4th Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2007)
J. Dingliana, F. Ganovelli (Editors)

A Physically Based Deformation Model for Interactive Cartoon Animation

M. Garcia[1] and J. Dingliana[2] and C. O'Sullivan[2]

[1] Rey Juan Carlos University, Spain
[2]GV2, Trinity College Dublin, Ireland

Abstract

We present an approach for automatic cartoon-style motion dramatization suitable for interactive realtime animation. The system is built upon a physically based deformation model previously discussed in [GMPR06] and achieves squash-and-stretch cartoon deformation relevant to the current object velocity by controlling the deformations in the physically based model. As an improvement over previous similar approaches, which largely provide geometrical solutions to the problem, our modified physics-based deformation approach handles more general cases.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

Cartoon style deformation, in particular squash-and-stretch, is a strategy often used by traditional animators to emphasize key-movements, create *ad hoc*, yet believable, motions and, in general, to make motions more interesting. This might be considered a variation on similar strategies used in non-photorealistic rendering, which exploit human perceptual discrepancies to filter out extraneous detail whilst simultaneously accentuating the saliency of more important features of an image. The goal of stylisation in both animation and rendering is to make imagery and animations more visually appealing, more accessible and more efficient at expressing their core message. Although stylised deformation in computer animation is not a new area of research, most techniques heretofore have dealt with the problem largely for noninteractive off-line applications.

In this paper, we present an approach for automated dynamic simulation, which achieves squash-and-stretch effects in interactive realtime animation. The dynamics based implementation implies not only that animation is, for the most part, natural looking, but also that the system is capable of robustly handling a very general range of simulation scenarios. The main improvement over other approaches is that we can handle multiple simultaneous collisions of squash-and-stretch objects with rigid or deformable objects, which may themselves be undergoing cartoon deformation. In addition, we provide support for relatively complex objects and we do not impose the restriction that our objects need to realign themselves with the velocity vector, allowing for a more natural behaviour in an interactive scene. This restriction can however be added relatively easily, if desired.

Figure 1: *Time-lapsed frames captured from realtime squash-and-stretch simulation. The object undergoes deformation due to its velocity as well as collisions.*

The rest of the paper is organized as follows: Section 2 provides an overview of related work in physically based modeling and cartoon animation, Section 3 and 4 describe

our implementation of deformable and squash-and-stretch dynamics respectively, Section 1 presents some results and we conclude in Section 6 with an evaluation of the system and directions for further work.

2. Related Work

2.1. Physically Based Animation

Physically based animation concerns itself with simulating, in the virtual world, the same properties and behaviors of objects that we perceive in the real world, using a model of known rules and constraints which govern the motion of objects in the animated environment. Such an approach to animation has the distinct advantage that not only do behaviors appear recognizable and oftentimes more realistic than *ad hoc* animation, but it becomes possible to generate self governing systems capable of reacting believably to user input and interaction. However, the task of modelling the infinitely complex physical world is one that can only be made tractable by significant amounts of simplifications to the models of objects or to the rules that govern them. On the other hand, there are cases to be made for intentionally generating motions in animation beyond those that are conventionally produced by an accurate dynamic simulator.

O'Sullivan et al [ODGK03] categorize distortions in physically based animation as being either aesthetic or unavoidable. *Unavoidable distortions* are effectively a lack of precision imposed on a system due to faults in the computational models, or because the system has to fall back to an imprecise approximated solution due to physical constraints *e.g.* memory, processing speed or time. Many distortions are imperceptible to viewers, probably because we have an imperfect notion of dynamic events of high dimensionality. *Aesthetic distortions*, on the other hand, are errors or inaccuracies that are deliberately introduced to generate simulations that achieve a certain objective, whether this be some interesting type of stylised movement or a specified goal state in a sequence of events.

Barzel et. al [BHW96] are amongst the first to suggest that complete accuracy, the ubiquitous goal in physically based animation, is not a stringent requirement for realism. A variety of distortions in dynamic events are largely imperceptible to the viewer, and this uncertainty can be exploited in order to save on processing effort or to provide flexibility and controllability for animators. Chenney and Forsyth [CF00] show how this can be used to deliver plausible simulations that satisfy goal states set by an animator. They propose a simulation model that incorporates plausible sources of uncertainty by sampling a number of simulations that achieve the target goal state. A probability score is applied to each simulation run and the animation designer or the application can then pick the most appropriate one. The simulation editing problem is also discussed by Popovic et. al [PSE*00], who calculate mathematically consistent simulation sequences that lead to a goal state specified by the animator.

Both the aforementioned methods deal with regular rigid body simulation by using conventional methods for integrating the trajectories of objects and only introducing uncertainty at discontinuities in the simulation process, i.e. at the point of collision. A more general form of stylisation, closer to that employed by traditional cartoon animators, can be obtained by perturbing not only the motion paths and temporal characteristics of animation, but also by manipulating the deformations generated by the dynamic simulation.

2.2. Stylistic Deformations

Lasseter [Las87] discusses how traditional principles, essentially aesthetic distortions, developed to make animation more realistic and entertaining [TJ81], can be applied to computer animation.

Amongst these, a commonly used technique is squash-and-stretch, which applies exaggerated deformation to moving objects to accentuate their movements and visually convey mass and rigidity. A number of previous approaches use different kinds of deformable modelling strategies to achieve the same effect including: FFD's [FvdPT97], implicit surfaces [OM94] [Wyv97] and purely geometric transformations [PW89] [CPIS02]. However, most previous approaches are largely designed for non-realtime application or for relatively simple scenes. For instance, geometry based deformation does not cater for collisions of a squash-and-stretch object with complex geometry, and most other approaches are not particularly suited to interactive applications, particularly if there is a potential for multiple simultaneous collisions.

More recently, Chenney et. al [CPIS02] provided a two-phase model for applying cartoon-style squash-and-stretch in realtime. Their approach employs uniform affine scaling for an object undergoing ballistic free-space motion and an empirical method for creating plausible stylised deformations during collisions. Their method caters for simultaneous collisions of a squash-and-stretch object with rigid objects but not for multiple cartoon objects simultaneously colliding with one another.

We wish to extend this approach by not only generating realtime deformations for multiple simultaneous collisions, but the deformations should also be utilized in generating the ensuing motion of the object after collision.

2.2.1. Deformable Object Simulation

Since physically based modeling of deformable objects was first introduced in the computer graphics literature, numerous approaches have been proposed. For further information on deformable simulation we direct the reader to [NAM*06]. A significant amount of recent research has attempted to address the problem of deformable simulation

at interactive rates. These approaches have generally targeted applications in computer games, surgery simulation, and cloth and hair simulation. In such applications interactivity is the main issue and in most of them a plausible solution is more important than an accurate one. A number of deformable simulation approaches exist, which can run at interactive rates including: mass-spring systems [BW98], mesh-free methods [Liu02] [MHTG05], finite element methods [MG04] [EKS03], finite differences [TPBF87], finite volumes [TSB*05] and boundary element methods [JP99]. For further information on deformable simulation we direct the reader to [NAM*06]. In this paper, we present a finite element approach that can run at interactive rates. We recommend [Bat96] for a complete treatise on the FEM theory.

3. The Deformable Model

To model the objects in our interactive scene, we employ a FEM based approach which is based on recently documented techniques. Our main contributions are the extensions discussed in section 4 but a short description of the underlying implementation is first provided here. For interactive rates, we use the co-rotational formulation of the finite element method. Further information about this method can be found in [MG04] [EKS03]. In our implementation we use the optimizations described in [GMPR06]. The main features of our method are:

- For efficiency, a linear strain tensor is used. This is only valid for measuring small displacements as it is not invariant to all rigid body motion. It is invariant to translations but not to rotations. This tensor can be described in terms of the deformation gradient as:

$$\xi = \frac{1}{2}(F + F^t - 2I) \tag{1}$$

  where F is the deformation gradient matrix, F^t its transpose and I the identity matrix.
- We use a linear constitutive equation. We relate the stress σ with the strain ξ using a constant matrix E:

$$\sigma = E\xi \tag{2}$$

  If the material is isotropic E can be computed using just two parameters: the Young modulus and the Poisson ratio. All the tests shown in this paper were made using isotropic materials.
- We compute the internal forces f_e^{in} of each element e using the stiffness matrix K_e, and $u_e = x_e(n) - x_e(0)$ is the displacement of the element nodes, where $x_e(n)$ is the position of the element nodes at the instant n and $x_e(0)$ is the initial position of the nodes. f_e^{in} is calculated in the un-rotated configuration of each element and then it is mapped again to the rotated configuration using the expression:

$$f_e^{in} = R_e K_e (R_e^t x_e(n) - x_e(0)) \tag{3}$$

  where R_e is the rotation of the element. The matrix K_e can be precomputed using the following expression:

$$K_e = \int_{vol_e} B_e^t E_e B_e dv_e = B_e E_e B_e vol_e \tag{4}$$

  where B_e is the matrix that relates the node displacement with the strain and vol_e is the element volume. B_e and E_e are constant for all the points inside the tetrahedral element. We calculate the deformation gradient F_e of the mesh elements at each iteration, decompose it using polar decomposition $F_e = R_e U$ and use the rotation computed to calculate the internal forces. This lets us deal with large displacements whilst using a linear strain tensor.
- We use an implicit integration scheme to compute the new state at each simulation loop. We intend that our system should cater for stiff objects as well as elastic objects, so we need a method that behaves stably in every situation. The implicit integration scheme requires us to solve the following system at each iteration:

$$G = M + hD + h^2 K \tag{5}$$

$$Gv(n+1) = Mv(n) + h(f^{in} - f^{ext}) \tag{6}$$

  where G is the system coefficient matrix, f^{in} are the internal forces, K the global stiffness matrix, M the mass matrix, D the damping matrix, h the time step between to iterations, $v(n+1)$ the velocity at the iteration $n+1$, $v(n)$ the velocity at the iteration n and f^{ext} the external forces. To solve this system we are going to use the two approaches proposed in [GMPR06]. The first approach is the more accurate but slower. It computes K as

$$K = \sum_e R_e K R_e^t \tag{7}$$

  using the preconditioned conjugate gradient method to solve the system and requires us to compute the element rotations at each iteration. The second solver uses the global mesh rotation instead of the element mesh rotations, calculating K as

$$K = \sum_e R_g K R_g^t \tag{8}$$

  where R_g is the global rotation matrix. This approach leads to a much faster solution, as described in [GMPR06]. In this paper we apply the squash-and-stretch method proposed to both approaches.

From the implementation point of view, our model is composed of three different geometrical layers as shown in Figure 2. The first is a finite element representation of the object consisting of a tetrahedral mesh, which is used in the simulation and control of the object. We choose tetrahedral elements because we can use their barycentric coordinates in a shape function. To accelerate the collision detection, we use a Bounding Volume Hierarchy made up of spheres. The sphere trees are generated using the algorithm described in [BO04] and are updated at each simulation step in relation to the deformations of the tetrahedral model, as explained

in [GBTM05]. Finally, in order to improve the visual appearance, we use a superficial mesh to render our models. This mesh is independent in complexity from the tetrahedral mesh and it can be of a much finer resolution to improve rendering quality. The vertices of this mesh are updated, at each render step, using the barycentric function as described in [MG04].

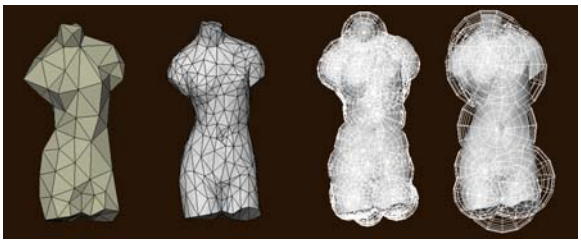

Figure 2: *Different models of the venus used in our multilayered representation: from left to right, the tetrahedral mesh, the surface mesh for rendering and two levels of the hierarchical sphere tree.*

4. Stylising Simulation

We propose a method that involves manipulating the physically based animation to stylise the movement of objects in the scene. We combine the physical solution computed by the physically based animation system with a geometrical deformation. Thus, our approach can be divided into two key parts. The first requirement is a method for controlling the deformation in our physically based model. This technique will be applied to drive the deformation of elastic objects in an interactive application. Once we have a means of controlling the deformations, we use this to apply an amount of squash-and-stretch to an animated object, depending on its velocity.

4.1. Controlling the Deformations

Once we have a physically based method for animating soft bodies, the next goal is to control deformations, whilst maintaining interactive frame rates. The method described in this section will give us complete control of the object's shape. A deformation field or an affine transformation can be used as input for this technique. In the following steps we are going to use this deformation control method to squash and stretch the object.

There are a number of previous works on trying to enforce control over a physically based animation. However, most previous approaches focus on animation control for non-interactive applications. In particular, many previous authors have dealt with controlling fluid [MTPS04] or rigid body motions [PSE*00] but the issue of controlling the deformations of a deformable body are much less widely explored. Irving et. al [ITF04] try to control the deformation using a plasticity model. This approach works well but is focused on non-realtime applications. We use this as starting point to develop our deformation control method. Müller et. al [MG04] propose a technique to model plasticity in a co-rotational FEM formulation. In their approach they decompose the total strain tensor into the plastic strain tensor and the elastic strain tensor. The model should recover from elastic deformations but not from plastic deformations. So they calculate the forces produced by the plastic strain and the internal forces produced by the total strain. Finally, they subtract the plastic forces from the internal forces.

Our approach for controlling deformations is based on this idea: We divide the total strain into elastic strain and the strain produced by the deformation desired by the user:

$$\xi^t = \xi^e + \xi^c \tag{9}$$

where ξ^t is the total strain tensor, ξ^e is the elastic strain tensor and ξ^c is the control deformation strain tensor.

In the first stage of the algorithm the strain tensor is computed. If the input into the algorithm is an affine transformation T all the mesh elements have the same control deformation strain tensor $\xi^c_e = \xi^c$. In the other case, if a displacement field is the input in to the algorithm, a different control deformation strain tensor has to be calculated for each element e, as explained below. Using the transformation matrix T to compute ξ^c, the rotation is first removed using polar decomposition $T = RU$ and then the strain tensor is computed:

$$\xi^c = \frac{1}{2}(U + U^t - 2I) \tag{10}$$

If the algorithm input is a displacement field $W(x)$, we compute a different ξ^c_e for each element. Using tetrahedral elements, and given that we know the initial position $x_e(0)$ and the final position $x_e(n) = x_e(0) + W(x_e(0))$ of the nodes of the element e, the deformation gradient can be computed as:

$$F_e = M_e(n) M_e^{-1}(0) \tag{11}$$

where the matrix $M_e(n)$ is computed using the vector $d_{e,i}(n)$

$$M_e(n) = [d_{e,1}(n) | d_{e,2}(n) | d_{e,3}(n)] \tag{12}$$

$$d_{e,i}(n) = x_{e,i}(n) - x_{e,4}(n) \tag{13}$$

where $x_{e,i}(n)$ is the desired position of the node i of the element e at the instant n. As in the other case the rotations should be removed ($F_e = R_e U_e$) and the strain tensor can be computed as:

$$\xi^c_e = \frac{1}{2}(U_e + U^t_e - 2I) \tag{14}$$

This technique can be used to control the shape of the object using a displacement gradient. Fortunately the squash-and-stretch deformations can be defined with an affine transformation T, so these kind of deformations do not need a different strain tensor for each element.

Finally, the stress σ_e^c and the force F_e^c are computed for each mesh element in the rotated configuration:

$$\sigma_e^c = E_e \xi_e^c \tag{15}$$
$$F_e^c = R_e B_e \sigma_e^c \tag{16}$$

F_e^c is subtracted from the total internal forces to make the object reach the new equilibrium position.

4.2. Squash-and-Stretch

Using the previously defined deformation control technique, we apply squash-and-stretch stylisation to our simulations with the goal of making them more appealing to the user. We stretch our model depending on the current velocity as is described by [CPIS02]. We extend their approach by adding elastic body physics and with a few modifications to the deformation variables.

Our algorithm basically consists of the following steps:

1. The velocity of the center of mass is computed v_c.
2. Depending on the modulus of the velocity, the scale factor s is computed:

$$v_\alpha = \begin{cases} |v_c| - \alpha, & if\, |v_c| - \alpha > 0 \\ 0, \end{cases} \tag{17}$$

$$s = \frac{|v_\alpha| s_m s_c + 1}{|v_\alpha| s_c + 1} \tag{18}$$

   where s_m is the maximum stretch factor, s_c is rate of stretch, v_c is the mass center velocity, v_α is a threshold velocity used to compute the scale factor s and α is a threshold value. s_c, s_m and α are defined by the user.
3. To compute the stretch deformation gradient S on the undeformed configuration, a rotation R_v is applied to the model to align with the velocity as in [CPIS02].

$$S = \begin{bmatrix} s & 0 & 0 \\ 0 & \frac{1}{\sqrt{s}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{s}} \end{bmatrix} R_v \tag{19}$$

4. Our model uses the Cauchy strain tensor to measure the deformations. This tensor is not invariant to rotations so, by using the polar decomposition, the rotations are removed from the matrix leaving a pure scale-shear matrix.
5. Finally, the control deformation forces are calculated using the technique described in the previous section.

The squash process is analogous so it will not be described in this paper, but some considerations must be taken into account:

- The squash process starts with a collision. The user must define which collisions are going to be squashed.
- When a squash is going to take place, the system controls the velocity and the rotation of the object. This is done by applying forces to modify angular and linear momenta of the object.

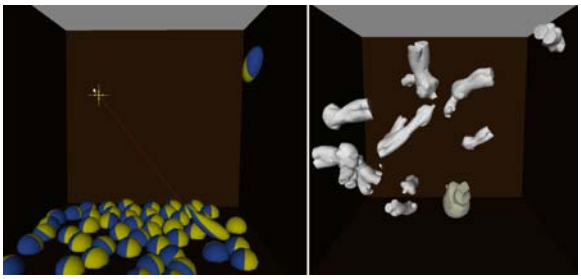

Figure 3: *Sample scenes, captured from interactive animation, showing multibody squash-and-stretch simulation*

5. Results

Table 1 shows how computational time is distributed between the different computational components per simulation time step. The simulation was run for several scenes of varying complexity, which was determined by the number of tetrahedral elements in the simulation. Sample scenes from the experimental scenarios can be seen in Figure 3. The models used, as shown in Figure 4, were a generic sphere model decomposed into tetrahedra, the Max Planck head model, and a Venus model. All timings are measured in milliseconds per simulation timestep and are averaged over the full simulation.

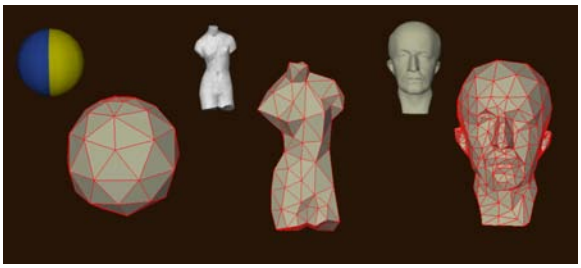

Figure 4: *Objects used in the simulation and the associated tetrahedral mesh layer: a generic sphere model, a venus model and the Max Planck head model.*

The tests were run on a DELL XPS700 3.73 GHz system with 2Gb of RAM and a 1Gb NVIDIA GeForce 7950GX2 graphics card. Table 1 shows that the method described can achieve results in real time. For this study, two implementations have been developed. The first one uses the object global rotation. This implementation is fast but sometimes it is not accurate enough. The other technique, the element rotation technique, is slower but more accurate [GMPR06]. We highlight only the squash-and-stretch aspects of the system

	Scenes						
Object Name	Planck Head		Sphere				Venus
No. Objects	1	2	1	10	20	40	1
No. Elements	1100	2200	107	1070	2140	4280	266
Global Rotation Solver (GRS)	2.10	4.42	0.20	1.79	3.44	6.59	0.45
GRS Stretch force computation	1.06	2.01	0.06	0.71	1.65	2.88	0.11
Element Rotation Solver (ERS)	24.97	49.01	1.50	X	X	X	4.58
ERS Stretch force computation	1.24	4.02	0.08	X	X	X	0.97
Element rotation computation	5.09	10.51	0.41	X	X	X	0.71

Table 1: *Mean time (in ms) spent per simulation timestep for each computational component of our system. Note that element rotation values were not recorded for the rotationally invariant sphere model.*

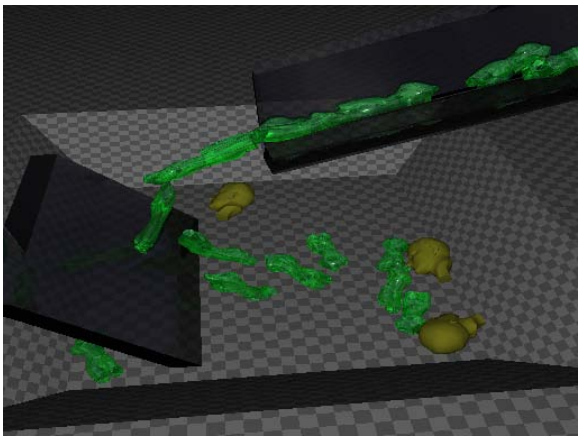

Figure 5: *A frame from demo animation generated by our system (rendered offline).*

here. The overall efficiency of the system is significantly influenced by the underlying deformation and collision detection system upon which the squash-and-stretch extensions are built. For a more detailed indication of the efficiency of our general deformation system, we direct the reader to [GMPR06]. It should also be noted that our squash-and-stretch may serve equally well as an extension to an alternative interactive deformation system that allows for some control of elasticity values.

Figure 5 shows a sample scene with a number of complex objects generated by our system. A comparison of a simple deformable simulation with a squash-and-stretch simulation generated by our system is shown in Figure 6.

6. Conclusions and Future Work

We have presented a physics-based simulation framework that successfully produces interactive cartoon-based deformations at realtime rates. The dynamics based solution has several advantages over purely kinematic or geometric solutions. One major advantage is that the solution is very robust for a wide range of cases. The system can cater for multiple simultaneous collisions with rigid objects or with other deformable bodies themselves undergoing squash-and-stretch and provides a reasonable range of motion stylisations, with potentially useful perceptual cues that may be relevant to enhancing animation.

Applying traditional animation techniques to computationally automated mechanisms opens up several important areas for further improvement. For instance, the system described in this paper relies on several parameters to control the amount of deformation. As our goal in this paper was mainly to prove the feasibility of realtime stylisation, we have simply outlined a few simulation parameters which can be adjusted to customize the animation but, arguably, may not be altogether intuitive for a human animator. Although it is possible to choose acceptable *ad hoc* values that create a range of reasonably pleasing visual results, we believe that there may be an optimal set of values that may be chosen to depict a certain style or mood. Whilst traditional artists rely on skill and experience to choose the best deformations, we believe there may be scope to determine these values using perceptual user studies as described in [ODGK03]. In addition, we believe that the quality of any such system, which seeks comparability to traditional production processes or, for that matter, to any alternate technique, can only objectively be determined through perceptual evaluation.

Furthermore, it may be important to investigate if alternate rendering styles might significantly affect a user's tendency to prefer cartoon deformations to more accurate physical simulation. Towards these ends, we have planned a series of experiments that will tests users' acceptance thresholds for various alternative animation scenarios. We wish not only to compare the relevance of different deformation parameters, but also to study whether a squash-and-stretch animation in general is perceptually as realistic, more immersive or generally more effective at engaging a viewer's attention.

Figure 6: *A comparison of normal deformable simulation (top) with our squash and stretch technique (bottom). Time-lapsed frames from a realtime simulation of the venus model.*

7. Acknowledgements

This work has been partially funded by the Government of the Community of Madrid (grants GR / SAL / 0940 / 2004 and S-0505 / DPI / 0235) and by Trinity College Dublin. The Max Planck head model was obtained from Igor Guskov's webpage at CALTECH.

References

[Bat96] BATHE L.-J.: *Finite Element Procedures*. Prentice Hall, NJ, USA, 1996.

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

[BO04] BRADSHAW G., O'SULLIVAN C.: Adaptive medial-axis approximation for sphere-tree construction. *ACM Trans. Graph. 23*, 1 (2004), 1–26.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. *Computer Graphics 32*, Annual Conference Series (1998), 43–54.

[CF00] CHENNEY S., FORSYTH D.: Sampling plausible solutions to multi-body constraint problems. In *Proceedings Siggraph 2000* (2000), pp. 219–228.

[CPIS02] CHENNEY S., PINGEL M., IVERSON R., SZYMANSKI M.: Simulating cartoon style animation. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2002), ACM Press, pp. 133–138.

[EKS03] ETZMUSS O., KECKEISEN M., STRASSER W.: A fast finite element solution for cloth modelling. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2003), IEEE Computer Society, p. 244.

[FvdPT97] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics 3*, 3 (/1997), 201–214.

[GBTM05] GARCIA M., BAYONA S., TOHARIA P., MENDOZA C.: Comparing sphere-tree generators and hierarchy updates for deformable objects collision detec-

tion. In *International Symposium on Visual Computing, ISVC05* (2005), pp. 167–174.

[GMPR06] GARCIA M., MENDOZA C., PASTOR L., RODRIGUEZ A.: Optimized linear fem for modeling deformable objects: Research articles. *Comput. Animat. Virtual Worlds 17*, 3-4 (2006), 393–402.

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), Eurographics Association, pp. 131–140.

[JP99] JAMES D. L., PAI D. K.: Artdefo - accurate real time deformable objects. In *Siggraph 1999, Computer Graphics Proceedings* (Los Angeles, 1999), Rockwood A., (Ed.), Addison Wesley Longman, pp. 65–72.

[Las87] LASSETER J.: Principles of traditional animation applied to 3d computer animation. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 35–44.

[Liu02] LIU G.: *Mesh Free Methods: Moving Beyond the Finite Element Method*. CRC Press, FL, USA, 2002.

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. In *GI '04: Proceedings of the 2004 conference on Graphics interface* (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004), Canadian Human-Computer Communications Society, pp. 239–246.

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM Press, pp. 471–478.

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIC Z., STAM J.: Fluid control using the adjoint method. *ACM Trans. Graph. 23*, 3 (2004), 449–456.

[NAM*06] NEALEN, ANDREW, MULLER, MATTHIAS, KEISER, RICHARD, BOXERMAN, EDDY, CARLSON, MARK: Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4 (December 2006), 809–836.

[ODGK03] O'SULLIVAN C., DINGLIANA J., GIANG T., KAISER M. K.: Evaluating the visual fidelity of physically based animations. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM Press, pp. 527–536.

[OM94] OPALACH A., MADDOCK S. C.: Disney effects using implicit surfaces. In *Proc. 5th Eurographics Workshop on Animation and Simulation* (1994).

[PSE*00] POPOVIC J., SEITZ S. M., ERDMANN M., POPOVIC Z., WITKIN A.: Interactive manipulation of rigid body simulations. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 209–217.

[PW89] PENTLAND A., WILLIAMS J.: Good vibrations: model dynamics for graphics and animation. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1989), ACM Press, pp. 215–222.

[TJ81] THOMAS F., JOHNSTON O.: *Disney Animation - The Illusion of Life*. Abbeville Press, 1981.

[TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 205–214.

[TSB*05] TERAN J., SIFAKIS E., BLEMKER S. S., NG-THOW-HING V., LAU C., FEDKIW R.: Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics 11*, 3 (2005), 317–328.

[Wyv97] WYVILL B.: Animation and Special Effects. *Introduction to Implicit Surfaces* (1997), 101–104. Edited by Jules Bloomenthal With Chandrajit Bajaj, Jim Blinn, Marie-Paule Cani-Gascuel, Alyn Rockwood, Brian Wyvill, and Geoff Wyvill.