PERGAMON

Computers & Graphics 24 (2000) 219–231

COMPUTERS &GRAPHICS

www.elsevier.com/locate/cag

Technical Section

General constrained deformations based on generalized metaballs

Xiaogang Jin[a], Youfu Li[b,*], Qunsheng Peng[a]

[a]State Key Lab of CAD&CG, Zhejiang University, Hangzhou, 310027, People's Republic of China
[b]Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong

Abstract

Space deformation is an important tool in computer animation and shape design. In this paper we present a new local deformation model based on generalized metaballs. The user specifies a series of constraints, which can consist of points, lines, surfaces and volumes, their effective radii and maximum displacements, and the deformation model creates a generalized metaball for each constraint. Each generalized metaball is associated with a potential function centered on the constraint. The value of the potential function drops from 1 on the constraint to 0 on the boundary defined by the effective radius. This deformation model operates on the local space and is independent of the underlying representation of the object to be deformed. The deformation can be finely controlled by adjusting the parameters of the generalized metaballs. We also present some extensions and the extended deformation model to include scale and rotation constraints. Experiments show that this deformation model is efficient and intuitive. It can deal with various constraints, which is difficult for traditional deformation models.

Keywords: Computer animation; Metaballs; Implicit surface; Constrained deformation

1. Introduction

Efficient and intuitive methods for three-dimensional shape deformations play an important role in both geometric modeling and computer animation. Although the shape of an object can be finely controlled by interactively adjusting the positions of its vertices or control vertices, to most users, this manipulation is tedious and inefficient. In the last decade, two efficient techniques, namely physically based modeling and spatial deformation, have been proposed to solve the problem.

Physically based modeling technique produces very realistic deformations of the elastic objects by solving complex differential equations [1–4]. After the users specify the physical attributes (such as mass, friction, external forces, etc.) of the objects, the technique automatically generates the deformations and motions of the objects without any user interaction. Although this technique is attractive, it suffers from some drawbacks. Firstly, the technique involves a large amount of computation. As a result, it is very hard to be used as a real-time interactive design tool. Secondly, the deformations produced by this technique are environment dependent. Finally, as there is no user interaction during the simulation, it is very difficult to control the deformations of the objects. These disadvantages have limited the application of the technique in geometric modeling and computer animation.

The idea behind the spatial deformation techniques is to deform the whole space in which the objects are embedded instead of directly manipulating the vertices or control vertices of these objects. The first spatial deformation model was proposed by Barr [5]. According to Barr's method the transformation matrix is no longer constant but a function depending on the position of the individual points to which the transformation is applied. Obviously, Barr's model is a global approach and hence

*Corresponding author. Tel.: +852-2788-8410; fax: +852-2788-8423.
E-mail addresses: jin@cad.zju.edu.cn (X. Jin), meyfli@cityu.edu.hk (Y. Li)

0097-8493/00/$ - see front matter
PII: S0097-8493(99)00156-9

difficult to deform the objects arbitrarily. The most popular spatial deformation technique is the free-form deformation (FFD) technique developed by Sederberg and Parry [6]. FFD is typically conducted by embedding an object to be deformed into a parametric space of a trivariate Bezier volume whose control points are organized as a lattice, the deformation of the object is obtained by moving the control points of the trivariate Bezier volumes. There have been many variants of FFD. Coquillart et al. extended the FFD technique to allow composite lattices in addition to parallelepiped [7]. Similar techniques based on B-spline volumes or rational Bezier volumes were also proposed by other authors [8,9]. MacCraken et al. developed a new FFD technique, in which the control points of the lattice can be arranged in arbitrary topology [10]. Wyvill et al. present a warping method for CSG/Implicit models [11]. Compared with FFD and its variants, axial deformation provides a more compact deformation method by using an intuitive curve which is easy to control [12,13]. By introducing domain curves which define the domain of deformation about an object, Singh et al. present a new geometric deformation technique which is related to axial deformation called *Wires* [14]. This method can also be used in animating figures with flexible articulations, modeling wrinkled surfaces and stitching geometry together.

Although FFD-based methods can achieve a variety of deformations, the user is forced to define some control points in the space to be deformed and then move these control points. This indirect interface may be unnatural for some applications. Hsu et al. addressed this problem and proposed a direct interface that involves solving a complex equation system [15], but its computational cost is high. Borrel and Bechmann developed a general deformation model in which the deformation is defined by some user-specified point displacement constraints [16]. The desired deformation is obtained by selecting a solution obeying the constraints. Nevertheless, the shape of the resulting deformation in this method is not strongly correlated to the constraints except that the constraints are satisfied. To overcome this problem, Borrel and Rappoport introduced a local deformation method which they term simple constrained deformation (Scodef) [17]. In Scodef, the user defines some constraint points, each of which is associated with a user-defined displacement and an effective radius. The displacement of any point to be deformed is the blend of the local B-spline basis functions determined by these constraint points. Note that 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. To extend the flexibility of the local deformation, however, deformation models based on line, surface, and volume constraints are desired. Borrel and Rappoport pointed out that their model could not be generalized to deal with these kinds of constraints.

Motivated by the concept of metaball, in this paper, we present a new constrained deformation model based on the special potential function distribution of generalized metaballs. In our method, constraints are generalized to include point constraints, line constraints, surface constraints and volume constraints. The user need only define 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 produces a local deformation and is independent of representation of the underlying objects to be deformed. The constraints generate some "bumps" shapes over the space based on the type of constraint and its associated potential function, and influence the final shape of the deformed object directly. The location and height of a bump are defined by a constraint and its influence space is determined by the constraint's effective radius. This method is very intuitive as the user can easily predict the deformed shape according to the constraints. For most constraints, the computations required by the technique can be achieved very efficiently and the deformations can be implemented in real-time on current SGI workstations.

2. Constrained local deformation based on generalized metaballs

Metaball modeling is regarded as a flexible technique for implicit surface modeling. It is very convenient for designing closed surfaces and provides simple solutions for creating blends, ramifications and advanced human character design [18–24]. A good introduction of metaball modeling and implicit surface can be found in [25]. According to the basic formulation proposed by Blinn and Nishimura [18,19], a free-form surface is defined as an isosurface of a scalar field which is generated from some field generating points. The field value at any point is determined by the distance to the generating points. The parameters available for each metaball include the position of the generating point, the potential function, etc.

Later Bloomenthal et al. extended the original idea to include other complex sources such as lines, surfaces and volumes [23,24], which are termed as skeletons. The skeleton-based model provides an intuitive way to define the desired shapes with implicit surfaces. Let C be the skeleton, $P(x, y, z)$ be a point in 3D space, $r(P, C)$ be the minimal distance from $P(x, y, z)$ to the individual points $Q(u, v, w)$ on the skeleton C:

$$r(P, C) = \inf_{Q \in C} \|P - Q\|. \tag{1}$$

Then the potential function associated with skeleton C can be defined as the composition of a potential function $f(r, R)$ which maps $\Re$ to $\Re$ and a distance function $r(P, C)$ which maps $\Re^3$to $\Re$ [26]

$$F(r(P, C), R) = f(r, R) \circ r(P, C), \tag{2}$$

where R is a specified distance called effective radius. Euclidean space is often adopted as the distance space for calculating $r(P, C)$ and

$$r(P, C) = \inf_{Q \in C} \sqrt{(x-u)^2 + (y-v)^2 + (z-w)^2}.$$

The field functions used for implicit surface modeling include Blinn's exponential function, Nishimura's piecewise quadric polynomial, Murakami's degree four polynomial and Wyvill's degree six polynomial [27]. In this paper, we adopt Wyvill's degree six polynomial as the finite potential function because this function blends well and can avoid the calculation of square root

$$f(r, R) = \begin{cases} -\frac{4}{9}\left(\frac{r}{R}\right)^6 + \frac{17}{9}\left(\frac{r}{R}\right)^4 - \frac{22}{9}\left(\frac{r}{R}\right)^2 + 1, & 0 \leqslant r \leqslant R, \\ 0, & r > R. \end{cases} \tag{3}$$

We extend the usage of metaball modeling to local space deformation. The field value of any point of an object is now defined as the weight of displacement from its original position. By interactively specifying the constraints and their effective radii, we can achieve various deformation effects. The constraints can either be points, lines, surfaces or volumes.

Let C be a constraint skeleton, R be the effective radius, and S be the corresponding distance surface

$$S = \{P(x, y, z) \in S | r(P, C) = R\}. \tag{4}$$

We define tuple $M = \langle S, f(r, R)\rangle$ as a generalized metaball based on skeleton C.

A general constrained deformation model based on generalized metaballs can then be defined. Let $P = (x, y, z)$ be a point in $\Re^3$, $Deform(P)$: $\Re^3 \to \Re^3$ be a deformation function which maps P to $Deform(P)$. Let C_i be a constraint which consists of points, lines, surfaces and volumes, $\Delta\mathbf{D}_i$ be its displacement, R_i be the effective radius of C_i. Then the deformation function effected by constraint C_i is defined as

$$Deform(P) = P + \Delta\mathbf{D}_i F(r(P, C_i), R_i). \tag{5}$$

Deformation model (5) has some nice properties. For $\forall P \in C_i$, we have

$$Deform(P) = P + \Delta\mathbf{D}_i F(0, R_i) = P + \Delta\mathbf{D}_i. \tag{6}$$

If the distance from P to constraint C_i is lager than R, we have

$$Deform(P) = P + \Delta\mathbf{D}_i F(R_i, R_i) = P. \tag{7}$$

Therefore, deformation function $Deform(P)$ yields a local deformation which satisfies the constraint precisely in the constraint C_i, and does not affect the points outside the effective radius of the constraint.

The above model can be easily extended to deal with multiple constraints. The deformation function for n constraints is defined as

$$Deform(P) = P + \sum_{i=1}^{n} \Delta\mathbf{D}_i F(r(P, C_i), R_i). \tag{8}$$

The "bumps" generated by the constraints are blended by the potential function. By adjusting the constraints and their effective radii, the required deformation can be achieved. Careful study shows that one constraint may sometimes impose deformation effect on other constraints although this does not prevent the application of the model. We say two constraints are disjoint if neither generalized metaball intersects the other constraint's skeleton. A set of constraints is disjoint if they are pairwise disjoint. Therefore for a disjoint set of constraints deformation model (8) can satisfy all the constraints. Model (8) has the following intuitive implication: the displacement of point P is the average of the displacements of the constraints weighted by their corresponding potential functions.

3. The computation of generalized metaballs

From Formula (1) we can see that the key for calculating the deformation function lies in the computation of distance function $r(P, C_i)$. If C_i is a point constraint, $r(P, C_i)$ is just the distance from point P to C_i, i.e. $r(P, C_i) = \|P - C_i\|$. When C_i is a line segment, a piece of surface, or a volume, the computations involved become complex. In the following, we give the computation methods for some typical cases.

3.1. Line segment constraint

Let C_i be a line segment determined by its end points $P_0 = (x_0, y_0, z_0)$ and $P_1 = (x_1, y_1, z_1)$, its length is l. We first transform this line segment into $\tilde{P}_0\tilde{P}_1$ on the x-axis by a transformation matrix T, where $\tilde{P}_0 = (0, 0, 0)$ and $\tilde{P}_1 = (l, 0, 0)$. For any point P we apply the same transformation T and obtain $\tilde{P} = (\tilde{x}, \tilde{y}, \tilde{z})$. As the distance function is independent of the coordinate system,

$$r(P, C_i{:}\, P_0P_1) = r(\tilde{P}, \tilde{P}_0\tilde{P}_1)$$

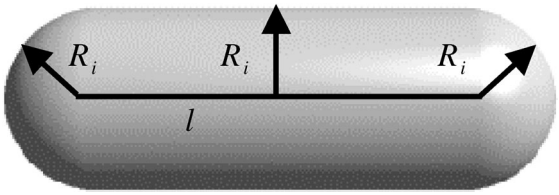


Fig. 1. The corresponding generalized metaball of a line segment.

i.e., the distance from P to C_i is just the distance from $\tilde{P}$ to $\tilde{P}_0\tilde{P}_1$, we have

$$r(\tilde{P}, \tilde{P}_0\tilde{P}_1) = \begin{cases} \sqrt{\tilde{x}^2 + \tilde{y}^2 + \tilde{z}^2}, & \tilde{x} < 0, \\ \sqrt{\tilde{y}^2 + \tilde{z}^2}, & 0 \leqslant \tilde{x} \leqslant l, \\ \sqrt{(\tilde{x} - l)^2 + \tilde{y}^2 + \tilde{z}^2}, & \tilde{x} > l. \end{cases}$$

In order to reduce the computation time, $\tilde{P}_0\tilde{P}_1$ and transformation T can be precomputed. As the potential function f is a function of r^2, the square root computation can be eliminated by computing $r^2(\tilde{P}, \tilde{P}_0\tilde{P}_1)$ instead of r. The corresponding generalized metaball of a line segment is a cylinder with two hemispheres at both ends as illustrated in Fig. 1. If the constraint C_i is a line, the computation becomes much simpler as $r(P, C_i)$ is just the distance from P to the line. The corresponding generalized metaball of a line is a cylinder whose radius and height are R_i and ∞, respectively.

3.2. Polyline constraint

Let constraint C_i be a polyline defined by $P_0P_1P_2 \ldots P_n$. For any line segment $P_{i-1}P_i$ $(i = 1, 2, \ldots, n)$, we can obtain $r(P, P_{i-1}P_i)$ by the line segment constraint method as described above. The distance between any space point P and C_i is the minimum of the obtained distances

$$r(P, C_i: P_0P_1P_2 \ldots P_n) = \min_{i \in [1,\, n]} \{r(P, P_{i-1}P_i)\}$$

3.3. Circle line constraint

Let C_i be a circle line whose radius is R_C. We first transform the circle line onto the xz plane by transformation matrix T, and its center is transformed into the origin. For any point P, we apply the same transformation matrix and obtain $\tilde{P} = (\tilde{x}, \tilde{y}, \tilde{z})$. From Fig. 2 we know $OB = R_C$, $O\tilde{P} = \sqrt{\tilde{x}^2 + \tilde{y}^2 + \tilde{z}^2}$, thus

$$r^2(P, C_i) = R_C^2 + \tilde{x}^2 + \tilde{y}^2 + \tilde{z}^2 - 2R_C\sqrt{\tilde{x}^2 + \tilde{z}^2}.$$

Its corresponding metaball is a torus whose major radius equals $R_C + R_i$ and minor radius equals $R_C - R_i$ as illustrated in Fig. 3.

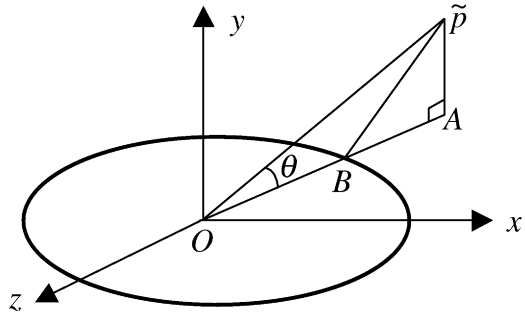


Fig. 2. Distance calculation for a circle line.

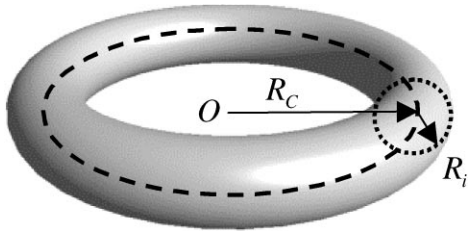


Fig. 3. Generalized metaball for a circle line.

3.4. n Degree Bezier curve constraint

Let C_i be a Bezier curve $\mathbf{R}(u)$ of degree n. The minimal distance from a space point P to C_i either lies in its end points, or lies in the points satisfying the equation:

$$(\mathbf{P} - \mathbf{R}(u)) \cdot \mathbf{R}_u(u) = 0.$$

This equation can be converted into a Bezier curve of degree $2n - 1$, then its roots can be solved by Bezier Clipping [27–29]. The value of the distance function is the minimum of the roots. It is easy to know the corresponding metaball is a generalized cylinder.

3.5. Disk constraint

Let C_i be a disk whose radius is R_C. We first calculate the distance r_1 from space point $P = (x, y, z)$ to the plane where the disk lies (see Fig. 4). If the perpendicular point of P lies within the disk, $r(P, C_i) = r_1$; otherwise we calculate the distance r_2 from P to the circle line, and set $r(P, C_i) = r_2$. The shape of the corresponding generalized metaball of a disk is shown in Fig. 5.

3.6. Planar polygon constraint

Let C_i be a planar polygon $P_0P_1P_2 \ldots P_nP_0$, the plane equation on which it lies is $Ax + By + Cz + D = 0$. Then the distance r_1 from a space point P to the plane is

$$r_1 = \frac{|Ax + By + Cz + D|}{\sqrt{A^2 + B^2 + C^2}}.$$

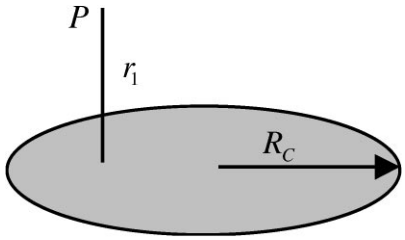


Fig. 4. Disk constraint.

Fig. 5. The generalized metaball for a disk.

Fig. 6. The generalized metaball for a square.

If the perpendicular point of point P lies inside the polygon, $r(P, C_i) = r_1$; otherwise we calculate the distance r_2 from P to the polyline $P_0P_1P_2 \ldots P_nP_0$, and set $r(P, C_i) = r_2$. Fig. 6 shows the shape of the generalized metaball for a square.

3.7. Sphere constraint

Let C_i be a sphere whose radius is R_C, its center is $O(x_C, y_C, z_C)$. Then the distance from space point P to the sphere is

$$r(P, C_i) = |\sqrt{(x - x_C)^2 + (y - y_C)^2 + (z - z_C)^2} - R_C|.$$

The cross section for its corresponding metaball is shown in Fig. 7.

3.8. Cylinder constraint

Let C_i be a cylinder whose radius is R_C and whose height is h. We first transform the cylinder so that its bottom surface lies on the xz plane and its center line coincides with z axis (see Fig. 8). By applying the transformation T to the space point P we obtain $\tilde{P} = (\tilde{x}, \tilde{y}, \tilde{z})$, and

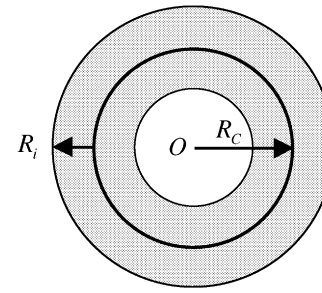


Fig. 7. The generalized metaball for a sphere.

$$r(P, C_i) = \begin{cases} \min(R_C - \sqrt{\tilde{x}^2 + \tilde{z}^2}, \tilde{y}, h - \tilde{y}), \\ \sqrt{\tilde{x}^2 + \tilde{z}^2} \leqslant R_C \&\& 0 < \tilde{y} < h, \\ -\tilde{y}, \\ \sqrt{\tilde{x}^2 + \tilde{z}^2} \leqslant R_C \&\& \tilde{y} \leqslant 0, \\ \tilde{y} - h, \\ \sqrt{\tilde{x}^2 + \tilde{z}^2} \leqslant R_C \&\& \tilde{y} \geqslant h, \\ \sqrt{\tilde{x}^2 + \tilde{z}^2} - R_C, \\ \sqrt{\tilde{x}^2 + \tilde{z}^2} > R_C \&\& 0 < \tilde{y} < h, \\ \sqrt{R_C^2 + \tilde{x}^2 + \tilde{y}^2 + \tilde{z}^2 - 2R_C\sqrt{\tilde{x}^2 + \tilde{z}^2}}, \\ \sqrt{\tilde{x}^2 + \tilde{z}^2} > R_C \&\& \tilde{y} \leqslant 0, \\ \sqrt{R_C^2 + \tilde{x}^2 + (\tilde{y} - h)^2 + \tilde{z}^2 - 2R_C\sqrt{\tilde{x}^2 + \tilde{z}^2}}, \\ \sqrt{\tilde{x}^2 + \tilde{z}^2} > R_C \&\& \tilde{y} \geqslant h. \end{cases}$$

The shape of outer surface of the generalized metaball for a cylinder constraint is shown in Fig. 9.

3.9. Sphere volume constraint

Let C_i be a sphere volume whose radius is R_C, its center be $O(x_C, y_C, z_C)$. Obviously $r(P, C_i)$ equals 0 if a space point P lies inside the sphere volume, otherwise the distance from P to the sphere volume is

$$r(P, C_i) = \sqrt{(x - x_C)^2 + (y - y_C)^2 + (z - z_C)^2} - R_C.$$

3.10. Cube volume constraint

Let C_i be a cubic volume, whose edge length is $2a$. We first apply transformation T so that the center of the cube

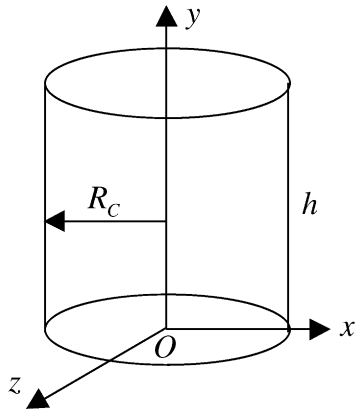


Fig. 8. Cylinder constraint.

Fig. 9. The outer surface of the generalized metaball for a cylinder constraint.

is at the origin, and its edges are parallel to the three coordinate axes. After applying the same transformation T to a space point P we get $\tilde{P} = (\tilde{x}, \tilde{y}, \tilde{z})$. If $|\tilde{x}| \leqslant a \&\& |\tilde{y}| \leqslant a \&\& |\tilde{z}| \leqslant a$, $r(P, C_i)$ equals to 0 as $\tilde{P}$ lies inside the cube. Otherwise the point nearest to $\tilde{P}$ either lies on the faces of the cube (6 cases), or lies on the edges of the cube (12 cases), or lies on the vertices of the cube (8 cases) according to the position of $\tilde{P}$. In each case, the distance can be calculated easily. The shape of the generalized metaball for a cube volume is shown in Fig. 10.

For those constraints which are not listed above, their distance functions can be calculated similarly. When the deformation is applied to an object, the distance function $r(P, C_i)$ must be calculated for any vertex P of the object, and thus the efficiency of the calculation of the distance function determines that of the algorithm. Note that the above deformation model is a local one. If the distance from a point on the candidate object to the constraint is larger than the effective radius of the constraint, this point is not affected. Thus we can adopt the bounding boxes or bounding spheres of the generalized metaballs to improve the efficiency of the algorithm. If a point does not lie inside the bounding boxes of the generalized metaball of a constraint, this constraint has no effect on the point and hence its distance function calculation can be eliminated.

Fig. 10. Generalized metaball for a cube volume.

4. Extensions

In the local deformation model discussed above we adopt Wyvill's six-degree polynomial as the potential function. This polynomial is in fact a special Bezier function. If we generalize the potential function to a Bezier function, more control freedoms can be obtained. The extended potential function can be rewritten as

$$f(r, R_i) = Bez_i(t) = \sum_{j=0}^{m} g_j B_j^k(t), \quad t \in [0, 1],$$

where g_0 is restricted to 1, g_m is restricted to 0, $t = r/R_i$, $t = 1$ if $r > R_i$. A user can use the remaining $m - 1$ control points $\{g_j\}_{j=1}^{m-1}$ to control the distribution of the potential function. The purpose of the restrictions on g_0 and g_m is to make metaballs blend well. Of course these restrictions can be removed if there is only one constraint or a user does not have the well-blend requirement. Moreover, we can deform both the constraint and its local area with distances less than R_U to a user-defined displacement ΔD_i by adjusting the Bezier function as

$$f(r, R_i) = \begin{cases} 1, & r \leqslant R_U, \\ Bez_i\left(\dfrac{r - R_U}{R - R_U}\right), & r > R_U. \end{cases}$$

The shape of the reformed Bezier function is shown in Fig. 11.

In the previous discussions the distance space we adopted is Euclidean distance, which is also known as spherical distance. The disadvantage of adopting such a distance is that the appearance of the resultant deformation is always of "bubble-shape". To alleviate this problem, other non-Euclidean metric spaces can be used to extend the variety of shapes of deformation. If we adopt n-norm metric space which is a straightforward

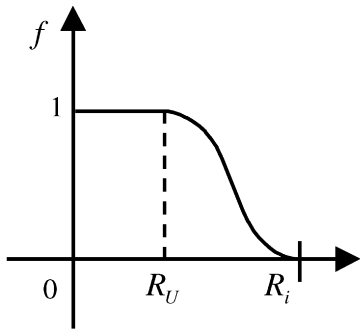


Fig. 11. Reformed Bezier function.

generalization of the Euclidean distance,

$$r(P_1, P_2) = \|((x_1, y_1, z_1), (x_2, y_2, z_2))\|_n$$
$$= (|x_2 - x_1|^n + |y_2 - y_1|^n + |z_2 - z_1|^n)^{1/n}$$

many interesting results can be obtained. If n equals 2 we obtain familiar Euclidean distance, the corresponding metaball for a point constraint is a sphere. If n equals 1,

$$r(P_1, P_2) = \|((x_1, y_1, z_1), (x_2, y_2, z_2))\|_1$$
$$= (|x_2 - x_1| + |y_2 - y_1| + |z_2 - z_1|)$$

we obtain Manhattan distance, and the corresponding metaball for a point constraint is a double pyramid. In the extreme case when $n \to \infty$, we have

$$r(P_1, P_2) = \|((x_1, y_1, z_1), (x_2, y_2, z_2))\|_\infty$$
$$= \max(|x_2 - x_1|, |y_2 - y_1|, |z_2 - z_1|).$$

One obtains city block distance, the corresponding metaball for a point constraint being a cube. By adopting different metric spaces, the influence range can be quite different. Therefore we can take n as an animatable parameter to adjust the influence range of a constraint. But adopting n as an animatable parameter usually requires some costly computations in $r(P_1, P_2)$. An alternative way is to linearly interpolate the Euclidean distance, the Manhattan distance and the city block distance to calculate other form distance in metric space. For example, if we calculate the distance in the following way:

$$(1 - u)\|((x_1, y_1, z_1), (x_2, y_2, z_2))\|_2$$
$$+ u\|((x_1, y_1, z_1), (x_2, y_2, z_2))\|_\infty$$

then, by animating parameter u from 0 to 1, the influence range will change from a sphere to a cube, but the computation involved is quite small. Blanc even presented some anisotropy distance functions such as axial distance function and radial distance function to control precisely the shape of the resulting soft object [26]. Introducing the distance functions into our deformation model can bring even more precise control of the influence range.

In the previous discussions, each space coordinate is treated symmetrically. To accommodate even finer control of the influence range, we can treat each space coordinate differently so as to provide asymmetric, nonisotropic space deformation around the constraints. For example, let R_{ix}, R_{iy}, R_{iz} be the effective radii for x, y, z axes, respectively, $C_i = (C_{ix}, C_{iy}, C_{iz})$, $P = (x, y, z)$, by redefining r/R_i in formula (3) as

$$\sqrt{\frac{(x - C_{ix})^2}{R_{ix}^2} + \frac{(y - C_{iy})^2}{R_{iy}^2} + \frac{(z - C_{iz})^2}{R_{iz}^2}}$$

we can achieve asymmetric space deformation.

5. Deformation by local rotation and scale

In Section 3, we discussed the shape deformation by local displacement or translation. A natural extension is to generalize the deformation model so that it can deal with local rotation and scale. Both of these kinds of deformations are of important use in computer animation. An intuitive interface can be set up by attaching a local coordinate system to each constraint C_i as illustrated in Fig. 12. Let the local coordinate system at C_i be $o'x'y'z'$, then a user moves, rotates and scales the coordinate system until all the translation, rotation and scale requirements are satisfied. Let the destination coordinate system be $o''x''y''z''$ and the transformation matrix from source coordinate system $o'x'y'z'$ to the destination coordinate system $o''x''y''z''$ be $\mathbf{M}$. $\mathbf{M}$ is made up of translation matrix $\mathbf{T}(D_x, D_y, D_z)$, scale matrix $\mathbf{S}(S_x, S_y, S_z)$ and rotation matrix $\mathbf{R}(\theta_x, \theta_y, \theta_z)$, i.e.,

$$\mathbf{M} = \mathbf{T}(D_x, D_y, D_z)\mathbf{S}(S_x, S_y, S_z)\mathbf{R}(\theta_x, \theta_y, \theta_z).$$

For any space point P to be deformed, we first transform it into the local coordinate system $o'x'y'z'$ and obtain P', then multiply P' with transformation matrix $\hat{\mathbf{M}}$ and

$$\hat{\mathbf{M}} = \mathbf{T}(\hat{D}_x, \hat{D}_y, \hat{D}_z)\mathbf{S}(\hat{S}_x, \hat{S}_y, \hat{S}_z)\mathbf{R}(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z),$$

where

$$(\hat{D}_x, \hat{D}_y, \hat{D}_z) = F(r(P, C_i), R_i)(D_x, D_y, D_z),$$

$$(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z) = F(r(P, C_i), R_i)(\theta_x, \theta_y, \theta_z),$$

$$(\hat{S}_x, \hat{S}_y, \hat{S}_z) = (1, 1, 1) + F(r(P, C_i), R_i)$$
$$\times (S_x - 1, S_y - 1, S_z - 1).$$

Suppose the obtained point be P''. Finally we transform P'' back to the global coordinate system and obtained the deformed image of point P. If there is only translation, i.e. $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z) = 0$ and $(\hat{S}_x, \hat{S}_y, \hat{S}_z) = (1, 1, 1)$, the result is the same as the deformation model discussed in Section 3.

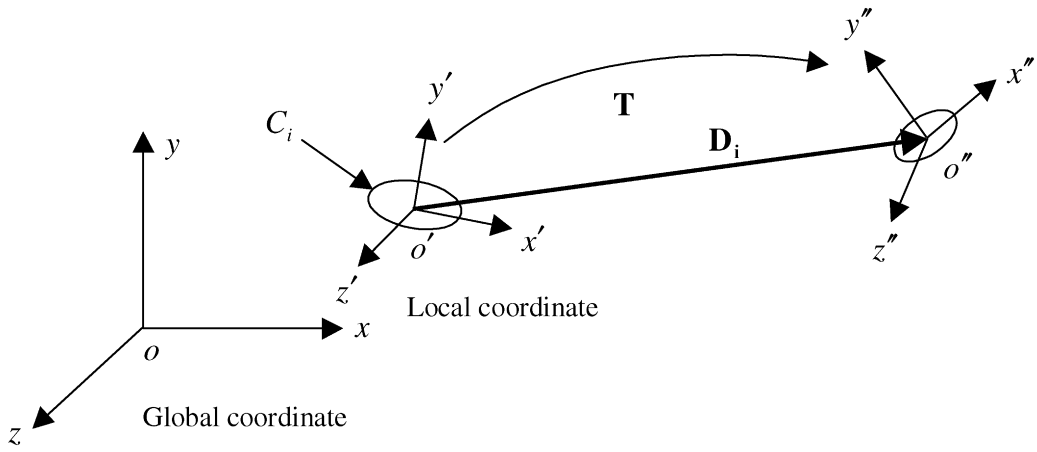


Fig. 12. Local coordinate system for a constraint.

In the previous discussion, we use Euler angles $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)$ to describe the rotation of the coordinate system. However, Euler angle representation suffers from several disadvantages [30]. Firstly, Euler rotations must be applied in a specified order because they do not commute. Secondly, it suffers from non-uniformity. A fixed change in Euler angles does not always yield the same amount of rotation change. Thirdly, Euler angle representation suffers from "gimbal lock". An alternative is to use quaternion instead of Euler angles [30]. Euler angles $(\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)$ can be converted into a quaternion

$$q = \left(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\mathbf{n}\right) = (w, x, y, z),$$

where θ is the rotation angle and $\mathbf{n}$ is the unit rotation axis. By converting quaternion q into rotation matrix $\mathbf{R}(\theta)$, we get

$$R(\theta) = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2xz+2wy \\ 2xy+2wz & 1-2x^2-2z^2 & 2yz-2wx \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 \end{bmatrix}.$$

In this case, the transformation matrix $\hat{\mathbf{M}}$ is refined as

$$\hat{\mathbf{M}} = \mathbf{T}(\hat{D}_x, \hat{D}_y, \hat{D}_z)\mathbf{S}(\hat{S}_x, \hat{S}_y, \hat{S}_z)\hat{\mathbf{R}}(\hat{\theta}),$$

where

$$\hat{\theta} = \theta F(r(P, C_i), R_i).$$

As quaternions interpolate only one angle instead of three Euler angles, it can generate smoother rotation interpolation and hence more fluid deformation. Given a set of quaternions, they can be spherically interpolated using a general construction scheme [30].

6. The animation of the deformations

The above deformation model can be conveniently applied to generate a deformation animation. We present two ways to simulate the deformation process of an object. The first is to apply a set of different constraints to the same object to obtain a sequence of deformed objects. Since these objects possess the same number of vertices and the same topology, we can blend them to generate the intermediate shapes by interpolating the corresponding vertices. The other method is to interpolate the corresponding parameters of the keyframe constraints to generate the intermediate constraints. The intermediate constraints are then applied to produce the deformation of the object for the intermediate frames. In fact, a constraint can be completely determined by parameter set Ω:

$$\Omega = \{C_i, \Delta\mathbf{D}_i, S_x, S_y, S_z, \theta, R_{ix}, R_{iy}, R_{iz}, g_1, g_2, \ldots, g_{m-1}\}.$$

Given the parameter set of the keyframes, traditional parametric key frame techniques can be used to generate the intermediate parameter set of the constraint.

Since both methods are based on parametric key frame techniques, which are provided by many animation systems, our deformation animation model can be conveniently incorporated into these animation systems.

7. Experiments

We implemented our algorithm on an SGI Indy Workstation. Fig. 13 shows the potential function distribution of the line constraint, disk constraint, square constraint, polyline constraint, point constraint adopting Euclidean distance and point constraint adopting Manhattan distance, respectively. All of them are obtained by applying

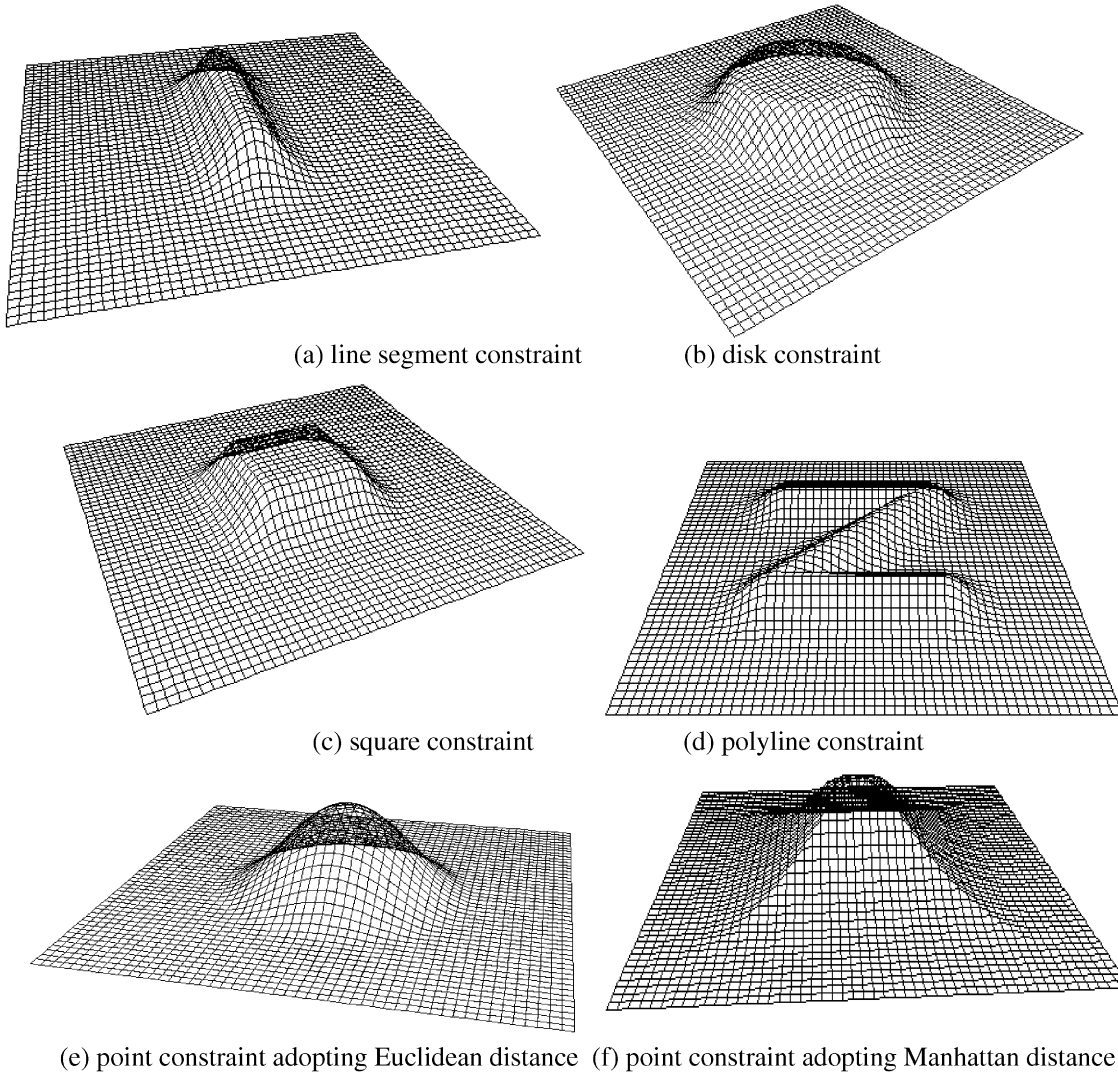


Fig. 13. The potential function distribution for some constraints.

the corresponding constraints to a grid. Fig. 14 is the wireframe of an undeformed cow, and Fig. 15 is the deformed cow by locating a line constraint on its back. Fig. 16(a) is an undeformed teapot. Fig. 16(b) is the deformed teapot by applying two plane constraints, with one put on its top and the other put on its left. Fig. 17 shows an undeformed cow and a plane constraint. The constraint is put on the right of the cow with a 45° to the xy plane where the cow lies. Fig. 18 shows the animation sequence obtained by animating the displacement of a plane constraint. From the two examples we can see that a plane constraint is like a magnet, it attracts the points within the influence range. Fig. 19 shows a "Z" deformed from a grid. There

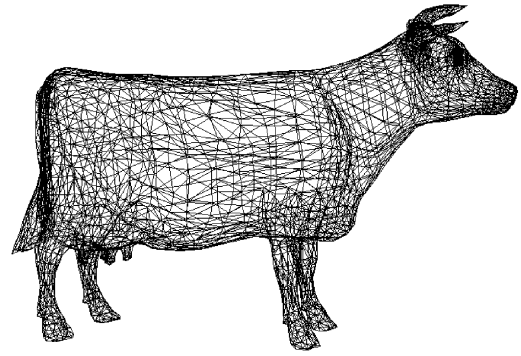


Fig. 14. The wireframe of an undeformed cow.

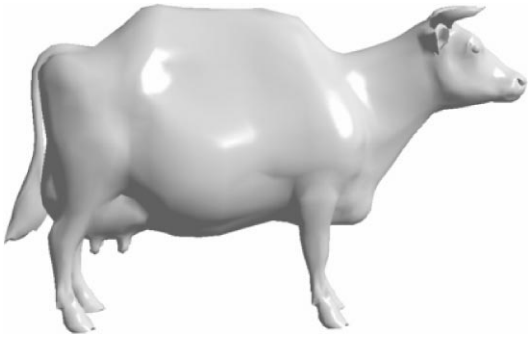


Fig. 15. Deformed cow by a line constraint.

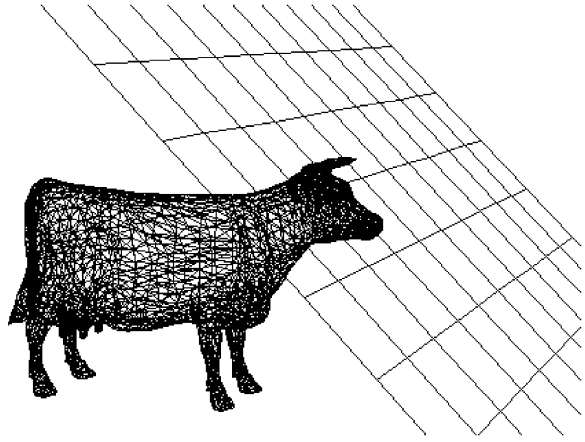
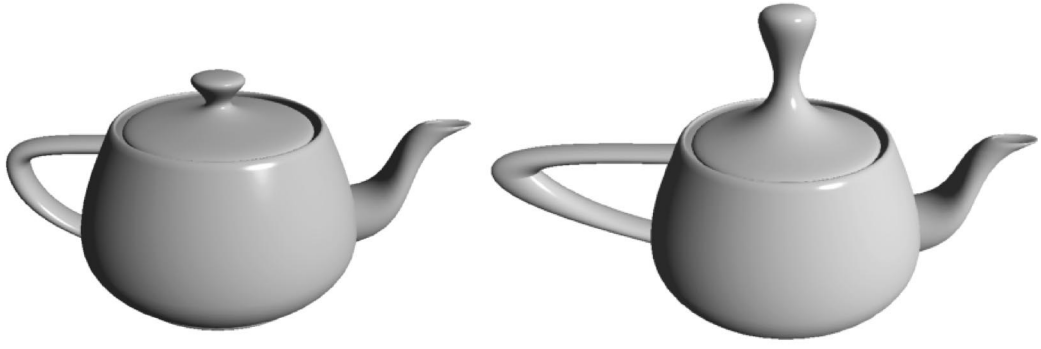


Fig. 17. Undeformed cow and a plane constraint.

are 12 point constraints corresponding to 12 metaballs in the environment. Compare this deformation with that in Fig. 13(d). Fig. 20 shows the undeformed cow and a sphere volume constraint, with the small sphere being the constraint and the big sphere being the generalized metaball indicating the influence range of the constraint. The line shows the displacement $\Delta\mathbf{D}$. The 3D-morphing sequence in Fig. 21 is achieved by animating $\Delta\mathbf{D}$ of the sphere volume constraint. We note that only the vertices of the cow which lie within the big sphere are deformed and other vertices are not affected at all. Fig. 22 shows the animation sequence by animating the scale constraint of a sphere volume constraint. All the points in the head of the cow satisfy the constraint. The 3D-morphing sequence in Fig. 23 is achieved by animating the rotation of the sphere volume constraint. The rotation constraint θ is -120° around axis $(\sqrt{2}/2, \sqrt{2}/2, 0)$. As the whole head of the cow is within the sphere volume, all the points on the head rotate -120° and hence the head maintains the same shape.

8. Conclusions

A general constrained deformation model is presented in this paper. After a user specifies a series of constraints which can consist of points, lines, surfaces and volumes, their effective radii and maximum displacements, the deformation model creates a set of generalized metaballs taking the constraints as the skeletons. Each metaball determines a local influence region and is associated with a local potential function. The potential function is centered at the constraint and falls to zero for points beyond the effective radius. We present our methods for calculating the distance functions for some typical constraints such as point, line segment, disk, Bezier curve, polygon, sphere volume, etc. One advantage of our deformation model is that it is independent of the representation of the underlying objects and can apply to both



(a) (b)

Fig. 16. The deformation of a teapot; (a) before deformation; (b) deformed teapot by two plane constraint.

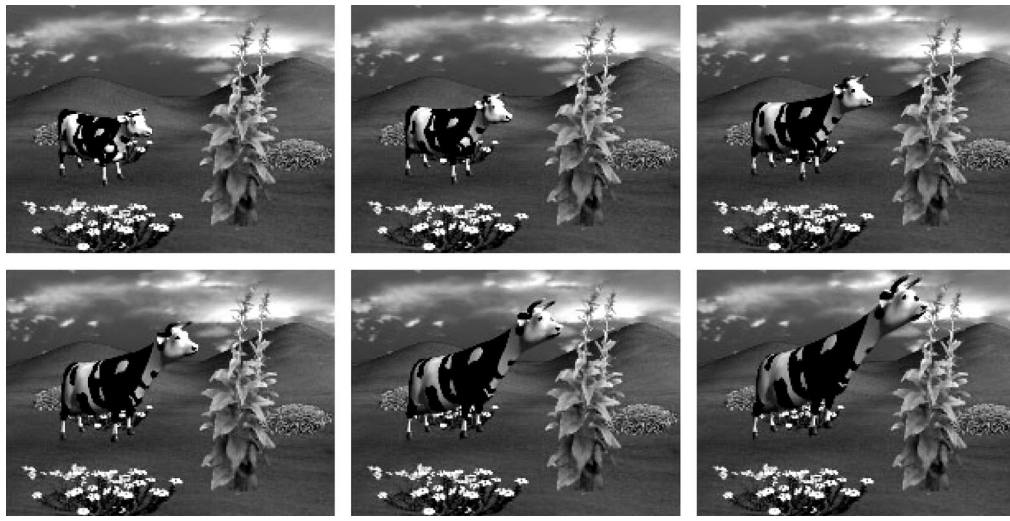


Fig. 18. 3D morphing sequence by animating the displacement of a plane constraint.

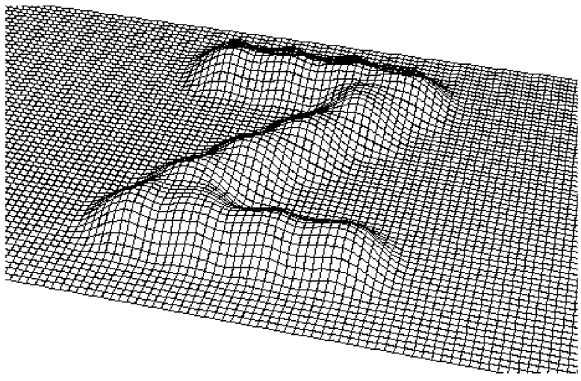


Fig. 19. Z deformed from a plane.

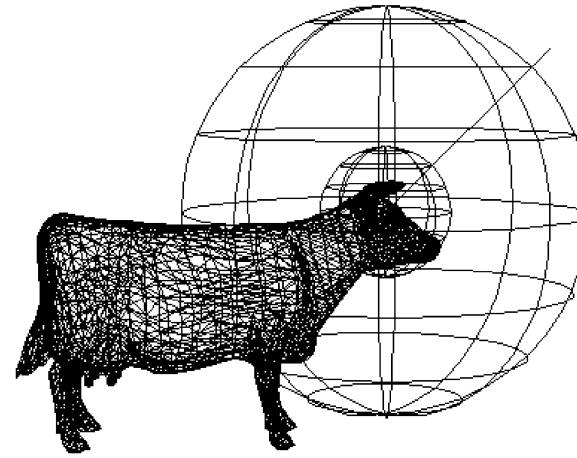


Fig. 20. Undeformed cow and a sphere volume constraint.

polygon mesh and parametric surfaces. For most of the useful constraints, the algorithm is of high efficiency because the calculation involved is simple, and can be implemented interactively on current SGI workstations. Compared with other deformation methods, this deformation model has the following features:

- *Generality*: This method cannot only deal with point constraint but also line, surface and volume constraints, which are difficult for traditional methods. The scale constraint and rotation constraint can also be dealt with in a systematic way.
- *Intuition*: For a specified constraint, a user can easily imagine the deformation effects caused by the constraint.
- *Locality*: Only points located in the local influence range are affected. Therefore it provides a useful tool for local shape adjustment.
- *Compatibility*: The deformation model can be easily incorporated into most existing animation systems.

Fig. 21. 3D morphing by animating the displacement of a sphere volume constraint.

Fig. 22. 3D morphing by animating the scale of a sphere volume constraint.

Fig. 23. 3D morphing by animating the rotation of a sphere volume constraint.

Acknowledgements

This work received support from the National Natural Science Foundation of China, Zhejiang Provincial Natural Science Foundation and City University of Hong Kong (grant No. 7000639). The authors are grateful to Dr. Jieqing Feng for his constructive suggestions.

References

[1] Terzopoulos D, Platt J, Barr AH, Fleischer K. Elastically deformation models. Computer Graphics 1988;21(4):205–14.
[2] Platt J, Barr AH. Constraints methods for flexible models. Computer Graphics 1988;22(4):279–88.
[3] Terzopoulos D, Fleischer K. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. Computer Graphics 1988;22(4):269–78.
[4] Terzopoulos D, Witkin A. Physically-based methods with rigid and deformable components. IEEE Computer Graphics & Applications 1988;8:41–51.
[5] Barr AH. Global and local deformation of solid primitives. Computer graphics 1984;18(3):21–30.
[6] Sederberg TW, Parry SR. Free-form deformation of solid geometric models. Computer Graphics 1986;20(4):537–41.
[7] Coquillart S. Extended free-form deformation: a sculpturing tool for 3D geometric modeling. Computer Graphics 1990;24(4):187–93.
[8] Kalar P, Mangli A, Thalmann M, Thalmann D. Simulation of facial muscle actions based on rational free-form deformations. Computer Graphic Forum 1992;11:59–69.
[9] Lamousin HJ, Waggenspack WN. NURBS-based free-form deformations. IEEE Computer Graphics & Applications 1994;14(9):59–65.
[10] MacCracken R, Joy KI. Free-form deformations with lattices of arbitrary topology. Computer Graphics 1996;26(3):181–8.
[11] Wyvill B, Overveld KV. Warping as a modelling tool for CSG/implicit models. In: Proceedings of shape modeling international'97, University of Aizu, Japan: IEEE Computer Society Press, 1997. p. 205–14.
[12] Lazarus F, Coquillart S, Jancence P. Axial deformations: an intuitive deformation technique. Computer Aided Design 1994;26(8):607–13.
[13] Chang YK, Rockwood AP. A generalized de Casteljau approach to 3D geometric modeling. Computer Graphics 1994;28(4):257–60.
[14] Singh K, Fiume E. Wires: a geometric deformation technique. Computer Graphics 1998;32(3):405–14.
[15] Hsu W, Hughes J, Kaufmann H. Direct manipulations of free-form deformations. Computer Graphics 1992;26(2):177–84.
[16] Borrel P, Bechmann D. Deformation of N-dimensional Objects. International Journal of Computational Geometry and Applications 1991;1(4):427–53.
[17] Borrel P, Rappoport A. Simple constrained deformations for geometric modeling and interactive design. ACM Transactions on Graphics 1994;13(2):137–55.
[18] Blinn JF. A generalization of algebraic surface drawing. ACM Transactions on Graphics 1982;1(3):235–56.
[19] Nishimura H, Hirai M, Kawai T. Object modeling by distribution function and a method of image generation. Transactions on IECE 1985;68-D(4):718–25.
[20] Wyvill G, McPheeters C, Wyvill B. Data structure for soft objects. The Visual Computer 1986;2:227–34.
[21] Wyvill B, Wyvill G. Field functions for implicit surfaces. The Visual Computer 1989;5:75–82.
[22] Shen J, Thalmann D. Interactive shape design using metaballs and splines. In: Brian Wyvill, Marie-Paule Gascuel, editors. Proceedings of implicit surfaces'95. France: Grenoble, 1995. p. 187–96.
[23] Bloomenthal J, Wyvill B. Interactive techniques for implicit modeling. Computer Graphics 1990;24(2):109–16.
[24] Bloomenthal J, Shoemake K. Convolution surfaces. Computer Graphics 1991;25(4):251–6.
[25] Bloomenthal J, Bajaj C, Blinn J, Cani-Gascuel M, Rockwood A, Wyvill B, Wyvill G. An introduction to implicit surfaces. Los Altos, CA: Morgan Kaufmann Publishers, 1997.
[26] Blanc C, Schlick C. Extended field functions for soft objects. In: Gascuel MP, Wyvill B, editors, Implicit Surfaces'95 Workshop, 1995. p. 21–32.
[27] Nishita T, Nakamae E. A method for displaying metaballs by using Bezier clipping. Computer Graphics Forum 1994;13(3):271–80.
[28] Schneider PJ. Solving the nearest-point-on-curve problem. In: Glassner AS, editor. Graphics Gems I, Academic Press, 1990. p. 607–611.
[29] Hart JC. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. The Visual Computer 1996;12:527–45.
[30] Shoemake K. Animating rotation with quaternion curves. Computer Graphics 1985;19(3):245–54.