Constrained deformations based on convolution surfaces

Yan Zhou[a] Xiaogang Jin[b] Jieqing Feng[b]
zhouyan@wznc.zj.cn jin@cad.zju.edu.cn jqfeng@cad.zju.edu.cn
[a]Department of Computer Science and Engineering, Wenzhou University, Zhejiang Province, 325003, P. R. China
[b]State Key Lab of CAD&CG, Zhejiang University, Hangzhou,310027, P. R. China

Abstract

Deformation is a powerful tool for shape modeling and computer animation. In this paper we present a new local deformation model based on convolution surfaces. After a user specifies a series of constraints, which can consist of line segments, arcs and quadratic curves, their effective radii and maximum displacements, the deformation model creates a set of convolution surfaces taking the constraints as the skeletons. Each convolution surface determines a local influence region and a field function associated with the constraint. Compared with the constrained deformation model based on generalized metaballs, our method can reduce the bulges emerged from the deformation. Furthermore, the user can finely control the deformation result by adjusting the weight distribution along the one dimensional convolution skeleton. This deformation model operates on the local space and is independent of the underlying representation of the object to be deformed. Experimental results show that our new deformation model is both efficient and intuitive.

Keywords: computer animation, computer modeling, convolution surface, constrained deformation

1. Introduction

Three-dimensional shape deformation is an important tool in both shape design and computer animation. Two efficient techniques, namely physics-based modeling and spatial deformation, have been proposed in this field.

Physics-based modeling uses physical simulation to obtain realistic shapes and motions. This technique is very promising but generally too expensive for real-time feedback. Furthermore, the representation of objects cannot be arbitrary.

The idea behind the spatial deformation is to operate on the whole space in which the objects are embedded instead of directly manipulating the vertices or control vertices of these objects. One of the first uses of spatial deformations was by Barr in [1], where the space deformations of twisting, bending and tapering were introduced. However, the general problems of arbitrarily shaped deformations are not addressed.

The most popular method to define spatial deformations is the free-form deformation (FFD) technique developed by Sederberg and Parry [2]. In the FFD technique, a user deforms an object by moving the control points of a trivariate Bézier volume whose control points are organized as a lattice. There have been many variants of FFD. Coquillart extended the FFD technique to allow composite lattices in addition to a cube [3]. FFD using rational Bézier volume or B-spline volumes were also proposed by other authors [4-5].Hsu et al. developed a version of FFD that allows direct manipulation [6], but its computational cost is high. MacCracken and Joy extended FFD to support more general lattices [7].

FFDs and their variants provide a high level of geometric control over the deformation, but they have the limitations in that the user must first define some control points around the region of space to be deformed, and then manipulate these control points. An object embedded within the lattice is deformed by the mapping defined by the variation of the lattice. While this type of technique is very useful for coarse-scale deformations of an object, it can be difficult to use for finer-scale deformations, where a very dense and customized control lattice shape is usually required. Arbitrarily shaped lattices can be cumbersome to construct

and it is often easier to deform the underlying geometry directly than to manipulate a dense control lattice.

Compared with FFD and its variants, axial deformations provide a more compact representation in which a one-dimensional primitive, such as a line segment or curve, is used to define an implicit global deformation [8-10]. By introducing domain curves which define the domain of deformation about an object, Singh and Fiume presented a new geometric deformation technique which is related to axial deformation called Wires [11].

Borrel and Bechmann developed a general deformation model in which the deformation is defined by an arbitrary number of user specified point displacement constraints [12]. The system lets the user select a solution obeying the constraints. However, the shape of the deformation is not strongly correlated with the constraints. To overcome this problem, Borrel and Rappoport introduced a local deformation method which they term Simple Constrained Deformation (Scodef) [13]. In Scodef, the user defines a set of constraint points, giving a desired displacement and radius of influence for each. Each constraint point determines a local B-spline basis function centered at the constraint point, falling to zero for points beyond the radius. The deformation achieved by Scodef is both local and intuitive and the constrained points can be directly located on the boundary surface of the object to be deformed. But their method could not be generalized to deal with other kinds of constraints, such as line, surface and volume, which are desired to extend the flexibility of the local deformation.

The works by Fowler and Gain support not only point displacement constraints, but also orientation constraints on points [14-15]. Llamas et al. developed Twister and Bender using a pair of 3D trackers to control a virtual ribbon and to deform it [16, 17]. Pauly et al used a similar deformation model on a system to edit point-sampled geometry [18]. Milliron et al. introduced a general framework for geometric warps in which the use of orientation constraints a also possible [19].

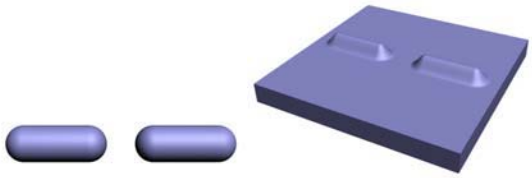
Jin et al. proposed a constrained deformation model based on the special potential function distribution of distance surfaces, which was termed as generalized metaballs in [20]. In Jin's method, constraints are generalized to include points, lines, surfaces and volumes. The user specifies a set of constraints, with desired displacements and an effective radius associated with each constraint. A generalized metaball is then set up at each constraint with a local potential function centered at the constraint falling to zero for points beyond the effective radius. The displacement of any point within the metaballs is a blend of these generalized metaballs. This deformation model is both efficient and intuitive. However, the generalized deformation model may suffer from bulges, which may occur when the user defines two adjacent constrains.

To alleviate the bulges in the constrained deformation based on distance surfaces, we propose a new constrained deformation model based on the special potential function distribution of convolution surfaces. The user specifies a set of constraints (which are termed as skeletons in convolution surfaces), their desired displacements and the radii of influence. Each constraint skeleton determines a convolution surface centered at the skeleton. The field of the convolution surface falls to zero for points beyond the radius. The displacement of any point within the convolution field is a blend of these convolution surfaces.

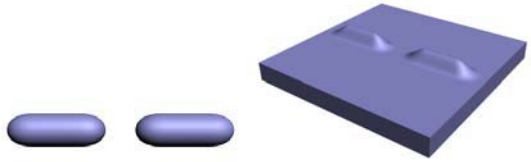
Because of the superposition property of the convolution surfaces, we can effectively reduce the bulges emerged from the generalized metaball based constrained deformation when the user specifies some adjacent constraint skeletons.

Figure 1 shows the deformation results adopting the field functions of distance surfaces and convolution surfaces respectively. The constraint skeletons used in this example are two line segments. From the figure 1(f) we can see, by adopting the convolution surface tool, the bulges appeared in 1(e) disappear.

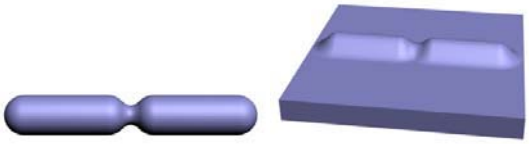
Our deformation mode is very efficient and intuitive, and it is independent of the underlying representation of the objects to be deformed. The computations required by the technique can be done very efficiently and real-time interactive deformation editing on current workstations is possible. Furthermore, the user can control the shape of deformation very conveniently by interactively adjusting the weight function along the skeleton defining the convolution surfaces.



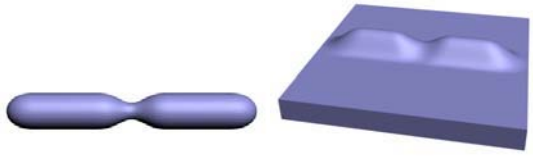
(a) The distance surface (left) and the corresponding deformation result (right) generated by two disjoint line segment skeletons.



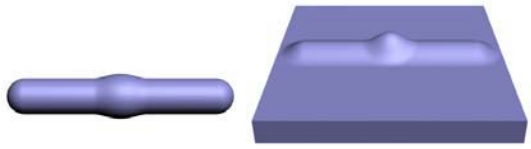
(b) The convolution surface (left) and the corresponding deformation result (right) generated by two disjoint line segment skeletons.



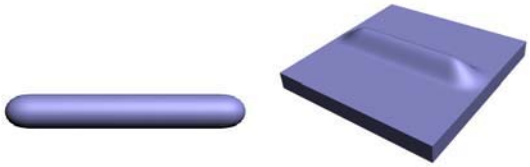
(c) The distance surface (left) and the corresponding deformation result (right) generated by two disjoint line segment skeletons that come closer.



(d) The convolution surface (left) and the corresponding deformation result (right) generated by two disjoint line segment skeletons that come closer.



(e) The distance surface (left) and the corresponding deformation result (right) generated by two connecting line segment skeletons that form one line segment.



(f) The convolution surface (left) and the corresponding deformation result (right) generated by two connecting line segment skeletons that form one line segment.

Figure 1 : The deformation results using the field functions of distance surfaces and convolution surfaces respectively.

The remainder of this paper is organized as follows. Section 2 presents the constrained deformation model based on convolution surfaces. Section 3 presents the computation of convolution surfaces for line segment, arc and quadratic curve skeletons with polynomial weight distribution. Some experimental results are shown in Section 4. Conclusions and future work are discussed in section 5.

2. Constrained deformation model based on convolution surfaces

Convolution surfaces are regarded as a flexible technique for implicit surface modeling. A convolution surface is an isosurface in a scalar field defined by convolving a skeleton, which can comprise points, line segments, curves, surfaces, or volumes, with a potential function. This approach overcomes the drawbacks of bulges and curvature discontinuity in distance surfaces. Convolution surfaces offer many desirable advantages, such as intuitive shape design, well-behaved blending and fluid topology changes with the underlying skeleton. Thus, they provide a very powerful and flexible representation for modeling complex objects.

Bloomenthal and Shoemake proposed convolution surfaces as a natural and powerful tool for implicit surface modeling in [21]. By convolving these skeletons with a three-dimensional (3D) low-pass Gaussian filter kernel, convolution surfaces overcome the problem of bulges and curvature discontinuity in distance surfaces. But Bloomenthal and Shoemake calculated the field function numerically based on point-sampling method, which unfortunately suffers from potential under-sampling artifacts and large storage. McCormack and Sherstyuk deduced analytical solutions for points, line segments, polygons, arcs and planes by employing a kernel function called Cauchy function [22, 23]. The analytical model for line-segment primitives derived by McCormack and Sherstyuk treats the weight distribution along the skeleton uniformly, thus modeling tapering or generalized cylindrical shapes requires specifying multiple line segments.

Jin et al. presented an analytical solution for line-segment skeletons convolved with the Cauchy function modulated by polynomial weighted distributions [24]. Later Jin and Tai

presented analytical solutions for line-segment, arc and quadratic curve skeletons with polynomial weighted distributions for most kernel functions [25, 26].

Let $\mathbf{P}(x, y, z)$ be a space point in $\mathbf{R}^3$, and let $f : \mathbf{R}^3 \to \mathbf{R}$ be a potential function describing the field generated by a single point $\mathbf{Q}$ in a skeleton g, then the field function of the convolution surface for the skeleton g is

$$F(\mathbf{P}) = \int_g f(\mathbf{P} - \mathbf{Q})ds \qquad (1)$$

where ds is the differential length of the skeleton, and f is called the convolution kernel function.

The kernel functions used for convolution surfaces include Gaussian, inverse linear, inverse squared, Cauchy, and polynomial functions. In this paper, we adopt quartic polynomial as kernel function because it leads to simplest computation. The quartic polynomial kernel is defined as

$$f(r^2) = \begin{cases} (1 - \frac{r^2}{R^2})^2, & r \le R \\ 0, & r > R \end{cases} \qquad (2)$$

where R is the effective radius of the kernel, r is the distance from the space point $\mathbf{P}$ to the skeleton.

By using a cubic control curve to define a polynomial distribution function $q(\mathbf{Q}) : \mathbf{R}^3 \to \mathbf{R}$ along a skeleton, and multiplying the field function of a point $\mathbf{Q}$ in the skeleton by $q(\mathbf{Q})$, the convolution model in Eq.(1) now becomes a weighted convolution surface model with polynomial weight distribution:

$$F(\mathbf{P}) = \int_g q(\mathbf{Q}) f(\mathbf{P} - \mathbf{Q})ds \qquad (3)$$

Computer vision research has shown that any 3D object can be defined entirely from a geometric skeleton [27], which implies that skeletons are natural abstractions for 3D objects. Convolution surfaces also provide us with a means to control the shape of an underlying modeling object by controlling its skeleton.

We extend the usage of convolution surface to local space deformation. The field function of a space point is taken as the weight of displacement. By interactively specifying the one dimensional constraints and their effective radii, we can achieve various deformation effects. The constraints are taken as the skeletons for convolution surfaces.

The general constrained deformation model based on convolution surfaces can then be defined. Let $\mathbf{P}(x, y, z)$ be a point in $\mathbf{R}^3$, $Deform(\mathbf{P}) : \mathbf{R}^3 \to \mathbf{R}^3$ be a deformation function which maps $\mathbf{P}$ to $Deform(\mathbf{P})$. Let g_i be a constraint skeleton, $deltD_i$ be its displacement, $F(\mathbf{P})$ be the field function of the convolution surface for the skeleton g_i. The deformation function controlled by constraint g_i is defined as

$$Deform(\mathbf{P}) = \mathbf{P} + deltD_i * F(\mathbf{P}) \qquad (4)$$

Because of the finite support of the kernel function in Eq.(2), deformation function $Deform(\mathbf{P})$ yields a local deformation. For any point on the skeleton whose distance to a point $\mathbf{P}$ is larger than $\mathbf{R}$, its field function contribution to $\mathbf{P}$ is zero. So we have

$$Deform(\mathbf{P}) = \mathbf{P} + deltD_i * F(\mathbf{P}) = \mathbf{P} \qquad (5)$$

To make the maximum displacement of the deformation coincides with $deltD_i$, which is defined by the user, we sample the skeleton uniformly by some points and take the maximum field of these points as the approximate maximum field of the whole skeleton. The field function generated by the skeleton is normalized using this maximum field. Then, for the point with maximum field, we have

$$Deform(\mathbf{P}) = \mathbf{P} + deltD_i * F(\mathbf{P}) = \mathbf{P} + deltD_i \qquad (6)$$

Because of the superposition property of convolution surfaces, which means summing the convolution surfaces generated by two separate skeletons yields the same surface as that generated by their combined skeleton, we can easily extend Eq.(4) to deal with multiple constraints. The deformation function for n constraints is defined as

$$Deform(\mathbf{P}) = \mathbf{P} + \sum_{i=1}^{n} deltD_i * F(\mathbf{P}) \qquad (7)$$

By adjusting the shape of the constraint skeletons, their effective radii and desired displacements, the required deformation can be achieved intuitively and interactively.

3. Field computation for convolution surface

From Eq.(7) we can see that the key for calculating the deformation function lies in the computation of the convolution function $F(\mathbf{P})$.

In the following, we give the computation methods of convolution surfaces for some typical skeletons, which are presented in [26].

3.1 Weight distribution control with cubic control curve

The cubic control curve used here is a one-dimensional Bézier curve. Assuming that the control points are $(\frac{i}{n}, q_i)$, $i = 0,1,2,3$, and based on the identity $\sum_{i=0}^{n} \frac{i}{n} B_{i,n}(u) = u[(1-u)+u]^n = u$, where $B_{i,n}(u)$ are Bernstein polynomials, the curve can be rewritten as $q(u) = \sum_{i=0}^{n} q_i B_{i,n}(u)$. The user can change the control curve by adjusting the Bézier control vertices q_0, q_1, q_2, q_3. With the kernel and the polynomial weight distribution function defined, for a one-dimensional skeleton with parameter t, we can now write the field of a point $\mathbf{P}$ of interest as

$$F(\mathbf{P}) = \int \{ \sum_{i=0}^{3} q_i B_{i,3}(u(t)) \} f(r^2(t)) dt \qquad (8)$$

By modulating the weight of the integration kernel along the skeleton, a convolution surface with varying radius can be achieved.

3.2 Line segment constraint

A line segment of length l with start point $\mathbf{b}$ and unit direction $\mathbf{n}$ can be represented parametrically as

$$\mathbf{L}(t) = \mathbf{b} + t\mathbf{n}, \ 0 \le t \le l \qquad (9)$$

Letting $\mathbf{d} = \mathbf{P} - \mathbf{b}$, the squared distance from the point $\mathbf{P}$ to a point on the line $\mathbf{L}(t)$ is given by

$$\begin{aligned} r^2(t) &= \| \mathbf{d} \|^2 + t^2 - 2t\mathbf{d} \bullet \mathbf{n} \\ &= (t-h)^2 + (d^2 - h^2) \end{aligned} \qquad (10)$$

where $d = \| \mathbf{d} \|$ and $h = \mathbf{d} \bullet \mathbf{n}$

Due to the finite-support of the convolution kernel function, we need to calculate the effective span of the line segment first. If the effective span is $[l_1, l_2]$, where $l_1 < l_2$, the field of a point $\mathbf{P}$ is as follows

$$\begin{aligned} F_{line}(\mathbf{P}) = q'_0 F^1_{line}(\mathbf{P}) + q'_1 F^t_{line}(\mathbf{P}) + \\ q'_2 F^{t^2}_{line}(\mathbf{P}) + q'_3 F^{t^3}_{line}(\mathbf{P}) \end{aligned} \qquad (11)$$

where

$$q'_0 = q_0,$$

$$q'_1 = \frac{1}{l}(-3q_0 + 3q_1),$$

$$q'_2 = \frac{1}{l^2}(3q_0 - 6q_1 + 3q_2)$$

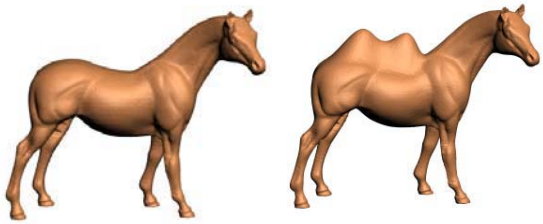
$$q'_3 = \frac{1}{l^3}(-q_0 + 3q_1 - 3q_2 + q_3),$$

and $F^{t^i}_{line}(\mathbf{P}), i = 0,1,2,3$, are the field functions of the line segment $\mathbf{L}(t)$ with weight distribution t^i defined as

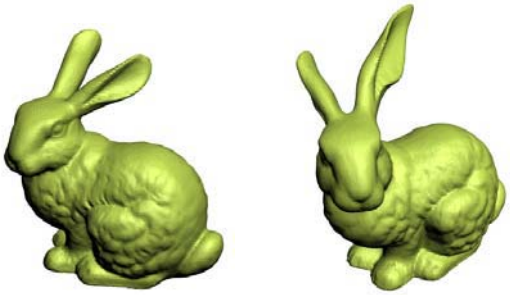
$$F^{t^i}_{line}(\mathbf{P}) = \frac{1}{R^4} \int_{l1}^{l2} t^i (t^2 - 2ht - (R^2 - d^2))^2 dt.$$

By applying integration techniques, $F^{t^i}_{line}(\mathbf{P})$ can be calculated easily [25].

Figure 2(a) shows the deformation result adopting one line segment constraint located on the back of the undeformed horse. By controlling the weight distribution along the line kernel, we can easily create the humps using just one line constraint, which would be difficult for deformation model based on distance field. Figure 2(b) shows the deformation result adopting two line segment constraints located on the ear of the bunny.



(a) undeformed horse and deformed horse



(b) undeformed bunny and deformed bunny

Figure 2 : The deformation results adopting line segment constraints

3.3 Arc constraint

Let $\mathbf{A}(t)$ be an arc defined in the arc's local z-aligned coordinate system,

$$\mathbf{A}(t) = (R_0 \cos t, R_0 \sin t, 0), \varphi_1 \le t \le \varphi_2 \quad (12)$$

where R_0 is the radius of the arc, and φ_1 and φ_2 are the starting and ending angles of the arc.

For an arbitrary arc in space, since the field is coordinate system independent, we may first transform a point $\mathbf{P}(x, y, z)$ into the arc's local z-aligned coordinate system and then perform the field computation.

Similar to line segment constraint, we should first calculate the effective spans of the arc skeleton. The number of effective spans of the arc may be zero, one or two.

Let $\Delta\varphi = \varphi_2 - \varphi_1$, and let *SpanNum* be the number of effective spans. If there is only one effective span, let it be $[\theta_1, \theta_2]$; if there are two of them, let them be $[\theta_1, \theta_2]$ and $[\theta_3, \theta_4]$. Then the arc's analytical field function for a point $\mathbf{P}(x, y, z)$ is

$$F_{arc}(\mathbf{P}) = \sum_{j=1}^{SpanNum} \{ q'_{0\,j} F^{1}_{arc}(\mathbf{P}) + q'_{1\,j} F^{t}_{arc}(\mathbf{P}) + q'_{2\,j} F^{t^2}_{arc}(\mathbf{P}) + q'_{3\,j} F^{t^3}_{arc}(\mathbf{P}) \} \quad (13)$$

The formulae of computing the convolution field function for span skeletons can be found in [26].

3.4 Quadratic curve constraint

Let the quadratic curve constraint be represented as

$$\mathbf{Q}(t) = (x(t), y(t), z(t)) = \sum_{i=0}^{2} \mathbf{Q}_i t^i, 0 \le t \le 1 \quad (15)$$

where $\mathbf{Q}_i = (Q_{ix}, Q_{iy}, Q_{iz})$ are vector coefficients. Quadratic curves that are represented in other parametric schemes, such as Bézier or B-spline, can be easily converted into this power basis form.

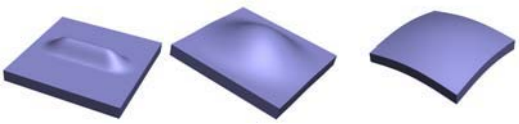
As with line segments and arcs, we should first obtain the effective span(s) of the curve. Then we will use the arc length as the integral parameter in the convolution model.

Here we also take the convolution formulae presented in [26].

4. Experimental results

We have implemented our deformation method on an Intel Pentium M 1.0 GHz computer with 768M main memory under the Windows XP operating system. Figure 3 shows some deformation results using line, arc and quadratic curve constraints. All of them are obtained by applying the corresponding constraints to a thin plate. Figure 3(a) shows the different deformation results adopting one line segment constraint with an increasing effective radius *R*. We can see that the whole object will be affected if *R* becomes big enough. In figure 3(b), the house is created with ten line segment constraints and one quadratic curve constraint, and the letters "house" are created with nine line segment constraints, one arc constraint, one quadratic curve constraint and one B-spline constraint.

In figure 4 we demonstrate an interesting example, in which an ellipsoid is deformed to a man's head under the interactive control of the user. Figure 4(a) is the original ellipsoid. Figure 4(b) is deformation result caused by three line segments. Figure 4(c) uses one quadratic curve and two line segments. Figure 4(d) uses three B-spline constraints for each eye. Figure 4(e) uses one B-spline constraint and one quadratic curve for the mouth. Figure 4(f) uses three line segments to make the chin thinner. Figure 4(g) uses three B-spline constraints to add hairs for the model. At last we get ears by two quadratic curves in figure 4(h).



(a) Deformation results adopting the same constraint with different effective radius.

(b) Deformation results using line, arc and quadratic curve skeletons

Figure 3 : Some deformation results using line, arc and quadratic curve constraints

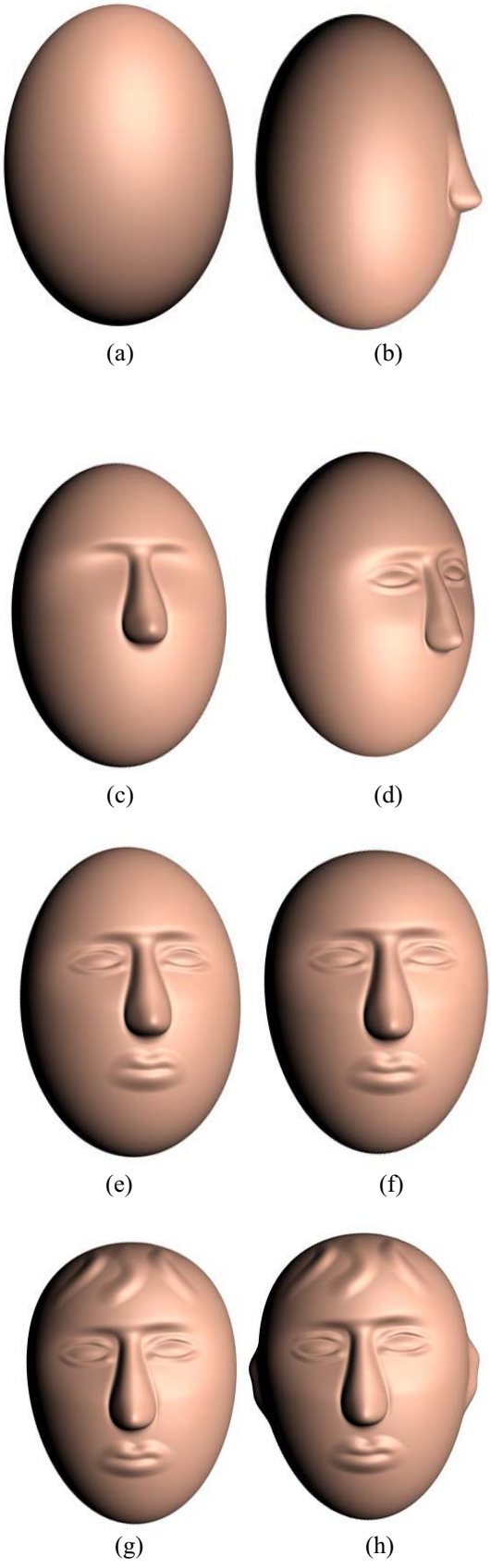


Figure 4 : The process of deforming an ellipsoid to a man's head

5. Conclusions and future work

A new constrained deformation model based on convolution surfaces is presented in this paper. After a user specifies a series of constraints, their effective radii and maximum displacement, the deformation model creates a set of convolution surfaces taking the constraints as the skeletons. Each convolution surface determines a local influence region and the field function associated with the constraint. Different from the constrained deformation model based on the distance surfaces, our method can effectively reduce the bulges appeared in the deformation when the user selects some adjacent constraint skeletons. Furthermore, the user can finely control the deformation result by adjusting the weight distribution along the convolution skeleton. Experimental results show that our deformation method is both powerful and intuitive, and can be implemented interactively on current PC.

The deformation model presented in the paper is based on line segment, arc and quadratic curve constraints. Deformations based on surface and volume constraints are our future work. We also plan to incorporate local rotation and scale constraints into our deformation model.

Acknowledgements

The authors are grateful to Wenguo Wu and Juncong Lin for their help on programming, Dr. Shengjun Liu for his constructive suggestions. and the anonymous reviewers of this paper. This work was supported by NSFC (No. 60573153), and Fok Ying Tung Education Foundation (No. 91069).

References

[1] A.H. Barr. Global and local deformation of solid primitives. Computer graphics, 18(3):21-30, 1984

[2] T. W. Sederberg, S. R. Parry. Free-Form Deformation of Solid Geometric Models. Computer Graphics, 20(4): 151–160, 1986

[3] S. Coquillart. Extended free form deformation: a sculpturing tool for 3D

geometric modeling. Computer Graphics, 24(4):187-193, 1990

[4] P. Kalar, A. Mangli, M. Thalmann, D. Thalmann. Simulation of facial muscle actions based on rational free-form deformations. Computer Graphic Forum,11:59-69, 1992

[5] H. J. Lamousin, W. N. Waggenspack. NURBS-based freeform deformations. IEEE Computer Graphics & Applications,14(9):59-65, 1994

[6] W. M. Hsu, J. F. Hughes, H. Kaufman. Direct Manipulation of Free-Form Deformations. Computer Graphics, 26(2): 177–184, 1992

[7] R. MacCracken, K. I. Joy. Free-Form Deformation With Lattices of Arbitrary Topology. Computer Graphics, 26(3):181-188,1996

[8] F. Lazarus, S. Coquillart, P. Jancence. Axial Deformations: An Intuitive Deformation Technique. Computer Aided Design, 26: 607–613,1994

[9] Q. Peng, X. Jin, J. Feng. Arc-length-based axial deformation and length preserving deformation. In: Thalmann N M, ed. Proc. Computer Animation'97,Geneva: IEEE Computer Society, 87-93, 1997

[10] Y. K. Chang, A. P. Rockwood. A generalized de Casteljau approach to 3D free-form deformation. Computer Graphics, 28(3):257-260,1994

[11] K. Singh, E. Fiume. Wires: A Geometric Deformation Technique. Computer Graphics,32(3): 405–414, 1998

[12] P. Borrel, D. Bechmann. Deformation of n-dimensional objects. International Journal of Computational Geometry and Applications, 1(4): 427-453, 1991

[13] P. Borrel, A. Rappoport. Simple Constrained Deformations for Geometric Modeling and Interactive Design. ACM Transactions on Graphics, 13(2): 137–155, 1994

[14] B. Fowler. Geometric Manipulation of Tensor Product Surfaces. In Proceedings of the 1992 Symposium on Interactive 3D graphics, 101–108, 1992

[15] J. E. Gain. Enhancing Spatial Deformation for Virtual Sculpting. PhD thesis, St. John's College, University of Cambridge, 2000

[16] I. Llamas, B. Kim, J. Gargus, J. Rossignac, C. D. Shaw. Twister: a Space-Warp Operator for the Two-Handed Editing of 3D Shapes. ACM Transactions on Graphics 22, 3, 663–668, 2003.

[17] I. Llamas, A. Powell, J. Rossignac, C. D. Shaw. Bender: A Virtual Ribbon for Deforming 3D Shapes in Biomedical and Styling Applications. ACM Symposium on Solid and Physical Modeling, 2005

[18] M. Pauly, Keiser, L. P. Kobbelt, M. Gross. Shape modeling with point-sampled geometry. ACM Transactions on Graphics 22, 3, 2003

[19] T. Milliron, R. J. Jensen, R. Barzel, A. Finkelstein. A Framework for GeometricWarps and Deformations. ACM Transactions on Graphics 21, 1, 20–51, 2002

[20] X. Jin, Y. F. Li, Q. Peng. General constrained deformations based on generalized metaballs. Computers & Graphics, 24(3):219-231, 2000

[21] J. Bloomenthal, K. Shoemake. Convolution surfaces. Computer Graphics. 25(4):251–256, 1991

[22] J. McCormack, A. Sherstyuk. Creating and rendering convolution surfaces. Computer Graphics Forum, 17(2):113–120, 1998

[23] A. Sherstyuk. Kernel functions in convolution surfaces: a comparative analysis. The Visual Computer, 15(4):171–182, 1999

[24] X. Jin, C.L. Tai, J. Feng, Q. Peng. An analytical convolution surface model for line skeletons with polynomial weighted distributions. Journal of Graphics Tools, 6(3):1–12, 2001

[25] X. Jin, C.L. Tai. Analytical methods for polynomial weighted convolution surfaces with various kernels. Computers & Graphics, 26:437-447, 2002

[26] X. Jin, C.L. Tai. Convolution surfaces for arcs and quadratic curves with a varying kernel. Visual Computer, 18: 530-546, 2002

[27] D. Attali, A. Montanvert. Computing and simplifying 2d and 3d semi-continuous skeletons of 2d and 3d shapes. Computer Vision and Image Understanding, 67(3): 261–273, 1997