Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2005)
K. Anjyo, P. Faloutsos (Editors)

Directable Animation of Elastic Objects

Ryo Kondo[1] Takashi Kanai[2] Ken-ichi Anjyo [3]

[1] Keio University, Japan
[2] RIKEN, Integrated Volume-CAD System Research Program, Japan
[3] OLM Digital, Inc.

Abstract

There is a crucial demand in the computer animation industry to make animations that blend animator-specified expressive motion with physics-based realism. We propose a novel framework to create directable animation of elastically deformable objects. The directable animation is created with animator-specified keyframes and the motion trajectory of the deformable object, while maintaining a plausible realism. Our framework mainly consists of two complementary approaches. The first is a method to control the time-varying geometry of an elastic object, using a loose key-framing technique. In our keyframing, we introduce an FEM-based elastic deformation algorithm that allows us to rearrange the elastic object motion, guided by the shape or pose specified at each keyframe. The second is a motion compensation technique, which allows us to rearrange the physical behavior of elastically deformable objects under a user-specified trajectory. The animation examples demonstrate that our framework provides plausibly realistic deformation animations with greater controllability and usability than existing approaches.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

In the computer animation industry, the ultimate goal for realistic animation is not a completely physics-based reality, but rather a plausible reality. Keyframe control is then the most primitive, intuitive methodology for an animator to create what he or she wants to make, according to his or her intention. On the other hand, physical simulation has also become widely used in making realistic animation. Keyframe control and physical simulation therefore represent two extremes in that the former is entirely intentional, whereas the latter dutifully obeys physical laws. As a crucial and practical demand, however, we wish to achieve both physics-based realism and user-specified expressive motion, in order to achieve a desired plausibly realistic, expressive animation. Trajectory control of physical objects is another important issue. Recent research has tackled the challenge of meeting such practical demands, as in the results for keyframing of smoke simulation [TMPS03, FL04, MTPS04, SY05] and for trajectory control of rigid body animation [PSE*00, PSE03]. However, these successes show only a first step toward a novel paradigm of making realistic physics-based animations that are still controllable by user's intention. In this paper we refer to such realistic, yet controllable animations as directable animations.

This paper presents a new framework for making elastic body animation directable. In our framework the elastic motion trajectory is first defined by the user. Animating elastic objects is then achieved by our keyframing technique. We introduce an FEM-based elastic deformation algorithm, which allows us to rearrange the elastic object motion, guided by the shape or pose specified at each keyframe. A motion compensation algorithm is also introduced to move the elastic body along a user-specified trajectory. Our prototype system and animation examples demonstrate that this framework can offer great controllability over a physical simulation using simple keyframes and curve editing, to make a directable animation. Though the resulting animations may not exhibit exact physics-based reality, we believe that the animations still provide plausible realism, derived from the above new algorithms.

The rest of the paper is organized as follows. In section 2 we briefly review the relevant related work. Section 3 presents our framework to make directable elastic body animations. In section 4 our loose keyframing method is detailed, along with the FEM-based deformation algorithm. In section 5 we describe our trajectory control method, featuring a novel motion compensation algorithm. In section 6 we show our experimental results and discuss the advantages and limitations of our approach. Section 7 presents conclusions and suggestions for future work.

2. Related Work

Relating to our animation framework, we review physically based approaches to generate motion for animation, emphasizing how to control physical simulation models to get desired animation.

A number of optimization-oriented techniques have been developed for animations of articulated figures, rigid bodies, or fluids. For example the space-time constraint approaches [WK88, GTH98, FvdPT01, Gle01] for articulated bodies have been well studied; these enable automatic generation of animation through kinematics. As for controlling fluids, especially smoke, several methods have been recently developed [TMPS03,FL04,MTPS04,SY05]. The basic ideas among them are to define a suitable error objective function and then to solve the optimization problem on that objective function, while a user specifies the desired density field in advance.

The methods for controlling the motion trajectory for rigid body animation have also been explored. Successful results include a method for generating plausible collision variations between objects [BHW96] and a method for multibody control using a Markov chain Monte Carlo algorithm [CF00]. These lead to a general approach where the motion trajectory of rigid bodies is controlled by changing the initial conditions or physical parameter values at an impact time. The interactive approaches using user-specified motion [PSE*00, PSE03] allow one to intuitively control the rigid motion trajectory.

In dealing with deformable objects, existing approaches have mostly focused on fast processing or on stability improvements, and not on controllability ([MDM*02,HSO03, HFS03,ITF04] for instance). In the realm of controlled physical animation, a mass-spring system was developed for modeling the muscles of artificial fish [TT94], where the rest length of the spring model is modified to simulate muscle deformation. An interactive simulation method has also been proposed for deformable objects with underlying skeletal structure [CGC*02], where a resting shape is introduced to simulate skeleton-based deformation. The idea of using a resting shape for deformation in these works is carried over into our method, but we use it for dealing with general deformation of elastic objects, without need for detailed muscle or skeletal structure of the objects. We also note that the previous approaches still stay in the realm of rigorous physical simulation so that they do not allow one to control deformation of soft objects in an intuitive and general way.

We propose a new methodology for intentionally controlling the time-varying shape and trajectory of elastic objects. In our formulation the resulting animation retains a degree of plausible realism, while allowing an animator to specify both the shape deformation and the trajectory of the elastic objects.

3. Overview of Directable Animation Framework

In this section, we describe our framework to establish directable animation using physically-based elastic deformation. Using the physically-based animation techniques described in Section 2 is the preferred means for deforming elastic objects with *physical realism*. However, using only physically-based animation is inadequate for plausible deformation for users. Since in the existing physically-based approaches, motions of elastic objects are automatically calculated according to given external forces and internal physical attribute parameters, these motions are very difficult to control. The only way to make the final motions plausible is to modify external forces or internal parameters, which are virtually impossible tasks.

On the other hand, in our situation the final results of motions are already *known*, that is, some physical behaviors of an object are partially specified in advance by the animator. In this case, one problem is how the whole behavior can be efficiently constructed from the known portions of the behavior of an object. In our framework, there are at least two functions which should be implemented to construct a plausible motion:

- A function with which the local geometry of an object at a given time can be edited as the user desires.
- A function with which the trajectory of an object can be edited as the user desires.

A framework involving the above two functions while using physically-based animation is required to establish plausible motion. It should, however, be noted that the fulfillment of this framework presents a trade-off between controllability and visual plausibility. As demonstrated later, our framework allows the user to easily generate plausibly realistic animation through interactive editing, while using a rigorous physics based motion as the initial input.

Figure 1 illustrates an overview of our framework for the editing of elastic body animation. This framework mainly consists of the following three phases:

Physically-based elastic body animation. We first calculate motions of objects using physically-based elastic body animation technique (Figure 1(a)). We use Müller et al.'s approach [MDM*02] for this calculation. In

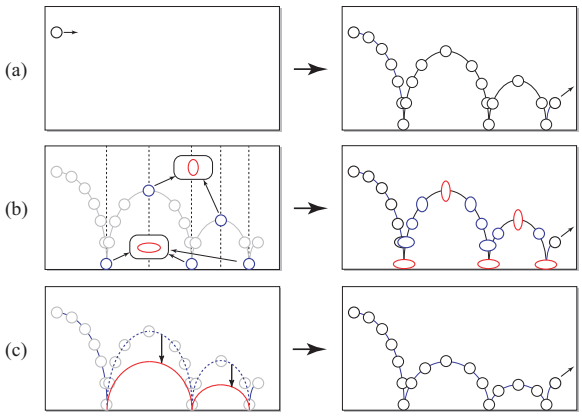

Figure 1: *Overview of our directable animation framework. (a) Physically-based elastic animation. (b) Deformation control of elastic bodies. (c) Trajectory control in elastic animation.*

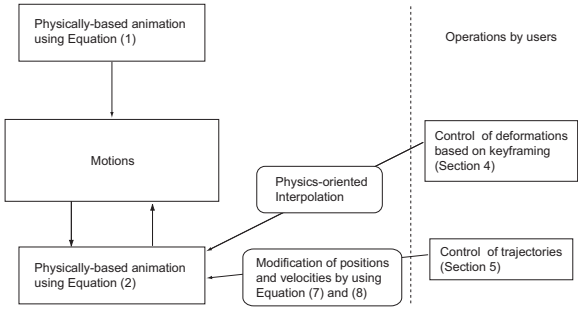

Figure 2: *A diagram of operations in our directable animation framework.*

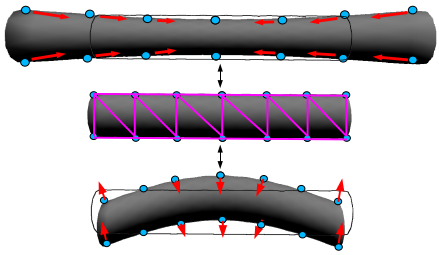

Figure 3: *The geometric meaning of elastic body animation.*

[MDM*02], objects are automatically decomposed into a set of tetrahedral elements in advance. The following differential equation is solved by using the implicit Euler method:

$$M\ddot{x} + C\dot{x} + K(x - o) = f_{ext}, \qquad (1)$$

where M, C are the mass matrix and the damping matrix, f_{ext} is an external force, x is the position of each vertex, and $\dot{x}$ and $\ddot{x}$ are the first and second derivatives of x with respect to time. K and o are the stiffness matrix and the original position of each vertex on an elastic object. In the process of motion calculation, we record the positions and velocities of each vertex for all time to quickly retrieve the shape of an object at a given time in the timeline of an animation.

Trajectory control of objects. We next edit the trajectory of a point element in an object and the motion of an animation is rearranged according to the modified trajectory (Figure 1(c)). This guarantees that such a point element is completely on the modified trajectory. The motions of other elements in an object are also rearranged without sacrificing their visual plausibility (described later in Section 5).

Deformation control of objects. The shape of an object at a given time is edited and the animation is rearranged according to a set of modified shapes (Figure 1(b)). For this editing, we utilize a paradigm of *keyframing*: We set a keyframe for the shape of an object at a given time t_i and modify its shape. We finally re-calculate the motion of an animation according to the shapes of keyframes. Note that in our keyframing, the set of specified keyframes are used as constraints deformation guides, instead of exact positional ones. To satisfy the visual plausibility of such an animation, we use our novel FEM-based deformation algorithm called *physics-oriented interpolation* (described later in Section 4).

In the above framework, the fundamental tool is physical simulation. The system allows the user to modify the physical simulation interactively; by modifying shape deformations and the motion trajectory are then performed interchangeably until the user's intention is achieved. By recalculating the physically-based animation after each modification, visual plausibility can be preserved.

Figure 2 illustrates the relationship between operations by users and the physically-based animation. First, physical motions of elastic objects are computed by solving Equation (1). For these physically-based motions, users can modify their shapes by keyframing or by editing their trajectories. Motions are recalculated using Equation (2) with the results of these modifications (described in Section 4). These modifications are reflected by building them into the solution of the differential equation. By making successive modifications, the desired animation can be created.

4. Keyframe Control of Deformation

In this section we describe our deformation control approach based on a paradigm of keyframing in physically-based elastic body animation.

In the animation sequences of elastic objects, the user sets the shape G_i to a keyframe at a given time t_i. We call

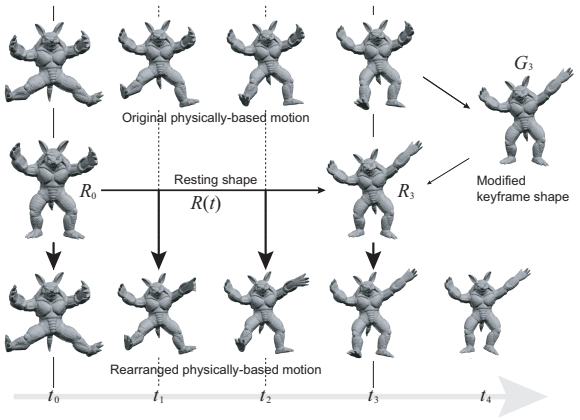

Figure 4: *Principle of our keyframe control for deformation.*

G_i a *keyframe shape*. Physically-based motions are then re-calculated for a set of keyframe shapes (t_i, G_i) $(i = 1 \ldots n$, n is the number of keyframes) defined by the user. On this setting, the re-calculated animation must not destroy the visual plausibility possessed by the original physically-based animation.

One possibility to address this condition is to use the solution of a two-boundary value problem [SGL99] or a space-time optimization technique [WK88, Coh92]. These solutions are effective only if the difference between two keyframe shapes is subtle. Moreover, these types of solutions require non-linear optimization approaches and these are often difficult to solve stably. In our case we take a different approach.

Before introducing the principle of our approach, we discuss the equation of motion (Equation (1)) for physically-based elastic body animation. In Equation (1), we particularly focus on the stiffness matrix K and the original position o. These parameters contribute strongly to the shape of an elastic object, and usually have constant values during the computation of an elastic body animation. Figure 3 shows the geometric meaning of these parameters. The original, non-stressed and non-strained shape of an elastic model (center shape in Figure 3) is at first defined. K and o are uniquely determined from this original shape. We call this shape the *resting shape*. When the resting shape is stressed or strained by instantaneous external forces, internal elastic forces of such a deformed shape are generated to restore the resting shape. A shape is then deformed according to attenuated internal elastic forces, and it finally settles to the equilibrium state as a resting shape.

Our basic idea here is to replace a resting shape at a keyframe to user-specified keyframe shape. By this, displacements to restore the replaced resting shape are generated at the time of a keyframe. Consequently, the object is deformed to approximate the user-specified keyframe shape. Note that the displacements between two neighbor keyframe shapes tend to be quite large. If we replace the resting shape with a user-specified keyframe shape only at the keyframes, unintended impulsive forces due to large displacements yielding undesirable vibrations are often generated.

To restrain such negative effects, we define here an interpolation function $R(t)$ between resting shapes so that their transitions can be gradually inserted into the elastic body animation. Time-varying stiffness matrix $K(t)$ and original position $o(t)$ are then calculated from $R(t)$. When recalculating elastic body animation, constant K, o are replaced by $K(t), o(t)$ in Equation (1) respectively,

$$M\ddot{x} + C\dot{x} + K(t)(x - o(t)) = f_{ext}. \tag{2}$$

We recalculate the elastic body animation using Equation (2) to get the final result, while employing the collision detection technique in [KK04].

Figure 4 illustrates the principle of our keyframe-based deformation control. In the original elastic body animation, a uniform resting shape R is used during the whole process of calculation. We next set up keyframe shapes and the user modifies them. In a user-defined keyframe shape (t_i, G_i), we replace the resting shape R_i with G_i, i.e., $R_i \leftarrow G_i$. We then calculate a function $R(t)$ which interpolates the set of resting shapes R_i to obtain $K(t)$ and $o(t)$. The equation of motion, Equation (2), is finally solved. This approach then not only reflects user-specified keyframe shapes in the elastic motions of objects, but also maintains the physically-based motions due to the recalculation of the elastic body simulation.

It should be noted that the keyframe shape and thus the replaced resting shape serve as *guides* to attract the current shape to the keyframe shape. Indeed, an approximated keyframe shape appears after a brief interval has elapsed after a keyframe. It can be seen in Figure 4 that a resting shape is set at a time t_3, while the effect of the keyframe actually appears at t_4, a bit after time t_3. From the user's point of view, it is not necessary to know about the resting shape R, that is, the user has only to modify a keyframe shape G. All calculations concerned with R are automatically processed by the system.

We also note that the elastic motions that are obtained by our method do not rigorously obey the laws of physics. Our method is physically oriented in the sense that we simply solve Equation (2), while controlling keyframe shape and motion trajectory. This means that our method enables the user to add his/her intention to the animation. As demonstrated later, this is a great advantage of our method over existing approaches in practical situations.

Physics-oriented interpolation

It is desirable that an interpolation function $R(t)$ for a set of resting shape $R_i (i = 1 \ldots n)$ generates intermediate objects

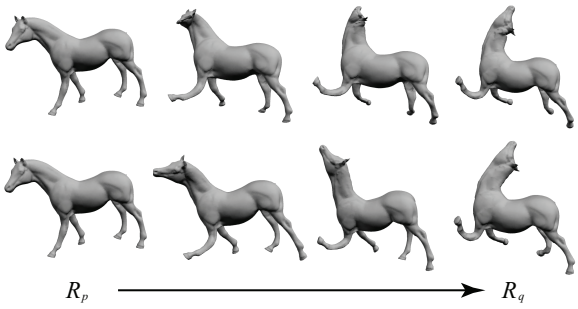

Figure 5: *Comparison of two interpolation approaches. Upper: linear interpolation. Bottom: our physics-oriented interpolation.*

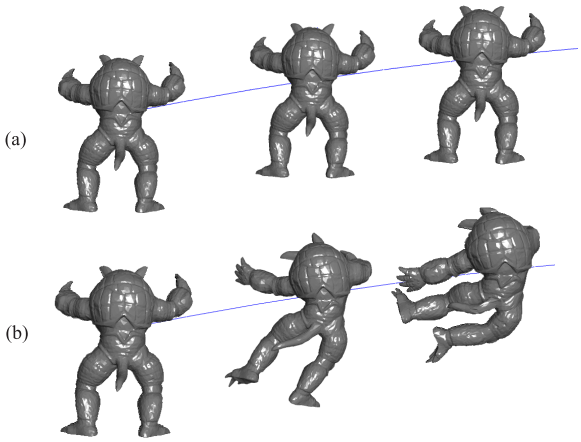

Figure 6: *Compensation results of an object for the constraint of its trajectory. (a) Compensation for a whole shape. (b) Compensation for a part of an object.*

preferably as close to elastic objects as possible. This means that interpolated objects do not suffer from the variation of volume or positional deviations. To realize such interpolation, we propose a novel FEM-based interpolation method called *physics-oriented interpolation*. Although a similar approach called as-rigid-as shape interpolation is presented by Alexa et al. [ACOL00], our approach is more closely related to physically-based elastic motions because we utilize the FEM-based solution method for elastic body animation described above.

We now discuss our approach to compute a function $R_{pq}(t)$ which interpolates two neighbor resting shapes R_p, R_q. To simplify, we assume that we only consider elastic deformations of R and omit the effects of rigid transformations such as translations or rotations. We also assume that there is no external force, i.e., $f_{ext} = 0$.

$R_{pq}(t)$ is then found by solving the following differential equation from R_p as an initial state:

$$M\ddot{x} + C\dot{x} + K_q(x - o_q) = 0, \tag{3}$$

where K_q, o_q denote a stiffness matrix and an original position of R_q respectively. This equation means that elastic forces are generated to restore a resting shape R_q from an initial state R_p. Here we also use [MDM*02] for solving Equation (3) in order to calculate interpolated objects in which distortions by the rotation term do not arise. Note that the temporal sampling used in Equation (3) may be coarser than that used for Equation (2). To reduce computation, in our implementation, the temporal sampling in Equation (3) is not as fine as that of Equation (2).

However, solving only Equations (3) has an issue. If displacements between R_p and R_q are quite large, huge internal restoring forces are generated at the beginning of computation. This yields vibrations which could make the computation unstable.

To overcome this issue, we restrain the restoring forces by amplifying the value of each element in C, a diagonal matrix which represents coefficients of damping (In our experiment, roughly five to ten times larger value is preferable). This setting dramatically increases attenuation forces in order to restrain unwanted vibrations.

Figure 5 shows the result of our physics-oriented interpolation. Compared to the result of linear interpolation in the upper figure, a natural interpolation is established which approximately preserves the volume of an object even for a large deformation between two resting shapes.

5. Trajectory Control

In this section, we describe our method to adapt the elastic body animation to follow a motion trajectory specified by the user. We regard such a trajectory as a type of a constraint. A point element in the object is fixed to the trajectory as a point constraint.

There are mainly two types of constraints, the constraints of position or rotation. We introduce here motion compensation algorithms for these two types of constraints.

Compensation for positions and velocities

For a point element in an elastic object, let the position and velocity both before and after the modification of trajectory be $x(t), v(t), x'(t), v'(t)$ respectively. The differences in their displacements are represented as follows:

$$\Delta v(t) = v(t) - v'(t), \quad \Delta x(t) = x(t) - x'(t),$$

We then build corrections for such differences into the solution algorithm for elastic body animation in Equation (2), i.e., for a position x^i and a velocity v^i at a time t_i,

$$\tilde{v}^i = v^i + \Delta v(t_i), \quad \tilde{x}^i = x^i + \Delta x(t_i). \tag{4}$$

We use $\tilde{v}^i, \tilde{x}^i$ instead of v^i, x^i to compute v^{i+1}, x^{i+1} in the implicit Euler method.

This compensation generates different modified shapes of an object according to which vertices to be updated. Figure 6 shows the results of different settings of updating vertices. In each figure, a dotted line indicates the modified trajectory. Figure 6(a) shows an example which constraints are set to the whole shape of an object. For the mass m_i of each vertex $v_i(t)$, a barycentric position $x(t)$ and velocity $v(t)$ are computed as follows:

$$v(t) = \frac{1}{m}\sum_i (v_i(t)\cdot m_i), \quad x(t) = \frac{1}{m}\sum_i (x_i(t)\cdot m_i),$$
$$m = \sum_i m_i.$$

Differences $\Delta v(t), \Delta x(t)$ are computed and applied to all vertices of an object. This yields the same amount of compensation to the whole shape, giving the effect of rigid transformation over a barycentric position. In contrast, Figure 6(b) shows another example in which constraints are applied to a part of elements of an object. Here we set a constraint on one of the tetrahedral elements around the barycenter of an object. In this case, the compensation is applied to only vertices of the target tetrahedron. Consequently, internal elastic forces are generated around the constrained point, and the object is deformed so that it is pulled to the current point on a trajectory.

Compensation for rotations

Next we describe the case when applying rotation constraints. Here we introduce a compensation for only a whole shape. This is because it is quite difficult to control the rotation constraint for each element of an object. As in Müller et al's approach [MDM*02], we suppose that each tetrahedral element has its own rotation matrix. We define the global rotation matrix of a barycentric point as follows:

$$R(t) = ortho(\frac{1}{m}\sum_i (R_i \cdot m_i)), \quad m = \sum_i m_i,$$

where m_i denotes the mass of a tetrahedron, *ortho(R)* is an orthonormalization for the matrix R. Let $R'(t)$ be the rotation matrix after the modification of a trajectory, then the correction matrix for rotation $\Delta R(t)$ is represented by,

$$\Delta R(t) = R'(t)R(t)^T.$$

Imitating the form of Equation (4), the compensation of a trajectory for a rotation constraint is given by:

$$\tilde{v}_i = v_m(t_i) + \Delta R(t_i)(v_i - v_m(t_i)), \quad (5)$$
$$\tilde{x}_i = x_m(t_i) + \Delta R(t_i)(x_i - x_m(t_i)),$$

where x_m and v_m mean the center of mass and its velocity, respectively.

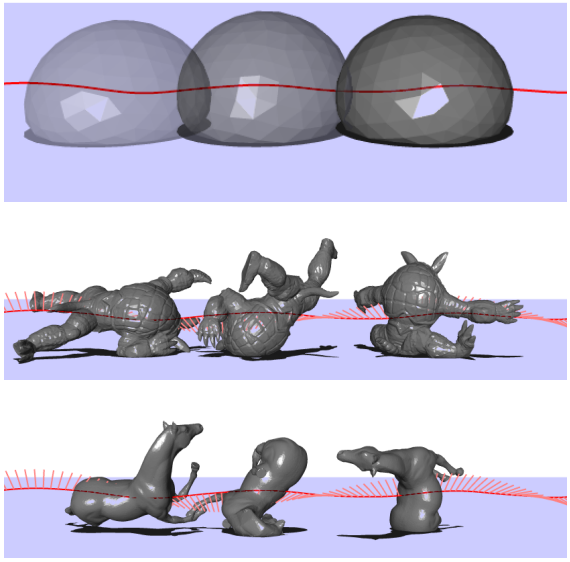

Figure 7: *Trajectory control results. A trajectory of an object is created by rolling a ball on the ground. Two diffrent objects are applied to the same trajectory. The red curve indicates the trajectory of the center of mass of the rolling ball, and the red arrows along the curve indicate its rotation directions over time.*

6. Results and Discussion

Our prototype system runs on a 2.5GHz Pentium 4 PC with 1G bytes of main memory, and the results are rendered on an NVIDIA GeForce FX Go 5600 graphics card. A coarse tetrahedral model of several hundred elements is used for solving the differential equations (2) and for dealing with collision response. In all of our examples described below, we consider the collision response only between objects and the ground. When a vertex of a simplified tetrahedral mesh penetrates the ground, we calculate a reaction force according to its depth by using a penalty method. Then a collision can be avoided by calculating Equation (2) with adding such a reaction force to an external force f_{ext}. The fine mesh model made of about 20,000 polygons is then used for rendering. The two models for the object are effectively synchronized by the technique in [KK04], so that we can edit and create the elastic motion at interactive rates.

Figure 7 shows a simple example of our trajectory control method. In this example, we first roll a ball on the ground, and record the trajectory of the center of the ball and its rotation. Two different objects (horse, Armadillo) are then made to follow the same trajectory and rotation. Each object has several collisions with the ground and the corresponding deformations occur during the animation while passing through the trajectory. This shows that our motion compen-

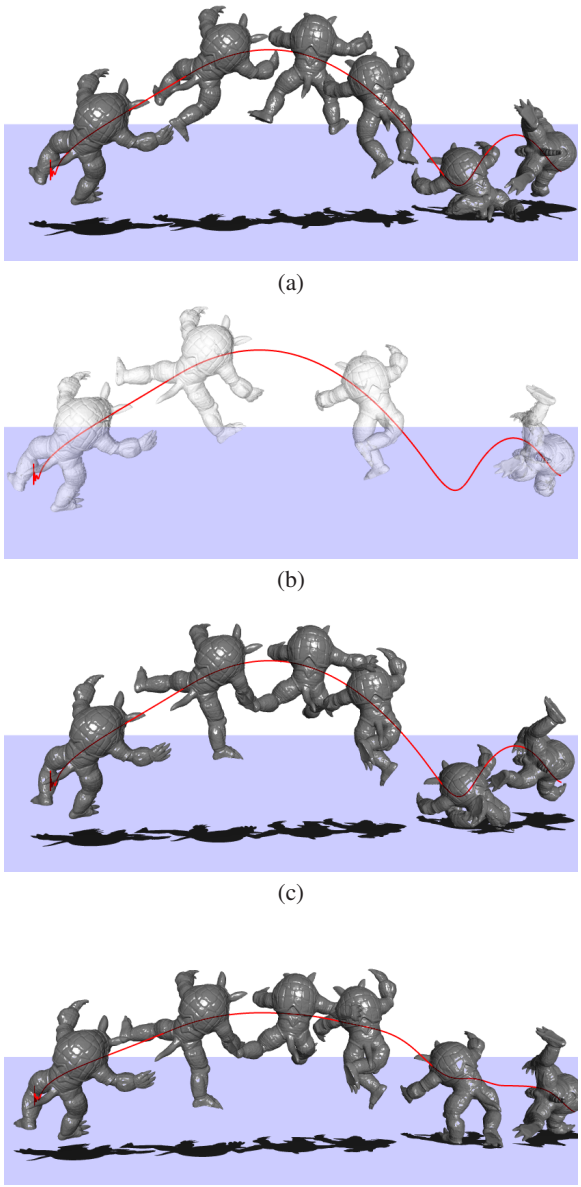

Figure 8: *Deformation control results. (a) the original elastic body animation. (b) keyframe shapes modified by the user. (c) the deformation control result. (d) the trajectory control result.*

sation algorithm preserves the physical realism of each object's elastic body animation.

Figures 8(a)-(c) present our deformation control method. We first calculate the original elastic body animation (Figure 8(a)) and record the position and velocity for each frame. The original elastic body animation tends to give the impression of a rubber doll being deformed, since the resting shape is set identical during the whole animation. Our intention is therefore to provide a lively character motion instead of the original purely passive physically-based animation. Four keyframe shapes are inserted in the time-line (Figure 8(b)). At each keyframe, the animated object is modified using our physics-oriented interpolation. Finally we re-calculate the object motion according to the keyframe shapes to generate the shape deformation animation (Figure 8(c)). The computation time needed for the interpolation is about 1 second on average. After making the keyframe animation, we can further modify the trajectory and then re-calculate the object motion to get another animation (Figure 8(d)). We then note that sometimes a keyframe shape may be seriously suffered from changing the trajectory. However, trajectory and shape deformation edits can be performed in any order and repeated, until the desired animation is obtained.

There are several points to keep in mind, in order to efficiently make a desired animation with our method. The first is keyframe shape delay. As described in Section 4, the keyframe shapes serve the role of guiding the deformation. Approximated keyframe shapes actually appear in the object deformation several frames after a keyframe. Such a delay over time depends on the deformation condition: In general, a large delay may occur for short intervals between two keyframes and with large displacement of the two keyframe shapes. The second is keyframe interval. In general, the keyframes should be set relatively far apart. For example, in making the animation in Figure 8, the keyframes are set every 8 frames (four keyframes for the 30-frame per second animation). A keyframe interval as short as three or less may cause physically incorrect motion. In addition, trajectory control should not be done in a radical way or over long timescales. If the user wants to do so, the initial physics-based animation can be altered. We also need more usability tests to make our approach more practical in the workplace. These are some limitations of our method, but we believe that the prototype system and the animation examples show the potential of our method for making directable animation of elastic objects.

Finally we make a few notes on how to control animation more precisely in our framework. First, our prototype system currently provides only a simple interface to modify an object's shape. A commercial modeling system would be helpful for more precise deformation control. Since our approach allows an animated object to be restored at each frame, modifying the keyframe shapes could be easily done with the commercial modeling system. Second, more automatic functions may be required in keyframing. For example, when a drastic collision occurs, adding keyframes before and after the collision would be helpful to avoid unnatural motion.

7. Conclusions and Future Work

In this paper we have proposed a framework to create directable animation of elastic objects. In our framework, users can intuitively control the trajectory of objects and shape deformations. The resulting animations still maintain the plausible realisms of the original physically-based motion.

In the deformation control, we have shown that our keyframing approach allows us to rearrange the elastic motion of objects guided by the specified shape at each keyframe. We have also proposed a novel FEM-based interpolation algorithm called physics-oriented interpolation which achieves a natural and approximately volume-preserving interpolation between two shapes. In the trajectory control, we have proposed a motion compensation algorithm to arrange elastic motion of an object along a trajectory. We have shown that this compensation provides a physically-based animation along a user-specified trajectory, while satisfying point constraints.

As the next step we would like to develop more sophisticated GUIs that can be intuitively manipulated for our framework, including motion control, deformation control, scene generation, and tetrahedral mesh generation. For future work, our approach should be generalized to deal with many deformable objects at a time, so that a wider variety of directable animation can be achieved.

Acknowledgments

We would like to thank Bill Baxter for his careful reading and comments. This research is partially supported by Japan Science and Technology Agency, CREST project.

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *Proc. SIGGRAPH '00* (2000), ACM Press, New York, pp. 157–164.

[BHW96] BARZEL R., HUGHES J. F., WOOD D. N.: Plausible motion simulation for computer graphics animation. In *Computer Animation and Simulation '96 (Proc. Eurographics Workshop)* (1996), Springer Wien, New York, pp. 183–197.

[CF00] CHENNEY S., FORSYTH D. A.: Sampling plausible solutions to multi-body constraint problems. In *Proc. SIGGRAPH '00* (2000), ACM Press, New York, pp. 219–228.

[CGC*02] CAPELL S., GREEN S., CURLESS B., DUCHAMP T., POPOVIĆ Z.: Interactive skeleton-driven dynamic deformations. In *Proc. SIGGRAPH '02* (2002), ACM Press, New York, pp. 586–593.

[Coh92] COHEN M. F.: Interactive spacetime control for animation. In *Proc. SIGGRAPH '92* (1992), ACM Press, New York, pp. 293–302.

[FL04] FATTAL R., LISCHINSKI D.: Target-driven smoke animation. In *Proc. SIGGRAPH '04* (2004), ACM Press, New York, pp. 441–448.

[FvdPT01] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Composable controllers for physics-based character animation. In *Proc. SIGGRAPH '01* (2001), ACM Press, New York, pp. 251–260.

[Gle01] GLEICHER M.: Comparative analysis of constraint-based motion editing methods. In *Graphical Models* (2001), vol. 63, Academic Press, pp. 107–134.

[GTH98] GRZESZCZUK R., TERZOPOULOS D., HINTON G.: Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proc. SIGGRAPH '98* (1998), ACM Press, New York, pp. 9–20.

[HFS03] HIROTA G., FISHER S., STATE A.: An improved finite element contact model for anatomical simulations. *The Visual Computer 19*, 5 (2003), 291–309.

[HSO03] HAUSER K., SHEN C., O'BRIEN J. F.: Interactive deformations using modal analysis with constraints. In *Proc. Graphics Interface 2003* (2003), Morgan Kaufmann, San Francisco, CA, pp. 247–256.

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *Proc. ACM SIGGRAPH Symposium on Computer Animation (SCA 2004)* (2004), ACM Press, New York, pp. 131–140.

[KK04] KONDO R., KANAI T.: Interactive physically-based animation system for dense meshes. In *Proc. Eurographics 2004 short paper* (2004), pp. 93–96.

[MDM*02] MÜLLER M., DORSEY J., MCMILLAN L., JAGNOW R., CUTLER B.: Stable real-time deformations. In *Proc. ACM SIGGRAPH Symposium on Computer Animation (SCA 2002)* (2002), ACM Press, New York, pp. 49–54.

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. In *Proc. SIGGRAPH '04* (2004), ACM Press, New York, pp. 449–456.

[PSE*00] POPOVIĆ J., SEITZ S. M., ERDMANN M., POPOVIĆ Z., WITKIN A.: Interactive manipulation of rigid body simulations. In *Proc. SIGGRAPH '00* (2000), ACM Press, New York, pp. 209–217.

[PSE03] POPOVIĆ J., SEITZ S. M., ERDMANN M.: Motion sketching for control of rigid-body simulations. *ACM Transactions on Graphics 22*, 4 (2003), 1034–1054.

[SGL99] SARTI A., GORI R., LAMBERTI C.: A physically based model to simulate maxillo-facial surgery from 3D CT images. *Future Generation Computer System, Elsevier Editions 15*, 2 (1999), 217–221.

[SY05] SHI L., YU Y.: Controllable smoke animation with guiding objects. *ACM Transactions on Graphics 24*, 1 (2005), 140–164.

[TMPS03] TREUILLE A., MCNAMARA A., POPOVIĆ Z., STAM J.: Keyframe control of smoke simulations. In *Proc. SIGGRAPH '03* (2003), ACM Press, New York, pp. 716–723.

[TT94] TU X., TERZOPOULOS D.: Artificial fishes: Physics, locomotion, perception, behavior. In *Proc. SIGGRAPH '94* (1994), ACM Press, New York, pp. 43–50.

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *Proc. SIGGRAPH '88* (1988), ACM Press, New York, pp. 159–168.