具有物理互動特性之混合形變模型

馬萬鈞
USC ICT
ma@ict.usc.edu

王怡華
國立臺灣大學
always@cmlab.csie.ntu.edu.tw

Graham Fyffe
USC ICT
fyffe@ict.usc.edu

陳炳宇
國立臺灣大學
robin@ntu.edu.tw

Paul Debevec
USC ICT
debevec@ict.usc.edu

ABSTRACT

在本論文中， 我們提出了一種利用質量彈簧系統（mass-spring system）中改變彈簧靜長度（rest length）來模擬混合形變模型（blendshape model）的形狀內插變化之新技術。應用質量彈簧系統來模擬物體的移動與變形，在動畫模擬的領域裡相當常見，尤其是在模擬粒子、衣物以及紡織物等物體時更是被大量使用。 而在混合形變模型的領域中，要找到兩個物體中間的形狀，除了最直覺的線性內差之外，也還有許多其他方法，但至今尚未有人使用質量彈簧系統來模擬中間的形狀。將質量彈簧系統導入混合形變模型除了可以保持原有之形狀內插的特性之外，中間產生出的模型由於同時具有質量彈簧系統的物理模擬特性，亦可提供與其他物體之互動使用。 為達此一目的， 我們首先分別將來源模型與目標模型建構出有一致性的兩個質量彈簧系統（即頂點、邊以及面都具有相同的拓樸結構）。要產生一個中間的形狀，我們首先創建一個新的質量彈簧系統，其結構也和來源模型及目標模型一致， 接著根據不同的權重，將相對應的彈簧做線性內插以得到一組新的彈簧靜長度。之後再經過計算，便可找出基於這些內插後，產生的彈簧靜長度在達到平衡時候的系統狀態。經由類似的步驟，此方法也適用於同時混合多種形狀。此外，結合適當的碰撞偵測與處理，內插所產出的中間模型亦可與其他的物體做互動。

Categories and Subject Descriptors

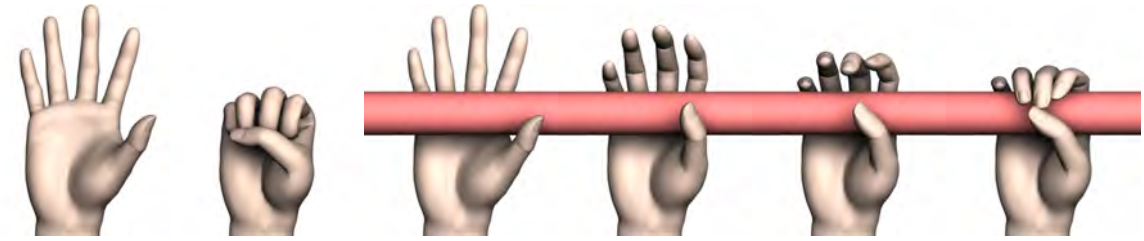
I.3.5 [**Computational Geometry and Object Modeling**]: Physically based modeling; I.3.7 [**Three-Dimensional Graphics and Realism**]: Animation; I.6.8 [**Types of Simulation**]: Animation

1. INTRODUCTION

Shape interpolation, or so-called blendshapes, is a crucial tool for many applications that requires a continuous geometrical shift between two or more input shapes. Keyframe animation of 3D characters is one such application. In each frame of an animation, the vertices are interpolated between the pre-stored keyframe shapes. Another typical application is facial animation. Shapes of different facial expressions, often referred to as key poses, are defined as deformations of the face geometry. A new shape of a desired expression can be fully or partially blended from those key poses. Nowadays, shape interpolation is a very common tool in most of the commercial 3D computer graphics software. The simplest shape interpolation is carried out through linearly interpolating vertex positions from one shape to another. However, the interpolation results are often not satisfactory when the deformation involves with large rotations. Figure 2(c) illustrates a typical "shrinking" effect observed when linearly interpolating two shapes (a) and (b) with a large rotation presents.

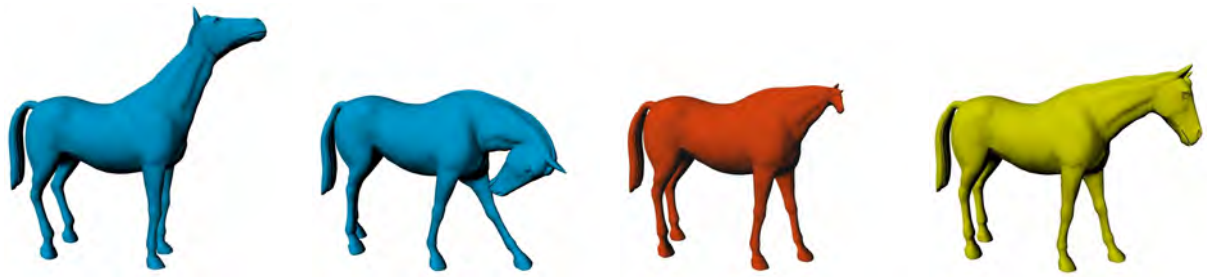
In this paper we introduce a new physically-motivated shape interpolation technique based on mass-spring systems. Our method seeks to associate shape interpolation with physical-based deformation. A key observation is that an equilibrium state of a mass-spring system minimizes local area/volume distortions through force balancing. As a result, local rigidity can be maintained during the deformation. The method begins with building a mass-spring system based on the structure of the input shapes (assuming both of them have the same topology). Specifically, the mass-spring system contains with three types of springs: (1) *structure springs* that model the elastic nature of the surface, (2) *bending springs* that maintain the rigidity of the shape, and (3) *internal springs* to approximately preserve volume. We interpolate the corresponding rest lengths of the springs based on the interpolation factor, and solve for the equilibrium state of the interpolated mass-spring system to produce the final interpolated shape. The proposed method yields more natural shape interpolations, which can be seen in Figure 2(d).

Shape interpolation techniques usually do not consider physical interaction. Any deformation caused by physical interaction has to be prepared with either a new shape that conforms to the physical interaction, or imitated by a post-editing process. The former usually is not applicable to arbitrary physical interaction otherwise there will be too many key poses need to be sculpted to accommodate all possible deformations and the number of key poses is typically limited. The later requires extra manual efforts or simulations to produce visually-pleasant results. For example,



(a) (b) (c) (d) (e) (f)

Figure 1: The proposed physically-motivated blendshape technique. (a) Source shape. (b) Target shape. (c-f) Interpolating input shapes while interacting with an obstacle.



(a) (b) (c) (d)

Figure 2: Linear interpolation versus our technique. (a) Source shape. (b) Target shape. (c) Linearly interpolated result with $\alpha = 0.5$. (d) Interpolated with the proposed method with the same α.

Borshukov [8] demonstrated both possibilities. Our shape interpolation method is based on mass-spring system and it is very straight forward to be associated with physical interaction. It can be easily carried out by applying external forces or additional constraints on the interpolation.

Our interpolation method is well suited for natural deformations due to the physically-motivated underlying mass-spring system. Simulating the equilibrium of the vertex positions is the main computational cost of the technique. The core computation is solving a sparse linear system, which can be accelerated with current graphics processor. We demonstrate our shape interpolation technique with a wide variety of shapes that exhibit complicated geometry and deformations.

Contributions. There are two substantial contributions in our work:

1. A solution which creates natural looking shape interpolation results based on the mass-spring systems. The proposed method requires no geometric analysis, articulated skeleton, or any manual intervention. Since it does not require a skeleton to drive the deformation, and thus is not limited to articulated shapes.
2. Physical interaction can be achieved under the same framework with additional collision detection and handling.

2. RELATED WORK

Our technique leverages a significant body of 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. Linear vertex interpolation (the "blendshapes" technique) is the most common method for shape interpolation. However, it suffers from artifacts such as causing the moving parts (e.g. arms and legs) to shrink and collapse. Shape interpolation can also be achieved using an articulated skeleton. The skeleton may be manually specified [33, 35], or automatically determined by finding near-rigid components of input shapes [11] or using the medial axis transform [7, 36]. Rohmer *et al.* [25] proposed a skinning method which exactly preserves (or controls) the volume of an object.

Other methods are free of using an articulated skeleton. One favorable trend is based on maintaining the rigidity criteria of local geometrical elements, or so-called "as-rigid-as-possible". [1] and [14] are typical examples of this method. Baxter *et al.* [5] proposed a solution for solving the rotation ambiguity arose from previous as-rigid-as possible approaches. Winkler *et al.* [34] 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 which conforms to user manipulation. Barbič *et al.* [4] 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, but our method simply interpolates spring rest lengths. Kondo *et al.* [16]

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 keyframe. Lewis and Anjyo [18] introduced a direct manipulation method for blendshape. This approach constrains any desired subset of vertices based on artist's direct manipulations and automatically infer the remaining points' positions. However, this method only produces shapes that are within the convex space of the original blendshape poses.

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. Teran *et al.* [30] 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. [19] 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 interpolates shape via physical simulation, presumably requiring fewer input shapes. Kilian *et al.* [15] presented an isometric deformation method based on Riemannian geometry, considering shape interpolation as a geodesic curve in shape space. Both techniques seem to be problematic to provide physical interaction capability because they are not modeled in the sense of classical mechanics.

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 [20], dynamics for animals [21], cloth [2, 10, 13], muscles [9, 22], and other deformable objects. Lee *et al.* [17] 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 for their low computational complexity.

Rest Length Animation. The idea of animating a mass-spring system by varying spring rest lengths is not new. Most of the methods simulate muscle activation through controlling the rest length of each spring. Raibert and Hodgins [24] 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 [31] 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 SPRING-SPACE INTERPOLATION

We interpolate shapes in a physically plausible way by interpolating the rest length parameters of a mass-spring system. This interpolation scheme consists of the following components:

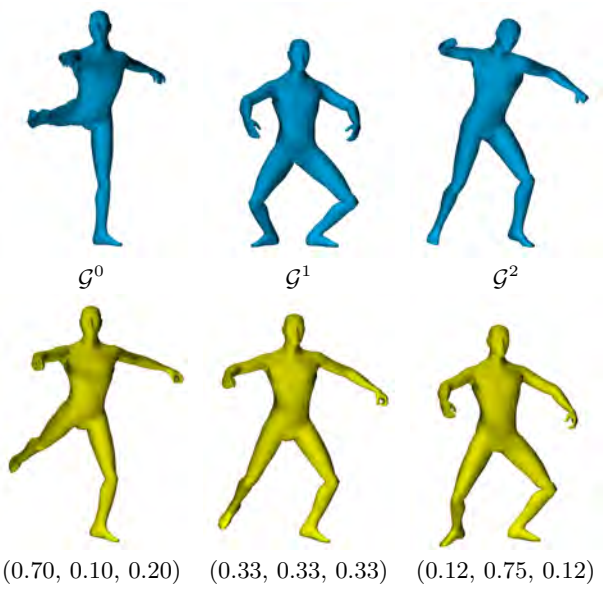


Figure 3: Blending between multiple shapes. The top row shows three source shapes and the bottom row shows blended results with corresponding weights shown as (w_0, w_1, w_2).

1. Given a set of input shapes, we start by constructing a mass-spring system based on their common structure.

2. Next, we interpolate rest lengths between two or more poses to deform the shape by an intermediate mass-spring system. The interpolated result is the quasi-static state of the system given the interpolated rest lengths.

A mass-spring system $\mathcal{M} = \langle \mathcal{V}, \mathcal{S} \rangle$ is defined by a collection of vertices $\mathcal{V} = \{v_i | i = 1...n_v\}$ connected by springs $\mathcal{S} = \{s_q | q = 1...n_s\}$. 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.

To perform shape interpolation, we first 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 two meshes $\mathcal{G}^0$ and $\mathcal{G}^1$ which are vertex-wise correspondence and have the same topology. Inherently the two consistent mass-spring systems have the same structure.

3.1 Solving Intermediate Shapes

The two mass-spring systems are initially set to be in their equilibrium, therefore $r_q^0 = \| v_{e_{q0}}^0 - v_{e_{q1}}^0 \|$, and r_q^1 can be computed similarly. 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,

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

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

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

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 formulation is based solely on quasi-static states. We also assume that each vertex has the same mass, therefore the mass term can be removed from the equation. 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:

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

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. We are ultimately interested in the equilibrium state, and that makes $f(x_{t+1}) = 0$. Eq. (3) now becomes:

$$J(x_t)\Delta x_t = -f(x_t). \quad (4)$$

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

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

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:

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

Generally matrix J is very sparse. There are both iterative (e.g. conjugate gradient) and direct methods for solving this sparse linear system. x_0 is initialized as the vertex positions of 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 $\bar{\mathcal{V}}$.

3.2 Blending Multiple Shapes

The proposed method can also be seen as blendshapes in spring-space. Consequently, many of the capabilities of classical blendshapes can be easily transfer to our method. For example, blending between multiple shapes in spring-space can be easily done by writing the rest length as a convex linear combination of the spring rest lengths from the input shapes:

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

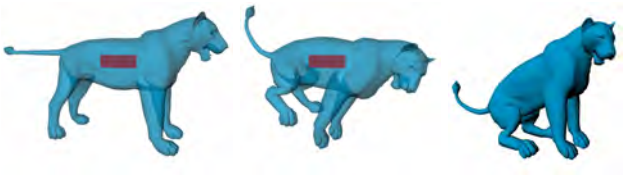
where $\sum_{i=0}^{n_b} w_i = 1$. This is illustrated in Figure 3 where the interpolated shapes (shown in yellow) are the results of linearly-blended spring rest lengths from three different input shapes (shown in blue).

3.3 Boundary Conditions

Boundary conditions must be specified in order to solve the equilibrium of a mass-spring system. Without setting the boundary conditions, the system will be under-constrained and the solution will not be unique. In a mass-spring system, boundary conditions are vertices which are fixed. Practically, 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 (and automatic) 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 input meshes. Alternatively, certain vertices on the surface can be manually marked as boundary conditions. During the interpolation, the positions of the marked vertices are then simply 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.

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 topology of the mass-spring system (as in Section 3), we connect additional springs to the vertices of the shape and 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. In our examples, we use a rectangular plane containing six springs as an anchor.
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



(a) (b) (c)

Figure 4: Using an internal anchor for boundary conditions. (a) A source shape with an anchor object (a rectangle plane, shown in red) inside. (b) The interpolated result. The vertices of the anchor act as the boundary conditions during the spring relaxation, making all the surface vertices move. (c) The interpolated result with a rigid transformation applied.

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 [28]. 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.

3.4 Structure, Bending and Internal Springs

We formulate the mass-spring system as a combination of *structure springs* and *bending springs* (similar to [10]). The structure springs model the elastic properties of the object's 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. In addition, we may add springs, or even extra vertices, inside the shape to help preserve volume and prevent the shape from collapsing on itself. Otherwise a large surface tends to be crumpled due to insufficient numerical precision. Constrained Delaunay tetrahedralization [29] is a good method for determining *internal springs*. To prevent springs with long rest lengths we can 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 3.3.

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 (e.g. generate larger force) at each step of the interpolation. To counterbalance this effect, we set the spring constant to be inversely proportional to the rest length:

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

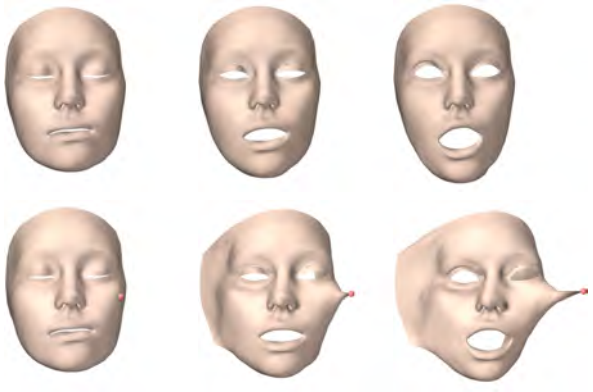


Figure 5: Physical interactions. The top row shows the original interpolation. The bottom row shows the same interpolation while a vertex is pulled away from the face. Parts of the mesh boundary are fixed as boundary conditions.

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 3.4) are involved in the intersection. In this case, we only use surface vertices as the boundary conditions.
3. Recompute the equilibrium state.

We only apply detection for the surface vertices (not internal vertices), so that the internal vertices never become fixed due to a collision and fail to preserve the internal structure. This produces a result which preserves the original shape's features as much as possible, while respecting to the collision constraints. Figures 1 and 5 show the results after physical interaction being applied.

5. RESULTS

Figure 6 shows five set of interpolated shapes (yellow) from selected interpolation steps between the input shapes (blue). The supplemental video provides additional examples of our technique. Our interpolation method can reproduce visually pleasant motions from just a few example shapes (e.g. the

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). 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.

opening and closing of the hand). The proposed method is easier to apply and also provides physical interaction capabilities. However its computational cost is higher than as-rigid-as possible methods.

Implementation. A sparse matrix direct solver is required to solve Eq. (3). We have adopted both PARDISO, a CPU-based solver [26, 27], and Cusp, a GPU-based solver [6]. A good mass-spring system framework can be found at [3]. We also use SOLID [32] for collision detection. Table 1 lists the statistics of each shape interpolation. We select a variety of 3D shapes, with the number of vertices range from 2k to 18k. 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 element inversion problem. We found that the element inversion problem is more likely to occur if the input shapes contains folds due to self-intersecting triangles.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel physically-motivated shape interpolation technique which linearly interpolates the rest lengths of a mass-spring system. The interpolated shape is then defined as the equilibrium state of the interpolated system. Our method requires neither additional geometrical analysis nor skeleton and interpolates shapes in a straightforward but physically plausible way. We showed the effectiveness of the method on a wide variety of shapes exhibiting natural deformations. The proposed method is capable of producing fully automated shape interpolation, with low distortion of surface area and volume. While our method does not guarantee volume preservation, it still maintains volume as much as possible due to the internal springs. These springs are necessary to prevent the shape from collapsing.

Several directions have been thought of for future improvements. The proposed interpolation method is computationally intensive due to the large linear system which must be solved to determine the equilibrium vertex positions. It is possible to enforce the positive definiteness of the global stiffness matrix and use a faster conjugate gradient method to solve the linear system as described in [30]. For physical interaction, we currently used a simplified method for collision handling (e.g. let the contact vertices be fixed by assigning them as the boundary conditions). For better simulation we should consider modeling friction force. We would also like to investigate the effect of heterogeneous spring constants that is analogous to [23].

7. REFERENCES

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

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

[3] J. Barbič. Computer graphics research code: 3D mass-spring system, 2009. `//www-bcf.usc.edu/~jbarbic/code/`.

[4] J. Barbič, M. da Silva, and J. Popović. Deformable object animation using reduced optimal control. *ACM Transactions on Graphics*, 28(3):53:1–53:9, 2009. (SIGGRAPH 2009 Conference Proceedings).

[5] W. Baxter, P. Barla, and K. Anjyo. Rigid shape interpolation using normal equations. In *Proceedings of the 6th International Symposium on Non-Photorealistic Animation and Rendering*, pages 59–64, 2008.

[6] N. Bell and M. Garland. Cusp: Generic parallel algorithms for sparse matrix and graph computations, 2010. `//cusp-library.googlecode.com/`.

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

[8] G. Borshukov. Making of the superpunch. In *ACM SIGGRAPH 2005 Courses*, pages 19:1–19:3, 2005.

[9] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. *ACM SIGGRAPH Computer Graphics*, 23(4):243–252, 1989. (SIGGRAPH 1989 Conference Proceedings).

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

[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. (SIGGRAPH 2006 Conference Proceedings).

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

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

[15] M. Kilian, N. J. Mitra, and H. Pottmann. Geometric

modeling in shape space. *ACM Transactions on Graphics*, 26(3):64:1–64:8, 2007. (SIGGRAPH 2007 Conference Proceedings).

[16] 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, 2005.

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

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

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

[20] 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.

[21] G. S. P. Miller. The motion dynamics of snakes and worms. *ACM SIGGRAPH Computer Graphics*, 22(4):169–173, 1988. (SIGGRAPH 1988 Conference Proceedings).

[22] 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, 1998.

[23] 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, 2006.

[24] M. H. Raibert and J. K. Hodgins. Animation of dynamic legged locomotion. *ACM SIGGRAPH Computer Graphics*, 25(4):349–358, 1991. (SIGGRAPH 1991 Conference Proceedings).

[25] 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, 2009.

[26] 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.

[27] 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. In *Journal of Computational Optimization and Applications*, volume 36, pages 321–341, 2007.

[28] K. Shoemake. Animating rotation with quaternion curves. *ACM SIGGRAPH Computer Graphics*, 19(3):245–254, 1985. (SIGGRAPH 1985 Conference Proceedings).

[29] H. Si and K. Gärtner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable*, pages 147–163, 2005.

[30] 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, 2005.

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

[32] G. van den Bergen. SOLID: Software library for interference detection, 2004. `//www.win.tue.nl/~gino/solid/`.

[33] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman. Context-aware skeletal shape deformation. *Computer Graphics Forum*, 26(3), 2007. (Eurographics 2007 Conference Proceedings).

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

[35] 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.

[36] S. Yoshizawa, A. G. Belyaev, and H.-P. Seidel. Free-form skeleton-driven mesh deformations. In *Proceedings of the 8th ACM Symposium on Solid Modeling and Applications*, pages 247–253, 2003.

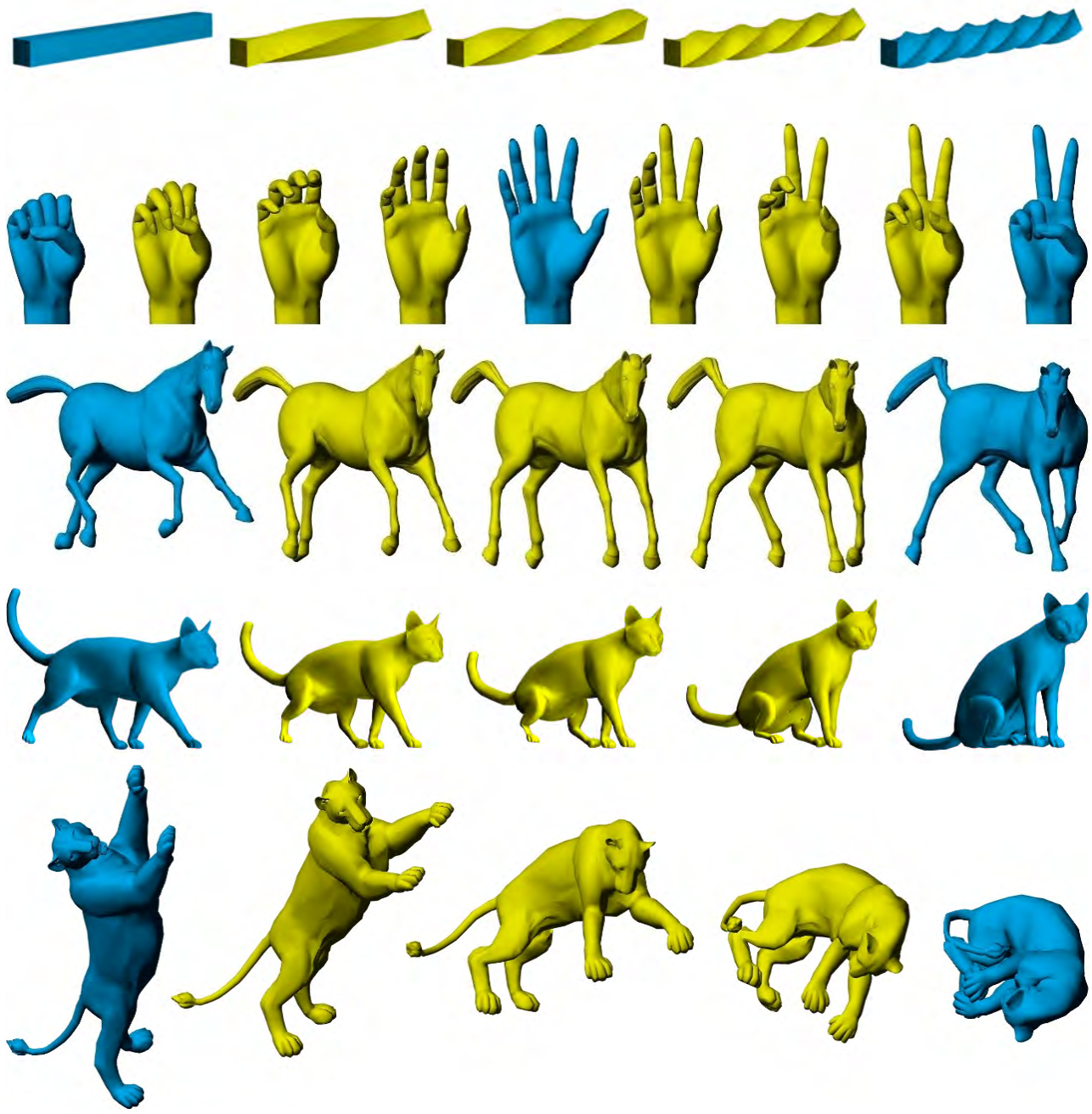


Figure 6: Results of shape interpolation using the proposed method. The input shapes are blue, while interpolated shapes are yellow. Table 1 lists the number of vertices, number of springs, and timings for each shape. The set of hand shapes in the second row is identical to the one used in Figure 1.