A Blendshape Model that Incorporates Physical Interaction

Wan-Chun Ma[1] Yi-Hua Wang[2] Graham Fyffe[3] Bing-Yu Chen[2] Paul Debevec[3]
[1]Weta Digital [2]National Taiwan University [3]USC Institute for Creative Technologies

Abstract

The linear blendshape technique has been intensively used for computer animation and games due to its simplicity and effectiveness. However it cannot describe rotational deformations and deformations due to self collision or scene interaction. In this paper, we present a new technique to address these two major limitations by introducing physical-based simulation to blendshapes.

The proposed technique begins by constructing a mass-spring system for each blendshape target. Each system is initialized in its steady state by setting the rest length of each spring as the edge length of the corresponding target. To begin shape interpolation, we linearly interpolate the rest lengths of the springs according to a given interpolation factor $\alpha \in [0,1]$. The interpolated shape is then generated by computing the equilibrium of the mass-spring system with the interpolated rest lengths. Results from our technique show physically-plausible deformations even in the case of large rotations between blendshape targets. In addition, the new blendshape model is able to interact with other scene elements by introducing collision detection and handling to the mass-spring system.

1 Introduction

The linear blendshape technique is a widely adopted solution for many applications that require an efficient geometrical deformation between two or more input shapes. One such applications is key frame animation. The shape at a given frame is obtained by directly interpolating the shapes from nearby key frames. Another typical application is facial animation. Various facial expressions, often referred to as blendshape targets or key poses, are modelled as input shapes. A new shape of a desired expression can be generated by fully or partially blending those input shapes. Despite its simplicity, the linear blendshape technique is still one of the mostly favored techniques for computer animation.

However, the linear blendshape technique exhibits two major drawbacks. First, the interpolated results usually are degraded when the deformation involves large rotations. Figure 1(c) illustrates a typical "shrinking" artifacts observed when linearly interpolating two shapes with a large rotation (e.g. bending) presents. More intermediate shapes can be introduced to reduce rotational artifacts. This requires extra time and storage budgets to prepare and save these shapes however. Second, the linear blendshape technique also ignores physical interaction. Any deformation caused by physical interaction has to be prepared by either pre-computing a new shape that conforms to the interaction based on physical simulation, or imitating the shape manually in a post-editing process. The former usually is not applicable to general physical interaction since a pre-computed shape would be needed for every possible deformation. The later requires experienced digital artists to produce visually pleasant results. For example, Borshukov [7] demonstrated both concepts.

In this paper we introduce a new physically-inspired blendshape technique that addresses the above-mentioned problems. Our method pro-

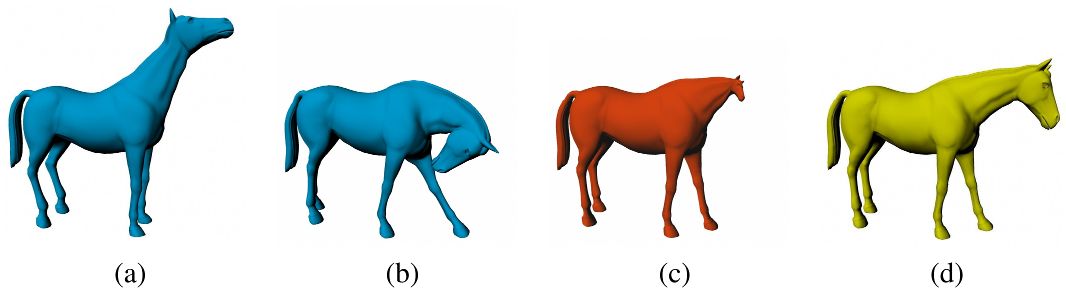

Figure 1: A comparison between linear blendshape and our technique. (a) Source shape. (b) Target shape. (c) Linearly interpolated with $\alpha = 0.5$. (d) Interpolated by the proposed method with the same α.

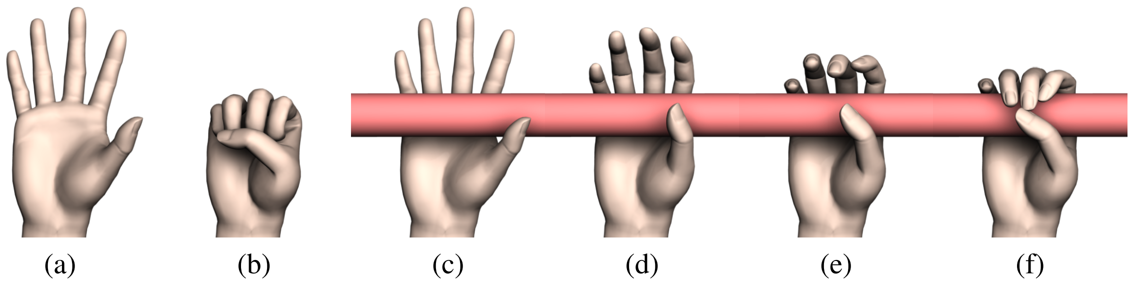

Figure 2: The proposed technique enables physically-plausible deformation. (a) Source shape. (b) Target shape. (c-f) Interpolating between the two shapes while interacting with a solid cylinder.

vides a physical underpinning for shape interpolation using mass-spring physics. A key observation is that the equilibrium state of a mass-spring system minimizes local area/volume distortions through force balancing. As a result, local rigidity can be maintained as much as possible during deformation. The underlying mass-spring system also provides a natural framework for physical interaction. It can be easily carried out by applying external forces and additional constraints based on collision detection.

The proposed method begins with building a mass-spring system for each input blendshape targets. The mass-spring system is initialized to its steady state by setting the rest length of each spring to the length of the corresponding edge. We then interpolate the rest lengths of the springs based on a given interpolation factor and solve for the equilibrium state of the interpolated mass-spring system, where the final vertex positions represent the interpolated shape. The equilibrium computation is the major cost of the technique. We demonstrate our blendshape technique with a wide variety of shapes that exhibit complicated geometry and deformations. The method yields more natural shape interpolations, which can be seen in Figure 1(d). Figure 2 shows interpolated results while interacting with a scene element.

Contributions. Two substantial contributions in our work are:

1. An algorithm that creates natural looking blendshape interpolation results based on interpolating the rest length of each spring of a mass-spring system. The proposed method requires no geometric analysis, articulated skeleton, or any manual intervention. Since it does not require a skeleton to drive the deformation, it is not limited to articulated shapes.

2. We provide a natural physical interaction capability, which has not been seen in traditional shape interpolation techniques, by applying additional collision detection and handling. The proposed shape interpolation exhibits correct deformation based on interaction with other scene elements and without requiring any pre-computation.

2 Related Work

Our technique relates to works for interpolating between shapes in two or three dimensions, mass-spring systems, and rest length animation.

Shape Interpolation and Deformation. Shape interpolation has been widely used for animating geometric deformation. The linear blendshape technique is the most common method for shape interpolation. Shape interpolation can also be achieved using an articulated skeleton. The skeleton may be manually specified [15, 35, 37], or automatically determined by finding near-rigid components of input shapes [11] or using the medial axis transform [6, 38]. Rohmer *et al.* [27] proposed a skinning method which exactly preserves (or controls) the volume of an object.

Other methods are free of using an articulated skeleton. One effective research trend is based on maintaining the rigidity criteria of local geometrical elements, or so-called "as-rigid-as possible" principle. [1] and [16] are typical examples of this approach. Baxter *et al.* [4] proposed a solution for solving the rotation ambiguity arising from previous rigid as-rigid-as possible approaches. Winkler *et al.* [36] interpolated edge lengths and dihedral angles of the input shapes, followed by a global multi-registration method to determine the best rigid transformation.

There are also methods that create shape interpolation that conforms to user manipulation. Barbič *et al.* [3] proposed a method for key frame animation based on an underlying physically-based simulation, which, similar to our approach, can also be driven by a mass-spring system. Both their method and ours compute an equilibrium state as the interpolated result. However, their method interpolates the deformation forces, which are either provided by a user or computed automatically based on key poses. Our method simply interpolates spring rest lengths. Kondo *et al.* [18] provided guided animations with dynamics. A user can have controls over trajectory and deformation. Their target trajectory is not obtained via proper shape interpolation, but just by pushing the object into next key frame. Lewis and Anjyo [20] introduced a direct manipulation method for blendshapes. This approach constrains any desired subset of vertices based on manual manipulations and automatically infers the remaining vertex positions. However, the method only produce shapes that are within the convex space of the original blendshape poses and has no physical meaning.

Other related works on shape interpolation or deformation include [12], which used example shapes to build a reduced deformable model which controls with mesh-based inverse kinematics. Galoppo *et al.* [14] introduced the concept of dynamic morph targets, to skeletally interpolate elastic forces, which allows control over geometry and elastic properties of an animated character. Teran *et al.* [32] solved quasi-static states of a finite element system for simulating deformations of nonlinear elastic materials. We also solve for quasi-static states and consider interaction with other rigid bodies but we only balance the mass-spring system with respect to varying the rest lengths.

Lewis *et al.* [21] regards shape interpolation as a scattered data interpolation problem in an abstract parameter (pose) space. Our method does not require high dimensional scattered data interpolation, and it interpolates shape with the support of physical simulation, presumably requiring fewer input shapes. Kilian *et al.* [17] presented an isometric deformation method based on Riemannian geometry, considering shape interpolation as a geodesic curve in shape space. Both of these previous techniques are unable to provide interaction capabilities because they lack an underlying physical approach.

Mass-Spring Systems. Mass-spring systems are commonly used for simulating physical behaviors. Generally, such systems are easy to implement and convenient to integrate with other techniques, such as collision detection. Applications that make use of a mass-spring system include surgical simulation [22], dynamics for animals [23], cloth [2, 10, 13], muscles [9, 24], and other deformable objects. Lee *et al.* [19] applied mass-spring systems to facial animation using a three-layered mesh to model the anatomy of human facial tissue. While finite element methods (FEM) can deliver more sophisticated and physically accurate analysis, mass-spring systems are attractive due to their low computational complexity.

Rest Length Animation. There are works that simulate muscle activation through controlling the rest length of each spring in a mass-spring system. Raibert and Hodgins [26] used rest length animation to simulate simple leg locomotion. Adjusting the rest length changes the force at each spring so that it is able to initiate or terminate its motion. Tu and Terzopoulos [33] constructed a mass-spring system of a fish body, and assigned some springs to be muscle springs driven by animated rest lengths. However, these earlier techniques were not applied to shape interpolation.

3 Blending Shapes with a Mass Spring System

The proposed blendshape technique consists of the following two steps:

1. Given a set of input blendshape targets, we construct a mass-spring system for each blendshape target based on its structure.
2. We interpolate rest lengths between two or more aforementioned mass-spring systems to generate a intermediate mass-spring system. The interpolated shape is the quasi-static state of the intermediate system.

3.1 Structure of the Mass Spring System

It is important to maintain the stability of the mass-spring system during interpolation, and this mostly depends on a good spring structure. Based on [10], we build the mass-spring system based on input triangulated mesh as a combination of *structure springs* and *bending springs*. The structure springs model the elastic properties of the mesh surface, connecting neighboring vertices. The bending springs define the bending and flexural properties of the material, and connect a vertex to secondary neighboring vertices (i.e., at a distance of 2). These bending springs help maintain the object's resting shape and preserve surface curvature.

Nevertheless, our experiments show that a large surface which consists of many small triangles tends to be crumpled during interpolation

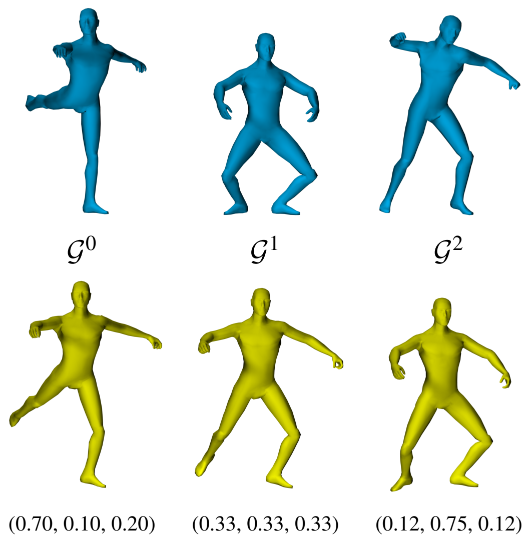

Figure 3: Blending between multiple shapes. The top row shows three source shapes and the bottom row shows blended results with corresponding weights shown as (w_0, w_1, w_2).

due to insufficient numerical precision. To solve this problem, we insert additional springs inside the shape to help preserve volume. We apply constrained Delaunay tetrahedralization [31] to determine where to insert these *internal springs*. Similar to the mesh refinement step in [1], to prevent springs with long rest lengths we can also perform mesh refinement by inserting new vertices. Those vertices have to be added to all the input shapes correspondingly. Certain additional vertices that are inside the input shapes can be assigned as an anchor as described in Section 3.4.

A mass-spring system $\mathcal{M} = \langle \mathcal{V}, \mathcal{S} \rangle$ is defined by a collection of vertices $\mathcal{V} = \{v_i | i = 1 \ldots n_v\}$ connected by springs $\mathcal{S} = \{s_q | q = 1 \ldots n_s\}$. Each spring $s_q \in \mathcal{S}$ connects two vertices $v_{e_{q0}}$ and $v_{e_{q1}}$, where $e_{q0} = \mathbf{e}(s_q, 0)$ and $e_{q1} = \mathbf{e}(s_q, 1)$ and function $\mathbf{e}$ returns the indices of spring s_q's two vertices. We enforce $1 \leq e_{q0} < e_{q1} \leq n_v$. In addition, s_q is characterized by a rest length $r_q = \mathbf{r}(s_q)$ and spring constant $k_q = \mathbf{k}(s_q)$, where functions $\mathbf{r}$ and $\mathbf{k}$ return the rest length and spring constant of spring s_q, respectively.

3.2 Blending between Two Shapes

We first introduce our technique by explaining the case of blending between two shapes $\mathcal{G}^0$ and

$\mathcal{G}^1$. We build two consistent mass-spring systems $\mathcal{M}^0 = \langle \mathcal{V}^0, \mathcal{S}^0 \rangle$ and $\mathcal{M}^1 = \langle \mathcal{V}^1, \mathcal{S}^1 \rangle$ from $\mathcal{G}^0$ and $\mathcal{G}^1$. The two mass-spring systems are initially set to be in their own steady states, i.e. $r_q^0 = \| v_{e_{q0}}^0 - v_{e_{q1}}^0 \|$ and $r_q^1 = \| v_{e_{q0}}^1 - v_{e_{q1}}^1 \|$. For each $\alpha \in [0,1]$, the interpolated result from the input shapes is the equilibrium state of a new mass-spring system $\bar{\mathcal{M}} = \langle \bar{\mathcal{V}}, \bar{\mathcal{S}} \rangle$, where for each spring,

$$\bar{r}_q = (1-\alpha) r_q^0 + \alpha r_q^1. \tag{1}$$

In an equilibrium state, the force $f(v_i)$ at each vertex v_i in $\mathcal{M}$ equals zero:

$$f(v_i) = \sum_{j \in \mathbf{n}(i)} k_q(\|v_i - v_j\| - r_q) \frac{v_i - v_j}{\|v_i - v_j\|} = 0, \tag{2}$$

where function $\mathbf{n}$ returns the indices of those vertices that are adjacent to v_i, and spring s_q connects v_i and v_j. Note that the velocity of each vertex can be ignored since our results are based solely on quasi-static states. We assume that each vertex has the same mass, therefore the mass term can also be ignored. To formulate Eq. (2) into a linear system, first we vectorize $\mathcal{V}$ into an one dimensional vector x as:

$$x = [\mathbf{x}(v_1), \mathbf{y}(v_1), \mathbf{z}(v_1), \cdots, \mathbf{x}(v_{n_v}), \mathbf{y}(v_{n_v}), \mathbf{z}(v_{n_v})]^T,$$

where $\mathbf{x}(v_i), \mathbf{y}(v_i), \mathbf{z}(v_i)$ are functions that return the X, Y and Z Cartesian coordinates of v_i. The system is then solved by using the Newton–Raphson method to determine the first order approximation of the optimal vertex configuration:

$$f(x_{t+1}) \approx f(x_t) + J(x_t)\Delta x_t, \tag{3}$$

where $x_{t+1} = x_t + \Delta x_t$ and $J(x_t) = \frac{\partial f}{\partial x}(x_t)$ is the global stiffness (Jacobian) matrix of f evaluated at the current vertex positions x_t. When the system is in its equilibrium state $f(x_{t+1}) = 0$. Eq. (3) now becomes:

$$J(x_t)\Delta x_t = -f(x_t). \tag{4}$$

The non-diagonal elements of the global stiffness matrix J_{ij} are defined by:

$$J_{ij} = J_{ji} = k_q \left(r_q \frac{\| d_{ij} \|^2 I - d_{ij} d_{ij}^T}{\| d_{ij} \|^3} - I \right),$$

where I is an identity matrix, $d_{ij} = v_i - v_j$, and spring s_q connects v_i and v_j; or else a 3×3 matrix of zeros if no such a spring exists. The diagonal elements ($i = j$) are defined as:

$$J_{ii} = -\sum_{j \in \mathbf{n}(i)} J_{ij}.$$

x_0 is initialized as the vertex positions of the source shape $\mathcal{V}^0$. We then iteratively solve Eq. (4) until $\|\Delta x_t\|$ is smaller than a threshold ($\bar{\mathcal{M}}$ reaches its equilibrium). The final vertex positions are then assigned to $\bar{\mathcal{V}}$. Generally matrix J is very sparse. There are both iterative (e.g. conjugate gradient) and direct methods for solving this sparse linear system.

3.3 Blending Multiple Shapes

The proposed technique can also be extended to blending multiple shapes by simply considering the interpolated rest length as a convex linear combination of the spring rest lengths from the input shapes:

$$\bar{r}_q = \sum_{i=0}^{n_b} w_i r_q^i,$$

where $\sum_{i=0}^{n_b} w_i = 1$. This is illustrated in Figure 3 where the interpolated shapes (shown in yellow) are the results of linearly-blended spring rest lengths from three different input shapes (shown in blue). Notice that this shares exact the same formulation for controlling blendshape targets as as the traditional linear blendshape technique does.

3.4 Boundary Conditions

Boundary conditions must be specified in order to solve the equilibrium of a mass-spring system. Without boundary conditions, the system will be under-constrained and the solution will not be unique. Simply speaking, boundary conditions are vertices which are fixed in a mass-spring system. In practice, this can be achieved by assigning Dirichlet boundary conditions to the global stiffness matrix, i.e., by replacing the block of the global stiffness matrix corresponding to boundary vertices with an identity matrix. The entries of the boundary vertices in f are replaced by zeros to enforce that the corresponding vertices do not move. A straightforward method to assign the boundary conditions is to find vertices which remain static between the source and target shapes. However, this is unlikely to apply to general blendshape targets. Alternatively, certain vertices on the surface can be manually marked as boundary conditions. During the interpolation, the positions of the marked vertices are then interpolated linearly. However, we found that this method often does not yield visually pleasing results, because these marked vertices still follow a linear trajectory during interpolation.

Similar to [8], we propose a more general method which allows all the surface vertices to move freely is to use an auxiliary object as an *anchor*, whose vertex positions act as the boundary conditions. For example, placing a rectangle around the center of mass of the object would keep this region fixed through the pose interpolation. We use the following procedure to add an anchor:

1. Select an anchor position inside the source shape (Figure 4(a)).

2. When building the structure of the mass-spring system (as in Section 3), we connect additional springs to the vertices of the anchor. All the springs belonging to the anchor are set as hard constraints by setting a very large spring constant. This enforces rigidity during the relaxation detailed in the next step.

3. The rest lengths of the internal springs that connect surface vertices to the anchor are computed from the source shape based on the user-assigned anchor positions. However, The rest lengths of these surface-anchor springs in the target shape still remains unknown. We have to determine the location of the anchor first. One possible solution is to reverse the roles of anchor and shape, i.e., we set all the vertices in the target shape as the boundary conditions, and copy the rest lengths of the springs connecting the anchor to the source, then determine the positions of anchor vertices within the target shape by solving the equilibrium.

4. Once the locations of the anchors are known, we can determine an optimal rigid transformation $T = \{R|t\}$ between the anchors of the source and target shapes such that $p_i^1 = Rp_i^0 + t$, where p_i^0 and p_i^1 are the positions of the source and target anchors' vertices, respectively.

5. For each step during the shape interpolation, we move the anchor according to the linearly-interpolated rigid transformation $T' = \{\mathbf{q}(I, R, \alpha)|\alpha t\}$, where $\mathbf{q}$ is the function that interpolates the identity matrix I and the rotation matrix R with a weight α using quaternions [30]. We subsequently fix the interpolated anchor vertices as the boundary conditions and compute the equilibrium state of the mass-spring system. To improve numerical stability, we solve for the equilibrium in the local coordinate frame of the (initial) anchor. Afterwards, we reapply the rigid transformation to both the anchor and the shape. This is illustrated in Figure 4.

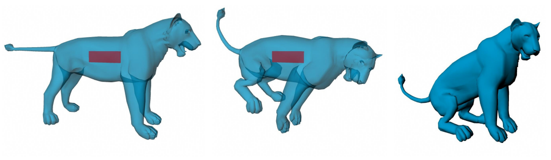

Figure 4: Apply an internal anchor as boundary conditions. (a) A source shape with an anchor object (a rectangular plane, shown in red) inside. (b) The interpolated result. The vertices of the anchor act as the boundary conditions during the spring relaxation, thus allowing free movement of all the surface vertices. (c) The interpolated result with a rigid transformation applied.

3.5 Spring Constants

The proposed shape interpolation method guarantees its results reach both of the input shapes as rest states of $\bar{\mathcal{M}}^i$. However, having a uniform spring constant for every spring leads to the problem that longer springs may have larger influences at each step of the interpolation ($f = k\Delta x$). To counterbalance this effect, we set the spring constant to be inversely proportional to the rest length:

$$k_q \propto \frac{1}{r_q}.$$

The strategy ensures every spring, no matter what its rest length is, contributes similar amount of force during the interpolation process.

4 Physical Interaction

Since the interpolation framework is based on physical-based simulation (mass-spring system), our method is able to physically interact with other objects during the interpolation. During each interpolation step, we perform collision detection between the mass-spring system and obstacles. An axis-aligned bounding box (AABB) tree structure is built for both objects to accelerate the detection. The following procedures are executed once collisions are reported:

1. Move the intersecting vertices back to the surface of the obstacle according to the penetration normals.
2. Enforce all the intersecting vertices as additional boundary conditions. However, sometimes the deformation might be too large such that both surface and internal vertices (as described in Section 3.1) are involved in the intersection. In this case, we only use surface vertices as the boundary conditions. The internal vertices never become fixed due to a collision in order to preserve the internal structure.
3. Recompute the equilibrium state.

This produces a result which preserves the original shape's features as much as possible, while respecting to the collision constraints. Figures 2 and 5 show interpolation results while physical interactions are applied.

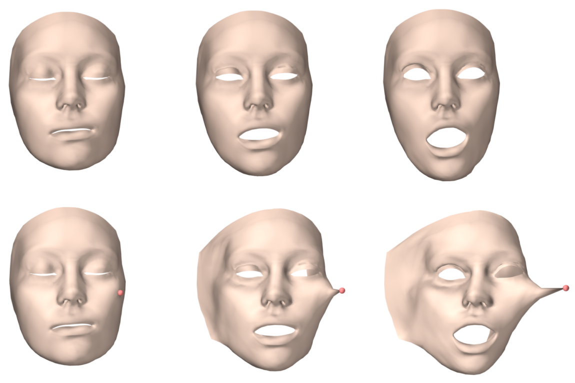
Figure 5: Physical interactions. The top row shows the original interpolation. The bottom row shows the same interpolation while a vertex is pulled away from the face. Parts of the mesh boundary are fixed as boundary conditions.

5 Results

Figure 6 shows several results (yellow) resulting from interpolation of the input shapes (blue). The supplemental video provides additional examples of our technique. Our interpolation method is able to produce visually pleasant motions from just a few example shapes (e.g. the opening and closing of the hand). It also successfully demonstrates physical interaction capabilities.

Implementation. A sparse matrix solver is required for Eq. (3). For our implementation we have tried both the CPU-based PARDISO [28, 29], and the GPU-based Cusp [5]. SOLID [34] is used for collision detection. Table 1 lists the statistics of each shape interpolation. All the results are generated on a desktop computer with a 2.66GHz Intel Core 2 Quad CPU, an NVIDIA Quadro FX 580 GPU, and 3.0GB main memory. The actual execution time for each Newton–Raphson step is to a large extent determined by the complexity of the mass-spring system.

Limitations. A mass-spring system may have more than one equilibrium. This leads to an element inversion problem. We found that the element inversion problem is more likely to occur if the input shapes originally contains folds due to self-intersecting triangles.

6 Conclusion and Future Work

In this paper, we presented a new blenshape technique where the interpolated shape is defined as the equilibrium state of a interpolated

Shape	n_v	n_s	t_{CPU}	t_{GPU}
Man	2.2k	21.6k	2.23	3.62
Face	8.1k	65.1k	40.11	3.67
Cat	7.3k	81.7k	67.58	13.67
Horse	8.5k	96.1k	81.36	15.95
Hand	18.6k	150.4k	153.99	26.28

Table 1: Statistics on the size of the input shapes and average running time per interpolation step (in seconds), assuming there are 36 steps for the interpolation between the source and target for all the experiments. n_v: number of vertices (including internal and anchor vertices), n_s: number of springs, t_{CPU}: running time with the PARDISO solver, t_{GPU}: running time with the Cusp solver.

mass-spring system. The proposed technique is fully automatic, requires no additional geometrical analysis or skeleton, and generates physically plausible shape interpolation with low distortion of surface area and volume. It also follows the linear blending control strategy used in traditional blendshapes. Digital artists who are familiar with traditional linear blendshape technique can potentially convert to our method without much efforts.

Our currently implementation does not achieve real-time performance, however for post production use this is not a critical issue. We are still exploring how to accelerate this technique. One possible method is to compute the deformation using a multi-resolution strategy, or to enforce the positive definiteness of the global stiffness matrix and use a faster conjugate gradient solver as described in [32]. Our current implementation for collision handling is also very simplified (e.g. let the contact vertices be fixed by assigning them as the boundary conditions). Friction force should be considered for better simulation. We would also like to investigate the effect of heterogeneous spring constants analogous to those in [25].

7 Acknowledgement

The authors would like to thank reviewers for their comments and suggestions. We are also grateful to the help from J.P. Lewis at Weta Digital and Jernej Barbič at University of Southern California during the preparation of this paper. This research was partially supported by the National Science Council, Taiwan under NSC98-2221-E-002-140-MY2.

References

[1] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *Proceedings of SIGGRAPH 2000*, pages 157–164.

[2] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 1998*, pages 43–54.

[3] J. Barbič, M. da Silva, and J. Popović. Deformable object animation using reduced optimal control. *ACM Transactions on Graphics*, 28(3):1–9, 2009.

[4] W. Baxter, P. Barla, and K. Anjyo. Rigid shape interpolation using normal equations. In *Proceedings of the 2008 International Symposium on Non-Photorealistic Animation and Rendering*, pages 59–64.

[5] N. Bell and M. Garland. Cusp: Generic parallel algorithms for sparse matrix and graph computations, 2010.

[6] J. Bloomenthal. Medial-based vertex deformation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 147–151.

[7] G. Borshukov. Making of the superpunch. In *ACM SIGGRAPH 2005 Courses*.

[8] C. Bregler, L. Loeb, E. Chuang, and H. Deshpande. Turning to the masters: motion capturing cartoons. In *Proceedings of SIGGRAPH 2002*, pages 399–407.

[9] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. *Proceedings of SIGGRAPH 1989*, pages 243–252.

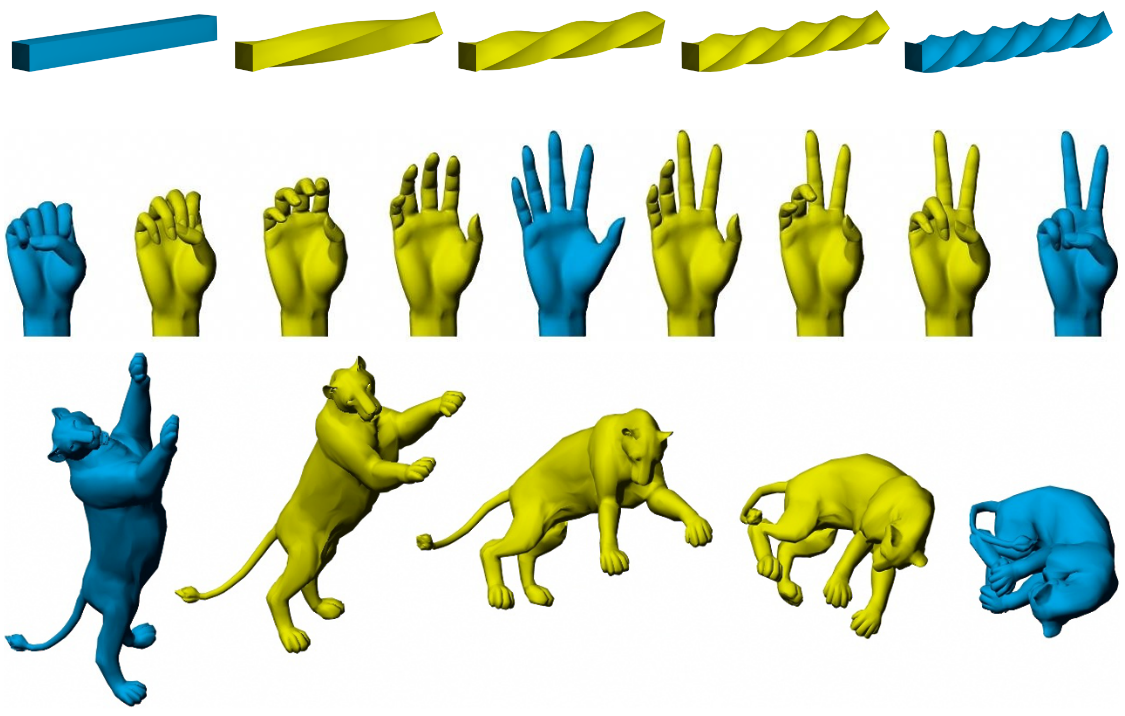

Figure 6: Results of shape interpolation using the proposed method. The input shapes are blue, while interpolated shapes are yellow. Table 1 lists the number of vertices, number of springs, and timings for each shape.

[10] K.-J. Choi and H.-S. Ko. Stable but responsive cloth. In *Proceedings of SIGGRAPH 2002*, pages 604–611.

[11] H.-K. Chu and T.-Y. Lee. Multiresolution mean shift clustering algorithm for shape interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):853–866, 2009.

[12] K. G. Der, R. W. Sumner, and J. Popović. Inverse kinematics for reduced deformable models. *ACM Transactions on Graphics*, 25(3):1174–1179, 2006.

[13] M. Desbrun, P. Schröder, and A. Barr. Interactive animation of structured deformable objects. In *Proceedings of the Graphics Interface 1999*, pages 1–8.

[14] N. Galoppo, M. Otaduy, W. Moss, J. Sewall, S. Curtis, and M. Lin. Controlling deformable material with dynamic morph targets. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, pages 39–47.

[15] J. Huang, X. Shi, X. Liu, K. Zhou, L.-Y. Wei, S.-H. Teng, H. Bao, B. Guo, and H.-Y. Shum. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics*, 25(3):1126–1134, 2006.

[16] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics*, 24(3):1134–1141, 2005.

[17] M. Kilian, N. J. Mitra, and H. Pottmann. Geometric modeling in shape space. *ACM Transactions on Graphics*, 26(3):1–8, 2007.

[18] R. Kondo, T. Kanai, and K.-i. Anjyo. Directable animation of elastic objects. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 127–134.

[19] Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *Proceedings of SIGGRAPH 1995*, pages 55–62.

[20] J. P. Lewis and K.-i. Anjyo. Direct manipulation blendshapes. *IEEE Computer Graphics and Applications*, 30(4):42–50, 2010.

[21] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of SIGGRAPH 2000*, pages 165–172.

[22] A. Liu, F. Tendick, K. Cleary, and C. Kaufmann. A survey of surgical simulation: applications, technology, and education. *Presence: Teleoperators and Virtual Environments*, 12(6):599–614, 2003.

[23] G. S. P. Miller. The motion dynamics of snakes and worms. pages 169–173.

[24] L. P. Nedel and D. Thalmann. Real time muscle deformations using mass-spring systems. In *Proceedings of the Computer Graphics International 1998*, pages 156–165.

[25] T. Popa, D. Julius, and A. Sheffer. Material-aware mesh deformations. In *Proceedings of the 2006 IEEE International Conference on Shape Modeling and Applications*, page 22.

[26] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. In *Proceedings of SIGGRAPH 1991*, pages 349–358.

[27] D. Rohmer, S. Hahmann, and M.-P. Cani. Exact volume preserving skinning with shape control. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 83–92.

[28] O. Schenk, M. Bollhofer, and R. A. Roemer. On large-scale diagonalization techniques for the Anderson model of localization. *SIAM Journal on Scientific Computing*, 28(3):963–983, 2006.

[29] O. Schenk, A. Wächter, and M. Hagemann. Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Journal of Computational Optimization and Applications*, 36:321–341, 2007.

[30] K. Shoemake. Animating rotation with quaternion curves. In *Proceedings of SIGGRAPH 1985*, pages 245–254, 1985.

[31] H. Si and K. Gärtner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 2005 International Meshing Roundtable*, pages 147–163.

[32] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 181–190.

[33] X. Tu and D. Terzopoulos. Artificial fishes: physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH 1994*, pages 43–50.

[34] G. van den Bergen. SOLID: Software library for interference detection.

[35] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman. Context-aware skeletal shape deformation. *Computer Graphics Forum*, 26(3):265–274, 2007.

[36] T. Winkler, J. Drieseberg, M. Alexa, and K. Hormann. Multi-scale geometry interpolation. *Computer Graphics Forum*, 29(2):309–318, 2010.

[37] H.-B. Yan, S. Hu, R. R. Martin, and Y.-L. Yang. Shape deformation using a skeleton to drive simplex transformations. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):693–706, 2008.

[38] S. Yoshizawa, A. G. Belyaev, and H.-P. Seidel. Free-form skeleton-driven mesh deformations. In *Proceedings of the 2003 ACM Symposium on Solid Modeling and Applications*, pages 247–253.