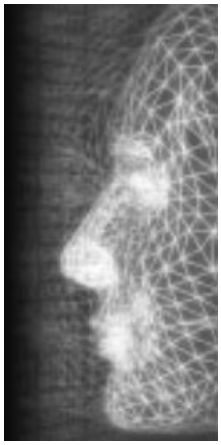
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2006; **17**: 479–489
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cav.150

Realistic human hand deformation



*By Jieun Lee, Seung-Hyun Yoon and Myung-Soo Kim**

We present a new approach to realistic hand modeling and deformation with real-time performance. We model the underlying shape of a human hand by means of sweeps which follow a simplified skeleton. The resulting swept surfaces are blended, and an auxiliary surface is then bound to the swept representation in the palm region. In the areas of this palm-control surface where bulges occur in certain poses of a real hand, the vertices are given their own trajectories, so that the palm forms realistic shapes as the joints bend. Palm lines can also be modeled as valleys in the skin by sketching them on a displacement map on the palm-control surface, and activitating them when appropriate joint movements take place. Self-intersections and collisions are detected using geometric primitives that are automatically generated from, and deform with, the sweeps and palm surface. Our algorithm runs in real time, and the naturalism of its results are demonstrated by comparative images of modeled and real hands, including several challenging poses.

Received: 10 April 2006; Revised: 2 May 2006; Accepted: 10 May 2006

KEY WORDS: human hand modeling; hand deformation; palm deformation; palm lines; sweep surfaces; freeform surface; displacement map; self-intersection; collision detection

Introduction

The hand distinguishes primates from other animals, and plays an important role in everyday life. As our sphere of action starts to extend from the real to the virtual world, naturalistic animation of human characters becomes increasingly important. Up to now, human animation research has mainly been focused on modeling the body and face. Our current work considers the hand, which has received much less research attention; and it is a topic that entails many difficult subproblems such as palm deformation and the handling of collisions between moving fingers. Hand animation is very important in VR as it provides the interface for human communication using gestures.

Skeletal-subspace deformation (SSD)[1] achieves skin deformation from a skeletal motion, but it has problems of buckling and collapsing in extreme poses. Example-based methods[2,3] produce realistic hand deformations in a limited range of poses, but they too have difficulty in handling extreme poses due to limitations in scanning. Many physically based methods of hand modeling cannot support real-time applications because of the amount of computation required. Realistic palm deformations, showing folding effects and palm lines, are also very difficult to realize using existing techniques. In this paper, we propose a new approach to hand modeling and deformation which provides practical solutions to these difficult problems and produces realistic hand deformations in real time.

The basis of our technique is the sweep-based human body deformation technique proposed by Hyun et al.[4] On top of that, we use a freeform surface to model the palm, which is how we achieve natural bulge and folding effects. Realistic palm lines can then be added using a displacement map of the freeform surface. Self-intersections and collisions are detected by an auxiliary approximation of the hand, composed of simple geometric primitives.

Our hand deformation procedure is summarized in Figure 1. The skeletal structure of Figure 1(b) is determined from the hand mesh model of Figure 1(a). Five sweeps, from the wrist to each finger and the thumb, are derived from the skeleton, as shown in Figure 1(c). Mesh vertices are then bound to the sweeps. Figure 1(d) shows a freeform surface that controls the palm deformation, and the palm vertices are bound to this surface. Palm lines can then be drawn by the user using a simple sketch

*Correspondence to: Myung-Soo Kim, School of Computer Science and Engineering, Seoul National University, San 56-1, Sillim-dong, Gwanak-gu, 151-744 Seoul, Korea. E-mail: mskim@cse.snu.ac.kr

Contract/grant sponsor: Korean Ministry of Information and Communication (MIC).

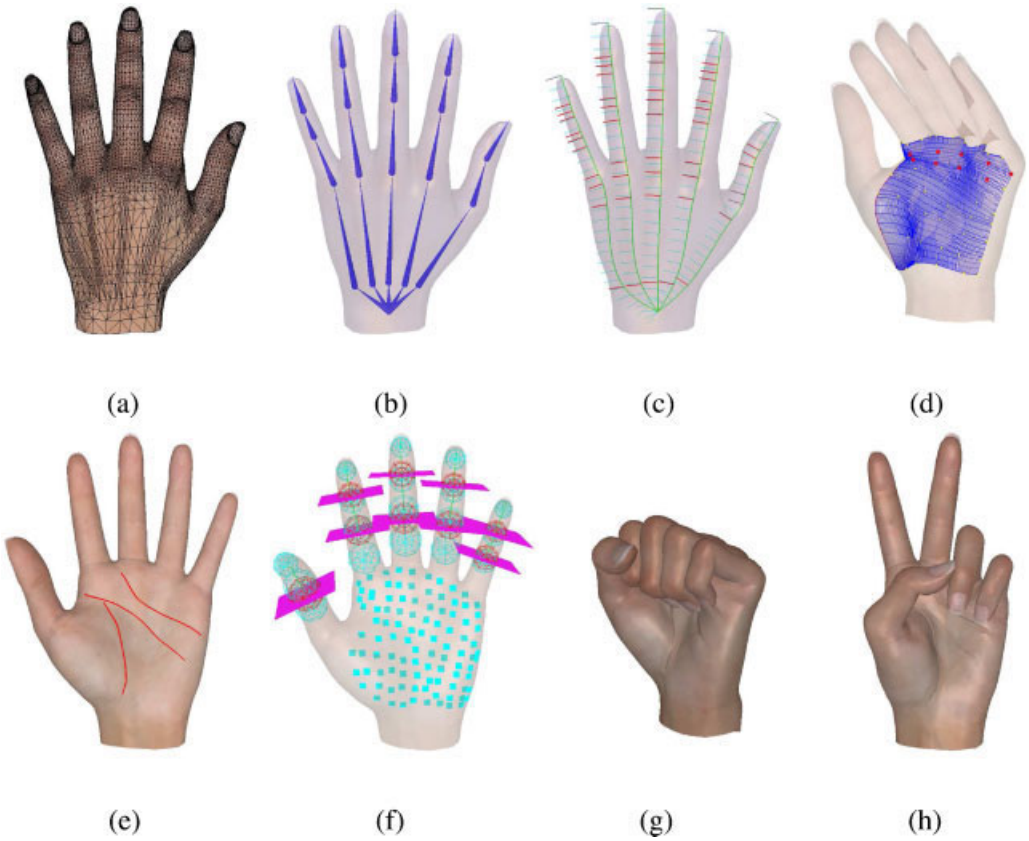


Figure 1. Overview of the hand deformation procedure: (a) hand skin mesh; (b) hand skeleton; (c) control sweeps; (d) palm-control surface; (e) palm lines; (f) geometric primitives for collision detection; and (g) and (h) resulting hand deformations.

interface, as shown in Figure 1(e), and these will change shape as the palm deforms. Finally, self-intersections are detected around finger joints, and collisions are detected among fingers and between the fingers and palm. Simple geometric primitives are used for collision detection, as shown in Figure 1(f). Figures 1(g) and (h) show expressive poses generated by this procedure.

The main contributions of our work can be summarized as follows:

- *Sweep-based hand deformation* produces *realistic hand shapes* even in extreme poses.
- *Realistic palm deformation* is achieved by a freeform surface that resolves the problems of bulge and crush.
- *Realistic palm lines* are generated using a displacement map of the palm surface.
- *Self-intersection* and *collision detection* are resolved using geometric primitives automatically generated from sweeps and the palm surface.
- The algorithm achieves *real-time performance*.

Previous Work

Coquillart[5] presented a sweep-based modeling technique that used profile curves and offsets of cross-sectional curves within the traditional sweeping operation of CAD. Lazarus et al.[6] applied the sweeping approach to freeform deformation using axial deformations. Their sweeping technique is based on a moving Frenet frame. Chang et al.[7] proposed more flexible sweeping techniques based on motions described by B-splines. Hyun et al.[4] presented a human body deformation technique using multiple sweeps. They represented the articulation of a body as sweep-based deformations, and introduced a vertex-blending technique to combine multiple sweeps.

Magnenat-Thalmann et al.[1] introduced a skeleton-driven technique known as SSD, in which each vertex of a hand mesh is bound to each effecting joint. The position of each vertex is determined by a blending technique. Our approach also follows a skeleton-driven paradigm,

but we resolve the buckling and collapsing problems of the SSD method by the use of control sweeps. Moccozet et al.[8] used Dirichlet free-form deformations to model a flexing hand. But this requires many control points to represent wrinkles and bulges over the whole hand. We avoid this problem by employing simple geometric tools for shape control.

Example-based shape deformation[9,10] has been adapted to hand deformations. Kry et al.[2] used principal component analysis to compute the eigendisplacements between range-scanned examples and hand models obtained by SSD, and their method allows low-dimensional displacements to be interpolated in real time using graphics hardware. Kurihara et al.[3] constructed an accurate skeleton of the hand from medical images and produced weighted pose-space deformations using examples of skin and skeleton. Their method generates various poses from a limited number of examples by assigning a different weight to each joint. Example-based methods have the obvious disadvantage that it is necessary to acquire examples, which are difficult to obtain for some extreme poses. For instance, a fist or a 'victory v' are very difficult to acquire by range scanning.

There are also many physically based methods of hand modeling, which rely on detailed anatomy and biomechanics[11–13]. These methods produce realistic motion of the hand and realistic deformation of bone, muscles and skin. But their ability to represent superficial details of the hand such as palm lines is limited. It is also difficult to support physics-based methods in real time because of the amount of computation that is necessary.

Bando et al.[14] modeled the wrinkles on the back of the hand and knuckles, whereas we concentrate on the palm of the hand, where the shape changes are more dramatic.

Hand Modeling

Skeleton

We create the fundamental shape of a hand using five sweeps, each running along the skeleton from the wrist all the way to the tip of each finger and the thumb. Figure 2(a) shows the skeletal structure that we use to avoid excessive complication: it includes only the major joints, the angles of which are dominant in forming hand shape. They are the carpometacarpal(CMC) joints near the wrist, the metacarpophalangeal (MCP) joints, the interphalangeal (IP) joint of the thumb, the proximal interphalangeal (PIP) joints and the distal interphalangeal (DIP) joints of the fingers.

Control Sweeps

Figure 2(b) shows five control sweeps that follow the shape of the skeleton. Each sweep runs from the wrist to the tip of a finger or the thumb and passes through the intermediate joints. A sweep is defined by two spline curves which interpolate the positions and orientations of a sequence of key frames (see the red tick marks in Figure 2(b)), which are allocated to each joint, with additional frames to smooth the sharp change of orientation over the knuckles. The sweep corresponding to each finger is generated by 12 key frames, while the thumb has 9. The trajectory of each sweep is a cubic B-spline curve which interpolates the origins of the key frames. The orientations of the key frames are represented as unit quaternions, and are linearly interpolated. When the user changes the joint angles to generate a new pose, the control sweeps reflect the changes to the key frames.

Binding the Mesh Vertices

A sweep can be regarded as a continuously moving frame. To bind a vertex in the skin mesh to a sweep, we need to find the time at which that vertex is contained in the cross-sectional plane of the moving frame and also find the polar coordinates of the vertex in that plane. A vertex V of the skin mesh is bound to the control sweep by the parameters (t, θ, d), where t is the time parameter of both the trajectory curve $T(t)$ and the quaternion curve $Q(t)$. The other parameters (θ, d) are the polar coordinates of V in the cross-sectional plane of the moving frame. Therefore, a vertex V_i with the parameters (t_i, θ_i, d_i) can be reconstructed as

$$V_i = R(t_i)\,[d_i \cos\theta_i \quad d_i \sin\theta_i \quad 0]^T + T(t_i) \qquad (1)$$

where $R(t)$ is a 3D rotation matrix corresponding to the unit quaternion $Q(t)$.

A vertex on the skin mesh can simultaneously be bound to more than one sweep. The vertices on the mesh that correspond to the fingers and thumb themselves are bound to a single sweep, while the vertices corresponding to the palm and the back of the hand are bound to multiple sweeps. For example, the red vertex V in Figure 2(b) is bound to the thumb sweep $\mathcal{S}_{\text{thumb}}$ with parameters $(t_{\text{thumb}}, \theta_{\text{thumb}}, d_{\text{thumb}})$, and also to the index

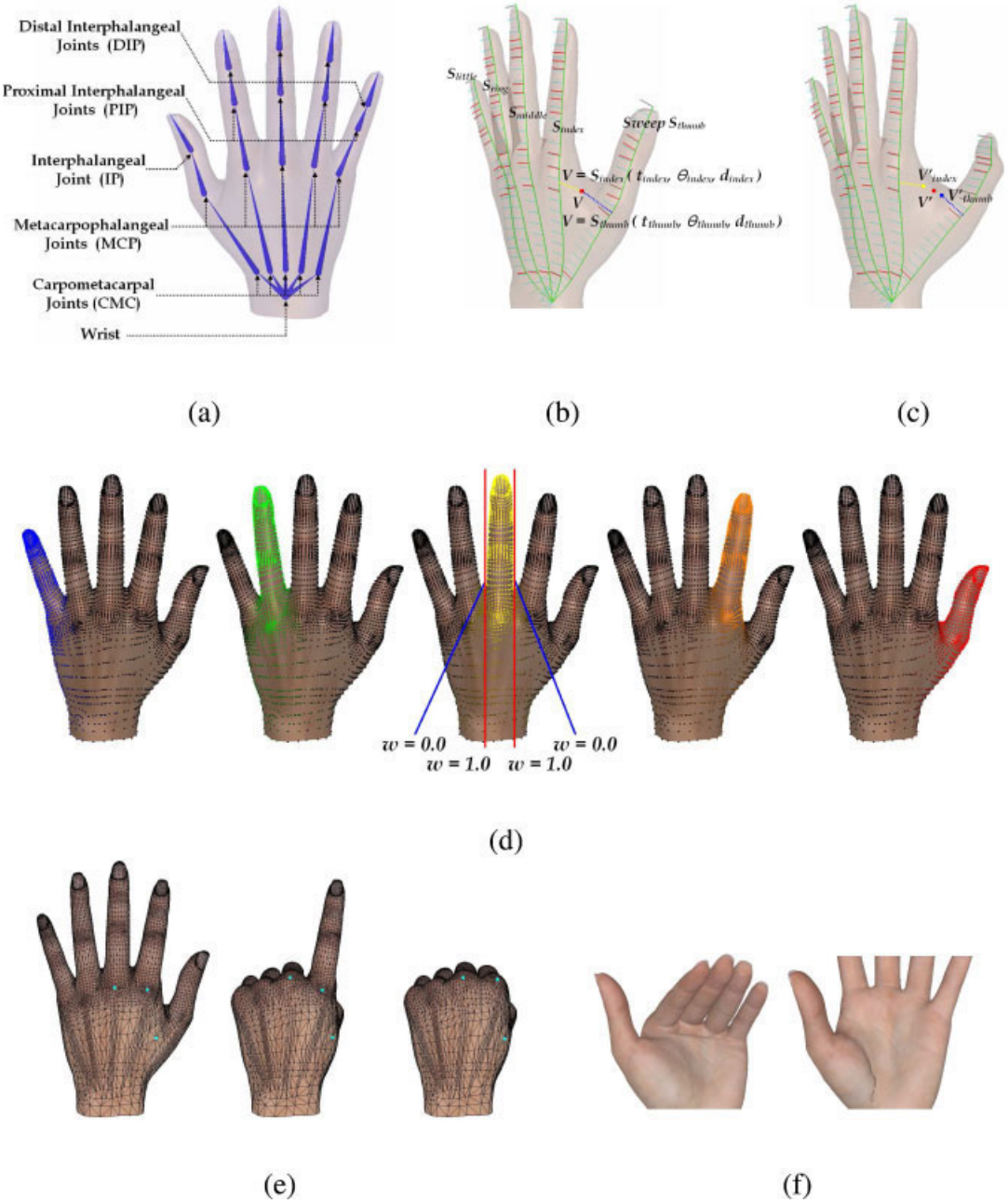


Figure 2. Hand deformation based on control sweeps: (a) hand skeleton; (b) control sweeps and vertex binding; (c) vertex reconstruction by blending control sweeps; (d) weight assignments for five control sweeps; (e) elastic movement of the skin in the sweep-based deformation; and (f) shortcomings in the sweep-based deformation.

finger sweep $\mathcal{S}_{\text{index}}$ with parameters $(t_{\text{index}}, \theta_{\text{index}}, d_{\text{index}})$. This means that the vertices in a finger or thumb are affected only by the deformation of that finger or thumb, but the vertices in the palm or the back of the hand are affected by the deformations of multiple digits.

Blending Sweeps

Suppose vertex V in Figure 2(b) is bound to both the thumb and the index finger sweeps, and the hand is deformed from the pose of Figure 2(b) to that of

Figure 2(c). We can reconstruct V after the pose change using Equation (1). The problem is that the two vertices V'_{thumb} and V'_{index}, reconstructed from two different sweeps, are not the same. But we can blend the two reconstructed results as follows:

$$V' = \frac{w_{\text{thumb}}}{w_{\text{thumb}} + w_{\text{index}}} V'_{\text{thumb}} + \frac{w_{\text{index}}}{w_{\text{thumb}} + w_{\text{index}}} V'_{\text{index}}$$

where w_{thumb} and w_{index} are weights that correspond to the thumb and index finger respectively. These weights are initialized to correspond to the relative distances between a vertex and its controlling sweeps in the rest pose. For example, in the case of the middle finger sweep, the weights are assigned so that they are inversely proportional to the distance from the red cylinder around the finger in Figure 2(d). The vertices inside the cylinder have a maximum weight of 1.0, and those outside the blue cone have no weight. Intermediate vertices have weights between 0.0 and 1.0. After weights have been assigned to all five sweeps, they are normalized to make the sum of the weights at each vertex 1.0, so as to achieve a convex combination. Figure 2(d) shows the results of weight assignments for five sweeps.

Hand Deformation

Hand deformation is realized through the following procedure. The hand skeleton articulates to follow the joint angles. The control sweeps are then updated based on the new skeleton. All mesh vertices are reconstructed from the updated control sweeps, and the vertices are then blended using the vertex-to-sweep weights. Figures 2(e) and (f) show some example sweep-based hand deformations. Figure 2(e) shows the deformation of the back of the hand while forming a fist pose. The elasticity of the skin is apparent, and the fingers and the back of the hand are realistic; but problems remain in the palm.

Palm Deformation

Using the sweep-based model, the palm appears to crush, as shown in Figure 2(f). Instead it should bulge when the MCP joints of the fingers and the CMC joint of the thumb bend.

Palm Control Surface

We propose a simple method of resolving this problem, by employing a freeform surface to control the palm deformation. This palm-control surface is a bi-cubic B-spline surface that interpolates a set of palm vertices which are updated after the sweep-based deformation. A changed palm-control surface is then produced by interpolating these new vertex locations. The vertices to be interpolated by the palm surface are selected experimentally, making sure that there are enough vertices near the branching points of the finger and thumb to create a surface which can control the palm accurately. We used 54 vertices (the colored vertices in Figure 3(a)). We started by generating a set of cubic B-spline curves using a chord-length parametrization for knot spacing (Figure 3(a)) and then constructed the palm-control surface by interpolating these curves (Figure 3(b)).

Binding the Mesh Vertices

Once the palm-control surface has been generated, all the vertices on the palm are bound to it. A palm vertex V is bound to the palm-control surface $S(u, v)$ using an orthogonal projection on to that surface. Let d denote the distance between V and $S(u, v)$. Using the binding parameters (u, v, d), we can now reconstruct the vertex as

$$V = S(u, v) + dN(u, v) \tag{2}$$

where $N(u, v)$ is the unit normal vector to $S(u, v)$.

Palm Deformation

To deform the palm, we first determine which interpolation vertices are moving. The red vertices in Figure 3(a) are moving interpolation vertices, and they are located in the crushing area (see Figure 2(f)) when the MCP joints bend. The moving interpolation vertices follow their own paths, while the remaining interpolation vertices (the green vertices in Figure 3(a)) are reconstructed from the sweep-based deformation. The moving interpolation vertices are designed to rise when the corresponding MCP joint bends, deforming the palm-control surface to achieve a natural bulging effect. We have only 10 moving interpolation vertices, and the correspondence between the moving interpolation vertices and the MCP joints, shown in Figure 3(c), allows a bulge to be localized to a nearby MCP joint. The final positions of the moving interpolation vertices are predefined based on the hand anatomy[15,16], and they correspond to the maximum bending angles of the MCP joints, while intermediate positions are determined by the corresponding joint angles.

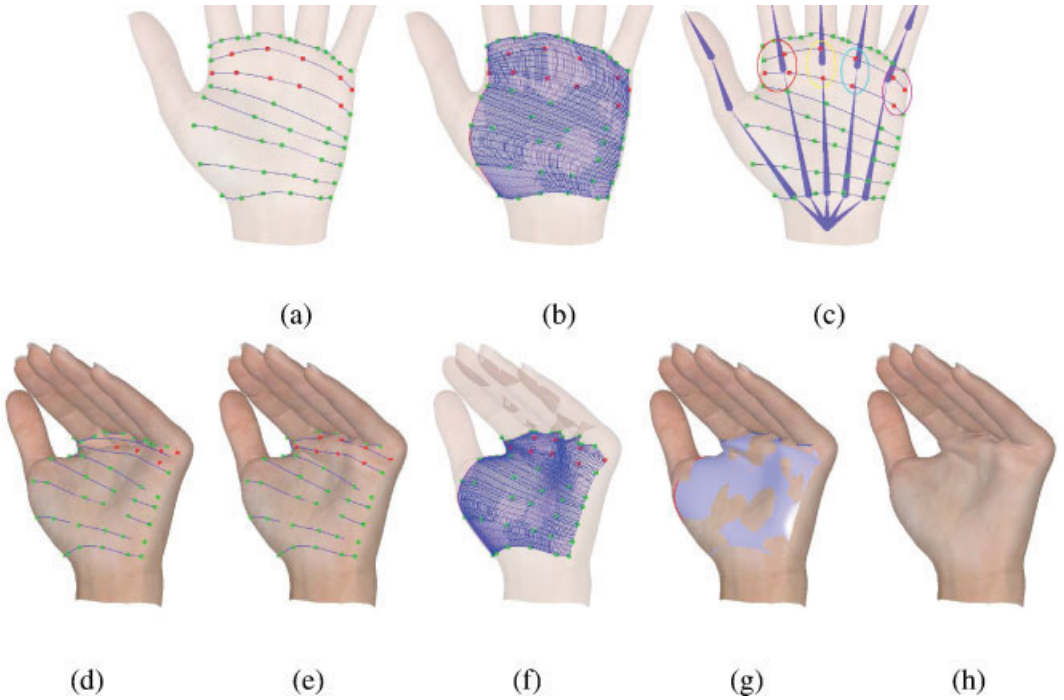


Figure 3. Palm deformation based on palm-control surface: (a) palm interpolation vertices (green and red vertices), moving palm interpolation vertices (red vertices) and palm interpolation curves; (b) the palm interpolation surface; (c) the correspondence between the moving interpolation vertices and the MCP joints; (d) palm interpolation vertices reconstructed using control sweeps; (e) modified moving interpolation vertices; (f) the palm-control surface modified using the interpolation vertices and curves; (g) deforming the palm-control surface and reconstructing palm vertices; and (h) a deformed palm.

Figures 3(d)–(h) illustrates palm deformation. Figure 3(d) shows what happens when all the interpolation vertices move to positions reconstructed from the sweeps, and the crushing problem occurs as described above. In Figure 3(e), however, the moving interpolation vertices are located at their predefined positions, resulting in different interpolation curves. From these curves, a new palm-control surface is generated, as shown in Figure 3(f). We now reconstruct the palm vertices using the new palm-control surface and Equation (2), and finally obtain the more accurately deformed palm shown in Figure 3(h).

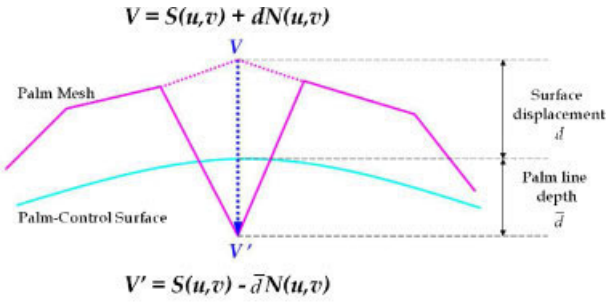
In this procedure, the binding parameters (u, v, d) of each palm vertex are fixed, even though the palm-control surface is re-created, so as to preserve the consistency of the bound vertices. The palm-control surface is in essence generated by the sweep deformation, except for the bulging area, thus we do not need to blend the result of the surface-based deformation with that of the sweep-based deformation.

Figure 6 shows several hand positions modeled using this procedure. The palm locally bulges around the bending MCP joints of the fingers and the crushing problem shown in Figure 2(f) is also resolved.

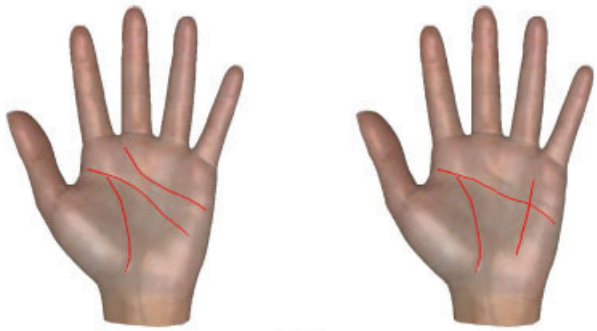
Palm Lines

Another important reason for using the palm-control surface is that it allows us to produce realistic palm lines, by offsetting the vertices on a palm line from the palm-control surface into the hand (Figure 4(a)). The displacement map that controls this process is edited by sketching the palm lines on the hand mesh (Figure 4(b)). Various different types of palm lines can easily be supported using this approach.

Palm lines can be activated or deactivated depending on the pose of the hand, by means of variable depths in the displacement map. The depth of the palm lines changes with the bending of the CMC joint of the thumb and the MCP joints of the fingers that contribute to forming palm lines. The following equation determines the depth of a palm line at a



(a)



(b)

Figure 4. Palm lines: (a) the representation of a palm line vertex and (b) different types of palm lines sketched by a user.

vertex V_i

$$\begin{aligned}\text{PalmLineDepth}(V_i) &= \text{Depth}_{\max} \times \left[\frac{\text{Ang(ThumbCMC)}}{\text{ThumbCMC}_{\max}} \times \text{Wgt}(V_i, S_{\text{thumb}}) \right. \\ &+ \frac{\text{Ang(IndexMCP)}}{\text{IndexMCP}_{\max}} \times \text{Wgt}(V_i, S_{\text{index}}) \\ &+ \frac{\text{Ang(MiddleMCP)}}{\text{MiddleMCP}_{\max}} \times \text{Wgt}(V_i, S_{\text{middle}}) \\ &+ \frac{\text{Ang(RingMCP)}}{\text{RingMCP}_{\max}} \times \text{Wgt}(V_i, S_{\text{ring}}) \\ &\left. + \frac{\text{Ang(LittleMCP)}}{\text{LittleMCP}_{\max}} \times \text{Wgt}(V_i, S_{\text{little}}) \right]\end{aligned}$$

where $\text{Depth}_{\max}$ is a constant specifying the maximum depth of palm line vertices, while the constants $\text{ThumbCMC}_{\max}$, $\text{IndexMCP}_{\max}$, $\text{MiddleMCP}_{\max}$, $\text{RingMCP}_{\max}$, and $\text{LittleMCP}_{\max}$ limit the maximum angles of the thumb CMC joint and the finger MCP joints. Ang() and Wgt() are functions that retrieve a joint angle and a vertex-to-sweep weight respectively. This formulation makes the palm lines dependent on the sweeps as well as on the joint angles and allows a palm line to be partially activated. Figure 6 shows some examples of this palm line representation. When the CMC joint of the thumb bends, only the vertical palm line between the thumb and the index finger is formed and the other palm lines are dormant. When the MCP joint of the ring finger bends, only the lines near this finger are formed.

Handling Self-intersections and Collisions

Self-intersections may occur in a hand model when the fingers and thumb touch a palm, when the fingers and thumb touch each other, or when bending joints rumple nearby skin. To detect these self-intersections, we form a simple approximation of the hand using spheres and planes. To detect self-intersections around the knuckles, we use the cross-sectional plane at each joint. These geometric primitives can be generated automatically from the sweep and surface representations, and they allow us to detect collisions efficiently.

Folding at Joints

To detect self-intersections around bending joints, we first compute the cross-sectional planes at the IP joint, PIP joints, and DIP joints (see the pink planes in Figure 5(b)). We will assume that a vertex V has a sweep parameter t_V and that corresponding joint has a parameter t_J. The vertex V is self-intersecting when one of the following conditions holds

- V is above the cross-sectional plane at the joint, while $t_V < t_J$, or
- V is below the cross-sectional plane at the joint, while $t_V > t_J$.

Self-intersections can then be resolved by projecting the self-intersecting vertices on to the cross-sectional plane. Figures 5(c) and (d) shows the situation before and after this process.

Collision among Fingers, Thumb and Palm

Simple geometric primitives are commonly used for collision detection in hand animation[17,18]. We use spheres and planes to detect self-intersections of fingers, thumb and palm. Spheres are created along each finger and thumb at sampled values of the sweep parameter t, and their diameters are automatically determined using the

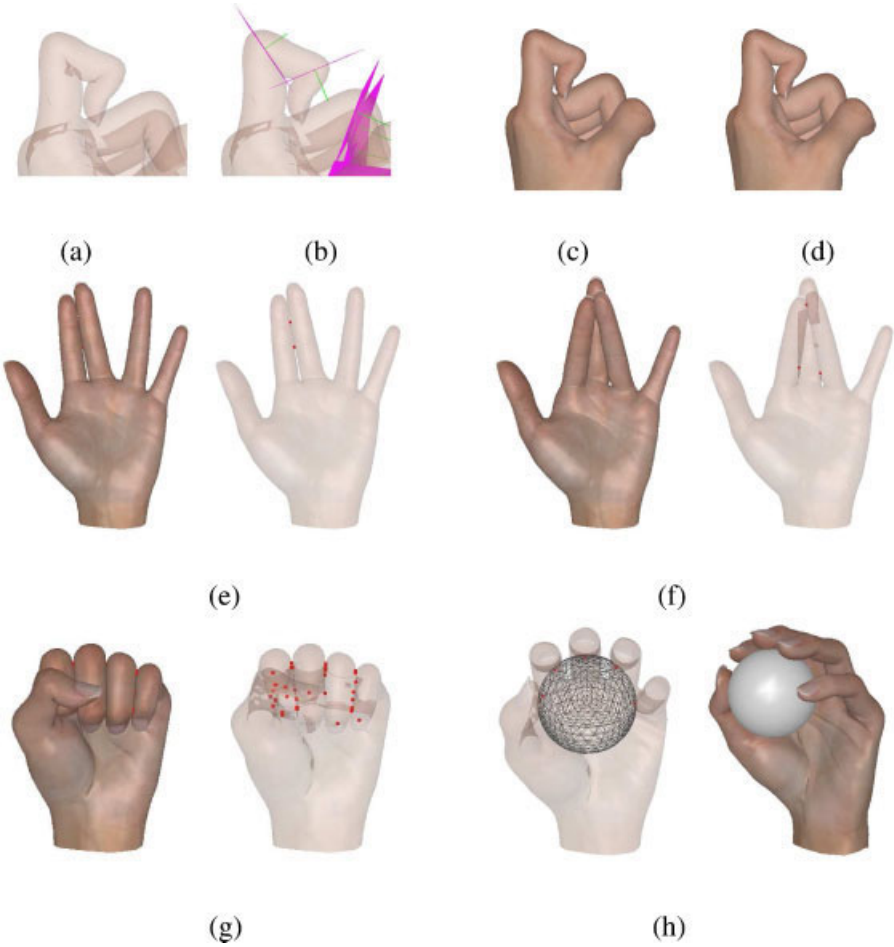


Figure 5. Handling self-intersections and collisions: (a) self-intersections at the joints of the index finger (transparent); (b) projection of the self-intersecting vertices on to the corresponding cross-sectional planes at the joints; (c) self-intersections at the joints of the index finger (opaque); (d) the result of eliminating self-intersections; (e) collision detection between the index finger and the middle finger; (f) collision detection between the index finger and the ring finger; (g) collision detection in the fist pose; and (h) collision detection between a hand and a ball.

sweep parameter d of the skin vertices. We add further spheres at each IP, PIP, and DIP joint (the spheres in Figure 1(f)). Tangent planes are created in a similar way at sampled values of the parameters u and v of the palm-control surface. Figure 1(f) shows both the spheres and the planes. We can control the sampling intervals to meet the speed and accuracy requirements of a specific application.

We begin collision detection by checking the joint angles so as to exclude unnecessary tests. Then we check for sphere-sphere intersections and sphere-plane intersections. Figure 5(e)–(g) shows the results of collision detection. The points in Figure 5(g) are contact points. Figure 5(e) shows the result of collision detection between the index finger and the middle finger, and Figure 5(f) shows the result of collision detection between the index finger and the ring finger. Collisions at the roots of fingers are nicely detected, as well as those at the tips. Figure 5(g) shows the detection of collisions that occur in forming the fist pose. Note how collisions are detected along each side of the fingers, and between the fingertips and the palm.

Collision Detection with Other Objects

We can apply the same method to detect collisions between a hand and other objects. Figure 5(h) shows a hand holding a ball.

Experimental Results

We implemented our human hand modeling and deformation algorithm in C++ on a Pentium IV 3.4 GHz computer with a 1 GB main memory. Our hand model has 7524 vertices and 15 017 triangles.

Figure 6 shows an array of hand deformation results achieved by our system for various different poses.

Figure 6. Comparative results of hand deformation in challenging poses.

Photographs of a real hand are provided for comparison. These results include the fist and 'victory v' poses which were very difficult to realize using previous methods.

Our system is able to animate a hand at about 19 frames per second. This includes palm deformation, palm line generation, and the elimination of self-intersections, but not the rendering process. The most time-consuming step in our approach is the evaluation of the palm-control surface, in particular the reconstruction of palm vertices using Equation (2). To reduce the computing time, we precompute values of the B-spline basis functions for the surface parameters (u, v) of each palm vertex, and then we multiply them by the varying control points of the surface at execution time.

Conclusion

Our three-level hand representation scheme provides a promising shape control mechanism for deforming human hand models. The use of sweeps and a freeform palm surface greatly simplify the control of changing hand shapes and provide realistic deformation results in a wide range of challenging poses. Representing palm lines as a displacement map is simple and incurs almost no computational overhead; moreover, it greatly improves the realism of the deformed palm shapes. Our technique also supports the elimination of self-intersections and efficient collision detection among fingers, thumb, and palm. In future work, we plan to extend our use of sweeps to collision detection, so as to improve its precision, and to apply our approach to hand modeling to a wider variety of challenging problems.

ACKNOWLEDGEMENTS

This work was supported by the Korean Ministry of Information and Communication (MIC) under the Program of IT Research Center on CGVR.

References

1. Magnenat-Thalmann N, Lemperriére R, Thalmann D. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings of Graphics Interface*, 1988, pp. 26–33.
2. Kry PG, James DL, Pai DK. Eigenskin: Real time large deformation character skinning in hardware. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2002, pp. 153–159.
3. Kurihara T, Miyata N. Modeling deformable human hands from medical images. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2004, pp. 355–363.
4. Hyun D-E, Yoon S-H, Chang J-W, Seong J-K, Kim M-S, Juttler B. Sweep-based human deformation. *The Visual Computer* 2005; **21**(8–10): 542–550.
5. Coquillart S. A control-point-based sweeping technique. *IEEE Computer Graphics and Applications* 1987; **7**(11): 36–45.
6. Lazarus F, Coquillart S, Jancéne P. Axial deformations: an intuitive deformation technique. *Computer-Aided Design* 1994; **26**(8): 607–613.
7. Chang T-I, Lee J-H, Kim M-S, Hong S-J. Direct manipulation of generalized cylinders based on B-spline motion. *The Visual Computer* 1998; **14**(5): 228–239.
8. Moccozet L, Magnenat-Thalmann N. Dirichlet free-form deformations and their application to hand simulation. In *Proceedings of Computer Animation*, 1997, pp. 93–102.
9. Lewis JP, Cordner M, Fong N. Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of ACM SIGGRAPH*, 2000, pp. 165–172.
10. Sloan P-PJ, Rose C, Cohen MF. Shape by example. In *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics*, 2001, pp. 135–144.
11. Albrecht I, Haber J, Seidel H-P. Construction and animation of anatomically based human hand models. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003, 98–109.
12. ElKoura G, Singh K. Handrix: Animating the human hand. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2003, pp. 110–119.
13. Tsang W, Singh K, Fiume E. Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005, pp. 319–328.
14. Bando Y, Kuratate T, Nishita T. A simple method for modeling wrinkles on human skin. In *Proceedings of Pacific Conference on Computer Graphics and Applications*, 2002, pp. 166–175.
15. Electronic Textbook of Hand Surgery. http://www.eatonhand.com/
16. The eSkeletons Project. http://www.eskeletons.org/
17. Huang Z, Boulic R, Magnenat-Thalmann N, Thalmann D. A multi-sensor approach for grasping and 3D interaction. In *Proceedings of Computer Graphics International*, 1995, pp. 235–254.
18. Mas Sanso R, Thalmann D. A hand control and automatic grasping system for synthetic actors. *Computer Graphics Forum* 1994; **13**(3): C167–C177.

Authors' biographies:

Jieun Lee is a Ph.D. student in the School of Computer Science and Engineering, Seoul National University. Her research interests include computer graphics and geometric modeling. Ms. Lee received her B.S. degree in Computer Science and Engineering from Ewha Womans University in 1997 and received her M.S. degree in Computer Science and Engineering from POSTECH in 1999. She worked at LG Electronics Institute of Technology as a research engineer from 1999 to 2002.

Seung-Hyun Yoon is a Ph.D. student in the School of Computer Science and Engineering, Seoul National University. His research interests include computer graphics and geometric modeling. Mr. Yoon received his B.S. degree in Mathematics from Hanyang University in 2001.

Myung-Soo Kim is a professor and the head of the School of Computer Science and Engineering, Seoul National University. His research interests are in computer graphics and geometric modeling. Prof. Kim received B.S. and M.S. degrees from Seoul National University in 1980 and 1982 respectively. He continued his graduate study at Purdue University, where he received an M.S. degree in applied mathematics in 1985 and M.S. degree and Ph.D. in computer science in 1987 and 1988, respectively. From then until 1998, he was with the Department of Computer Science, POSTECH, Korea. Prof. Kim serves on the editorial boards of Computer-Aided Design, Computer Aided Geometric Design, Computer Graphics Forum, and the *International Journal of Shape Modeling*. He has also edited several special issues of journals such as *Computer-Aided Design*, *Graphical Models*, the *Journal of Visualization and Computer Animation*, *The Visual Computer*, and the *International Journal of Shape Modeling*. Recently, together with Gerald Farin and Josef Hoschek, he edited the *Handbook of Computer Aided Geometric Design*, North-Holland, 2002.