Surface Deformation Using the Sensor Glove

Lizhuang Ma [1 2] Rynson W.H. Lau [2] Jieqing Feng [1] Qunsheng Peng [1] Janis Wong [2]

[1] State Key Lab. of CAD&CG, Zhejiang University, Hangzhou 310027, P.R. China

[2] Computer Graphics and Media Laboratory, Department of Computing, The Hong Kong Polytechnic University, Hong Kong

Abstract

Intuitive 3D surface control and deformation are crucial to CAD/CAM. To do this in a virtual environment, however, the technique must be very efficient. A common method for shape deformation is the free-form deformation (FFD) method, in which the complete object is deformed by deforming a 3D grid of the object. In this paper, we propose an intuitive method for surface deformation based on deforming a hand surface, which is basically a bicubic B-spline surface interpolating or approximating key data points of a sensor glove (i.e. finger joints and palm center of the user's hand). By setting up a corresponding mapping between the virtual object being deformed and the hand surface, the object can be deformed with the control of the sensor glove. As the user flexes his/her fingers, the object changes its shape accordingly. Such control can be local or global. For local deformation, we introduce a region filter function which imposes locality on the mapping/deformation. The new algorithm is made very efficient through incremental update. It is also intuitive as if the user were using his hand to deform the object directly. Experimental results show the potential of the new method.

1. Introduction

Interactive surface design and deformation have been extensively studied in both CAD/CAM and computer graphics. After the surface or object has been created, subsequent modifications are likely necessary. One common way to modify the shape of a free-form surface is to modify its control points one by one [5,10,11]. However, the modification process becomes tedious if the surface or object is composed of a large number of patches with many control points. Thus interactive tools for manipulating a set of control points or sampled points are preferred. Existing deformation methods and linear transformation methods include [1,4,12]. In virtual reality, the user immerses into a computer synthesized environment and interacts with some virtual objects inside it. The most natural tool for interaction is the user's hand. To modify the shape of a surface or object, one may want to do it simply by flexing one's hand and fingers. The sensor glove is a tool designed for capturing the motion of the user's hand. Commercial products include VPL's DataGlove, Virtual Technologies' CyberGlove, and Mattel's PowerGlove. They all have sensors that measure some or all of the finger joint angles. In addition to capturing the user's hand gesture, a 3D position sensor may also be used to track the 3D position of the user's hand inside the virtual environment. Our idea is to make use of the sensor glove in the deformation process so that the shape of a virtual object can be modified intuitively. A surface called hand surface is created, which is a bicubic B-spline surface interpolating or approximating key data points of the glove (i.e. the user's hand) including finger joints and palm center. This surface is used to control and deform the shape of the virtual object. By setting up a corresponding mapping between the object surface to be deformed and the hand surface, the deformation can be handled in an incremental manner and the shape of the object surface changes according to the shape of the sensor glove. To allow local deformation, we also introduce a region filter function $W(u,v)$ which imposes locality on the mapping/deformation. With this filter, we may modify the object surface either globally or locally. In order to eliminate the illness state of the deformed surfaces, an α blending parameter is used between the initial and the active normals of the hand surface. When α decreases from 1 to 0, the corresponding surface deformation becomes more and more sensitive to the active normal of the hand surface. The new method allows surface deformation in an easy and intuitive way. We show some experimental results of the new method towards the end of the paper.

The rest of the paper is organized as follows. Section 2 briefly discusses some general concepts on B-spline curves and surfaces. Section 3 presents our method of surface deformation using the sensor glove, and section 4 discusses some techniques for accelerating the proposed method. Section 5 presents some results and discusses the strengths and limitations of the method. Finally, section 6 presents conclusions of this work and discusses some possible future work.

2. B-spline Curves and Surfaces

A B-spline curve in 3D is a vector-valued piecewise polynomial function of the form:

$$C(t) = \sum_{i=0}^{n} P_i N_{i,k}(t)$$

where P_i are Euclidean control points. $N_{i,k}(t)$ are the normalized B-spline basis functions of order k or degree $k-1$, and are defined over the knot vector $T = \{t_0, t_1, ..., t_{n+k}\}$.

For $k=1$: $N_{i,k}(t)=\begin{cases}1 & \text{for} \quad t_i \le t < t_{i+1}\\ 0 & \text{otherwise}\end{cases}$

For $k>1$: $N_{i,k}(t)=\frac{t-t_i}{t_{i+k-1}-t_i}N_{i,k-1}(t)+\frac{t_{i+k}-t}{t_{i+k}-t_{i+1}}N_{i+1,k-1}(t)$

A B-spline curve has some well-known properties. For example:

- Local support. $N_{i,k}>0$ for $t\in(t_i,t_{i+k})$ and vanishes elsewhere.
- Derivative formulas.

$$\begin{cases}\frac{d}{dt}N_{i,k}(t)=\frac{k-1}{t_{k+i-1}-t_i}N_{i,k-1}(t)-\frac{k-1}{t_{k+i}-t_{i+1}}N_{i+1,k-1}(t)\\ \frac{d^r}{dt^r}[C(t)]=(k-1)(k-2)\ldots(k-r)\sum_{i=0}^{n-r}P_i^{[r]}N_{i,k-r}(t)\end{cases} \quad (1)$$

where

$$P_i^{[r]}=\begin{cases}P_i & r=0\\ \frac{P_i^{[r-1]}-P_{i-1}^{[r]}}{t_{k+i-r}-t_i} & \text{otherwise}\end{cases}$$

If the knot vector of an order k B-spline curve has k multiple end knots, it is called a Bézier type B-spline. A Bézier type B-spline interpolates the two end control points and is tangential to the control polygon at the end points. A bicubic B-spline surface is defined similarly with spline basis and its control points $P_{i,j}$, $i=0,1,\ldots,m$ and $j=0,1,\ldots,n$.

$$H(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n}P_{i,j}N_{i,3}(u)N_{j,3}(v)$$

The derivative of a B-spline surface can be computed in a similar way.

3. Surface Deformation with the Sensor Glove

Human fingers have flexion-extension and lateral abduction-adduction as illustrated in Figure 1. The process of transforming raw sensor data into finger joint angles is called "glove calibration". The calibration method differs between flexion sensors, and abduction/adduction ones. The sensor calibration method uses a least-square formula that was given by [6].

$$\theta(r)=a+br+c\ln(r)$$

where r is the raw sensor reading, $\theta(r)$ is the flex angle of the finger. a, b, and c are calibration constants. Most of the sensor glove in their standard configuration do not measure the flexion angle θ_1 of the distal joints, except for the thumb. One way to determine θ_1 is to take into account the coupling that exists between θ_1 and θ_2 over the range of grasping motions. Experimental measurements showed that the general coupling formula is of the form [2]:

$$\theta_1=a_0-a_1\theta_2+a_2\theta_2^2$$

where the parameters $a_0, a_1,$ and a_2 depend on the hand characteristics of individual user.

To simplify the presentation, we number the five fingers, thumb, index, middle, ring and pinkie as fingers *1, 2, 3, 4* and *5*, and let the joint points of finger i be $Q_{i,1}$, $Q_{i,2}$, $Q_{i,3}$, $Q_{i,4}$, $Q_{i,5}$. If the parameters of the middle finger L_1, L_2, L_3, L_4 are given, which depend on the characteristics of the user's hand, the joint points $Q_{3,1}$, $Q_{3,2}$, $Q_{3,3}$, $Q_{3,4}$, $Q_{3,5}$ can be calculated according to the measured angles θ_2 and θ_3, and the local coordinate of the palm. The joint points of other fingers can be determined similarly.

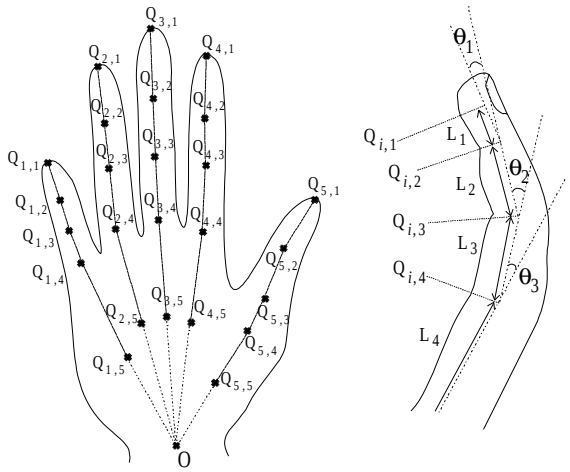

Figure 1. Finger parameters of the sensor glove.

3.1 Constructing the Hand Surface

To construct the hand surface which interpolates the measured joint points $Q_{i,j}$, $i,j=1,\ldots,5$, we use the bicubic B-spline surface. The knot vectors, $U = \{u_0, u_1,\ldots, u_9, u_{10}\}$ and $V = \{v_0, v_1,\ldots, v_9, v_{10}\}$ in U and V directions respectively, are calculated with chord length. If we let L_u^i, L_v^j be the average chord lengths in U and V directions such that

$$\begin{cases}L_u^i=\frac{1}{5}\sum_{j=1}^{5}\left|Q_{i+1,j}-Q_{i,j}\right| & i=1,2,3,4\\ L_v^j=\frac{1}{5}\sum_{i=1}^{5}\left|Q_{i,j+1}-Q_{i,j}\right| & j=1,2,3,4\end{cases}$$

Then the knot vectors U and V are chosen as follows:

$$\begin{cases}u_0=u_1=u_2=u_3=0 \quad \text{and} \quad u_7=u_8=u_9=u_{10}\\ v_0=v_1=v_2=v_3=0 \quad \text{and} \quad v_7=v_8=v_9=v_{10}\\ u_{3+l}=u_{l+2}+L_u^l \qquad l=1,2,3,4\\ v_{3+l}=v_{l+2}+L_v^l \qquad l=1,2,3,4\end{cases}$$

For convenience, we normalize the knot vectors such that $u_{10} = v_{10} = 1.0$. The domain of parameter (u, v) is thus a unit square region. If the hand surface $H(u,v)$ is a bicubic B-spline surface:

$$H(u,v)=\sum_{i=0}^{6}\sum_{j=0}^{6}P_{i,j}N_{i,3}(u)N_{j,3}(v)$$

Then from the interpolating conditions, $H(u_i, v_i) = Q_{i,j}$, $i=1,\ldots,5$. According to the end condition, four rows of virtual data points $Q_{i,j}$, $i,j=0,6$, should be computed such that the hand surface satisfies the given end condition [3,9]. The following equation can then be obtained:

$$\boldsymbol{Q} = \boldsymbol{APB} \tag{2}$$

where $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{A}$ and $\boldsymbol{B}$ are 7×7 matrices defined as:

$$\boldsymbol{P}=(P_{i,j}),\quad \boldsymbol{Q}=(Q_{i,j}),\quad \boldsymbol{A}=(N_{i,3}(u_j)),\quad \boldsymbol{B}=(N_{j,3}(v_i))$$

The surface for interpolating all the given data points and satisfying the specific end conditions can be obtained in a two-step process. First, piecewise cubics $F_i(v)$, $i=1,\ldots,5$, are used to interpolate the data points $Q_{i,j}$, $j=1,\ldots,5$, with the knot vector $V = \{v_0, v_1,\ldots, v_9, v_{10}\}$ and to satisfy the specific end condition, which the quadratic end condition is chosen here. Second, $S(u,v)$ is obtained to interpolate the 5 curves $F_i(v)$ as shown in Figure 1.

In some situations, we may want to use a hand surface to approximate the key data points of the sensor glove. For fast computation, the hand surface is represented by a Bézier type B-spline surface and its control points are simply set equal to the data points. The knot vectors in U and V directions are also chosen as the chord length. However, one more row of data points are added in U direction so that the resulting hand surface approximates well the shape of the glove. The added data points are obtained by extending the last segments with a ratio β_i, which is chosen according to the characteristics of the user's hand:

$$Q_{i6} = Q_{i5} + \beta_i(Q_{i5} - Q_{i4})$$

Thus the hand surface can be obtained very efficiently as follows.

$$\begin{cases} H(u,v) = \sum_{i=0}^{4}\sum_{j=0}^{5} P_{i,j} N_{i,3}(u) N_{j,3}(v) \\ P_{i,j} = Q_{i+1,j+1} \qquad i = 0,\ldots,4,\ j = 0,\ldots,5 \end{cases} \tag{3}$$

3.2 Setting up the Corresponding Mapping

In the FFD method [12], a 3D bounding volume is used to deform a virtual object. In our method, however, surface deformation can be considered as a mapping of the hand surface to the surface of the object. It is, therefore, necessary to set up a well-defined correspondence mapping between the two. A planar region or the so-called base surface, RH_0, is defined. It is the co-domain of the hand surface in flat state whose control points are all in a plane π_H. The object to be deformed is embedded in the extended 3D parametric space of RH_0. The local coordinate of the hand surface is defined as follows. Let the unit normal vector of π_H be N_H^0 and C_H be the center point of the palm. If R is any point on π_H, then the equation of π_H is $N_H^0(R-C_H)=0$. Let $P=(x_p, y_p, z_p)$ be a control point or sampled point of the object surface. When P is projected on to π_H, its projection $P'=(x_p', y_p')$ with respect to π_H can be obtained as shown in Figure 2. If the projection P' of P is within RH_0, then its parameter (u_p, v_p) on the base surface RH_0 can be calculated easily by approximation method. An acceleration method can be designed similar to the computation of normal vectors (discussed in section 4). When the projection P' of P is outside RH_0, P will not be modified.

When the sensor glove changes its shape, the hand surface $H(u,v)$ changes accordingly. P is then transformed to a new position Q by the following mapping:

$$Q = H(u_p, v_p) + d_p N(u_p, v_p) \tag{4}$$

where $N(u_p, v_p)$ is the unit normal vector of $H(u,v)$ at (u_p, v_p) and d_p is the directional distance of P to the base surface RH_0. The geometric meaning of the above deformation is that the object is deformed along the normal directions as if it were in the force field formed by the hand surface. When P is on the positive side of the hand surface, i.e. $(P' - P)\bullet N_H^0 > 0$, then $d_p > 0$; otherwise $d_p \le 0$.

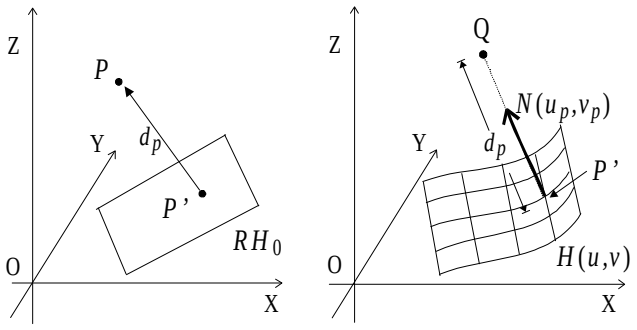

Figure 2. The correspondence between P, P' and Q.

3.3 Defining the Blending Parameter

Although Equation (4) can map a hand surface to the object surface, it is sensitive to the normal variation of the hand surface, especially if the glove is far away from the object surface. The object surface will change rapidly if the normal of the hand surface varies quickly. In some extreme cases, the resulting surface is far from satisfactory. For example, the modified objects may be self-intersected as shown in images *ii-b* and *ii-c* of Plate 1. A modified mapping method is introduced here to solve this problem.

Initially, the user extends the sensor glove to a flat state. The base surface RH_0 and the local coordinate of the hand surface are set up. The normal vector, N_H^0, of the initial hand surface is recorded, and the deformation of an object can be made with respect to this direction. An α blending parameter can be used to relate the current normal of the hand surface at (u_p, v_p) to N_H^0. The mapping shown in Equation (4) is then modified as follows.

$$Q = H(u_p, v_p) + d_p[(1-\alpha)N(u_p, v_p) + \alpha N_H^0]_I \tag{5}$$

where $[V]_I$ denotes the unit normal vector of V. By selecting different values of α, the user can achieve different effects. When $\alpha=0$, the mapping shown in Equation (5) degenerates to Equation (4). When $\alpha=1$, the mapping is similar to the FFD method in which the deformation is a linear extrusion of the hand

surface along N_H^0. When α decreases from 1 to 0, the corresponding mapping becomes more and more sensitive to the active normal vector of the hand surface. The deformation of the virtual object also becomes less and less coherent to the shape of hand surface.

3.4 Global Surface Deformation

In global surface deformation, all sample points or control points of the surface are modified. This can be done by scaling the projection region RH_0 so that all the projected points will lie inside of RH_0. This is similar to scaling down the size of the object to be modified or scaling up the size of the hand surface.

3.5 Local Surface Deformation

In local surface deformation, only part of a surface is modified. There are two approaches to attain this goal: localizing the object surface $R(u,v)$ to be modified, or localizing the effect of the mapping in Equation (5). The first approach depends on the representation of $R(u,v)$. It requires $R(u,v)$ being a free-form surface. The second approach is independent of the representation of $R(u,v)$. However, it is slightly more complex.

Localizing the object surface

An active region of the surface is specified in its parameter space or in the 3D space. If a parameter region $\Omega = [u_1, u_2; v_1, v_2]$ is specified, the surface $R(u,v)$ is subdivided into sub-surfaces along parameter values $u = u_1, u_2$ and $v = v_1, v_2$. This can also be achieved by knot insertion algorithm [3,7]. The parametric region Ω can be defined with the sensing glove by, for example, specifying the two diagonal points of the region on the object surface. After the surface is localized, the sub-surface in the active region Ω can be modified by the sensor glove.

Localizing the mapping

The effect of the mapping in Equation (5) can be localized by introducing a region filter. To simplify the discussion, we will describe the 1D filter function first. We specify the effective region, $R_s = [a,b]$, and the fillet region, $R_f = [e,f]$, of the hand surface as shown in Figure 3. This fillet region is introduced to preserve the continuity of the deformed object surface. For the region filter function, $F(u)$, it is 1 when inside region R_s, 0 when outside the fillet region R_f, and a cubic function which joins smoothly between $F(u)=1$ and $F(u)=0$. If $B_{i,3}(u) = \binom{3}{i}(1-u)^{3-i}u^i$, $u \in [0,1]$, *i=0,1,2,3,4*, are the cubic Bernstein-Bézier basis functions, then $F(u)$ can be represented as follows:

$$F(u) = \begin{cases} 0 & u \notin [e,f] \\ 1 & u \in [a,b] \\ B_{2,3}(\frac{u-e}{a-e}) + B_{3,3}(\frac{u-e}{a-e}) & u \in [e,a] \\ B_{0,3}(\frac{u-b}{f-b}) + B_{1,3}(\frac{u-b}{f-b}) & u \in [b,f] \end{cases}$$

In fact, $F(u)$ can be represented as a Bézier curve by introducing related control points, for instance,

$$(e,0), (e+\frac{1}{3}(a-e),0), (e+\frac{2}{3}(a-e),1), (a,1)$$

over the interval [e, a]. $u_i = e + \frac{i}{3}(a-e)$, *i=0,1,2,3*, are often called the Bézier coordinates of the four control points over [e, a] as shown in Figure 3.

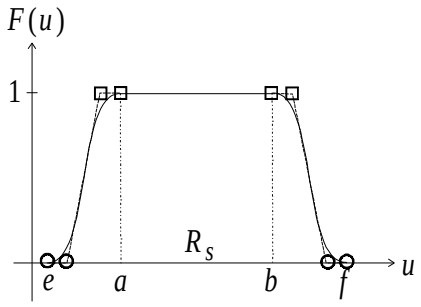

Figure 3. The smooth region filter function $F(u)$ and its Bézier representation in [e, a]. ("○" and "□" denote the Bézier coordinates at $F(u)=0$ and 1 respectively.)

In the 2D case, the region filter $W(u,v)$ is a generalized tensor product function of $F(u)$ and $F^*(v)$. The smooth region filter $W(u,v)$ is 1 when inside the interval [a, b; c, d], 0 when outside the filter region [e, f; g, h], and bicubic (or cubic by linear) Bézier functions which join smoothly with each other over the transit regions as shown in Figure 4.

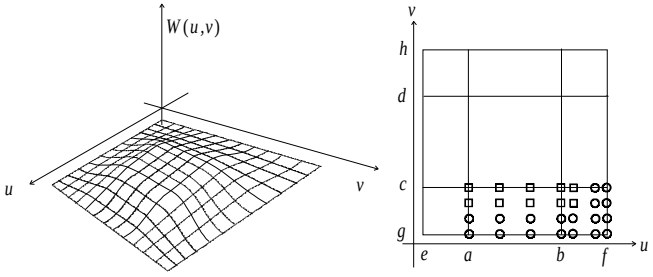

Figure 4. The region filter function $W(u,v)$ and its Bézier control points. ("○" and "□" denote the Bézier coordinates of the control points when $W(u,v)=0$ and 1 respectively.)

$W(u,v)$ can be represented as follows:

$$W(u,v)=\begin{cases}0 & u\notin[e,f;g,h]\\ 1 & u\in[a,b;c,d]\\ B_{2,3}(\frac{u-e}{a-e})+B_{3,3}(\frac{u-e}{a-e}) & (u,v)\in[e,a;c,d]\\ B_{0,3}(\frac{u-b}{f-b})+B_{1,3}(\frac{u-b}{f-b}) & (u,v)\in[b,f;c,d]\\ B_{2,3}(\frac{v-g}{c-g})+B_{3,3}(\frac{v-g}{c-g}) & (u,v)\in[a,b;g,c]\\ B_{0,3}(\frac{v-d}{h-d})+B_{1,3}(\frac{v-d}{h-d}) & (u,v)\in[a,b;d,h]\\ \sum_{i=2}^{3}\sum_{j=2}^{3}B_{i,3}(\frac{u-e}{a-e})B_{j,3}(\frac{v-g}{c-g}) & (u,v)\in[e,a;g,c]\\ \sum_{i=0}^{1}\sum_{j=2}^{3}B_{i,3}(\frac{u-b}{f-b})B_{j,3}(\frac{v-g}{c-g}) & (u,v)\in[b,f;g,c]\\ \sum_{i=2}^{3}\sum_{j=0}^{1}B_{i,3}(\frac{u-e}{a-e})B_{j,3}(\frac{v-d}{h-d}) & (u,v)\in[e,a;d,h]\\ \sum_{i=0}^{1}\sum_{j=0}^{1}B_{i,3}(\frac{u-b}{f-b})B_{j,3}(\frac{v-d}{h-d}) & (u,v)\in[b,f;d,h]\end{cases}$$

Now, if the object surface is deformed due to the deformation of the hand surface, a point *P* on the object surface before the deformation will become $Q^L(P)$ after the deformation as follows:

$$Q^L = W(u_P, v_P)Q + (1 - W(u_P, v_P))P$$

where *Q* is obtained from the mapping Equation (5). The parameter value (u_p, v_p) is calculated at the beginning. If it is outside the fillet region, Q^L is simply equal to *P*.

4. Accelerating the Computation

In this section, we discuss some techniques for accelerating the proposed algorithm. They include techniques for speeding up the computation of the surface normal, which is needed in constructing the hand surface and in deforming the object surface, and for incremental updating the hand surface and the object surface.

4.1 Fast Computation of the Surface Normal

In our implementation, the hand surface $H(u,v)$ is subdivided adaptively according to its local curvature and then approximated by planar triangles. The normal vector of the hand surface at coordinate (u_p, v_p) can then be approximated by weighted average of the normals at nearby vertices of the corresponding triangle. Assuming that $H(u_p, v_p)$ is within triangle $T_1T_2T_3$, and $T_i = H(u_i, v_i)$, *i=1,2,3*, the normal vector N_p at $H(u_p, v_p)$ can be calculated as follows (referring to Figure 5).

$$N_p = [w_1N_1 + w_2N_2 + w_3N_3]_I$$

where N_i is the normal vector of $H(u,v)$ at points T_i, and (w_1, w_2, w_3) is the barycentric coordinate of (u_p, v_p) with respect to triangle defined by $P_i = (u_i, v_i)$, *i=1,2,3*.

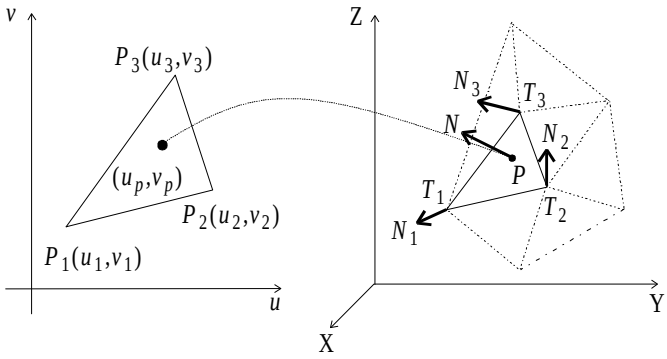

Figure 5. Approximation of normal vector at (u_p, v_p).

4.2 Incremental Update of the Hand Surface

Our method deforms an object based on a hand surface, which represents the user's hand. Because the shape of this hand surface may change in every frame during the deformation, we may need to construct a hand surface during every frame time. This can be expensive to do. To overcome this limitation, we apply an incremental technique to update the hand surface as it deforms instead of reconstructing it in every frame. Because the finger parameters are constants during the deformation process, ***A*** and ***B*** in Equation (2) are constant matrices. Their inverse matrices can therefore be pre-computed. When there are only a few data points, $P_{i,j}$, changed, the hand surface can be updated very efficiently. Let ***D*** be a matrix whose elements are all zero except for the element $\delta_{i,j}$ at location (i, j). When $P_{i,j}$ is moved by an amount $\delta_{i,j}$, the coefficient matrix, ***P***, containing the control point set will change as follows:

$$\boldsymbol{P'} = \boldsymbol{P} + \boldsymbol{A}^{-1}\boldsymbol{D}\boldsymbol{B}^{-1}$$

Since ***D*** has only one none zero element, the above equation has only $n + n^2$ multiplications of scalar with vector which are much less than $n^3 + n^2$ in general cases. Here *n=7* is the matrix order. A similar technique was proposed in our recent paper [8].

4.3 Incremental Update of the Object Surface

If the shape of the hand surface changes by only a small amount, for example, the movement of only one control point or one finger joint, then the corresponding surface point will be changed accordingly from *Q*(P) to *Q*'(P). *Q*' can be evaluated by a simple approximation formula when the data points obtained from the sensor glove are considered to be the control points of the hand surface as in Equation (3). Since the object to be deformed often has a large number of sample points, the following technique will accelerate the updating of the object surface. Suppose that a

control point $P_{i0,j0}$ of the hand surface is moved by a small amount $\delta_{i0,j0}$. Let H_u, H_v, H'_u, H'_v denote the partial derivatives of the hand surface before the deformation $H(u,v)$ and after the deformation $H'(u,v)$ at coordinate (u_p, v_p). Let also that

$$\begin{cases} \delta_{i0,j0} = a_0 H_u + b_0 H_v + c_0 H_u \times H_v \\ G(u,v) \equiv N_{i0,3}(u) N_{j0,3}(v) \end{cases}$$

The partial derivatives of *G(u,v)*, G_u, G_v can be obtained from Equation (1). Then the following results can be derived:

$$\begin{cases} H'(u_P, v_P) = H(u_P, v_P) + \delta_{i0,j0} G(u_P, v_P) \\ H'_u \times H'_v = H_u \times H_v + (H_u G_v + G_u H_v) \times \delta_{i0,j0} \\ Q' \approx Q + (\delta_{i0,j0} + \alpha d_P) G(u_P, v_P) \\ \qquad + d_P (1-\alpha)[N + c_0 (H_u G_v + G_u H_v) \times N]_I \end{cases} \quad (7)$$

When the hand surface is approximated with triangles by subdivision method, a look up table can be set up. The look up table stores the pre-computed quantities (u_i, v_i), $H(u_i, v_i)$, $H_u(u_i, v_i)$, $H_v(u_i, v_i)$, $N(u_i, v_i)$, $i=1,2,\ldots,n$, at vertices of triangles where *n* is the total number of vertices. All the above quantities at arbitrary coordinate (u_p, v_p) are weighted averages of the corresponding quantities at three vertices of respective triangle containing (u_p, v_p) as shown in Figure 5 and Equation (6). The detailed formulas are omitted here. It is clear that H', H'_u, H'_v can be obtained from H, H_u, H_v with simple computations. Thus Equation (7) involves less computation for updating the deforming object surface.

5. Results and Discussions

We have implemented the new method in C on an SGI workstation. Plate 1 shows some initial results produced with different blending parameter values. The images in columns *i* and *ii* are generated with a=0, columns *iii* and *iv* with a=0.5, and columns *v* and *vi* with a=1.0. Images in columns *i*, *iii* and *v* are in wireframe rendering while those in columns *ii*, *iv* and *vi* are corresponding images in shaded rendering. All images in row *a* show the original object and the hand surface. Images in row *b* show the deformed object and the hand surface when lifting up the thumb slightly. Images in row *c* show the deformed object and the hand surface when the thumb, index and middle fingers were up slightly. Images in row *d* show the deformed objects when local deformation is used via the region filter. Images *ii-b* and *ii-c* (with a = 0), the shape of the deformed cup is very sensitive to the active normal of the hand surface and the object is self intersected along the rim of the cup. Images *iv-b* and *iv-c* (with a = 0.5), the shape of the cup becomes less sensitive to the active normal of the hand surface. Images *vi-b* and *vi-c* (with a = 1), the deformed shape of the cup is independent of the active normal of the hand surface. Initial results demonstrated the potential of the new method in the following aspects:

1. Our method is more intuitive and convenient to use than existing methods such as Pigel's method [10,11] and FFD method [12], especially when a lot of data are involved in an interactive environment. This is because the modification tool in our method is a hand surface rather a free form volume. An object can be modified intuitively with one or several fingers moving up or down simultaneously.
2. The hand surface is used as a control tool rather than a tri-cubic Bernstein-Bézier tensor volume. As such, this method has a much lower computational cost, especially when the acceleration techniques discussed in section 4 are incorporated.
3. A region filter $W(u,v)$ is used to impose locality on the mapping/deformation. With this filter, local or global deformation can be realized easily without changing the (first order) smoothness. In this way, the deformation mapping is independent of the representation of the surfaces or objects to be modified.
4. Additional parameters are available to meet with different demands including the blending parameter a and the region parameters a, b, c, d; e, f, g, h in the region filter $W(u,v)$.

6. Conclusions

In this paper, we have introduced a new method for deforming surfaces using a sensor glove. It is by creating a hand surface either interpolating or approximating key data points of the glove. The deformation of the object surface can be achieved by deforming this hand surface. The new method is efficient and intuitive. We have also demonstrated some initial results of the method.

The work presented in this paper only concerns about how to deform an object surface. We are currently developing an editing system for object creation and modification based on the new method. We are also considering how to handle the situation when there are multiple objects to be deformed at the same time.

Acknowledgments

We are very grateful to the members of the State Key Lab of CAD&CG and the graphics research group of the Department of Computing, The Hong Kong Polytechnic University, for their helps in preparation of this paper. This project is supported in part by China National Excellent Young Science Foundation, Huo Ying-dong Foundation for Young Teachers and Zhao Guang-biao Foundation for High Technology Developments, and by the University Central Research Grant, #351/084.

References

[1] A. Barr, "Global and Local Deformation of Solid Primitives", *ACM Computer Graphics (SIGGRAPH'84)*, **18**(3), pp. 21-34, 1984.

[2] G. Burdea , J. Zhuang, E. Roskos, D. Silver and N. Langrana, "A Portable Dextrous Master with Force Feedback," *Presence: Teleoperators and Virtual Environments*, **1**(1), pp. 18-28, 1992.

[3] G. Farin, "Curves and Surfaces in Computer Aided Geometric Design," 3rd ed., Academic Press, 1993.

[4] J. Feng, L. Ma and Q. Peng, "A New Free-Form Deformation Through the Control of Parametric Surfaces," *Computer & Graphics*, **20**(4) , 1996.

[5] G. Fog, "B-Spline Surface System for Ship Hull Design," in V. Banda and C. Kuo (Eds.), *Computer Applications in the Automation of Shipyard Operation and Ship Design*, North-Holland, pp. 359-366, 1985.

[6] J. Hong and X. Tan, "Teleoperating the Utah/MIT Hand with a VPL DataGlove I. DataGlove Calibration," *Proceedings of IEEE*, pp. 1752-1757, 1988.

[7] J. Hoschek and D. Lasser (Translated by Larry L. Schumaker), *Fundamentals of Computer Aided Geometric Design*, A K Peters, 1993.

[8] F. Li, R.W.H. Lau and M. Green, "Interactive Rendering of Deforming NURBS Surfaces", To appear in the *Proceedings of Eurographics'97*, Sept. 1997.

[9] L. Ma, Q. Peng and J. Feng, "Explicit Formulas for Bicubic B-spline Surface Interpolation," *CAD/Graphics'95*, SPIE, 1995.

[10] L. Piegl, "Modifying the Shape of Rational B-Splines. Part 1: Curves," *CAD*, **21**(8), pp. 509-518, 1989.

[11] L. Piegl, "Modifying the Shape of Rational B-Splines. Part 2: Surfaces," *CAD*, **21**(9), pp. 538-546, 1989.

[12] T. Sederberg and R. Parry, "Free-Form Deformation of Solid Geometric Models," *ACM Computer Graphics (SIGGRAPH'86)*, **20**(4), pp. 151-160, 1986.