Realistic Human Hand Deformation[1]
(Extended Abstract)

Jieun Lee, Seung-Hyun Yoon, and Myunng-Soo Kim
Seoul National University, KOREA

ABSTRACT

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 activating 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.

CR Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling.

INTRODUCTION

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) [6] achieves skin deformation from a skeletal motion, but it has problems of buckling and collapsing in extreme poses. 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 methods [4, 5] produce realistic hand deformations in a limited range of poses, but they too have difficulty in handling extreme poses due to limitations in scanning. There are also many physically based methods of hand modeling, which rely on detailed anatomy and biomechanics [1]. These methods produce realistic motion of the hand and realistic deformation of bone, muscles and skin. But They 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.

Hyun et al. [3] 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. The basis of our technique is the sweep-based human body deformation technique proposed by Hyun et al. [3]. 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.

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

[1] A full version of this paper is appeared in the proceedings of Computer Animation and Social Agents 2006.

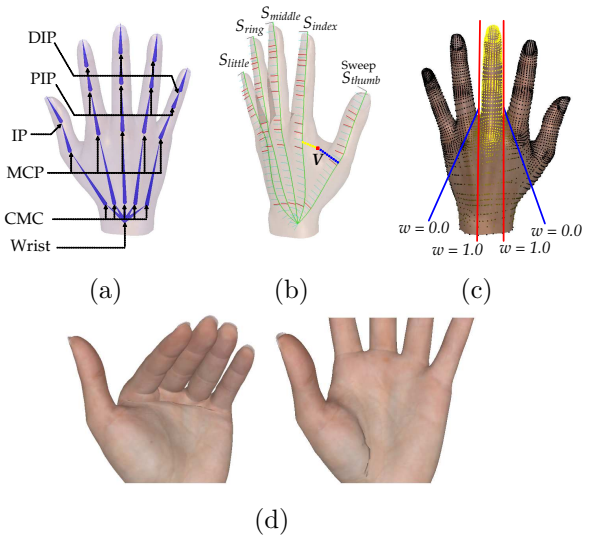

Figure 1: Hand modeling and deformation based on sweeps: (a) hand skeleton, and (b) control sweeps and vertex binding, (c) weight assignment for middle finger sweep, and (d) shortcomings in the sweep-based deformation.

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**.

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 1(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 1(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 1(b)), which are allocated to each joint, with additional frames to smooth the sharp change of orientation over the knuckles. When the user changes the joint angles to generate a new pose, the control sweeps reflect the changes to the key frames.

Binding the skin 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 orientation 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

$$V_i = R(t_i)\begin{bmatrix} d_i\cos\theta_i & d_i\sin\theta_i & 0 \end{bmatrix}^T + T(t_i), \qquad (1)$$

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. 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 1(b) is bound to both the thumb and the index finger sweeps, and the hand pose changes. We can reconstruct V after the pose change using Equation (1). The problem is that the two vertices, reconstructed from two different sweeps, are not the same. But we can blend the two reconstructed by weighted sum. The 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 1(c). 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.

Hand deformation. Hand deformation is realized through the following procedure. The hand skeleton articulates to follow the joint angles. The control

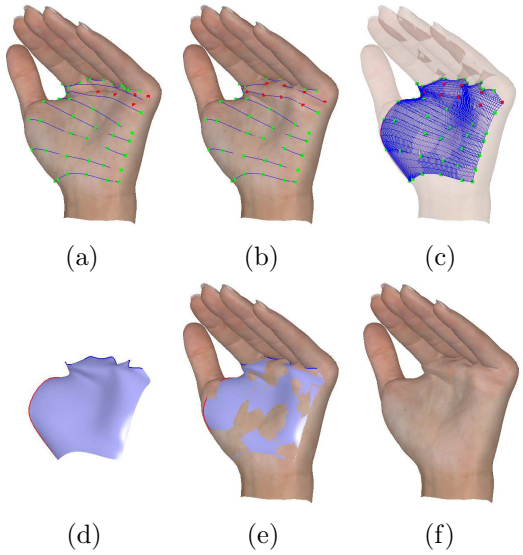

Figure 2: Palm deformation based on spline surface: (a) palm interpolation vertices reconstructed using control sweeps, (b) modified moving interpolation vertices (red), (c) and (d) palm-control surface modified using the interpolation vertices and curves, (e) reconstructing the palm vertices, and (f) a deformed palm.

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. Sweep-based hand deformation generates reasonably good results in global shape, but problems remain in the palm.

PALM DEFORMATION

Using the sweep-based model, the palm appears to crush, as shown in Figure 1(d). 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. We used 54 vertices (the colored vertices in Figure 2(a)). We started by generating a set of cubic B-spline curves by interpolating these vertices and then constructed the palm-control surface by interpolating these curves.

Binding the skin 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 2(a) are moving interpolation vertices, and they are located in the crushing area when the MCP joints bend. The moving interpolation vertices follow their own paths, while the remaining interpolation vertices (the green vertices in Figure 2(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. In Figure 2(b), moving interpolation vertices result in different interpolation curves, and, from these curves, a new palm-control surface is generated, as shown in Figure 2(c) and (d). 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 2(f).

We have only 10 moving interpolation vertices, and the correspondence between the moving interpolation vertices and the MCP joints, 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, and they correspond to the maximum bending angles of the MCP joints, while intermediate positions are determined by the corresponding joint angles.

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 4 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 1(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. The displacement map that controls this process is edited by sketching the palm lines on the hand mesh. Hence, various different types of palm lines can easily be supported using this approach.

The depth of the palm lines changes with the sweep weights as well as angles of joints that contribute to forming palm lines. It allows a palm line to be partially activated. Figure 4 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

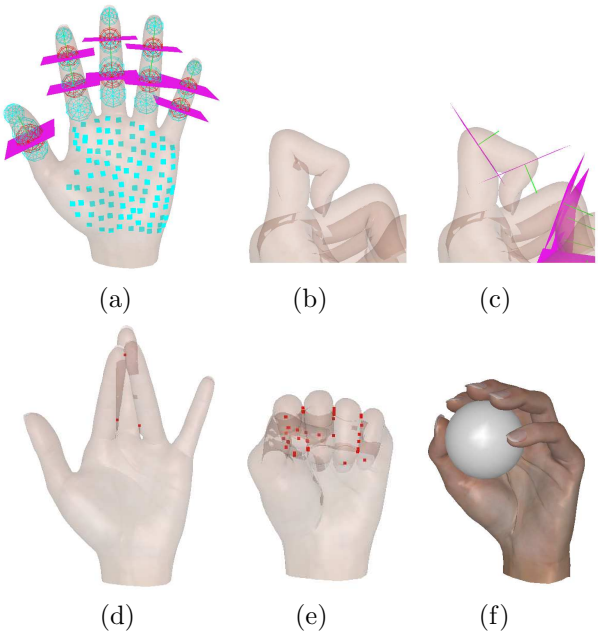

Figure 3: Handling self-intersections and collisions: (a) geometric primitives, (b) self-intersections at the joints of the index finger, (c) projection of the self-intersecting vertices on to the corresponding cross-sectional planes at the joints, (d) and (e) collision detection among fingers and palm, and (f) collision detection between a hand and a ball.

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 (See Figure 3(a)).

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 3(c)). 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 3(c) shows the result after this process.

Collision among fingers, thumb and palm. Simple geometric primitives are commonly used for collision detection in hand animation [2, 7]. 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 sweep parameter d of the skin vertices. We add further spheres at each IP, PIP, and DIP joint (the red spheres in Figure 3(a)). Tangent planes are created in a similar way at sampled values of the parameters u and v of the palm-control surface. Figure 3(a) 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. Figures 3(d) and (e) show the results of collision detections between the index finger and the middle finger, collision detections between the index finger and the ring finger, collision detections in forming the fist pose. The red points are contact points.

Collision detection with other objects. We can apply the same method to detect collisions between a hand and other objects. Figure 3 (f) shows a hand holding a ball.

EXPERIMENTAL RESULTS

We implemented our human hand modeling and deformation algorithm in C++ on a Pentium IV 3.4GHz computer with a 1GB main memory. Our hand model has 7,524 vertices and 15,017 triangles.

Figure 4 shows an array of hand deformation results achieved by our system for various different 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. 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 layered 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.

ACKNOWLEDGEMENT

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

REFERENCES

[1] I. Albrecht, J. Haber, H.-P. Seidel. Construction and Animation of Anatomically Based Human Hand Models. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 98-109, 2003.

[2] Z. Huang, R. Boulic, N. Magnenat-Thalmann, D. Thalmann. A Multi-sensor Approach for Grasping and 3D Interaction. In *Proceedings of Computer Graphics International*, pages 235-254, 1995.

[3] D.-E. Hyun, S.-H. Yoon, J.-W. Chang, J.-K. Seong, M.-S. Kim, B. Juttler. Sweep-based Human Deformation. *The Visual Computer*, 21(8-10):542-550, 2005.

[4] P. G. Kry, D. L. James, D. K. Pai. Eigenskin: Real Time Large Deformation Character Skinning in Hardware. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 153-159, 2002.

[5] T. Kurihara, N. Miyata. Modeling Deformable Human Hands from Medical Images. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 355-363, 2004.

[6] N. Magnenat-Thalmann, R. Lemperrière, D. Thalmann. Joint-Dependent Local Deformations for Hand Animation and Object Grasping. In *Proceedings of Graphics Interface*, pages 26-33, 1988.

[7] R. Mas Sanso, D. Thalmann. A Hand Control and Automatic Grasping System for Synthetic Actors. *Computer Graphics Forum*, 13(3):C167-C177, 1994.

[8] L. Moccozet, N. Magnenat-Thalmann. Dirichlet Free-Form Deformations and their Application to Hand Simulation. In *Proceedings of Computer Animation*, pages 93-102, 1997.

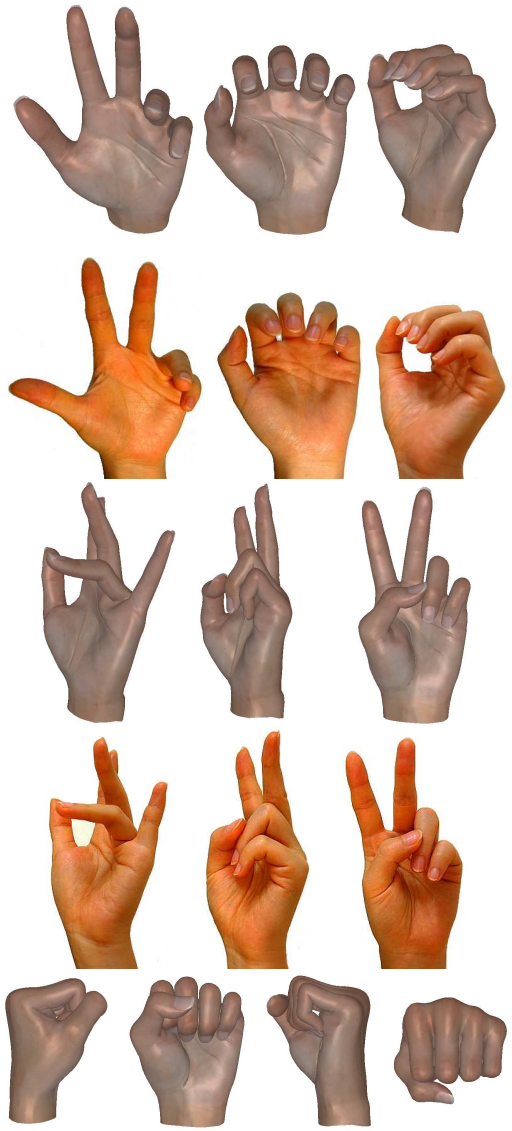

Figure 4: Comparative results of hand deformation in challenging poses.