

Animating the Human Muscle Structure

Graphical simulations of human muscle motion and deformation are of great interest to medical education. In this article, the authors present a technique for simulating muscle deformations by combining physically and geometrically based computations to reduce computation cost and produce fast, accurate simulations.

Graphical reproductions of human anatomical structures and their motions are of significant interest to the medical illustration and simulation fields. With advances in computer graphics technology, researchers can render medical illustrations not only on paper but also digitally, allowing multimedia interactivity between users and a virtual human body.

The computer graphics research community has made efforts to offer increasingly realistic visual fidelity. Yet, most computer illustrations are primarily passive—that is, they're digital reproductions of their paper counterparts. Although users can view the anatomical elements interactively through HTML-like links—with sound and text descriptions, for example—most available systems can't depict the human anatomical structure's motion.

Skeletal muscles deform when the body moves. Simulating the muscles' movement and deformation represents a tremendous technological challenge because of their complex shapes and material

properties. Although some illustration systems can visualize some basic movements, they primarily display movements as prerecorded animations.

In this article, we present a technique that simulates muscle motion and deformation based on the muscle's physical properties. Our objective is to provide a computationally efficient method that displays a muscle's deformed shape as it moves for use in medical illustration and education.

System Architecture

Our prototype system consists of four functional modules: skeletal animation, muscle acquisition, muscle deformation, and rendering. The skeletal-animation module's main function is to animate the human skeletal structure using recorded or hand-animated motion data. The muscle-acquisition module constructs a full 3D geometric model for each muscle. We used commercially available muscle models as well as muscle shapes from the Visible Human Project (VHP)¹ to ensure model accuracy. The muscle-deformation module is the most important because it deforms muscles based on their mechanical properties and reactions. Our technique, however, differs from full-scale simulations for medical treatments in which medical personnel must consider the muscles' detailed shapes and interactions, making accuracy essential. Our intent is to mimic defor-

1521-9615/07/\$25.00 © 2007 IEEE
Copublished by the IEEE CS and the AIP

XIAOSONG YANG, JIAN CHANG, AND JIAN J. ZHANG
Bournemouth University

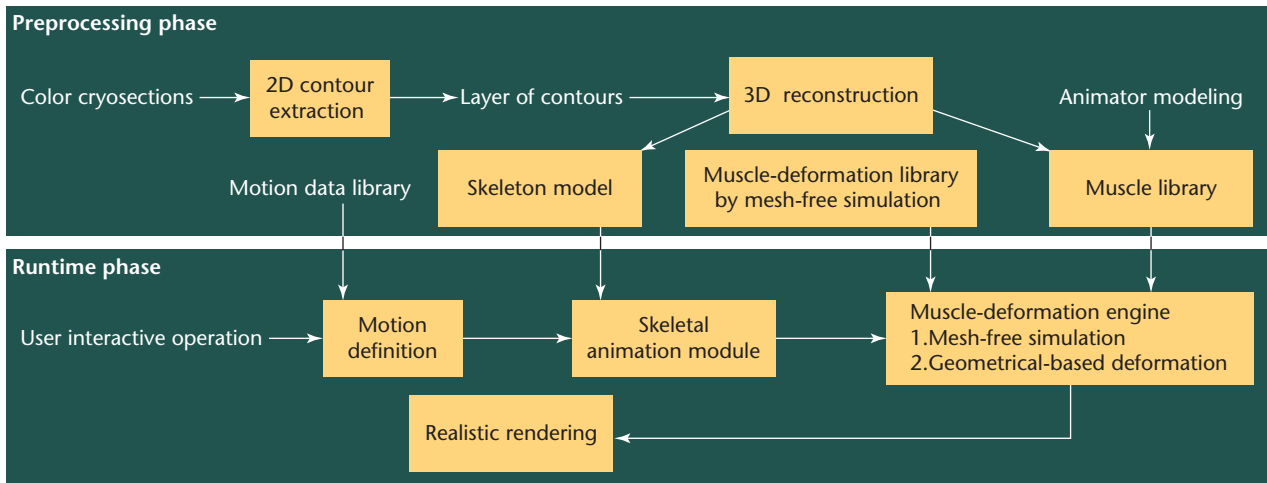


Figure 1. System architecture. Our prototype system consists of two phases: preprocessing and runtime simulation. The preprocessing phase includes the modeling of muscle mesh and deformation poses to ensure fast simulation responses in the runtime phase. The runtime simulation phase includes muscle deformation, collision detection, and realistic rendering.

mations efficiently and convincingly rather than representing deformations with high accuracy because the heavy computing costs incurred aren't justified for medical illustration. Finally, the rendering module displays the anatomical structure.

To maintain fast graphical frame rates and simulation accuracy, we used a hybrid simulation method. Specifically, we used a mechanical simulation technique, the *mesh-free method*, to compute the large deformations of various muscles, which lets us formulate several key muscle-deformation poses accurately for storage in the deformed-muscle library. We can then produce muscle poses by blending existing deformed muscle shapes during runtime. Using this method, we gain computation speed and simulation accuracy for realistic and fast responses.

Figure 1 shows our system's software architecture, which we developed with OpenGL and the C++ programming language; the overall system flow consists of a preprocessing and a runtime simulation phase. In the preprocessing phase, we construct all muscle models from the VHP data set or from hand-modeled muscle meshes. We include precomputed muscle-deformation poses by using the mesh-free method to achieve fast simulation responses. The runtime operations include physically based muscle deformations from the mesh-free method, collision detection, and realistic rendering.

We provide two different computation engines from which users can choose based on their requirements for realism, accuracy, and response speed. We developed our prototype system on a

Pentium 4 3-GHz PC with 1 Gbyte of RAM and a 128-Mbyte DDR ATI Radeon 9800 Pro graphics adapter. The CPU handles the computation, including the mesh-free simulations, and the GPU is responsible for realistic rendering.

Libraries

For our prototype system, we can generate all muscle models in the preprocessing phase and store them in the muscle library. We developed the geometric muscle models or meshes from Ultimate Human 3D data (www.cgcharacter.com/ultimatehuman.html) and VHP's 2D color cryosection images using a 3D surface reconstruction method. Because it provides an accurate source of human anatomy, the VHP ensures that the muscle models are developed accurately. Figure 2 briefly describes the modeling process.

Image Processing and Data Segmentation

The first step in simulating muscle deformation realistically is to acquire accurate 3D meshes. We used the VHP's color cryosection images to model each individual muscle model. With the effective methods^{2,3} currently available for contour extraction based on seed growing, we can automatically extract muscle contours and edit them interactively. Once this is completed, we store the muscle contours in the contour library indexed by VHP slice number and muscle name.

3D Surface Reconstruction

A simple 3D reconstruction algorithm works fine

for muscles with only two extremity points, but bifurcated muscles require special treatment. To address their complexity, we introduce an extra layer of contours around the split point to link the contours on two adjacent slices in a way similar to A.B. Ekoule's method⁴ and use S. Ganapathy and T.G. Dennehy's method⁵ to construct the surface triangle meshes.

Because muscles are solid objects, we must also consider their internal substances, so we distribute volume points inside each muscle as well as surface points on the outside. The mesh-free method computes and captures the muscle deformations using the volume and surface points as sampling points.

We smooth the meshes to improve the visual quality of the muscle objects' surface models. Smoothing is a technique that regulates the nodal positions to improve the representations' smoothness. Smoothing doesn't affect the model's topology but will alter the surface geometry to a small extent. A common and effective technique is Laplacian smoothing,⁶ which moves a point to a new place according to the average positions of its neighbors.

Motion Data

Unlike other traditional passive muscle illustration systems, our system represents deformed muscles when the virtual human body is in motion. Users can interactively operate on the virtual human muscle structure, for example, by specifying a specific motion to investigate. We have two very important *gearwheels* for animating the muscles. The first is a motion library containing basic motion patterns (this library self-optimizes through a procedure-recording function that lets users record and store frequently used motions; users can also interactively manipulate the underlying skeleton to define motion patterns). The second gearwheel is the animation engine, which calculates the muscle's final deformation.

Muscle Deformation

To maintain simulation accuracy, we exploit a physically based mesh-free method to simulate muscle deformation and contraction. This method, however, is computationally intensive and can't meet our real-time requirements. To overcome this drawback without causing a noticeable loss of accuracy, we combine the mesh-free method with geometric blending. Similar to skinning,⁷ we construct a muscle-deformation library consisting of several possible muscle-deformation poses. For each muscle, we use mesh-free computation

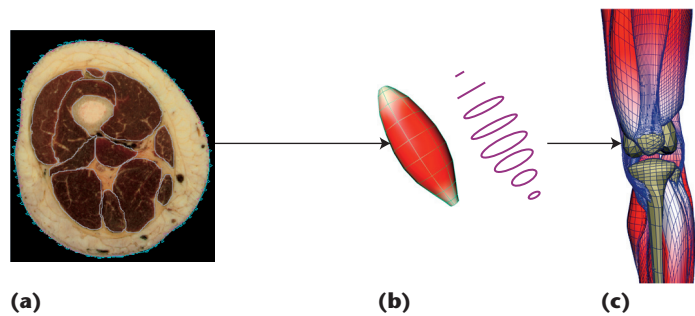


Figure 2. Muscle modeling. The modeling of each muscle consists of three steps: (a) extraction of muscle contours from the color cryosections images, (b) reconstruction of 3D surface model from a series of 2D contours, and (c) preparation of 3D volume models for mesh-free simulation.

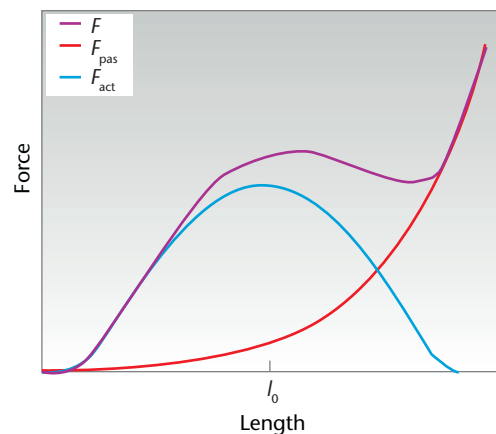


Figure 3. Force-length muscle relationships.⁸ When a muscle deforms, its force F contains two parts, the passive force F_{pas} and the active force F_{act} .

to create accurate deformations for a few typical poses. With these deformation samples in place, we can produce new muscle poses by blending these samples during the runtime phase. Thus, when the underlying skeleton moves, the muscle samples blend to generate a deformation appropriate for the skeleton position. For good blending results, samples should be evenly spaced in the deformation space and the distance between the samples should be kept small to ensure that the errors caused by the blending process are as small as possible.

Each individual muscle has its own deformation space, which is determined by the underlying skeletal bone linked to the muscle. From our knowledge of human motion, we know that

Table 1. Timing for our system.

	Active muscles	Interpolation and rendering (runtime phase)	Frame rates per second
Figure 4a	3	19 ms	52
Figure 4b	6	22 ms	42
Figure 4c	23	37 ms	27

each joint has a number of degrees of freedom (DOF) and there's usually a limit for each rotation. Most human muscles are linked by only two bones—for example, the forearm's extensor carpi ulnaris involves two types of deformations, one to stretch or compress when the hand lifts or drops and the other to twist the forearm when the hand and forearm perform a twisting action. The first action is pretty simple, so two muscle samples are sufficient, but the second action involves twisting up to 180 degrees in the most extreme positions, so we place six muscle samples in the deformation space. Other muscles, such as the pectoralis major, trapezius, and latissimus dorsi, span across several bones. We can simplify the muscles' movement into two major movement dimensions: arm up and down (six muscle samplings) and arm front and back (three samplings), thus the total samplings for these muscles is 18. We distribute each muscle's samplings based on an analysis of its attachment to the underlying skeleton. Because we don't include the muscle-muscle contact reaction in our current implementation, we can calculate each muscle separately and store the results in the muscle-deformation library.

Mesh-Free Muscle Deformation

Figure 3 shows a muscle's force-length relationship. Muscle forces contain passive components that act in a nonlinear fashion similar to other soft tissues, and active parts act due to changes in the microcontractile element, called myofilament.

We consider the muscle's strain-energy distribution in two parts. One is the contribution of the ground substance, W_m , and the other is the energy in the muscle fibers, W_f .⁹

$$W(\mathbf{C}) = W_m(\mathbf{C}) + W_f(\mathbf{C}, a), \quad (1)$$

where \mathbf{C} is the right Cauchy–Green deformation tensor, and a is the active ratio of the muscle that varies from zero to one.

Researchers often model the ground substance as isotropic hyperelastic material. The fiber con-

tribution is assembled by the active part, W_{act} , and the passive part, W_{pas} , following the experimental observation of Figure 3:

$$W_f(\mathbf{C}, a) = W_{pas}(\mathbf{C}) + W_{act}(\mathbf{C}, a). \quad (2)$$

Instead of using the traditional finite element approach, in which volume is divided into elements with meshes, we use the mesh-free method to solve the muscle-deformation problem numerically. The volume is discretized into a finite number of point samples. With the mesh-free method, we achieve the trade-off between efficiency and accuracy simply by distributing more or fewer points in the volume. The discretized points carry all physical properties—such as location, density, deformation, and velocity—and each point's state is affected by its neighbors. By finding the physically balanced states among all the points, we can numerically simulate muscle behavior.

Force per unit volume at point \mathbf{x} is defined as the negative gradient of the strain-energy density with respect to this point's displacement

$$\mathbf{F} = -\frac{\partial W(\mathbf{C}, a)}{\partial \mathbf{u}}, \quad (3)$$

where \mathbf{u} is the displacement at a given point. The differentiation operation of a function at a given point is approximated with the *moving least squares method*, which numerically reconstructs a continuous function from the sampling points, via computation with all its neighbors. Thus, in our mesh-free method, each point would have an associated force \mathbf{F} , which updates the point's physical state. The acceleration of one discretized point is the force \mathbf{F} multiplying the small volume that the point represents after dividing the mass distributed on that point.

Typically, at a given time t , we know the displacement distribution \mathbf{u}_t and the velocity \mathbf{v}_t , which deduces deformation tensor \mathbf{C} . Then we can compute the distribution of the strain energy $W_t(\mathbf{C}, a)$, and finally force \mathbf{F}_t at this point. Given a small timestep Δt , we update the velocity $\mathbf{v}_{t+\Delta t}$

RELATED WORK IN MUSCLE DEFORMATION

Over the past two decades, computerized muscle modeling and deformation has become one of the primary areas of research for human character animation. We can broadly classify existing methods into two groups: geometrically and physically based methods.

Ferdi Scheepers¹ and Jane Wilhems² used simple geometric primitives to approximate the appearance of fusiform muscles using the geometrically based method. Scheepers also presented a general model that consists of tubular bicubic patches.¹ Although these methods tried to mimic physical reality by considering properties such as volume preservation, they used models that were too simple to represent the human muscle structure's complexity.

John Chadwick and colleagues presented the first physically based technique in which muscles were deformed by embedding them into a free-form deformation (FFD) lattice.³ Chadwick used Hookean springs to simulate the dynamics of physical muscle tissues. The springs deformed the FFD lattice, which then deformed the embedded muscles. Although this represented progress, simple-spring systems don't simulate the complex deformation of muscles very well. Luciana Procher-Nedel⁴ introduced the idea of action lines for muscle deformation, which are easy to implement, but the limitations are similar to simple-spring systems. David Chen⁵ presented a biomechanical muscle model, which discussed a single muscle working in isolation, using the finite element method; Scott Delp's work also addressed single muscles.⁶ Alexander W.J. Gielen⁷ and Can A. Yucesoy⁸ modeled the stress and deformation relationship of highly detailed muscles using nonlinear solid mechanics. However, these methods—despite being theoretically more accurate than earlier techniques—have very high computing costs, making them impractical for interactive illustration applications.

In contrast to the finite element method, which relies on a properly meshed model, the mesh-free approach uses only unconnected points and has shown promise in several

engineering applications. Gui-Rong Liu⁹ provides a good review of the applied mesh-free techniques in engineering. In recent years, works by Jian Chang and Jian Zhang¹⁰ and Matthias Müller¹¹ have extended the idea of mesh-free computation to graphical uses.

References

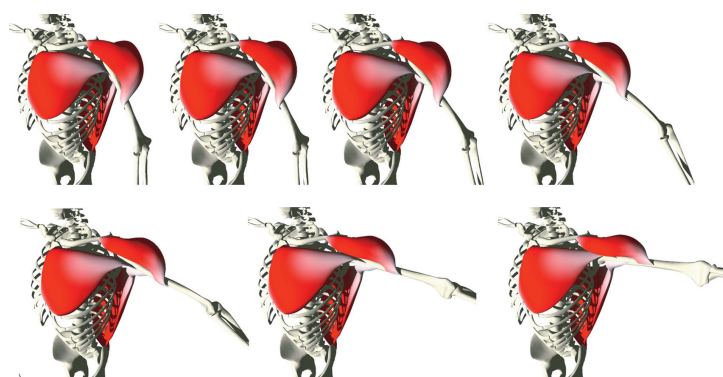
1. F. Scheepers et al., "Anatomy-Based Modeling of the Human Musculature," *Proc. 24th Ann. Conf. Computer Graphics and Interactive Techniques (SIGGRAPH 97)*, ACM Press/Addison-Wesley, 1997, pp. 163–172.
2. J. Wilhems, "Animals with Anatomy," *IEEE Computer Graphics and Applications*, vol. 17, no. 3, 1997, pp. 22–30.
3. J. Chadwick, D. Haumann, and R. Parent, "Layered Construction for Deformable Animated Characters," *Proc. 16th Ann. Conf. Computer Graphics and Interactive Techniques (SIGGRAPH 89)*, ACM Press, 1989, pp. 243–252.
4. L. Procher-Nedel and D. Thalmann, "Real Time Muscle Deformation Using Mass-Spring Systems," *Proc. Computer Graphics Int'l (CGI 98)*, IEEE CS Press, 1998, pp. 156–165.
5. D.T. Chen and D. Zeltzer, "Pump It Up: Computer Animation-Based Model of Muscle Using the Finite Element Method," *Proc. 19th Ann. Conf. Computer Graphics and Interactive Techniques (SIGGRAPH 92)*, ACM Press, 1992, pp. 89–98.
6. S.L. Delp and J.P. Loan, "A Graphics-Based Software System to Develop and Analyze Models of Musculoskeletal Structures," *Computers in Biology and Medicine*, vol. 25, no. 1, 1995, pp. 21–34.
7. A.J.W. Gielen, P.H.M. Bovendeerd, and J.D. Janssen, "A Finite Element Approach for Skeletal Muscle Using a Distributed Moment Model of Contraction," *Computer Methods in Biomechanics and Biomedical Eng.*, vol. 3, 2000, pp. 231–244.
8. C.A. Yucesoy et al., "Three-Dimensional Finite Element Modeling of Skeletal Muscle Using a Two-Domain Approach: Linked Fiber-Matrix Mesh Model," *J. Biomechanics*, vol. 35, no. 9, 2002, pp. 1253–1262.
9. G.R. Liu, *Mesh Free Methods: Moving Beyond the Finite Element Method*, CRC Press, 2002.
10. J. Chang and J.J. Zhang, "Mesh-Free Deformations," *Computer Animation and Virtual Worlds*, vol. 15, nos. 3–4, 2004, pp. 211–217.
11. M. Müller et al., "Point Based Animation of Elastic, Plastic, and Melting Objects," *Proc. 2004 ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA 04)*, Eurographics Assoc., 2004, pp. 141–151.

as $\mathbf{v}_t + \Delta \mathbf{v}$ and the displacement $\mathbf{u}_t + \Delta \mathbf{u}$ as $\mathbf{u}_t + \Delta \mathbf{v} \Delta t$. Here, $\Delta \mathbf{v}$ is computed with the traditional implicit integration¹⁰ scheme by solving the following equation

$$\left(\mathbf{I} - \Delta t \frac{1}{\rho} \frac{\partial \mathbf{F}}{\partial \mathbf{v}} - \Delta t^2 \frac{1}{\rho} \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \right) \Delta \mathbf{v} = \Delta t \frac{1}{\rho} \left(\mathbf{F}_t + \Delta t \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \mathbf{v}_t \right), \quad (4)$$

where \mathbf{I} is a 3×3 identity matrix, and ρ is the mass density at the point. We approximate all the differentiation operations using the moving least squares method. Step by step, we numerically simulate a muscle's physical changes through its movement history.

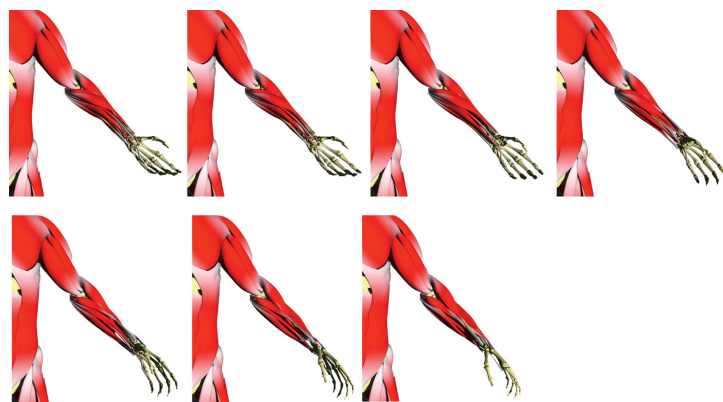
As far as the implementation is concerned, a muscle's two ends are rigidly attached to the bone, so let's assume that one bone is rigidly transformed. We can select some points distributed at



(a)



(b)



(c)

Figure 4. Muscle-deformation illustrations. (a) Pectoralis major, deltoid, and latissimus dorsi muscles following the shoulder movement. (b) Bending left arm. (c) The forearm following a twist movement.

one end of the muscle and regulate their positions to the respective bone's movement.

For contact deformation, the movement of a point along the surface normal is restricted by penalty forces to prevent penetration, and the

movement in the tangent plane is subject to friction forces. In our implementation, we consider the contact between muscles and bones, which is modeled as a contact problem between soft and rigid objects. Tendon deformation is modeled in the same way as muscles, but its strain energy contains only passive parts, which take the form of hyperelasticity.

Results

In our prototype system, we have more than 3,000 geometrical surface models including muscles, ligaments, and tendons, as well as a skeletal bone structure. The motion library contains data for basic motions, including walking, running, and jumping, and more complex ones, such as dancing. The muscle-deformation library is still under construction, but so far, we've accomplished the deformation of the major muscles of the limbs and torso. Figures 4a, 4b, and 4c show some deformation pictures.

To illustrate a muscle's motion and deformation interactively, response speed is essential. In the runtime phase, the morphing of muscle samples (called *interpolation*) and surface rendering take the most time. However, both processes involve very simple computations and are effectively supported by current GPU hardware. The only module that involves high computational overhead is the construction of the muscle-deformation library, which is settled in the preprocessing phase.

For the mesh-free computation, each muscle is represented with 400 discretized points. Normally, it takes no more than 3 seconds to generate a sample shape for a single muscle. If we use a coarse model with 200 points, we can reduce the computational time to one-third. Table 1 shows the computing time it took to create our muscle examples in Figure 4.

Although our technique's primary application is medical illustration, the technological developments we present in this article can easily apply to other uses. By considering accurate physical properties, for example, we can reproduce detailed anatomical modeling and complex interactions between soft tissues. This is useful for medical simulation and is potentially valuable for medical diagnosis of muscle- and joint-related conditions, such as those resulting from sports injuries.

Our future work will examine the complex interactions between soft tissues and the devel-

opment of a theoretically more realistic contact model to represent the phenomena that occur between muscles and other soft tissues.

Acknowledgments

Our research is funded by the British Arts and Humanities Research Council grant B/RG/ANS263/APN12727. We're also grateful to Autodesk for its donation of the Maya software licenses.

References

1. M.J. Ackerman, "The Visible Human Project," *Proc. IEEE*, vol. 86, no. 3, 1998, pp. 504–11.
2. R. Adams and L. Bischof, "Seeded Region Growing," *IEEE Trans. Pattern and Machine Intelligence*, vol. 16, no. 6, 1994, pp. 641–647.
3. J. Fan et al., "Automatic Image Segmentation by Integrating Color-Edge Extraction and Seeded Region Growing," *IEEE Trans. Image Processing*, vol. 10, no. 10, 2001, pp. 1454–1466.
4. A.B. Ekoule, F.C. Peyrin, and C.L. Odet, "A Triangulation Algorithm from Arbitrary Shaped Multiple Planar Contours," *ACM Trans. Graphics*, vol. 10, no. 2, 1991, pp. 182–199.
5. S. Ganapathy and T.G. Dennehy, "A New General Triangulation Method for Planar Contours," *ACM SIGGRAPH Computer Graphics*, vol. 16, no. 3, 1982, pp. 69–75.
6. P. Hansbo, "Generalized Laplacian Smoothing of Unstructured Grids," *Comm. Numerical Methods in Eng.*, vol. 11, no. 5, 1995, pp. 455–464.
7. A. Mohr and M. Gleicher, "Building Efficient, Accurate Character Skins from Examples," *ACM Trans. Graphics*, vol. 22, no. 3, 2003, pp. 562–568.
8. D. Winter, *Biomechanics and Motor Control of the Human Movements*, 2nd ed., Wiley, 1990.
9. X. Zhou and J. Liu, "NURBS-Based Galerkin Method and Application to Skeletal Muscle Modeling," *Proc. 2005 ACM Symp. Solid and Physical Modeling (SPM 05)*, ACM Press, 2005, pp. 71–78.
10. W.H. Press et al., *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., Cambridge Univ. Press, 1992.

Xiaosong Yang is a research fellow at the National Centre for Computer Animation, Bournemouth University, UK. His research interests include 3D modeling, animation, real-time rendering, virtual reality, virtual surgery simulation, and computer-aided design. Yang has a PhD in computing mechanics from Dalian University of Technology, China. Contact him at xyang@bournemouth.ac.uk.

Jian Chang is a research fellow at the National Centre for Computer Animation, Bournemouth University, UK. His main research interests include computer animation, physically based simulation, and geometric modeling. Chang has a PhD in computer graphics from Bournemouth University. Contact him at jchang@bournemouth.ac.uk.

Jian J. Zhang is a professor of computer graphics and

director of the Computer Animation Research Centre at Bournemouth University, UK. His research interests include computer graphics, computer animation, computer games, physically based simulation, medical simulation, and visualization. Contact him at jzhang@bournemouth.ac.uk.

Silver Bullet Security Podcast series

Sponsored by



digital and
SECURITY & PRIVACY

Check out the Silver Bullet Security Podcast with host

Gary McGraw, author of *Software Security*, *Exploiting Software*, and *Building Secure Software*! This free series features in-depth interviews with security gurus, including:

Avi Rubin of Johns Hopkins • Marcus Ranum of Tenable Security • Mike Howard of Microsoft • Bruce Schneier of Counterpane Internet Security • and more!



Stream it online or download to your iPod...

www.computer.org/security/podcasts

Writers

Visit www.computer.org/cise/author.htm.

Letters to the Editors

Send letters to Jenny Stout, Lead Editor, jstout@computer.org.

Provide an email address or daytime phone number.

On the Web

Access www.computer.org/cise/ or <http://cise.aip.org>.

Subscribe

Visit https://www.aip.org/forms/journal_catalog/order_form_fs.html or www.computer.org/subscribe/.

Subscription Change of Address (IEEE/CS)

Send an email to address.change@ieee.org. Please specify *CISE*.

Subscription Change of Address (AIP)

Send general subscription and refund inquiries to subs@aip.org.

Missing or Damaged Copies

Email help@computer.org. For AIP subscribers, email claims@aip.org.

Reprints of Articles

For price information or to order reprints, email cise@computer.org or fax +1 714 821 4010.

Reprint Permission

Contact William Hagen, IEEE Copyrights and Trademarks Manager, at copyrights@ieee.org.

www.computer.org/cise/ or <http://cise.aip.org>

How to
Contact **CiSE**