In Journal of Visualization and Computer Animation, John Wiley & Sons, **11**(3), pp. 155-166, July 2000.

Virtual 3D Sculpting

Janis P.Y. Wong [†] Rynson W.H. Lau [†] Lizhuang Ma [‡]

[†] Department of Computer Science, City University of Hong Kong, Hong Kong

[‡] State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou, P.R. China

Abstract

This paper presents a virtual sculpting method for interactive 3D object deformation. The method is based on the use of an electronic glove. A parametric *control hand surface* defined by an O*pen-Uniform B-Spline tensor product surface* is first created to model the hand gesture. The geometric attributes of the object in the Euclidean 3D space are then mapped to the parametric domain of the *control hand surface* through a R*ay-Projection* method. By maintaining the distance between the mapped pairs, change of hand gesture can be efficiently transferred to control the deformation of the object.

1 Introduction

Most deformation systems developed so far make use of traditional input devices. They rely greatly on the skillfulness of the user to manipulate the system, and they pay little attention on the interface for interaction with the object. Most of the methods developed concentrate on deforming free-form surfaces by modifying either the control points or the sample points one by one. The operations become tedious when these methods are applied to complicated surfaces having a large number of control points. Besides, these deformation systems usually work with interfaces constrained by some intermediary devices such as keyboards, mice and joysticks. These devices with only one, two or three degrees of freedom are often ill-suited for the complicated modeling and deformation tasks such as artistic sculpting. In order to facilitate more intuitive interactions within the virtual world, it is desirable to be able to detect all the degrees of freedom of the human hand by sensing individual finger motions. Therefore, we believe that the glove device is the most natural interactive input tool for object modification or deformation. It consists of sensors for measuring the movement of each finger. Commercial models include VPL DataGlove, Virtex *Cyber*Glove^{TM}, Mattel's PowerGlove and Exos Dexterous Hand Master. These gloves have sensors that measure the angles of some or all finger joints. Some of these gloves also work with 3D trackers to locate the position of the user's hand. The DataGloves from VPL uses the hand as the user's manipulative extension into the virtual environment [1]. By pantomiming reaches and grabs, the user causes the hand to reach and grab objects in the virtual environment, and can move around in the virtual space by pointing in the desired direction and "flying" to the destination by recognizing finger postures. The major advantage of this model of interaction is naturalness. However, in this model, the glove functions as little more than a 3D joystick with several buttons where little of the dexterity and naturalness that characterize our hands have been explored to the modeling tasks.

In this paper, we propose a method for virtual sculpting based on the use of a glove device. The outline of the rest of the paper is as follows. Section 2 compares existing object modeling and deformation algorithms. Section

3 presents some of the techniques used in our method in detail. Section 4 presents some results from our prototype and discusses the performance of our method. Finally, section 5 concludes the paper and presents possible future works.

2 Backgrounds

Many techniques and systems have been developed for object modeling and deformation. Here, we discuss some of the important ones and compare their differences.

Solid modeling is mainly concerned with assembling primitive objects or scenes, not free-form models. An example is JDCAD [2]. This is a system built on the MR Toolkit [3]. It introduces interaction techniques based on the use of a hand-held bat, which is a six-degree-of-freedom 3D magnetic tracking device, and a keyboard. Unlike the work described in this paper, the system is used for the creation of mechanical components by assembling primitive volumes using the technique of Constructive Solid Geometry (CSG). Kurmann and Engeli in [4] proposed a spatial approach for interactive modeling to support architectural design in a VR environment. By using the mouse or other 3D input devices, the user may interact with the system in 3D. This enables the user to formulate design ideas in 3D space. Similar to JDCAD, this system allows assembling of predefined building elements or furniture for architectural design. It also introduces the concept of positive (solid) and negative (space) volumes to spatial modeling for the architects to construct design ideas. This method is useful for outlining vague ideas during the conceptual design phase where conceptual decisions are made and major constraints are established. However, it is not suitable for artistic free-form deformation.

Free-Form Deformation algorithms, on the other hand, change the geometric attributes of an object flexibly under some restrictions related to the properties of the object (e.g. continuity). The most well known method for object deformation is the **Free-Form Deformation (FFD)**. It was first proposed by Sederberg and Parry [5]. Since then, there has been a lot of enhancement works from the original FFD [6,7]. Basically, all of them deform an object by deforming the space around it. The object is first embedded in or mapped to a 3D solid lattice defined by some parametric function. Deformation of the object can then be achieved by deforming the control points of the 3D lattice. Free-form deformation is a powerful modeling technique that can deform surface primitives of any types or degrees. However, deforming an object through manipulating control points is not intuitive to use.

Direct Manipulation of FFDs [8] allows the user to move the sample points on the object model and automatically computes the necessary alteration to the control points of the FFDs. However, this technique involves the computation of least squares, which are computationally very expensive. In addition, the problem could be very complex when more than one vertex point is to be moved at the same time, and the solution may not be unique. Similar to other FFDs, this technique only tackles the modification of control points one by one. The FFD technique has been implemented with an elastic input device which has the same shape as the control volume (a cube) and provides tactile feedback [9]. The 3D model embedded can be deformed by deforming the cubic input device. Although the cubic input device provides a better interface, the nature of the underlying algorithm still hinders the naturalness of the deformation interaction.

Kameyama in [10] discussed a rubber pad with a tactile sensor sheet, which can detect user's shape deforming actions. The system creates grid surface data and provides a method for converting the input shape data into a solid modeling format. The limitations of the method are that the initial shape of the surface to be deformed has to be the same as that of the tactile sensor sheet, i.e. flat, and only vertical deformation of the surface grid is allowed. These limitations imply that the system has a very limited scope of applications and is not suitable for highly complex modeling tasks like sculpting.

Galyean and Hughes [11] proposed a volumetric approach for virtual sculpting with a 3D tracker. They described it as a 3D painting system in which the object is represented by voxel data. Sculpting is induced by modifying the values in the voxel array, similar to the paint brush function of a "paint" program, by moving the 3D tracker through space. The resultant voxel data is converted to a polygonal surface using a marching-cube algorithm. The models created are said to be free-form and may have complex topology. This approach is useful for controlling the shape by modifying a point or a small part of an object, but not suitable for global deformations. Li et al. [12] also described a prototype system for creating and editing NURBS surfaces. Instead of developing a deformation system, their objective was to provide an efficient technique for rendering deformable NURBS surfaces. Recently, they have extended their work for the rendering of any deformable parametric free-form surfaces [13].

3 The Virtual Sculpting Method

The main objective of our research is to develop a ***Virtual 3D Sculpting*** system to facilitate object deformation in a virtual environment with natural and real time interaction similar to sculpting in the real world. To modify the shape of an object, the users may prefer to simply flex the hand and the fingers. Therefore, the glove device that captures the hand gesture by tracking finger bend angles is believed to be the most natural and the most expressive input device to facilitate the task. Here, we present a method for virtual sculpting in 3D space based on the use of the *Cyber*Glove^{TM} for direct object/surface modeling or deformation. The main idea of the algorithm is to make use of the data collected from the *Cyber*Glove^{TM} to create a parametric ***control hand surface***, which is basically an O*pen-Uniform bicubic B-Spline tensor product surface*. An object to be deformed is mapped to this *control hand surface* by a novel ***Ray-Projection*** method. To improve the performance of this mapping process, a ***Two-Pass Projection*** technique is developed. By maintaining the mapping relationship between the *control hand surface* and the object, the change in the user's hand gesture can be effectively passed to control the deformation of the object. In the following subsections, we present the method in detail.

3.1 Detection and Deduction of Finger Joints Positions

In our implementation, we have adopted the *Cyber*Glove^{TM} from Virtual Technologies for hand gesture input and the FASTRAK system from Polhemus for hand position and orientation input. The *Cyber*Glove^{TM} hand model could be considered as an articulated rigid body system composed of rigid bone segments connected by joints. A human hand consists of 17 active joints and has 29 degrees of freedom: 23 degrees of freedom in the hand joints above the wrist, and 6 degrees of freedom in the free motion of the palm.

To identify the data points that define the hand gesture, we index the data points as $J_{a,b}$ where $a = 0,...,6$ and $b = 0,...,6$. The inner data points $J_{a,b}$ where $a = 1,...,5$ and $b = 1,...,5$ represent the main gesture of the hand. The first dimension refers to the five fingers, i.e., thumb, index, middle, ring and pinkie. The second dimension refers to the first, second and third data points of each finger starting from the fingertip. Due to the limitations of the *Cyber*Glove, only 5×3 finger joint positions can be determined directly from the glove device; other joint positions are deduced from the measured joint positions.

The software library of the *Cyber*Glove^{TM} provides the transformation matrices of the 5×3 articulated rigid segments comprising the virtual hand model [14]. These matrices hold the transformations to successive joint origins. Hence, the matrix for the coordinate system of the next joint can be obtained by applying the specific transformation matrix to the current joint point. From these transformation matrices, we can determine the 5×3 finger joint points.

The *Cyber*Glove^{TM} we use in this work has only 18 sensors and it does not explicitly measure the positions of the joints nearest to the fingertips. We predict these data points by taking into account the coupling existing between finger joint angles θ_5 and θ_4 shown in Figure 3-1(b). Experimental measurements show that the general coupling formula is of the form [15]:

$$\theta_5 = a_0 - a_1\theta_4 + a_2\theta_4^2$$

where θ_5 is the flexion angle of the distal joint of any of the fingers, except for the thumb and the parameters $a_0, a_1,$ and a_2 depend on the hand characteristics of individual users.

The other 5 data points defining the lower palm of the hand are deduced from the wrist and the joints nearest to the wrist. By using a Polhemus tracker, the wrist position and the hand orientation can be obtained. The first data point position of each finger is then set to the position two-third from the wrist to the joint nearest to it as shown in Figure 3-1(a).

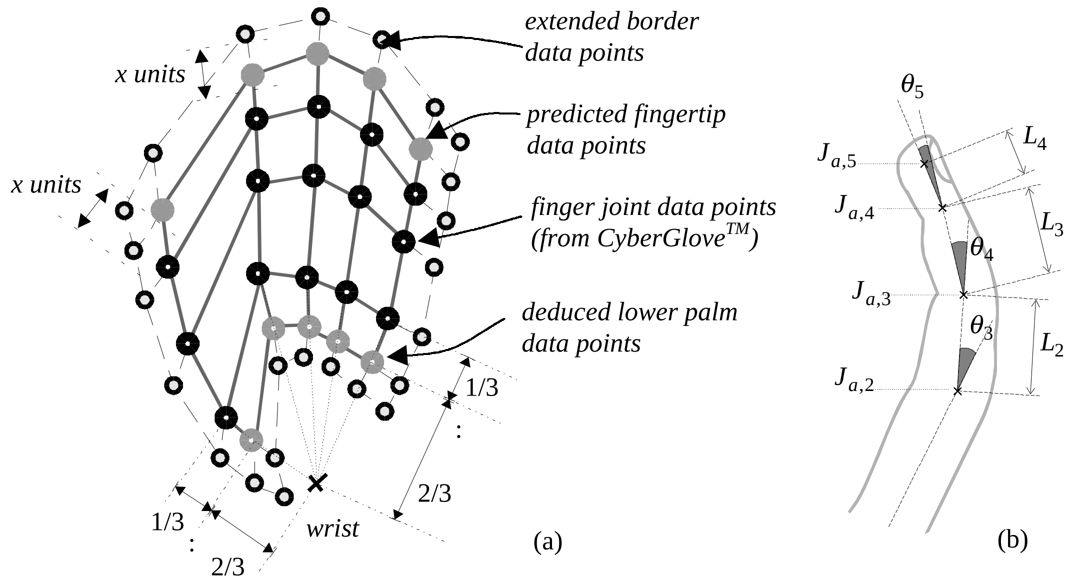
extended border data points
predicted fingertip data points
finger joint data points (from CyberGlove^{TM})
deduced lower palm data points
x units
x units
1/3
:
2/3
1/3
:
2/3
wrist
(a)
θ_5
$J_{a,5}$
L_4
$J_{a,4}$
L_3
θ_4
$J_{a,3}$
θ_3
L_2
$J_{a,2}$
(b)

Figure 3-1 (a) Finger joint points and, and (b) the parameters of a finger.

Now, we have 5×5 data points representing the feature gesture of the hand. We then extend the data points to 7×7 simply by extending the last segments in each of the dimensions on the edge by *x* unit(s). This border part is used to model the elastic property of the object. The more elastic the object is, the smaller is the parameter *x* and hence the narrower is the border. This can effectively preserve the continuity of the object when local modification is applied to it. We will explain this in detail in section 3.4.

3.2 Construction of the Control Hand Surface

To construct the parametric *control hand surface*, we have employed a technique called **interpolating method** [16]. This is to fit an *Open-Uniform B-Spline tensor product surface* $H(u_{a,b}, v_{a,b})$ by:

- Allocating knot vectors $U=\{0,0,0,0,\bar{u}_1,\bar{u}_2,\bar{u}_3,\bar{u}_4,\bar{u}_5,1,1,1,1\}$ and $V=\{0,0,0,0,\bar{v}_1,\bar{v}_2,\bar{v}_3,\bar{v}_4,\bar{v}_5,1,1,1,1\}$ in the two dimensions, and
- Resolving 9×9 unknown control points $P_{i,j}$, where $i=0,...,8$ and $j=0,...,8$,

such that the surface is able to interpolate all 7×7 data points, $J_{a,b}$, at some parametric coordinate $(u_{a,b}, v_{a,b})$, where $0\le u_{a,b}, v_{a,b}\le1$. Therefore,

$$H(u_{a,b}, v_{a,b}) = J_{a,b} = \sum_{i=0}^{m}\sum_{j=0}^{n} Nu_i^3(u_{a,b})Nv_j^3(v_{a,b})P_{i,j} \qquad (3\text{-}1)$$

where $Nu_i^3(u)$ and $Nv_j^3(v)$ are the cubic basis functions at particular parameter values u in *U*-dimension and v in *V*-dimension, respectively. $P_{i,j}$ are the control points of the hand surface that fulfil the condition of interpolating $J_{a,b}$. Equation (3-1) can be rewritten in the matrix form as:

$$\begin{bmatrix} J_{0,0} & \cdots & J_{0,6} \\ \vdots & \ddots & \vdots \\ J_{6,0} & \cdots & J_{6,6} \end{bmatrix} = \begin{bmatrix} Nu_0^3(\bar{u}_0) & \cdots & Nu_8^3(\bar{u}_0) \\ \vdots & \ddots & \vdots \\ Nu_0^3(\bar{u}_6) & \cdots & Nu_8^3(\bar{u}_6) \end{bmatrix} \cdot \begin{bmatrix} P_{0,0} & \cdots & P_{0,8} \\ \vdots & \ddots & \vdots \\ P_{8,0} & \cdots & P_{8,8} \end{bmatrix} \cdot \begin{bmatrix} Nv_0^3(\bar{v}_0) & \cdots & Nv_0^3(\bar{v}_6) \\ \vdots & \ddots & \vdots \\ Nv_8^3(\bar{v}_0) & \cdots & Nv_8^3(\bar{v}_6) \end{bmatrix} \qquad (3\text{-}2)$$

To improve the efficiency of the calculation, we do not determine the inverse of the matrices implicitly. Instead, we simplify the problem by considering the tensor product nature of the *control hand surface*. We first introduce a set of intermediate control points, $E_{a,b}$ where $a=0,...,6$ and $b=0,...,8$, such that:

$$\begin{bmatrix} E_{0,0} & \cdots & E_{0,8} \\ \vdots & \ddots & \vdots \\ E_{6,0} & \cdots & E_{6,8} \end{bmatrix} = \begin{bmatrix} Nu_0^3(\bar{u}_0) & \cdots & Nu_8^3(\bar{u}_0) \\ \vdots & \ddots & \vdots \\ Nu_0^3(\bar{u}_6) & \cdots & Nu_8^3(\bar{u}_6) \end{bmatrix} \cdot \begin{bmatrix} P_{0,0} & \cdots & P_{0,8} \\ \vdots & \ddots & \vdots \\ P_{8,0} & \cdots & P_{8,8} \end{bmatrix} \qquad (3\text{-}3)$$

Then, Equation (3-2) becomes:

$$\begin{bmatrix} J_{0,0} & \cdots & J_{0,6} \\ \vdots & \ddots & \vdots \\ J_{6,0} & \cdots & J_{6,6} \end{bmatrix} = \begin{bmatrix} E_{0,0} & \cdots & E_{0,8} \\ \vdots & \ddots & \vdots \\ E_{6,0} & \cdots & E_{6,8} \end{bmatrix} \cdot \begin{bmatrix} Nv_0^3(\bar{v}_0) & \cdots & Nv_0^3(\bar{v}_6) \\ \vdots & \ddots & \vdots \\ Nv_8^3(\bar{v}_0) & \cdots & Nv_8^3(\bar{v}_6) \end{bmatrix} \qquad (3\text{-}4)$$

The 2D interpolation problem is now reduced to two sets of 1D interpolation problems. Each *r*th row of Equation (3-4) is a 1D interpolation problem of an isoparametric curve $C_r^v(v)$ with knot vector

$\{0,0,0,0,\bar{v}_1,\bar{v}_2,\bar{v}_3,\bar{v}_4,\bar{v}_5,1,1,1,1\}$ such that it interpolates the data points as $C_r^v(\bar{v}_i) = J_{r,i}$. After solving the intermediate control points $E_{a,b}$ in Equation (3-4), $P_{i,j}$ in Equation (3-3) can be solved in a similar way. Hence, by considering the tensor product nature of the *control hand surface*, the 2D interpolation problem in Equation (3-2), which represents 7×7 linear equations, is now broken down into $7 + 7$ 1D curve interpolation problems.

By carefully selecting the knot values $\bar{u}_a$ and $\bar{v}_b$ that define the surface, the system of equations can be further simplified. Recall the local support properties of the B-Spline surface that for each data points $J_{i,j}$ at the interior knots defining the *control hand surface,* there are only three nonzero cubic basis functions as shown in Figure 3-2.

$$S(\bar{u}_i) = N_i^3(\bar{u}_i)E_i + N_{i+1}^3(\bar{u}_i)E_{i+1} + N_{i+2}^3(\bar{u}_i)E_{i+2} \tag{3-5}$$

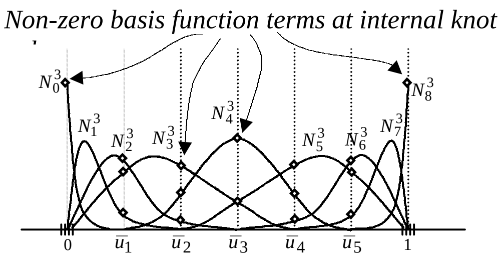

Figure 3-2 Basis functions of cubic B-Spline.

We approximate the knot values at each data point by the chord length ratios between successive data points [17]. These ratios are averaged row by row and column by column accordingly in the two dimensions. The interior knots of the knot vectors are set to coincide with the averaged chord length ratio of the data points. As a result, the matrix of the B-Spline basis functions can effectively be reduced to a tridiagonal matrix and the system of equations can be resolved efficiently by numerical methods [18].

3.3 Mapping of Object Vertices to the Control Hand Surface

The mapping method we developed is referred to as *Ray-Projection*. In this method, we consider the object to be deformed as embedded in the extended 2D parametric space of the *control hand surface*. To establish the mapping, rays are projected from a point called the center of projection, P_c, through each of the geometric attributes (say object vertices), V_a, of the virtual object to be deformed onto the *control hand surface* as shown in Figure 3-3. With this projection, V_a in the Euclidean 3D space is mapped to (u_a, v_a) in the 2D parametric space of the *control hand surface* as shown in Figure 3-4. The mapping relationship remains unchanged during the deformation process.

However, computing the intersection of a ray with a parametric surface is computationally very expensive. In order to speed up these intersection calculations, we propose to approximate the parametric surface by a polygonal hand model, which we refer to as the *3D triangulated hand model*. This model is constructed by triangulating the data points $J_{i,j}$.

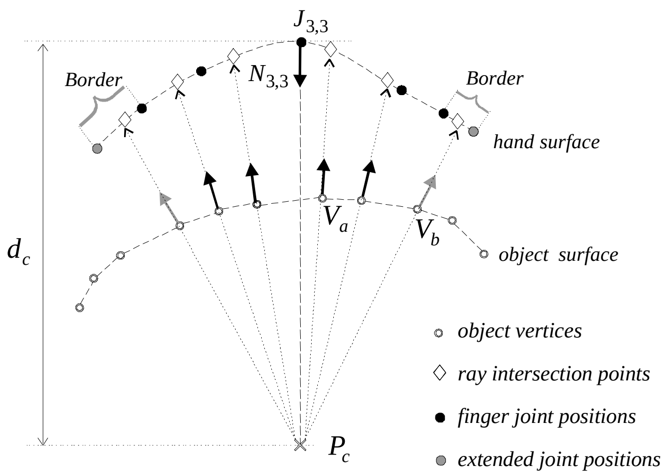

Figure 3-3 Ray-Projection.

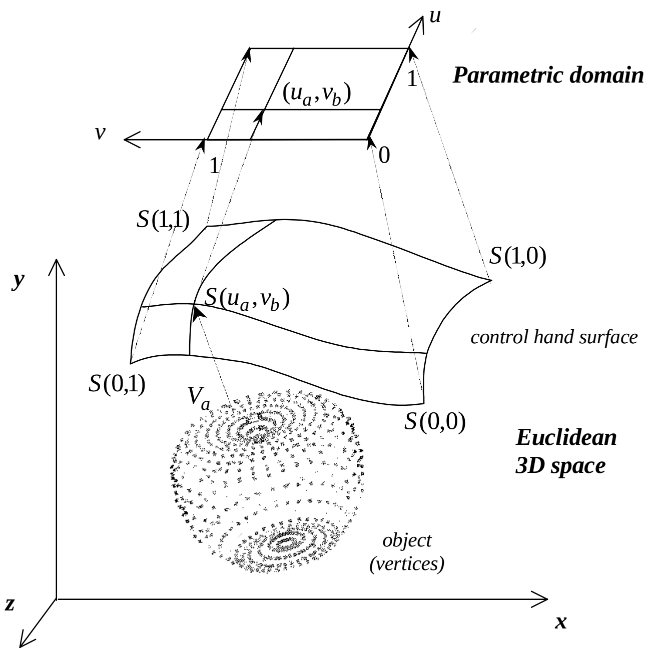

Figure 3-4 Mapping of the spatial domain of the object surface to the parametric domain of the control hand surface.

Recall that the hand is modeled by a tensor product surface interpolating through these data points. Therefore, we tessellate the surface by joining neighboring data points in *U* and *V* dimensions as shown in Figure 3-5. This results in a tetrahedron mesh with 6×6 tetrahedrons. Since the four vertices of a tetrahedron may not be on a single plane, we further break each of the tetrahedrons into two triangles. As a result, a *3D triangulated hand model* with $6 \times 6 \times 2$ triangles, $tri_{i,j,\#}$, is formed and each triangle is defined on a plane $T_{i,j,\#}$,

$$T_{i,j,\#} : (P \cdot N_{i,j,\#}) + d_{i,j,\#} = 0 \qquad (3\text{-}6)$$

where $i = 0,...,5$, $j = 0,...,5$ and $\# = 0,1$. P is any point on the plane $T_{i,j,\#}$. $d_{i,j,\#}$ is the offset of the plane from the origin, and $N_{i,j,\#}$ is the normal of the plane.

To determine the intersection of a ray with the *3D triangulated hand model*, we first define a ray R_a projected from P_c through vertex V_a as:

$$R_a : P = V_a + k \cdot [V_a - P_c]_I \tag{3-7}$$

where P is a point on R_a at some scalar variable *k*. The *ray-intersection point* of R_a and $T_{i,j,\#}$ can be calculated as follows:

$$V_a^{proj} = V_a - \frac{d_{i,j,\#} + N_{i,j,\#} \cdot V_a}{N_{i,j,\#} \cdot [V_a - P_c]_I} [V_a - P_c]_I \tag{3-8}$$

Since a projected ray intersecting with one of the $6 \times 6 \times 2$ planes does not necessarily mean that it intersects with the corresponding triangle in the *3D triangulated hand model*, clipping tests are necessary to determine if V_a^{proj} falls inside $tri_{i,j,\#}$ or not. Therefore, in the worst case, for an object consisting of *n* vertices, the mapping process may involve a maximum of $6 \times 6 \times 2 \times n$ projections, intersection tests and clipping operations. To minimize the number of projections and intersection tests, we separate the ray-intersection calculations into two passes. In pass 1, we determine which triangle of the *3D triangulated hand model* the ray intersects. In pass 2, the *ray-intersection point* is determined and the parametric coordinate is assigned accordingly.

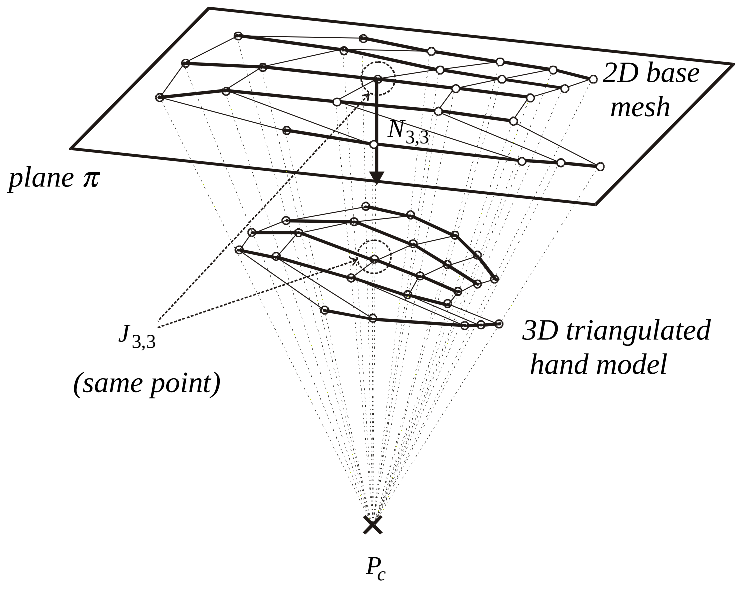

Figure 3-5 Two-Pass Projection.

Pass 1: Determination of the Intersection Triangle

To determine the triangle that a ray intersects at the *3D triangular hand model*, we first define a plane π through the center of the palm $J_{3,3}$ and having the same normal as that of the center of palm, $N_{3,3}$. By ray-projecting all 7×7 data points onto π, a corresponding *2D base mesh* on the plane is constructed as shown in Figure 3-5. Each object vertex V_a is then ray-projected to π and the triangle on the *2D base mesh* that the ray intersects is determined. This intersecting triangle on π indicates the intersection of the corresponding triangle on the *3D triangulated hand model*.

Pass 2: Determination of the Ray-Intersection Point

A second ray-projection is made to find out the *ray-intersection point* on the corresponding triangle of the *3D triangulated hand model* by solving k in Equation (3-7). Afterwards, the point is transformed to the 2D local coordinate defined by the corresponding plane $T_{i,j,\#}$ and the parametric coordinate (u,v) of the vertex is evaluated by the *barycentric coordinate* method [19].

3.4 Sculpting Transformation

We now go into the detail on how the object can be deformed. Figure 3-6 shows how the change of hand gesture can be passed to deform the object vertices. Consider at time 0 when an object is mapped to the *control hand surface*. Let the initial position of a vertex be V_a^0, the parametric coordinate that V_a^0 mapped to by using *ray-projection* be (u_a,v_a) and the corresponding *mapped ray-intersection point* on the *control hand surface* be $S^0(u_a,v_a)$. V_a^0 can be written as:

$$V_a^0 = S^0(u_a,v_a) + \left|V_a^0 - S^0(u_a,v_a)\right| \cdot \left[V_a^0 - S^0(u_a,v_a)\right]_I \qquad (3\text{-}8)$$

where $V_a^0 - S^0(u_a,v_a)$ is the vector from $S^0(u_a,v_a)$ to V_a^0. $\left[V_a^0 - S^0(u_a,v_a)\right]_I$ and $\left|V_a^0 - S^0(u_a,v_a)\right|$ are the unit vector and the magnitude, respectively.

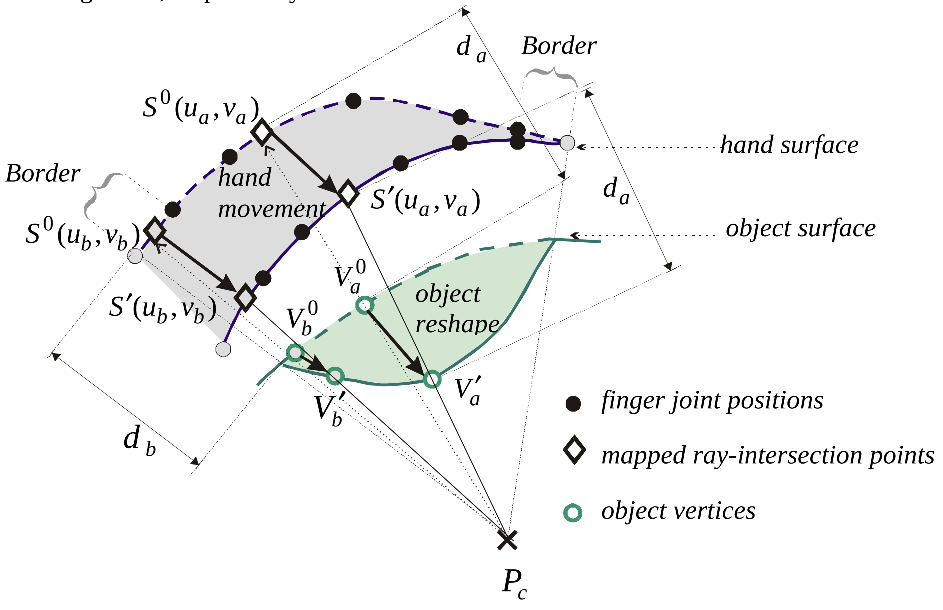

Figure 3-6 Object Vertex Modification.

By the above definition of *ray-projection,* the unit vector $\left[V_a^0 - S^0(u_a,v_a)\right]_I$ equals $\left[P_c - S^0(u_a,v_a)\right]_I$. Let $d_a^0 = \left|V_a^0 - S^0(u_a,v_a)\right|$. At time 0, the relationship of V_a^0 and $S^0(u_a,v_a)$ can be rewritten as:

$$V_a^0 = S^0(u_a,v_a) + d_a^0 \cdot \left[P_c - S^0(u_a,v_a)\right]_I \qquad (3\text{-}9)$$

Here, d_a^0 represents the distance between the *mapped ray-intersection point,* $S^0(u_a,v_a)$, and the object vertex, V_a^0, at the time that the mapping is established, i.e. time 0. We maintain this mapping by keeping d_a^0

unchanged throughout the deformation process.

After this relationship is established, deformation can be initiated by changing the hand gesture i.e. $S(u,v)$. Let us consider the situation at time *t*, that the change of the *control hand surface* leads to the change of the Euclidean 3D coordinate of $S^0(u_a,v_a)$ to $S^t(u_a,v_a)$. The *sculpting transformation* for the object vertex at time *t* can be evaluated accordingly as

$$V_a^t = S^t(u_a,v_a) + d_a^0 \cdot \left[P_c - S^t(u_a,v_a)\right]_I \tag{3-10}$$

Recall that the *control hand surface* is constructed with a border by extending *x* unit(s) from the edge of the finger joint data. This prevents undesirable sudden change in the object surface and maintains the continuity at the edge when local deformation is applied to the object. At the border, where $u \in [0,\bar{u}_1]$, $u \in [\bar{u}_5,1]$, $v \in [0,\bar{v}_1]$ or $v \in [\bar{v}_5,1]$, we introduce a weighting function which is a cubic Bernstein-Bézier basis function, $w(u,v)$, as shown in Figure 3-7. This weighting function is for preserving the continuity of the deformation along the border and the result is shown in Plate 1. The corresponding *sculpting transformation* at the border region is given by:

$$V_a^t = w(u_a,v_a)\cdot\left(S^t(u_a,v_a) + d_a^0 \cdot \left[P_c - S^t(u_a,v_a)\right]_I\right) + \left(1 - w(u_a,v_a)\right)\cdot V_a^0 \tag{3-11}$$

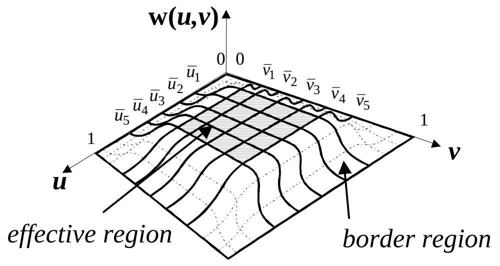

Figure 3-7 Illustration of the Smooth Weighting Function $w(u,v)$.

3.5 Control of Sculpting Region

With *ray-projection*, the sculpting region can be very flexible. We can change it by changing the location of the center of projection, P_c. Three possible deformation effects can be produced.

When P_c is set to infinity in the direction of the normal of the palm, all the projection rays may be considered as parallel to each other. This is referred to as *parallel projection* and is shown in Figure 3-8(a). The sculpting region will be the same size as the *control hand surface*. Under this circumstance, $d_a^0\left[P_c - S^t(u_a,v_a)\right]_I$ equals $V_a - S(u_a,v_a)$, and the *sculpting transformation* becomes

$$V_a^t = S^t(u_a,v_a) + (V_a - S(u_a,v_a)) \tag{3-12}$$

That is, $\Delta V_a = \Delta S(u_a,v_a)$. This implies that the change of the *control hand surface* applies directly to the vertices of the object and it provides a similar effect of touching and deforming the object.

Magnified Sculpting can be produced by setting the center of projection, P_c, to the back of the *control hand surface* as shown in Figure 3-8(a). This enlarges the size of the effective region and magnifies the magnitude of deformation that the hand passed to the object. Plate 2 demonstrates the *Magnified Sculpting* experiment.

If the center of projection, P_c, is set to in front of both the *control hand surface* and the object being deformed, *Reduced Sculpting* is resulted and this is illustrated in Figure 3-8(b). This will reduces the size of the effective region and the magnitude of the deformation. Experimental result for R*educed Sculpting* is demonstrated in Plate 3.

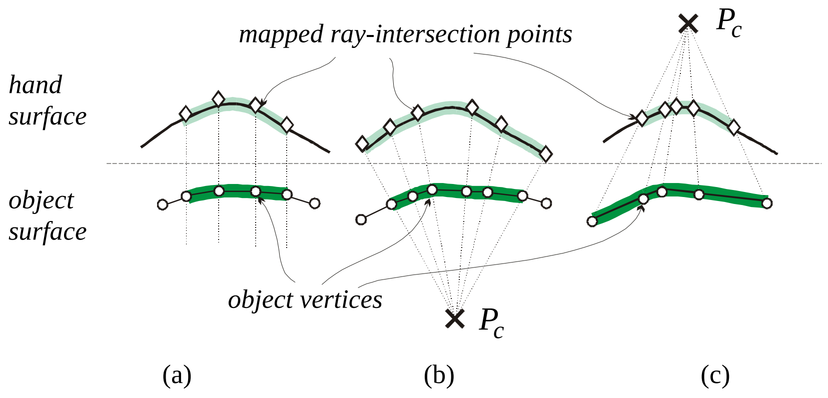

Figure 3-8 Sculpting Regions: (a) Parallel Projection, (b) Reduced Sculpting, and (c) Magnified Sculpting.

4 Results and Discussions

We have implemented the algorithm in C++ with OpenInventor and OpenGL. We tested it on a SGI Indigo^2 workstation with a 200MHz MIPS 4400 CPU and the Extreme graphics accelerator using 5 different models. Here we show and discuss some of the experiments.

Plate 2 shows the experiment of grasping a teapot model and Plate 3 demonstrates the deformation of a human face model. Each window shows the *control hand surface*, the object being deformed and a colored cluster in the middle. There are five colored articulated sticks representing the finger segments of the user’s hand. The red one is the thumb and the green one is the pinkie. The surface with colored points distributed across is the *control hand surface*. It is divided as tetrahedron mesh. The colored points on the *control hand surface* and the colored vertices on the object indicate the corresponding mapping between them. (For simplicity, only those regions that exert deformation effect on the object are shown.) This helps recognize the exact control of different regions of the *control hand surface* applied to the object. Here, the white points corresponding to the *border region* and other colored points reveal the deformation control of the *effective region* to the object. The cluster of colored points located between the *control hand surface* and the object is for easy reference only.

In Plate 2, the center of projection is set at the back of the hand which produces *magnifying* control. In Plate 3, the center of projection is set in front of both the hand and the object and thus *reducing* the deformation effect exerted on the object. The upper windows of both plates show the mapping of the *control hand surface* to the object before the deformation starts and the models are in their initial shape. The lower windows show the sculpting effects induced by changing the hand gesture. Using our prototype system, we experienced sculpting control on several models. Plate 4 shows some of the output models resulted from the experiments.

To evaluate the performance of our method, we measured the processing time for the two main stages during sculpting. Stage 1 concerns with the ***mapping of object vertices to the control hand surface***. Stage 2 is the ***object deformation process***. Some of the performance results are shown in Table 4-1. From the table, the performance of stage 1 is about 0.5ms per vertex and that for stage 2 is about 0.05ms per vertex. In addition, by employing the interpolation method described in section 3.2, the construction time for the *control hand surface* is reduced from approximately 1s down to below 0.01s. These results show that our method could provide a real time interactive experience for object deformation and we believe that the new method is efficient enough for the task of *Virtual 3D Sculpting*.

Model	No. of Vertices	--- Stage 1 ---		--- Stage 2 ---	
		Total Time	Average time per vertex	Total Time	Average time per vertex
Button	701	0.35s	0.50ms	0.035s	0.050ms
Teapot (high res.)	2081	1.00s	0.48ms	0.110s	0.055ms
Teapot (low res.)	529	0.26s	0.49ms	0.030s	0.055ms
Apple	867	0.45s	0.52ms	0.050s	0.050ms
Face	2278	0.86s	0.38ms	0.130s	0.055ms

Table 4-1 Performance of the Method.

5 Conclusion

We have presented the ***Virtual 3D Sculpting*** method for interactive 3D object deformation, based on the use of the glove device. The main idea of our method is to create a parametric *control hand surface* defined by an *Open-Uniform B-Spline tensor product surface*, interpolating through the data points that defined the user's hand. Each geometric attribute of the object in the Euclidean 3D space is mapped to the parametric domain of the *control hand surface* through a R*ay-Projection* method. By maintaining the distances of all the mapped pairs, change of hand gesture can be effectively passed to control the deformation of the object. In addition, the *ray-projection* method also allows the size of the effective deformation region to be changed simply by changing the position of the center of projection.

The major contributions of the new method can be summarized as follows:

- The new method is one of the few that make use of essentially all the degrees of freedom available in glove devices to control something other than an animated hand.
- The method provides a way for simultaneously controlling a set of feature points of the object.
- The method is efficient and can be applied to objects of any representations.

As a future work, we are currently developing an object modeling and editing system based on the method developed here. The system is designed to allow the creation of complex objects by stitching together multiple surface patches and the editing of surface properties such as colors and textures.

References

[1] D. Sturman and D. Zeltzer. *A Survey of Glove-based Input*. IEEE Computer Graphics and Applications, pp. 30-39, 1994.

[2] J. Liang and M. Green. *JDCAD: A Highly Interactive 3D Modeling System.* Computers & Graphics, **18**(4), pp. 499-506, 1994.

[3] C. Shaw, M. Green, J. Liang, and Y. Sun. *Decoupled Simulation in Virtual Reality with MR Toolkit*. ACM Transactions on Information Systems, **11**(3), pp. 135-142, May 1993.

[4] D. Kurmann and M. Engeli. *Modeling Virtual Space in Architecture*. Proceedings of ACM Symposium on Virtual Reality Software and Technology, pp. 77-82, July 1996.

[5] T. Sederberg and S. Parry. *Free-Form Deformation of Solid Geometric Models.* ACM Computer Graphics, **20**(4), pp. 151-160, August 1986.

[6] S. Coquillart. *Extended Free-Form Deformation: A Sculpting Tool for 3D Geometric Modeling*. ACM Computer Graphics, **24**(4), pp. 187-193, August 1990.

[7] H. J. Lamousin and W. N. Waggenspack. *NURBS-Based Free-Form Deformations.* IEEE Computer Graphics and Applications, pp. 59-65, 1994.

[8] W. Hsu, J. Hughes, and H. Kaufman. *Direct Manipulation of Free-Form Deformation*. ACM Computer Graphics, **26**(2), pp. 177-184, July 1992.

[9] T. Murakami and N. Nakejima. *Direct and Intuitive Input Device for 3-D shape Deformation*. ACM CHI'94, pp. 465-470, 1994.

[10] K. Kameyama. *Virtual Clay Modeling System*. ACM Symposium on Virtual Reality Software and Technology, pp. 197-200, Sept. 1997.

[11] T. Galyean and J. Hughes. *Sculpting: An interactive volumetric modeling technique*. ACM Computer Graphics, **25**(4), pp. 267-274, 1991.

[12] F. Li, R. Lau, and M. Green. *Interactive Rendering of Deformating NURBS Surfaces*. EUROGRAPHICS'97, 16(3), pp. 47-56, September 1997.

[13] F. Li, R. Lau. *Real-Time Rendering of Deformable Parametric Free-Form Surfaces*. ACM Symposium on Virtual Reality Software and Technology (to appear), December 1999.

[14] *Virtual*Hand^{TM} *Software Library Reference Manual.* Virtual Technologies.

[15] 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.

[16] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design – A Practical Guide*, 4th Edition. Academic Press, 1997.

[17] L. Piegl and W. Tiller. *The NURBS Book.* Springer-Verlag, 1995.

[18] G. Engeln-Mullges and F. Uhlig. *Numerical Algorithms with C.* Springer, pp. 89-92, 1996.

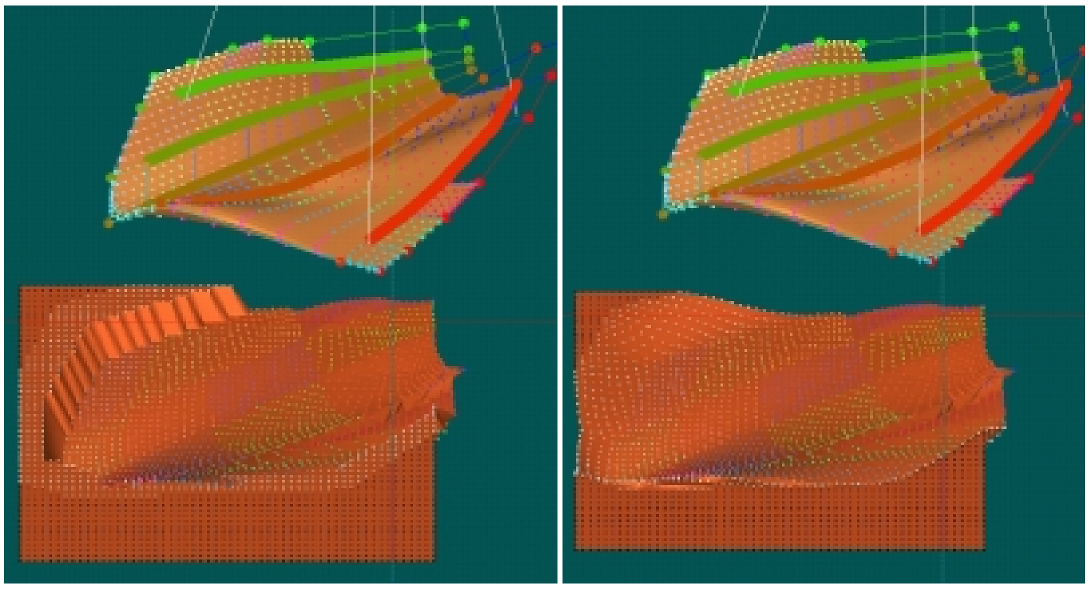

Plate 1 Effects of Applying Different Weightings at the Border.

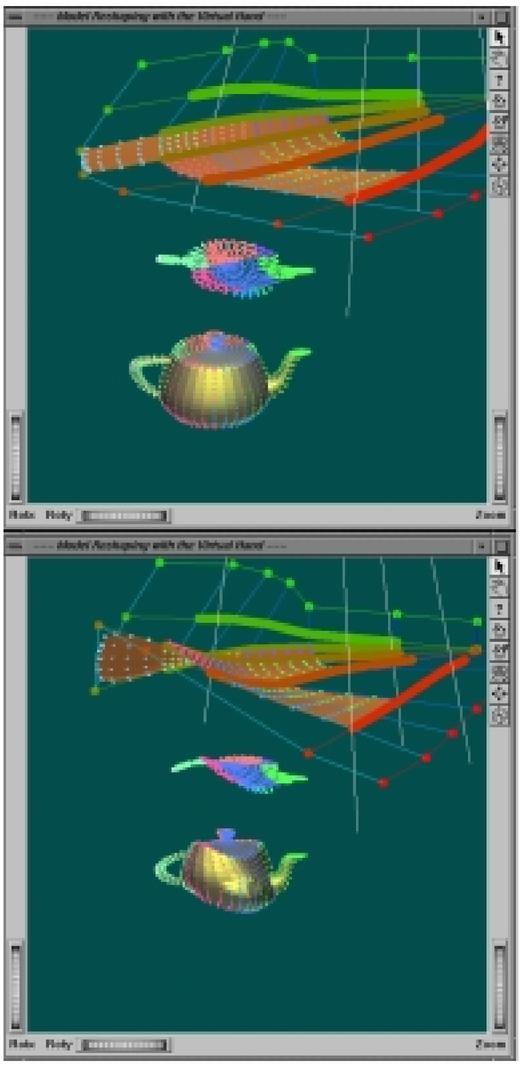

Plate 2 Magnified Sculpting.

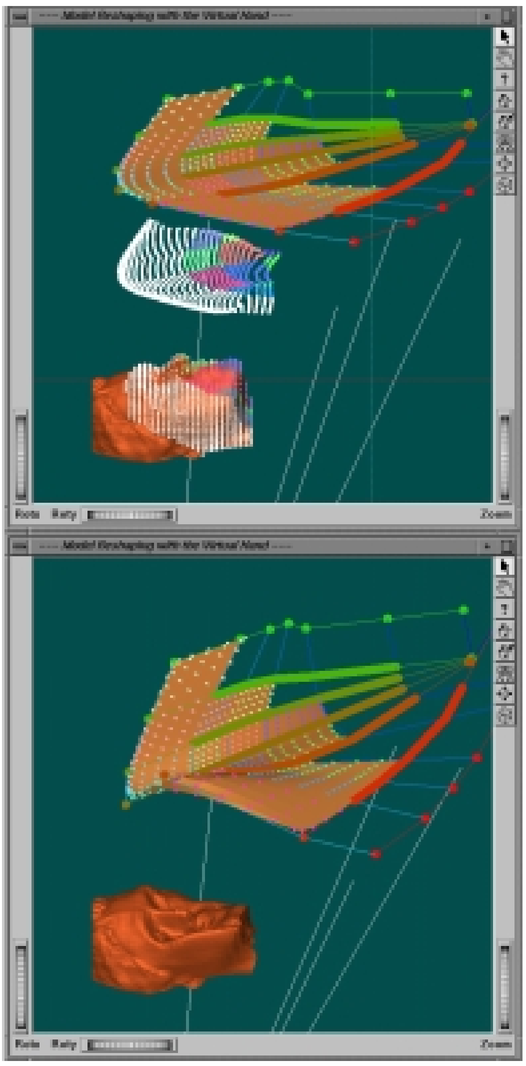

Plate 3 Reduced Sculpting.

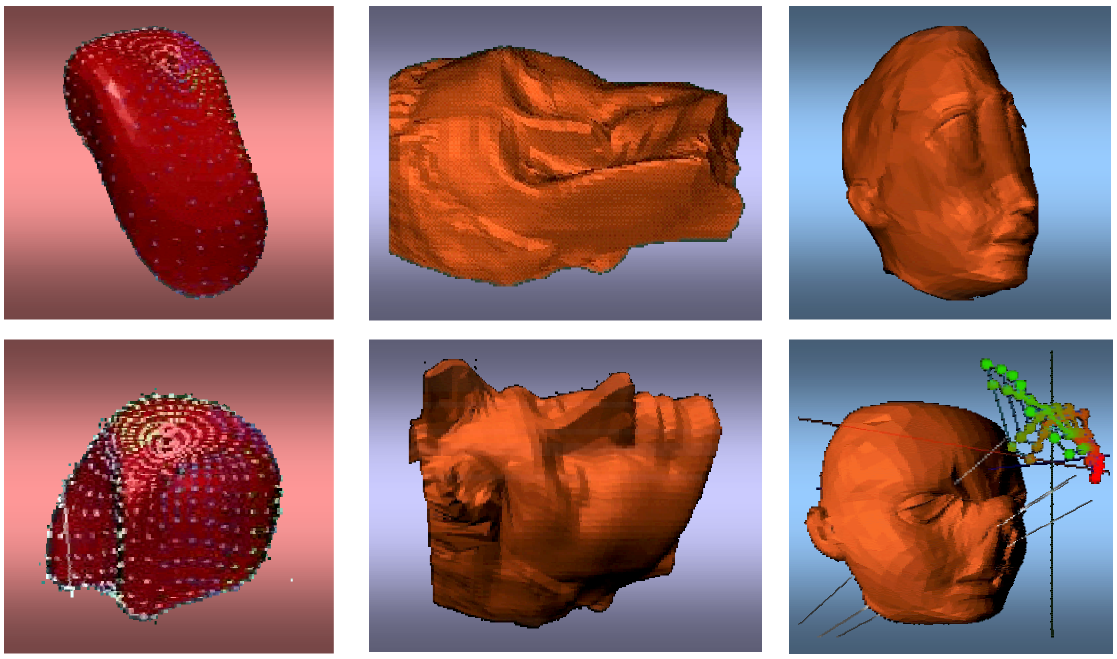

Plate 4 Some Outputs.