Accepted for the *2004 International Conference on Cyberworlds, Nov. 18-20, 2004*
(Proceedings is to be published by the IEEE press.)

3D Freeform Design: Interactive Shape Deformations by the Use of CyberGlove

Masatake Sato, Vladimir Savchenko, Ryutarou Ohbuchi
University of Yamanashi, 4-4-37 Takeda Kofu-shi, Yamanashi, 400-8510, Japan
E-mails: g04mk012@ccn.yamanashi.ac.jp, vsavchen@k.hosei.ac.jp, ohbuchi@acm.org

Abstract

This paper presents an approach to realize a tool for interactive and intuitive deformation of 3D shapes. The approach combines a fast algorithm for 3D free form deformation that employs Compactly Supported Radial Basis Functions (CSRBFs) with a high degrees-of-freedom input device of hand joint angles, namely, the CyberGlove. While shape deformation algorithm by using CSRBFs allows for fast and reasonable shape deformations, it is not a simple task to position the CSRBFs so that desired shape deformations can be created. In our approach, the 10 CSRBFs are attached to the tip of the fingers of the CyberGlove worn on both hands of the users. This allows the user an intuitive manipulation of 3D shapes in a manner somewhat similar to clay modeling. While the shape deformation algorithm does not produce physically accurate deformation, the deformation is fast, plausible and quite useful. For example, using a 32k polygon mesh, a deformation involving all the 15k vertices can be produced at about 30 frames-per-second.

1. Introduction

Creating and editing 3D shapes is not easy, whether the tool is conventional and real, as in clay modeling and wood sculpting, or computerized and virtual, as in computer aided geometric modeling (CAGD) for industrial or entertainment purposes. While the instance of capturing 3D shapes of real objects has increased, for example by a laser range scanner, captured models often require various manipulations and editing.

A recently popular approach is to combine a virtual modeling tool, e.g., a CAGD software, with so-called "virtual reality (VR)" user-interface devices. These VR user interfaces devices try to exploit multimodal, high bandwidth, high degrees-of-freedom (DOF) human input-output capabilities. For instance, He et al [1] employed DataGlove, a device that captures hand finger joint angles and hand positions, as an input device for a real-time manipulation of 3D shape models. They utilized neural-network based hand gesture recognizer to enter some of the commands. A problem with this approach is that these gestures are not "natural" to human beings and requires some effort to learn.

An approach to tackle this problem is to employ various tangible tools that conceptually matches to intended operations, as employed by Schkolne et al. [2]. They employed DataGlove-like input device to specify surface shape. They also used tangible (physically real) devices that are directly coupled to specific operation(s). A kitchen tong is used to grab and move objects, and a pair of tongs is used to grab and stretch/shrink objects. A ball is used as an eraser, and a custom made "magnet device" that attaches to a palm is used to specify deformation of objects in combination with the bending of the fingers.

While these tangible devices do provide sensation of touching and/or holding, they do not provide force feedback that accompanies real world manipulation of physical objects, e.g., chiseling or bending. Haptic displays are display devices that produce various force based on computer output. An example is the Phantom [3], a single-point-of-contact force feedback device that employs a lightweight mechanical arm. FreeForm Concept [4] is a good example of 3D shape modeling software that effectively uses the Phantom devices. These systems run a physical (to a certain extent) simulation of force that would be created if a surface is touched or poked by the tip of the virtual devices. A 2D painting system by Baxter et al [5] employed the Phantom to create force feedback that would have resulted if a paintbrush with oil paint is drawn on a canvas. However, the Phantom is a single Point-Of-Contact (POC) input/output device, which makes it difficult to feel and specify a line or a surface at once. In fact, it is inherently difficult to create a force feedback device having a multiple POC and a high DOF.

Another approach to 3D shape deformation is a model-based one that aims for realistic deformations. In the example of the facial animation, one could employ anatomically and physiologically correct models of skin, fat, muscles and skull. If the model is accurate enough, the model could only generate reasonable deformations (given enough computational power). Classic examples of this approach can be found in [6, 7, 8]. The mass and spring (and damper) model and the finite element methods

A part of this work has been done in Hosei University by Vladimir Savchenko.

[9] are the most used method to simulate physics into these model-based shape deformation/animation systems. One of the primary issues in this approach is finding reasonable parameters for the model and the model's components. For example it is quite difficult to find elasticity and the dumping coefficients of a muscle, for they vary from person to person, and they change due to fatigue and other factors.

While realistic models have their own advantages, a physically inaccurate model of shapes and deformation could prove quite important. In some applications such as a puppet animation the artist deals with a "rubber" puppet to produce attractive deformations. It is important to notice that, nevertheless, such deformations pose some sort of charm. In some sense there is "the aesthetics of the imperfect," a set of rules for producing deformations is not defined. The art of imperfect triumphed in Japanese ceramics. Another case for the non-realistic deformation model is time (or computational power) critical applications, a prominent example of which being a real-time game on a game console. In these time-critical applications, one cannot be spending too much CPU cycles on simulating shape deformations.

This paper proposes and discusses the method and preliminary results of combining a physically inaccurate, yet reasonable 3D surface shape deformation algorithm with a Multiple POC (MPOC), high-DOF VR input device. Our shape deformation algorithm does not employ physically-based simulation and is computationally quite efficient. In our algorithm, the applications space mapping technique based on Compactly Supported Radial Basis Functions (CSRBFs) generates deformations. The 10 fingertips of the CyberGlove are the centers of the 10 CSRBFs that bring shape deformations. To specify shape deformation, we employed a MPOC input device CyberGlove [12] so that a complex deformation of 3D shapes can be specified at once by using the 10 POC. Contributions of this paper can be summarized as follows;

- Intuitive and interactive control of 3D surface shape deformation thorough the multiple points-of-contact manipulation by using the CyberGlove hand finger joint angle measurement device.
- A very fast shape deformation by using a scattered data interpolation algorithm that employs locally supported radial basis function, or CSRBF. A CSRBF is attached to each fingertip to induce "force field" that determines the magnitude of deformation.

To realize the first contribution, the data from GyberGlove should be used effectively. Thus, we propose two types of methods to generate parameter from the CyberGlove.

In the next section, we describe the overview of the system as well as algorithms for deformation and shape manipulation by using CyberGlove. We present examples of deformation and some timing results in Section 3. We conclude the paper in Section 4.

2. The algorithm

2.1. Overview

By applying motion of human hands to fast shape deformation method using CSRBF, we could accomplish the intuitive manipulation of shape deformation in real time. When an operator moves his/her fingertips, the 3D shape near the fingertips changes along the trace of the fingertips. Our system gets the posture data from human hands via CyberGlove, our developed VR Hand module (see section2.2) converts the data to proper parameters in order to generate shape deformation. To generate plausible shape deformation, getting vectors from real hands' postures is important. We designed two types of deformation algorithms called Anchored and Unanchored deformation (see Section 2.3.1 and 2.3.2 for more detail).

In the Anchored deformation, start points of control vectors are acquired only once from the hands' posture at the beginning of deformation. Thus the start points are generally fixed. On the other hand, the end points are acquired every time when the hands posture is changed. In the Unanchored deformation, both the start and end points of control vectors are updated every time when the hand's posture is changed. Both modules are based on CSRBF deformation algorithm which consists of following steps (see section 2.4 for detail):

- Sort the start points of control vectors.
- Find displacement vector for each point. Assign starting and ending positions to each point to find displacement vector.
- Construct a SLAE per RBF.
- Solve the SLAE for the blending coefficients α_i.
- Evaluate the displacement.

2.2. Computing positions of fingers

This section describes the structure of CyberGlove and developed VR Hand module. VR Hand module gets the data from CyberGlove, and outputs the hand posture as numerical data.

2.2.1. CyberGlove

The CyberGlove is a hand finger-joint angle measurement device similar to DataGlove, and is worn like a glove over hands. This CyberGlove provides up to 22 high-accuracy joint-angle measurements and uses proprietary resistive bend-sensing technology to accurately transform hand and finger motions into real-time digital joint-angle data [12].

We used the Virtual Hand Suite 2000 from the Virtual Technologies Inc. for the post processing and calibration of the joint angle data [13, 14].

The Virtual Hand Suite 2000 includes "Virtual Hand Toolkit" (VHT) which provides C++ library for software development. The library includes predefined graphical hands for VR simulation, methods for acquiring device data, and network support methods for transmitting the device data from a host computer to another computer. We used VHT for acquiring joint-angle-data.

2.2.2. VR Hand

The VHT contains a method to generate (and visualize) posture data using the angle data from CyberGlove. But the method is developed using OpenGL. From a viewpoint of compatibility, it is not good to use a module which is depending on particular graphics library. To avoid the problem, we developed our own VR hand, which does not depend on a particular graphics library and consists of three modules: one for description of a hand skeleton model, another one is a method to obtain the angle data from CyberGlove using VHT, and the third one is a method to output the calculated posture data of the skeleton. Our VR hand outputs the numerical posture data. The data can be visualized using graphics libraries such as OpenGL (see Figure 1(a)). In the developed software, we used VTK for visualizing the posture output.

The developed VR hand includes a skeleton model of a hand, which is a linked list structure and consists of three types of nodes; Parent, PalmJoint, and Joint. Constructed VR hand structure is shown in Figure 1(b). Note that the fingertips' next pointers are NULL.

Angle data from the CyberGlove is applied to the VR hand and its new posture is calculated. By calculating the posture from the wrist-side joint to fingertip-side joint, calculation cost can be minimized.

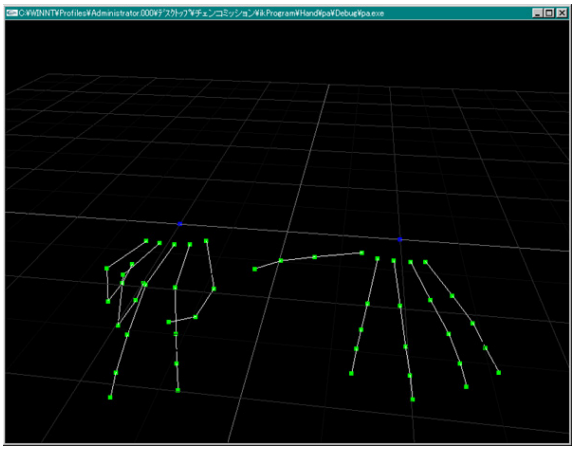

Figure 1(a). Visualized skeleton hands by using OpenGL

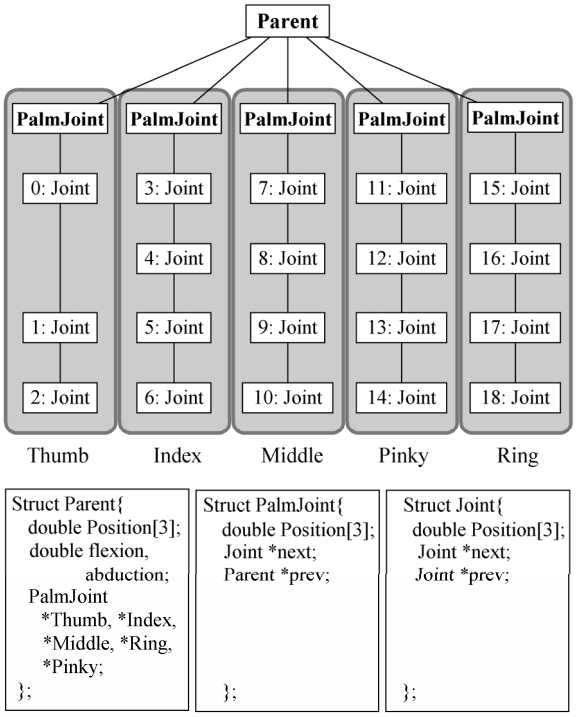

Figure 1(b). Structure of skeleton hand model.

The posture is calculated from the default shape every time when the new angles are acquired from the CyberGlove and the skeleton model never overwrites its default shape. It has to be done because of consideration for an error; the error causes dissonance between the posture of operator's hand and that of VR hand.

2.3. Two types of deformation

For plausible shape deformation, we present two types of deformation modules that use vector of fingertip's motion effectively. We call these methods Anchored and Unanchored deformation. As described in section 2.1, these modules are based on CSRBF deformation algorithm (see section 2.2 for more detail), but differ in getting control vectors from hands' posture data.

2.3.1. Anchored deformation

We propose an algorithm which uses only end points of the control vectors generated by VR Hand. The start points $\vec{s}_i$ are stationary and only the end (destination) points $\vec{d}_i$ are updated. A shape is deformed from the original state according to the updated vector. The start points are updated only at the beginning of the deformation process and the original shape data is updated only when the proper control vectors are determined. Stationary start points provide three

advantages. First of all, it allows to skip sorting of scattered data, secondly, we are able to recycle left-hand side of a SLAE, and the third one is to save time for updating the shape data. The Anchored deformation is almost the same as CSRBF deformation algorithm with the exception of that the algorithm includes additional steps:

- Define start points of control vectors.
- Sort the start points of control vectors.
- Update end points of control vectors.
- Construct the right side of a SLAE per RBF (if it is the first step, entire SLAE are constructed).
- Solve the SLAE for the blending coefficients α_i
- Evaluate the displacement.
- Save current shape data and return to the first step of CSRBF deformation, or go to the third step.

In case of using CyberGlove, 10 fingertips' coordinates from VR Hand define the start points at first. Only the end points are constantly update during the Anchored deformation.

2.3.2. Unanchored deformation

The Unanchored deformation, which is based on CSRBF deformation algorithm, realizes the complicated shape deformations along the curves of trajectories of the fingertips. While the Anchored deformation can recreate similar deformed shapes easily, it is difficult to create complicated shape deformations due to the straight-line control vectors.

Straight-line segments approximate the curves of the trajectories (the accuracy of the approximation is depending on how many times can CyberGlove acquire angle data in a unit time). The new fingertips' coordinates updates the start points while the start points are replaced by the previous end pints. Along with the update of the start and end points, CSRBF deformation evaluates the displacements and update the current shape data. The Unanchored deformation consists of following steps:

- Update start and end points of control vectors.
- Sort the start points of control vectors.
- Construct a SLAE per RBF.
- Solve the SLAE for the blending coefficients α_i.
- Evaluate the displacement.
- Update the shape data and go to the first step.

In case of using CyberGlove, the newest 10 fingertips' coordinates from VR Hand update the end points and the previous end points update the start points.

2.4. Shape deformation by using CSRBF

We employed the shape deformation algorithm proposed by Kojekine et al [10]. The algorithm warps the space to be deformed. The extent of the warp and the magnitude of the deformation in computed by the scattered data interpolation algorithm based on CSRBF.

The main idea of shape deformation by using CSRBF is to warp 3D space around start points of vectors toward the direction determined by the direction vectors. The extent of the warp space is defined by support of the CSRBF. The displacement of a point within the radius is weighted by the distance from start points of the control vectors.

2.4.1. CSRBF deformation theory

CSRBFs multivariate interpolation provides $\mathbf{R}^3 \to \mathbf{R}^3$ mapping. Suppose a set of distinct control points $X = \{x_1,....,x_N\} \subseteq \mathbf{R}^3$ is given. Suppose further, we know the values of deviations $g_1,\ldots g_N$ at the control points and we search for a continuous function that interpolates these values at the control points. Then the CSRBF interpolation has the form

$$Sg, X(\vec{x}) = \sum_{i=1}^{N} \alpha_i \phi(\|\vec{x} - \vec{x}_i\|) + p(\vec{x}) \quad , \qquad (1)$$

where $\|\cdot\|$ denotes the usual Euclidean norm in $\mathbf{R}^3$, and p is a low degree polynomial. The coefficients α_i and the polynomial p are determined by the interpolation conditions

$$Sg, X(\vec{x}_i) = g_i, \ 1 \le i \le N, \qquad (2)$$

$$\sum_{i=1}^{N} a_i q(\vec{x}_j) = 0 \qquad (3)$$

for all polynomials q of degree deg(q) $\le$ deg(p).

In [10], positive definite and compactly supported radial functions [11] are applied that have the form

$$\phi(r) = \begin{cases} \psi(r), & 0 \le r \le 1 \\ 0, & r > 1 \end{cases} \qquad (4)$$

Where $\Psi(r)$ is an univariate, and its radius of support is 1. From given two data sets in $\mathbf{R}^3$, $\vec{s}_i = \{x_s^i, y_s^i, z_s^i\}$ for a non-deformed object, and $\vec{d}_i = \{x_d^i, y_d^i, z_d^i\}$ for the deformed object, we construct a space mapping $T : \mathbf{R}^3 \to \mathbf{R}^3$ which is a CSRBF interpolation of form (1) in each of its components. By determining a linear polynomial $p(\vec{x}) = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 z$, condition (2) and (3), we can construct a system of linear algebraic equations (SLAE) for calculations of $\alpha_i, \beta_0, \beta_1, \beta_2, \beta_3$. Since function ϕ is compactly supported, the resulting SLAE has a sparse matrix, which provides simple solution (to be described in Section 2.4.2).

2.4.2. CSRBF deformation algorithm

Using two data sets $\vec{s}_i$ and $\vec{d}_i$ in $\mathbf{R}^3$, we can construct vectors in $\mathbf{R}^3$ from the displacements of the two data sets. We call these vectors control vectors; the start points correspond to $\vec{s}_i$, the destination points correspond to $\vec{d}_i$. Using the displacements of the control vectors, CSRBF deformation is executed.

The CSRBF deformation algorithm consists of following steps:

- Sort the start points of control vectors.
- Find displacement vector for each point. Assign starting and ending positions to each point to find displacement vector.
- Construct a SLAE per RBF.
- Solve the SLAE for the blending coefficients α_i.
- Evaluate the displacement.

In the sorting step, start points of inputted control vectors are sorted using octree data structure [15] to create a band-diagonal sub matrix **A**. Differences between start points and end points of control vectors is the right-hand side of SLAE.

In the constructing step, the sub matrix **A** is constructed as band-diagonal using sorted data in the previous step. Band diagonal structure of the sub matrix **A** provides two advantages. One is to reduce memory consumption, another to provide simple, reliable and fast solution by using combination of block Gauss solution and Cholesky decomposition [16]. In the evaluation step, SLAE solution provides displacements of all points within a radius of support. Hereafter these steps are called "CSRBF deformation".

3. Implementation

The overall structure of developed software shown in Figure 2 combines the C++ CyberGlove library, VTK and CSRBF deformation algorithms discussed in [10] to provide shape modifications.

3.1. Structure of the software

The developed software consists of two modules: VR hand module and the CSRBF deformation module. Both deformation types commonly use the same CSRBF deformation module.

VR hand module acquires CyberGlove's angle data via VHT and then applies the data to the hand-skeleton model. The skeleton model bends the joints synchronized with operator's hand. After calculating a new posture, VR hand module outputs fingertips' coordinates. The coordinates are used to define control vectors.

Both Anchored and Unanchored deformations acquire the coordinates of fingertips. In the Anchored deformation, end points of control vectors are always updated to the acquired coordinates. The start points are updated only when the operator changes the start points. In the Unanchored deformation, $\vec{s}_i$ and $\vec{d}_i$ are determined when a VR hand enters within the radius of support of a 3D object. The Anchored deformation updates an original 3D shape in respect to the operator's actions, while the Unanchored deformation constantly updates the shape. Figure 2 shows the structure of the developed software.

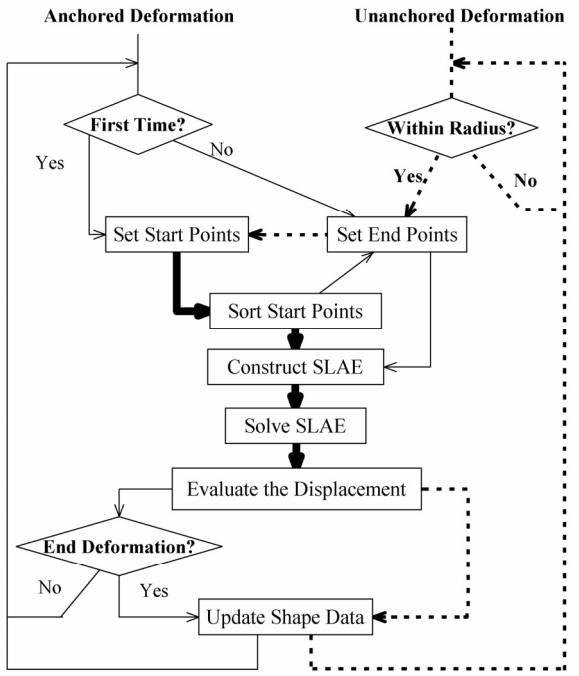

Figure 2: Flowchart shows the structure of the developed software. Solid lines and broken lines represent control flow and data flow, respectively. Bolded lines are followed by both Anchored and Unanchored deformations. "Within Radius?" checks whether the current end points contain the 3D shape's vertices within the radius of support or not.

3.2. Deformation examples and timing results

In this paper, we examined the developed system using the "Moai" model. Figure 4 and Figure 5 show various pictures of the "Moai" model generated in a few seconds by an untrained operator. Our system generates a huge number of potential shapes and these shapes can be used to assist cartoon animation. The set of 10 control vectors, five for each hand, was used for the deformation. Obviously, producing a useful deformation requires skill and training, as well as tuning of parameters such as the

radius of support for the CSRBF. Otherwise, a user may produce excessively plastic deformations, for example.

We also tried to measure the performance of shape deformation algorithm without the CyberGlove. The throughput of the data generated by the CyberGlove is not very high, not fast enough to push the shape deformation algorithm on contemporary CPUs. We thus computed the positions of the 10 CSRBF centers, without relying on the slow data input from the CyberGlove that uses RS-232C for the communication with the computer. For this experiment, we used planer polygonal models, whose number of vertices ranged from 2038 to 65536. We deformed the model by using 10 CSRBFs whose positions are computed by a simple algorithm, not captured. We measured the deformation update rate, including the display overhead, by varying the polygon count of the model and the radius of the CSRBF. We run this set of experiments on a PC with an Athlon 64 3400+ CPU and a mid-low end graphics accelerator card (Nvidia Quadro4 380 xgs)

The results of the experiments are shown in Table 1 and in Figure 3. Figure 3 shows that Anchored deformation is faster than Unanchored deformation with same number of polygons, and with same radius of support. As shown in Table 1, a 65536 polygon model can be deformed at 28 frames per second, more than an interactive rate, with the radius of support of 80. As the radius of support increases, the number of vertices under the influence of the CSRBF increases. The radius of support of 80 implies that all the polygons (and vertices) are under the influence of all the CSRBFs; that is, the worst case. A better frame rate resulted if the radius of support is smaller.

4. Conclusion and future works

This paper proposed a shape deformation system that allows interactive and intuitive manipulation of 3D shape. The algorithm combined a multiple-point-of-contact (MPOC) shape manipulation metaphor with a efficient and fast shape deformation algorithm. The shape is deformed by the force field generated by the ten locally supported radial basis functions (RBFs) attached to the fingertips, whose locations are tracked by using a pair of CyberGlove. The strength of the field generated by the scattered data interpolation algorithm [10] determines the magnitude of deformation. The direction of deformation is computed by one of two algorithms based on the positions of the fingertips tracked by using the CyberGlove.

Compared to the single POC interaction, multiple POC interaction of the proposed method allows more intuitive manipulation of 3D shapes somewhat reminiscent of clay modeling. Also, a very lightweight shape deformation algorithm allows interactive deformation of fairly complex models at an interactive rate. For example, a model having 32 k polygons (or 15 k vertices) can be deformed interactively at the throughput of more than 30 frames per second on a PC with an Athlon 64 3400+ CPU and a mid-low end graphics accelerator card.

The usability and the intuitiveness of the proposed method need still be improved if the method is to be used in a character animation application.

For example, assume an animation of a face. Producing such facial expressions as a smile or a surprise is not easy using the current system. This in part results from excessive degree of freedom built in the media (the face) to be manipulated. Introduction of some form of geometric constraints, e.g., jaw and skull bones, could make the deformation more constrained and well behaved as a facial animation. If the system is to be used for manipulating a face of a deformable puppet, there is another practical issue to be solved. Currently two hands are occupied to deform the face, leaving none to manipulate hand, torso, and other parts of the body. One could find a hand-to-facial shape mapping that requires only one hand. One could also employ a foot pedal for an extra control, e.g., specifying the starting point for the type 1 deformation.

Another issue is the way deformation direction is specified from the sequence of fingertip positions using the type 1 or the type 2 methods. While these two methods are useful, we need to find another mapping, as many of the possible applications of the proposed algorithm can't be covered by these two mappings.

We may also wish to increase the number of points of contact further, e.g., to cover the palm of a hand, so that the interaction become more intuitive and natural for the manipulation of shape.

These improvements would create a system that combines the features of geometric modeling, shape deformation, and character animation, as in the case of clay animation. Such an approach would be different from, for example, a pose space deformation described in [17].

Acknowledgement

Special thanks to Nikita Kojekine for his numerous advices during the development of the software.

5. References

[1] S. He, Y. Kawamura, K. Tanaka, N. Abe, J. Zheng, and H. Taki, "A Virtual Reality System for Exterior Design", In proceedings The 17th International Conference on Artificial Reality and Tele-existence, Dec. 3-5, Tokyo, 1997, 192-199.
[2] Steven Schkolne, Michael Pruett, Peter Schröder, "Surface Drawing: Creating Organic 3D Shapes with the Hand and Tangible Tools", Proc. CHI 2001.
[3] Phantom, SensAble Technologies.

[4] FreeForm Concept system, SensAble Technologies.
[5] W. Baxter, V. Scheib, M. Lin, and D. Manocha, "DAB: Interactive Haptic Painting with 3D Virtual Brushes", Proc. ACM SIGGRAPH 01, August 2001, pp. 461-468.
[6] N. Magnenat-Thalmann, N. E. Primeau, and D. Thalmann, "Abstract muscle actions procedures for human face animation." The Visual Computer, 3, 1988, 290-297.
[7] Y. Lee, D. Terzopoulos, and K. Waters, "Realistic modeling for facial animation." Computer Graphics, 29, 1995, 55-62.
[8] D. Terzopoulos and K. Fleischer, "Modeling inelastic deformation: Viscoelasticity, plasticity, fracture", Computer Graphics, 22(4), Proc. ACM SIGGRAPH'88 Conference, Atlanta, GA, August, 1988, 269-278.
[9] D. C. Zienkiewicz and K. L. Taylor, *The Finite Element Method*, Butterworth-Heinemann, 2000.
[10] N. Kojekine, V. Savchenko, M. Senin, I. Hagiwara, "Real-time 3D Deformations by Means of Compactly Supported Radial Basis Functions", Short papers proceedings of Eurographics EG2002, ISSN 1017-4565, 2002, 35-43.
[11] H. Wendland, "Piecewise Polynomial, Positive Defined And Compactly Supported Radial Functions of Minimal Degree." AICM, 4:389–396, 1995
[12] Immersion Corporation, 2004, www.immersion.com
[13] Virtual Technologies, VirtualHand Suite V1.0 User's guide, 1999.
[14] Virtual Technologies, VirtualHand Suite V1.0 Programmer's guide, 1999.
[15] H. Samet, "The Design and Analysis of Spatial Data Structures", Addison-Wesley Pub Co., 1986.
[16] A. George, J. W. H. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[17] J. P. Lewis, M. Cordner, N. Fong, "Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-Driven Deformation", Proc. of SIGGRAPH 2000, 165-172, 2000

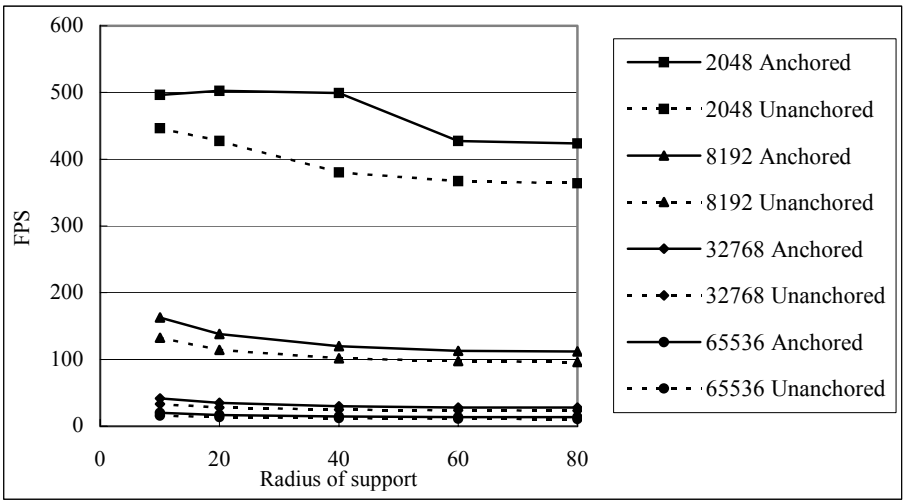

Figure 3. Relation between FPS and Radius of support with different polygonal model (see Table 1 for the detail).

Radius of support / Models	Anchored deformation					Unanchored deformation				
	10	20	40	60	80	10	20	40	60	80
#poly. 2048 #vert. 1089	496.3 (698)	502.1 (2414)	499.2 (7342)	427.4 (10124)	423.6 (10890)	446.3 -	427.6 -	380.2 -	367.0 -	364.0 -
#poly. 8192 #vert. 4225	162.6 (2646)	138.0 (9526)	119.7 (28686)	113.1 (39476)	112.0 (42250)	132.6 -	114.4 -	101.5 -	97.5 -	96.1 -
#poly. 32768 #vert. 16641	41.3 (10510)	34.8 (37562)	30.0 (113282)	28.2 (115812)	28.0 (166410)	32.9 -	28.2 -	24.9 -	23.9 -	23.4 -
#poly. 65536 #vert. 33025	20.0 (20790)	16.7 (74482)	14.4 (225714)	13.6 (309476)	13.4 (330250)	15.9 -	13.5 -	11.9 -	11.4 -	10.9 -

Table 1. Benchmark results of Anchored and Unanchored deformation with 10 control vectors. The upper value of each cell is FPS, and the lower value is the number displacement calculation by CSRBF. If *m* points of the model are placed within *n* control vectors' radius of support, the number of calculation is *m***n*. In Unanchored deformation, at maximum, the number of calculation is the same as that of Anchored deformation.

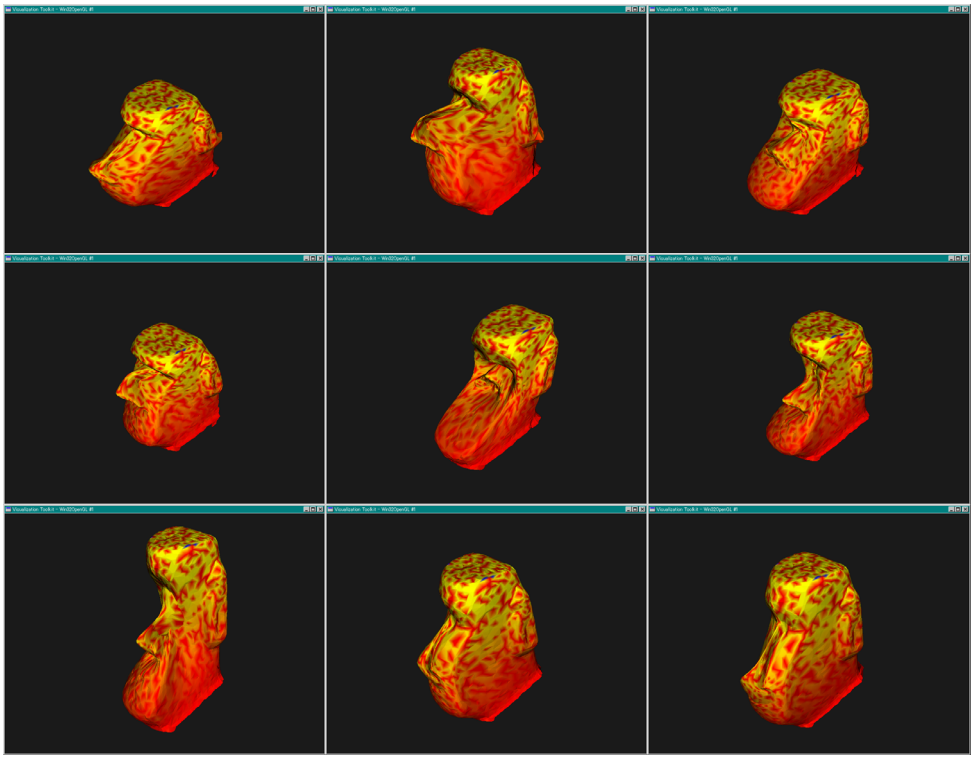

Figure 4. Examples of Anchored deformation deformations of the "Moai" model (head fragment, 7152 polygons).

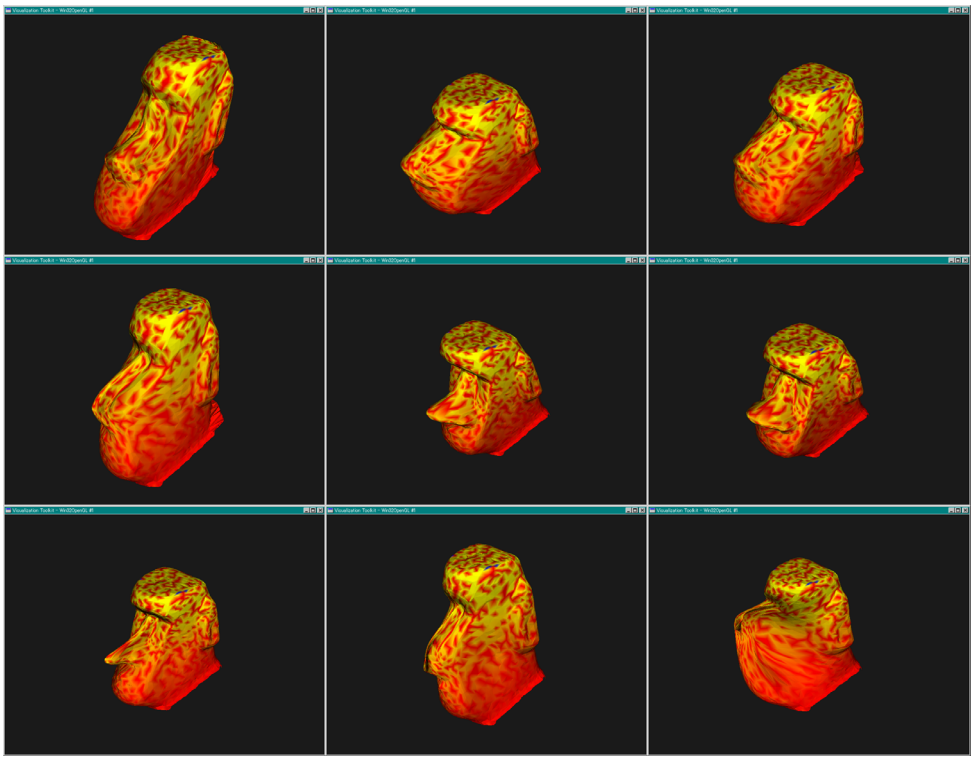

Figure 5. Examples of Unanchored deformation deformations of the "Moai" model (head fragment, 7152 polygons).