex. Let $\mathbf{p}_A^t = (\mathbf{p}_1^t|\mathbf{p}_2^t|\mathbf{p}_3^t)$, then the surface-based deformation gradient of this triangle $\bar{\mathbf{D}}_A^t$ can be calculated by:

$$\bar{\mathbf{D}}_A^t = \mathbf{p}_A^t\bar{\mathbf{G}}_A \tag{13}$$

and

$$\bar{\mathbf{G}}_A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & 0 \end{pmatrix}\left(\mathbf{p}_1^0 - \mathbf{p}_3^0|\mathbf{p}_2^0 - \mathbf{p}_3^0|\mathbf{n}_A\right)^{-1} \tag{14}$$

where $\mathbf{n}_A$ is the normal vector of the triangle in undeformed state and $\bar{\mathbf{G}}_A$ is the linear gradient operator.

For the target interpolated deformation gradient, considering the triangle A referred above, we first calculate the transformation of this triangle from start to end $\mathbf{D}_A$ (3×3), using method described in [3] which involves the fourth vertices. Then, according to [4], we interpolate the rotational and scale/shear components of $\mathbf{D}_A$ separately:

$$\mathbf{g}_A^t = \mathbf{R}_A^t(t\mathbf{S}_A), \tag{15}$$

where $\mathbf{g}_A^t$ is the as-rigid-as-possible interpolated deformation gradient matrix at time t. $\mathbf{R}_A$ and $\mathbf{S}_A$ are gotten by polar factorization $\mathbf{D}_A = \mathbf{R}_A\mathbf{S}_A$ and $\mathbf{R}_A^t$ is the rotational interpolation of $\mathbf{R}_A$ with $\mathbf{I}$.

Notice that the surface-based gradient $\bar{\mathbf{D}}_A^t$ discards the normal component[19] and for our intermediate gradient constraint, the target interpolated gradients should also remove the degree of freedom along the normal direction:

$$\bar{\mathbf{g}}_A^t = \mathbf{g}_A^t - \mathbf{g}_A^t\mathbf{n}_A\mathbf{n}_A^\top. \tag{16}$$

We use this as-rigid-as-possible deformation gradient as a guidance constraint at the the intermediate frame:

$$E_{grad}{}^t(A) = \|\mathbf{p}_A^t\bar{\mathbf{G}}_A - \bar{\mathbf{g}}_A^t\|_F^2. \tag{17}$$

In practice, we find that under the key pose and rotation constraints, we can efficiently optimize the interpolation by only constraining the deformation gradient of intermediate frame $t = 0.5$.

Summarizing all the constraint energies on all m triangles with repacking the gradient operator matrices and target gradient vectors into $9m \times 3n$ matrix $\mathbf{G}$ and $9m \times 1$ vector $\bar{\mathbf{g}}^{0.5}$, we can get the global form of gradient guidance constraint energy in terms of velocities parameters:

$$\begin{aligned} E_{grad} &= \sum_A E_{grad}{}^{0.5}(A) \\ &= \|\mathbf{G}\mathbf{P}^{0.5} - \bar{\mathbf{g}}^{0.5}\|^2 \\ &= \|\frac{1}{2}\mathbf{G}\tilde{\mathbf{R}}\left(\frac{\mathbf{w}}{2}\right)\mathbf{v} - \left(\bar{\mathbf{g}}^{0.5} - \mathbf{G}\mathbf{P}^0\right)\|^2 \\ &\triangleq \|\frac{1}{2}\mathbf{G}\tilde{\mathbf{R}}\left(\frac{\mathbf{w}}{2}\right)\mathbf{v} - \mathbf{g}\|^2. \end{aligned} \tag{18}$$

4 Implementation

In this section, we presents the implementation details on solving the optimization problem of Equation (8) for trajectory warping.

Gauss-Newton Algorithm We adopt Gauss-Newton method to solve this optimization problem. Note that the objective function is a sum of squares, we can rewrite it in dot product form of vector $\mathbf{F}$, i.e., $E(\mathbf{v}, \mathbf{w}) = \mathbf{F}^\top\mathbf{F}$. Moreover let $\mathbf{J}(\mathbf{v}, \mathbf{w})$ be the Jacobian matrix of vector $\mathbf{F}(\mathbf{v}, \mathbf{w})$. Base on this configuration, the whole optimization is performed in an iterative way:

$$\left\{ \begin{array}{rcccl} \mathbf{v}^{k+1} & = & \mathbf{v}^k & + & \delta_v^k, \\ \mathbf{w}^{k+1} & = & \mathbf{w}^k & + & \delta_w^k, \end{array} \right. \tag{19}$$

where (δ_v^k, δ_w^k) is obtained by solving following linear least squares problem

$$\arg\min_{\delta_v^k, \delta_w^k} \left\| \mathbf{J}(\mathbf{v}^k, \mathbf{w}^k) \begin{pmatrix} \delta_v^k \\ \delta_w^k \end{pmatrix} + \mathbf{F}(\mathbf{v}^k, \mathbf{w}^k) \right\|^2. \tag{20}$$

Initial Value To obtain the initial value $(\mathbf{v}^0, \mathbf{w}^0)$, we first calculate the local frame variety of vertex x_i by averaging rotation components of its neighboring triangles $\mathcal{N}_i$:

$$\mathbf{w}_i^0 = \frac{1}{|\mathcal{N}_i|}\sum_{A_j \in \mathcal{N}_i} \log(\mathbf{R}_j). \tag{21}$$

Here $\mathbf{R}_j$ is the rotation component of the deformation gradient $\mathbf{D}_j$ of triangle A_j and extracted by polar decomposition $\mathbf{D}_j = \mathbf{R}_j\mathbf{S}_j$. When the shape deformation involves mainly the rotation component, the integration of the rotation vector $\mathbf{w}_i$ will be close to the variety of local frame around this vertex. Therefore $\mathbf{w}_i^0$ is a good estimation of per-vertex angular velocity. Hence we assemble all of them as the initial angular velocity vector $\mathbf{w}^0$ in our solver.

The initial value of the displacement velocity is calculated by:

$$\mathbf{v}^0 = \left(\tilde{\mathbf{R}}(\mathbf{w}^0)\right)^{-1}(\mathbf{P}^1 - \mathbf{P}^0). \tag{22}$$

We find that under most cases, the non-linear optimizations converge within 10 iterations by adopting these initial values.

Weighting Schema To select meaningful weights, we first calculate a base set of weights α_0,β_0 and γ_0 such that Hessian matrices of energy terms E_{pos}, E_{rot} and E_{grad} have same Frobenius norms. Then, because the pose constraint is expected to be a hard constraint for our shape interpolation application, we set the weight $\alpha = \alpha_0$ as the largest weight in our optimization. For the weights of the compatibility and intermediate gradient constraints, we found that a large range of values work well and a minimal amount of example-specific tuning is required. In this paper, we use the setting of $\beta = 0.1 \cdot \beta_0$ and $\gamma = 0.05 \cdot \gamma_0$ for most of the examples.

Post-processing After solving the optimization, we can compute the velocities $(\mathbf{v}, \mathbf{w})$ for performing the interpolation. However in the velocity optimization (Equation (8)), pose energy E_{pose} acts as a soft constraint which cannot be exactly ensure pose interpolation. Therefore we need to adopt a simple post-processing. That is we modify displacement velocities to directly satisfy the pose interpolation constraint (Equation (2)):

$$\mathbf{v}' = \left(\tilde{\mathbf{R}}(\mathbf{w})\right)^{-1} (\mathbf{P}^1 - \mathbf{P}^0). \qquad (23)$$

After this, we can ensure that the trajectories interpolate shape $\mathcal{M}^1$ when $t = 1$. Usually the changes brought in by the post procedure can not be noticed visually because the residuals are very small when the optimization converges.

5 Results and discussions

Our framework can be applied to various scenarios and applications. In this section we will first show several applications of our method. Then performance will be reported as well. Finally, we will discuss an extension of our trajectory warping technique for better handling an intrinsic complex articulated motion.

5.1 Applications

Two shapes interpolation Figure 1 shows pose interpolation between two armadillo poses. Our method can handle the large deformations of the legs. Figure 4 shows the morphing result between two head models. The results of our method are as good as those of Poisson Interpolation [5] and the comparisons between poisson method and ours can be found in the attached video.

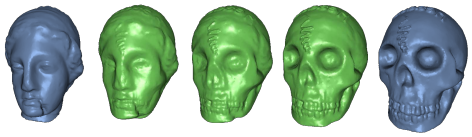

Figure 4: The morphing result between head models.

Multi-pose interpolation Given K multiple poses $\mathcal{M}^k$ $(k \in \{1, \ldots, K\})$ and one reference pose $\mathcal{M}^r$, we can calculate the vector pairs $\{\mathbf{v}^k, \mathbf{w}^k\}$ for all poses. Consider the poses $\mathcal{M}^k$ as a pose space refer to $\mathcal{M}^r$, the vector pairs $\{\mathbf{v}^k, \mathbf{w}^k\}$ could be viewed as the coordinates of these poses in this high dimension space. The multi-pose interpolation could be transformed into interpolating the vector pairs. Given a set of interpolation weights ξ_k $(k \in \{1, \ldots, K\})$, the coordinates of the interpolated pose is:

$$\{\mathbf{v}^{inter}, \mathbf{w}^{inter}\} = \{\sum_{k=1}^{K} \xi_k \mathbf{v}^k, \sum_{k=1}^{K} \xi_k \mathbf{w}^k\}. \qquad (24)$$

And the new pose could be reconstructed in our framework with Equation 6:

$$\mathbf{P}^{inter} = \mathbf{P}^0 + \tilde{\mathbf{R}}(\mathbf{w}^{inter})\mathbf{v}^{inter}. \qquad (25)$$

Figure 5 shows the multi-pose interpolation between armadillo poses. The top three poses are given and the first one is used as the reference pose. The triangle areas with red points indicate the interpolation weights. The three corners correspond to the input poses and the weights are $(1, 0, 0), (0, 1, 0), (0, 0, 1)$. The red point inside the triangle area indicates the weight as its barycentric coordinate. The bottom four poses are interpolated using the weights shown by the above triangle figures.

Because our reconstruction procedure is very fast, the users can interactively change the weights and get the interpolated shape in real-time. The effects of this real-time multi-pose interpolation are demonstrated in the attached

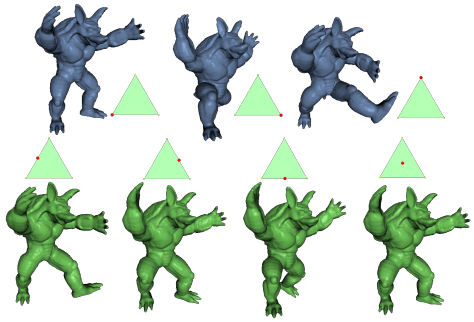

Figure 5: Multi-pose interpolation between armadillo poses.

video. Users can create new animation interactively by interpolating the existing poses in real-time.

5.2 *Timing and comparison*

Table 1 shows the statistics of the demos in this paper, including the complexity of the models and the performance of the algorithm. All the timings in the table are measured on an Intel Pentium Dual 2.0GHz (only one core used), 3.0GB RAM machine with a NVIDIA GForce 9800GT graphics card.

The third column in the table shows one iteration time of our non-linear optimization solver and the fourth column shows the iteration times before the solver converges. The fifth column shows the run-time interpolation performance on GPU. Here we implement the per-vertex integration (Equation (6)) using CUDA and bind a VBO for rendering the deformed models. After we upload the velocity pairs to GPU at the beginning, only the interpolation coefficient t need to be passed to GPU every frame, while for multi-pose interpolation, the set of interpolation weights ξ_k need to be passed.

Compared to the time cost of poisson method (the sixth column), our method is several orders of magnitude faster. The time cost of the poisson method includes the cost of back-substitution and constructing the right-hand side value of one frame, but not includes the cost for Cholesky factorization, which is done in pre-computation.

5.3 *Multi-basis trajectory extension*

Note that in the simplified trajectory function (Equation (6)), we made strong assumption on

Model in Fig	#Tri Num	Iter Opti(s)	Iter Num	GPU (μs)	Poisson (ms)
Fig1	40k	16.2	3	34.2	189.5
Fig4	48k	19.9	6	39.6	169.5
Fig6	4k	0.6	4	16.8	11.6
Fig7	40k	17.3	9	74.5	160

Table 1: The statistics of demos in the paper.

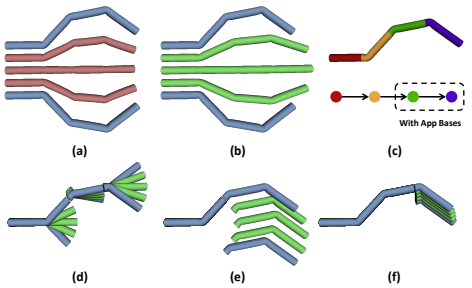

Figure 6: Interpolation results of cylinder model with intrinsic articulated structure.

steady shape deformation which fixes the angular and displacement velocities. Although it is sufficient for many cases and ensure our methods efficiency, this assumption limits the expression ability of the trajectory for many complex cases. Actually, we find that when the interpolation motion has intrinsic complex articulated structure, the user are very sensitive to the rigidity preserving of each part and our current basic formulation may not achieve the 'desired' result (Figure 6(a)).

The motion of the end effector of a complex articulated structure is actually composed by all the motions of linked joints. For expressing this composite motion, we extend the basic formulation by appending additional velocity-pairs, i.e.,

$$\mathbf{d}_i(t) = t\tilde{\mathbf{R}}(t\mathbf{w}_i)\mathbf{v}_i + t\sum_{k\in S_i}\tilde{\mathbf{R}}(t\mathbf{w}_k^{app})\mathbf{v}_k^{app}, \tag{26}$$

where w_k^{app} and v_k^{app} are velocity basis introduced for expressing the motion of joint k and S_i is the indices set of the joints which contribute to the motion of vertex i.

For a model whose interpolation motion may involve intrinsic complex articulated structure, we first extract its articulated segmentation and structure (6(c)). Similar to [16], we use mean-shift clustering algorithm for extracting the near-rigid components, and manually handle the joint part for completing a segmentation. Then, one

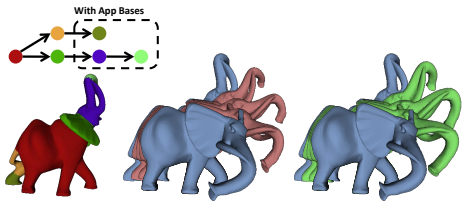

Figure 7: Elephant model interpolation results.

segment is chosen to be the root part and the articulated structure can be generated based on the connected relations. We assume the root part to be near-static during the interpolation (otherwise we can align the models by the root part first), and append bases for presenting the joint motions whose node distances to the root node are larger than 2. As shown in Figure 6(c), we add two new bases for solving.

We use the same optimization strategy in Equation (8) for solving the appended velocity bases $(\mathbf{w}^{app}, \mathbf{v}^{app})$ with the basic bases $(\mathbf{w}, \mathbf{v})$ at one time, where $\mathbf{w}^{app}$ and $\mathbf{v}^{app}$ are the packed vectors of all appended angular and displacement velocities respectively. Based on the new formulation in Equation (26), the derivations of the new constraint energies of key pose and intermediate gradient are straightforward. For the rotation energy term, note that the rotation component of the local frame relates to the sum of the displacement velocities on all basis and the rotation vectors of the basic bases impact the per-vertex local frames. The new rotation energy is then formulated as following:

$$E'_{rot} = \|\mathbf{w} - \mathbf{W}\mathbf{v} - \mathbf{W}^{app}\mathbf{v}^{app}\|^2, \qquad (27)$$

where $\mathbf{W}^{app}$ is a matrix which repacks the corresponding terms in $\mathbf{W}$.

As shown in Figure 6(b), our method greatly improves the result, while only increases a little cost because only few more variables need to be solved. The bottom row pictures show the result of the displacement integration results of different bases. (d) is the result only integrates the basic bases, while (e) and (f) are the individual integration results of the first and the second appended bases respectively.

Figure 7 shows the interpolation result of elephant model. The leftmost picture shows the segmentation and the articulated structure of the model. The middle picture shows the result only employs the basic trajectory formulation, in which the front part of the nose has some shrinking. The right picture is the result using our multi-basis extension.

6 Conclusion

This paper proposed a fast shape interpolation technique which aims at the performance demanding key frame animation applications such as game and virtual reality. The basic idea is to compute proper parameters of a close-form non-linear vertex trajectory representation in the pre-computing stage and interpolate the intermediate shape vertices in parallel in the run-time. Our method demonstrates that the intermediate shape of as-rigid-as-possible shape interpolation can be reconstructed in GPU without solving any equation.

As we adopt the optimization strategy to find the best results in the parameter space of the trajectory formulation, our method can not avoid self-intersection restrictively. Our method may not work well on the cases that the given shapes are extremely different, such like genus-n-to-m shape morphing [20]. Another limitation of our work is high computational cost at the pre-computing stage. Although it only need to be calculated once, the bottleneck of solving large non-linear optimization problem limits our method from very huge models. We will investigate the possibility of adopting multi-grid solver to overcome the problem in the future.

Acknowledgements

This work is supported in part by National Natural Science Foundation of China (No. 61070073). We also thank the anonymous reviewers for their helpful comments and suggestions.

References

[1] Vladislav Kraevoy and Alla Sheffer. Cross-parameterization and compatible remeshing of 3d models. *ACM Trans. Graph.*, 23(3):861–869, 2004.

[2] John Schreiner, Arul Asirvatham, Emil Praun, and Hugues Hoppe. Inter-surface mapping. *ACM Trans. Graph.*, 23(3):870–877, 2004.

[3] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. In *ACM SIGGRAPH*, pages 399–405, 2004.

[4] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *ACM Trans. Graph.*, pages 157–164, 2000.

[5] Dong Xu, Hongxin Zhang, Qing Wang, and Hujun Bao. Poisson shape interpolation. In *SPM '05*, pages 267–274, 2005.

[6] Min Gyu Choi and Hyeong-Seok Ko. Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE TVCG*, 11(1):91–101, 2005.

[7] Marc Alexa. Recent advances in mesh morphing. *Computer Graphics Forum*, 21(2):173–196, 2002.

[8] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2):105–114, 2003.

[9] Alla Sheffer and Vladislav Kraevoy. Pyramid coordinates for morphing and deformation. In *3DPVT '04*, pages 68–75, 2004.

[10] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.

[11] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph.*, 24(3):479–487, 2005.

[12] Martin Kilian, Niloy J. Mitra, and Helmut Pottmann. Geometric modeling in shape space. *ACM Trans. Graph.*, 26(3), 2007.

[13] Kevin G. Der, Robert W. Sumner, and Jovan Popović. Inverse kinematics for reduced deformable models. *ACM Trans. Graph.*, 25(3):1174–1179, 2006.

[14] Doug L. James and Christopher D. Twigg. Skinning mesh animations. In *SIGGRAPH '05*, pages 399–407, 2005.

[15] Wei-Wen Feng, Byung-Uck Kim, and Yizhou Yu. Real-time data driven deformation using kernel canonical correlation analysis. In *SIGGRAPH '08*, pages 1–9, 2008.

[16] Hung-Kuo Chu and Tong-Yee Lee. Multiresolution mean shift clustering algorithm for shape interpolation. *IEEE TVCG*, 15(5):853–866, 2009.

[17] Min Gyu Choi, Seung Yong Woo, and Hyeong-Seok Ko. Real-time simulation of thin shells. *Comput. Graph. Forum*, 26(3):349–354, 2007.

[18] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. Mesh-based inverse kinematics. *ACM Trans. Graph.*, 24(3):488–495, 2005.

[19] M. Botsch, R. Sumner, M. Pauly, and M. Gross. Deformation transfer for detail-preserving surface editing. In *Vision, Modeling & Visualization*, pages 357–364, 2006.

[20] Tong-Yee Lee, Chih-Yuan Yao, Hung-Kuo Chu, Ming-Jen Tai, and Cheng-Chieh Chen. Generating genus-n-to-m mesh morphing using spherical parameterization. *Comput. Animat. Virtual Worlds*, 17:433–443, 2006.