THE CAO EN SURFACE: A NEW APPROACH TO FREEFORM GEOMETRIC MODELS

G. Wyvill,[1] Cao En,[1,2] and A. Trotman[1]

The Cao En surface provides a new way to describe freeform models for computer graphics. In essence, it is a mapping from a sphere, or hypersphere, to a corresponding, closed, parametric surface. However, the same technique can create surfaces of arbitrary topology, perform smoothing by interpolating surface normals, calculate projections of a surface, define distortions, and interrogate texture functions. The mapping can be described as the function of a two-layer neural network. This structure suggests that it could be implemented efficiently in special-purpose parallel hardware. By separating the concepts of blending and geometry, the Cao En surface has enabled us to better understand the modelling process.

1. Introduction

In this paper we present a new kind of geometric model for freeform surfaces, the Cao En surface. Why do we need it?

Parametric patches, such as Bezier or B-spline surfaces, can approximate arbitrary surfaces, but in common use, they have several drawbacks. If you use patches at all, you must decide where the boundaries are to be. Choosing the number and placement of patches has received little or no attention in the literature, yet it is the first and often the most difficult problem in a given design. Most objects consist of one or more closed surfaces. A patch, by its nature, is not closed, so the onus is placed on the designer to organize a network of patches to fit the desired shape and enforce conditions to close the surface.

An attractive alternative is to use blended, implicit surfaces (Blinn 1982, Wyvill 1986, Nishimura 1985). Soft objects created this way are guaranteed to be closed, 3D objects and can be incorporated easily into a CSG system (Wyvill 1990). But the efficient construction of soft objects is still a mysterious art. If you try to create a simple skeleton of key points, the resulting surface will show a bump, roughly corresponding to each key point. Smoother results can be had by choosing key points that create general quadric fields, and relatively few of these can be combined to produce a variety of shapes. But this is a process requiring a great deal of skill and an understanding of the underlying mathematics. It is no longer intuitive.

A parametric patch defines a mapping from a finite plane to a bounded, curved surface in 3D space. The Cao En surface, in its simplest form, can be thought of as a mapping from the surface of a sphere to an arbitrary, closed surface in 3D space. Intuitively, it is like taking a spherical balloon and stretching the surface into another shape. To describe the stretching operation, we specify a set of points on the original sphere and a corresponding set of points on the target surface. The resulting surface is guaranteed closed and topologically identical to the sphere. Like other parametric surfaces, it can intersect itself and it does not necessarily separate inside and outside regions of space.

2. Definition of the Cao En Surface

Let x represent a vector, $(x_1, x_2, \ldots x_m)$ on the surface of a unit hypersphere. Thus,

$$x_1^2 + x_2^2 + \ldots x^2_m = 1 \qquad (1)$$

[1]Department of Computer Science, University of Otago, Dunedin, New Zealand.

[2]On leave from the Department of Electronic Engineering, Beijing University of Aeronautics and Astronautics, Beijing, People's Republic of China.

Published in Programmirovanie, No. 4, pp. 4-16, July-August, 1992. Original article submitted June 15, 1992.

0361-7688/92/1804-0135$12.50

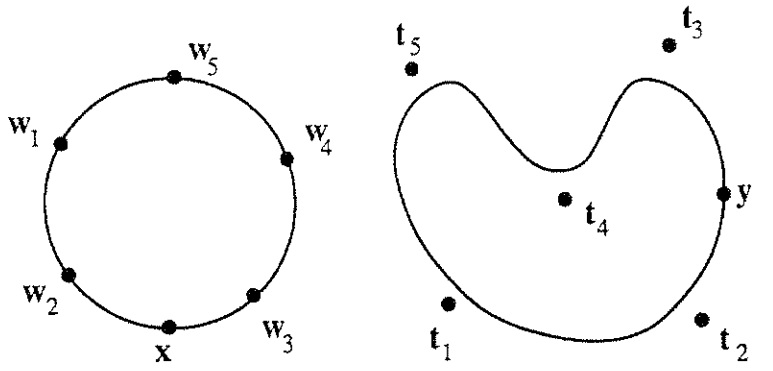

Fig. 1. Each w_i corresponds to a point t_i in the target space. A point y on the target surface corresponds to a source point x on the parametric sphere.

Let K_i, $1 \le i \le n$ represent one of n *key points*. With K_i is associated a weight vector $w_i = (w_{i,1}, w_{i,2}, ...w_{i,m})$, and a blending function F_i.

Let G_i, $1 \le i \le p$ represent one of p *shape generators*. With each G_i is associated a weight vector $v_i = (v_{i,1}, v_{i,2}, ...v_{i,n})$.

For each K, we can calculate a value

$$k_i = F_i(\mathbf{x}.\mathbf{w}_i) \tag{2}$$

and these k values make a vector, $\mathbf{k} = (k_1, k_2, ...k_n)$.

For each G_i, we can calculate a value

$$y_i = \mathbf{k}.\mathbf{v}_i \tag{3}$$

and these p values make a vector, $\mathbf{y} = (y_1, y_2, y_p)$.

The vector y represents a position in p-dimensional space that is a mapping from the vector x in m-dimensional space. The set of points y that correspond to the set of all points x on the surface of the hypersphere defines a Cao En surface.

In the case where $m = p = 3$, this defines a mapping from a sphere to a closed surface in 3D using n key points.

For computer implementation, it is convenient to regard the vectors w_i as comprising a matrix W of points distributed on the surface of a hypersphere which we call the blending matrix. Similarly, the vectors v_i comprise a matrix V, which we call the shape matrix.

3. Understanding the Definition

Each w_i represents a point on the original hypersphere. The dot product $\mathbf{x} \cdot \mathbf{w}_i$ is a measure of how close the point x is to w_i. The blending function F_i determines how quickly the influence of w_i drops off as x moves away. For a given i, the p dimensional vector $t_i = (v_{1,i}, v_{2,i}, ...v_{p,i})$ represents a point in the target space corresponding to the point w_i. The Cao En surface does not, in general, include t_i, but will normally pass close to it (see Fig. 1). We can think of the points w_i as source points that map approximately to the corresponding target points t_i. This fits with the idea of stretching a rubber sphere into a new shape. We stretch it to try and make each w_i move to t_i and the result is a smooth surface that approximates this fit.

4. The Blending Function F_i

Every pointy on the target surface is a weighted sum of the shape vectors t_i $1 \le i \le n$. For a point x close to some w_i, we would expect the corresponding point y to be close to t_i. This is achieved if k_i is equal to one when $\mathbf{x} = w_i$ and close to zero when x is reasonably distant from w_i. The dot product $\mathbf{x} \cdot \mathbf{w}_i$ is a cosine function of the angular distance from x to w_i. The purpose of the blending function F_i is to change this cosine function into something closer to our needs. In principle, you can have a different F_i for each key point. So far, we have not needed such flexibility. If we introduce a parameter q_i for each key point, then the function

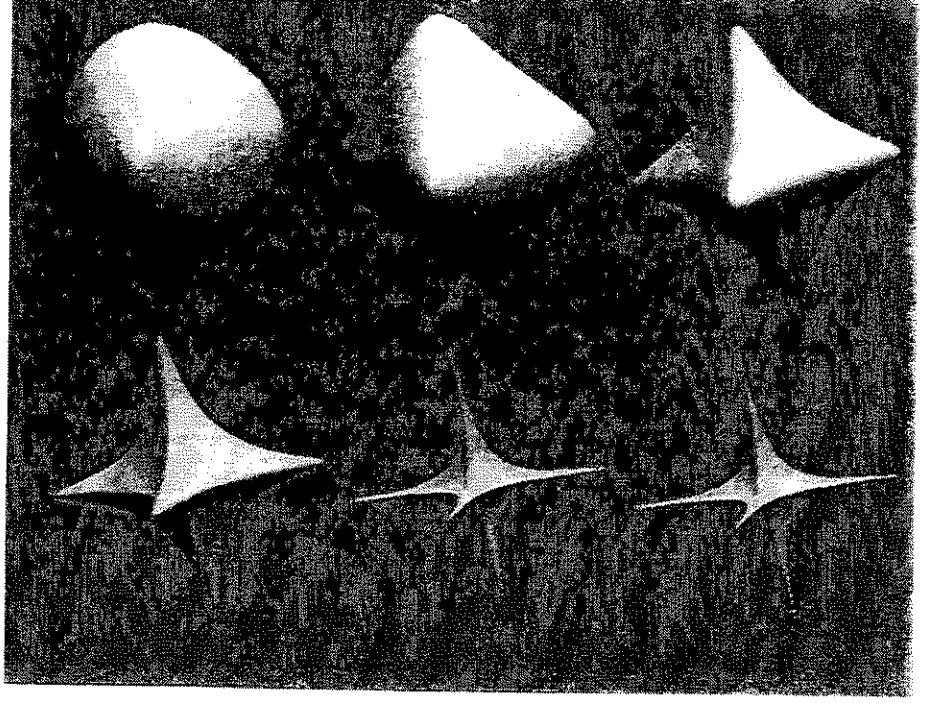

Fig. 2

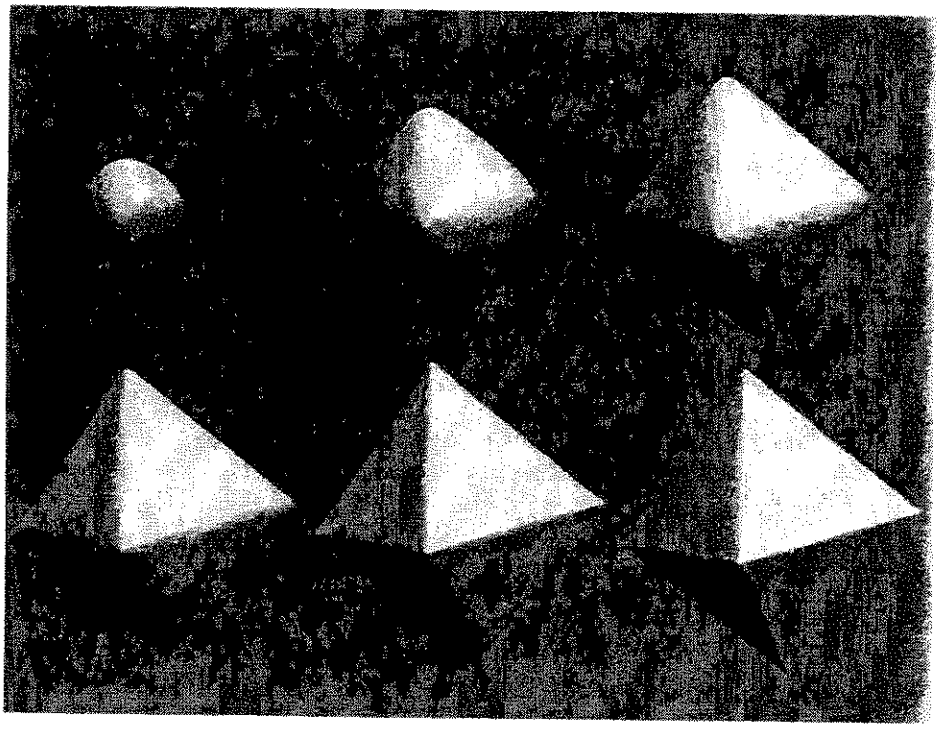

Fig. 3

$$F_i(s) = \begin{cases} s^{q_i} & s > 0 \\ 0 & \text{elsewhere} \end{cases} \tag{4}$$

produces a suitable blending effect. We have also used

$$F_i(s) = e^{(q_i(s-1))} \tag{5}$$

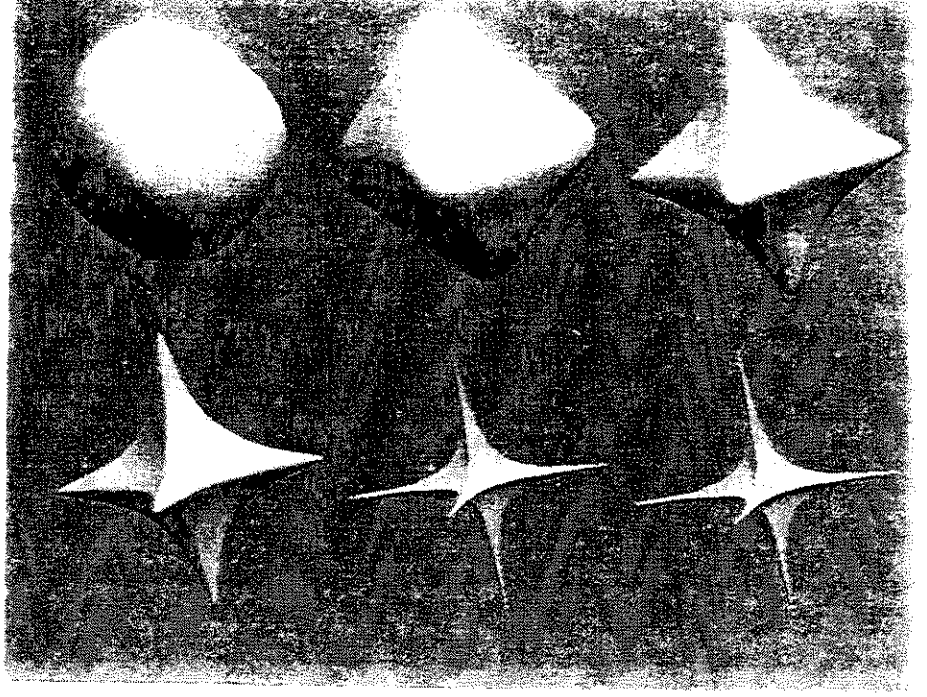

Fig. 2

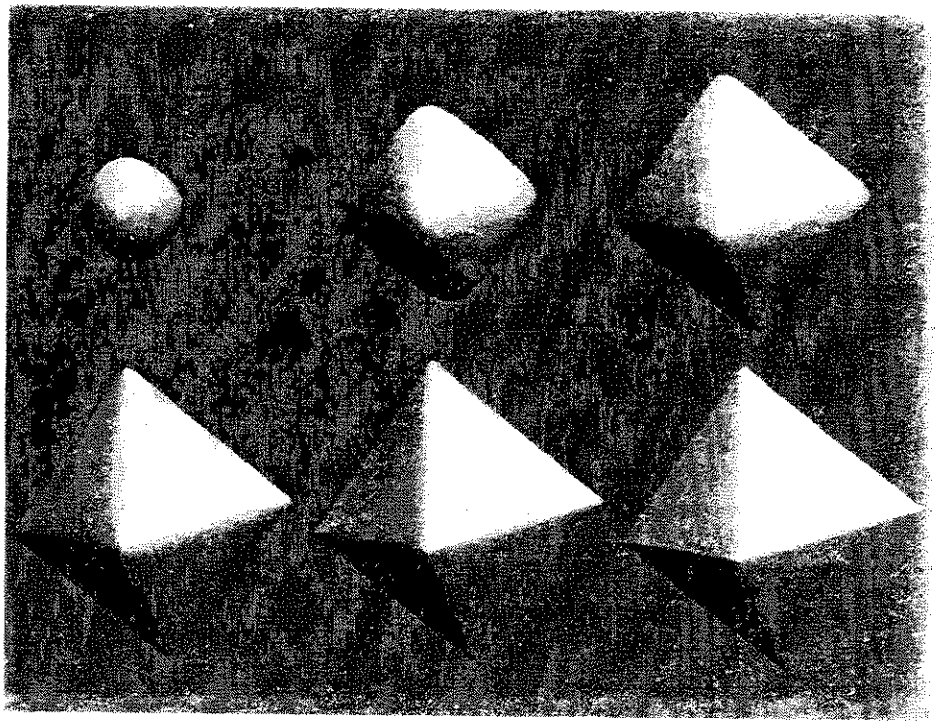

Fig. 3

$$F_i(s) = \begin{cases} s^{q_i} & s > 0 \\ 0 & \text{elsewhere} \end{cases}$$

produces a suitable blending effect. We have also used

$$F_i(s) = e^{(q_i(s-1))}$$

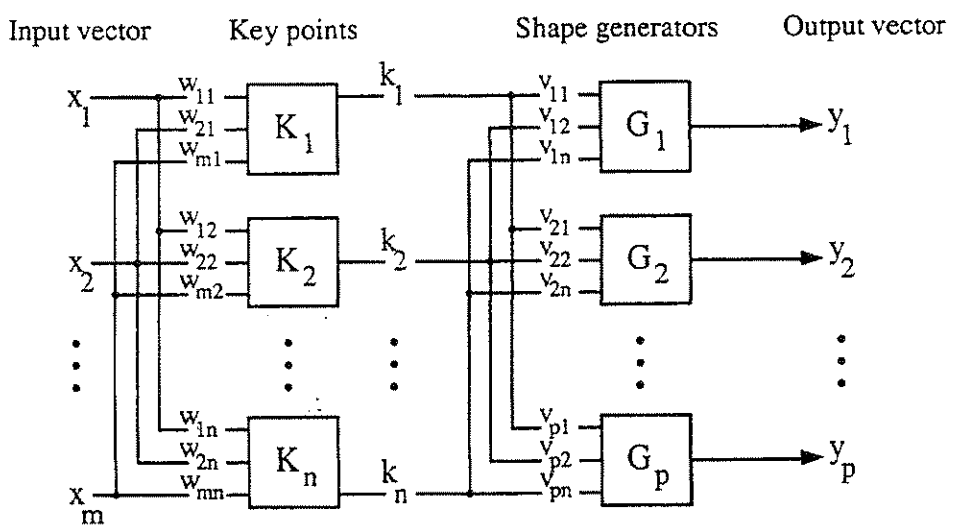

Fig. 4. The neural network form of the mapping function.

In each case, a suitable value of q_i must be chosen. The effect of varying this value is shown in Fig. 2 for the simple case where W is equal to V. The six target points lie at the points where the coordinate axes intersect the unit sphere and the value of q_i for each target point is the same. The six shapes correspond to values of q_i from 1 to 6. The value $q_i = 1$ produces the most rounded shapes, and $q_i = 6$, the most spiky.

It is also possible to use a rational blending function where

$$k_i = (\mathbf{x} \cdot \mathbf{w}_i) \Big/ \sum_{i=1}^{n} F_i(\mathbf{x} \cdot \mathbf{w}_i) \tag{6}$$

Figure 3 shows the effect of altering q_i in this case. The target points are the same as in Fig. 2. As the value of q_i increases, the shape becomes less round, approximating an octahedron.

5. Local Control

The functions of (4) and (5) above produce nonzero k_i values even when the points x and w_i are widely separated. Because of this, the whole surface is affected by a change in any w_i. If we want more local control of shape, we have to choose a function $F_i(s)$ that falls to zero at some particular value of $s = \mathbf{x} \cdot \mathbf{w}_i$. This is an area that needs further investigation. The surface could be generated more efficiently if only a subset of the key points were needed for each calculation.

6. Neural Network

We can visualize the mapping as the effect of an artificial neural network (Fig. 4). The key points and shape generators form two independent layers of nodes. The w and v vectors are then seen as weights for the inputs of these nodes. The Cao En surface was originally conceived as something produced by a neural network (Hertz 1991) and the visualization is so convenient that we have come to refer to the mapping and associated algorithm as Can En's network or simply the network.

From this visualization, we can see that a Cao En surface could be produced efficiently with parallel hardware.

7. Surface Normals

For rendering purposes we need to calculate the normal at any point of a Cao En surface. For this calculation, we consider only the simple case of mapping a sphere to a surface embedded in 3D space. In polar coordinates, a point $\mathbf{x} = (x_1, x_2, x_3)$ on the surface of a sphere may be represented by the angles θ, ϕ, where

$$x_1 = \cos(\theta)\,\sin(\phi),\quad x_2 = \sin(\theta)\,\sin(\phi),\quad x_3 = \cos(\phi) \tag{7}$$

Small movements, $\delta\theta$ and $\delta\phi$, lie in the surface of the sphere and the corresponding vectors $\delta y/\delta\theta$ or $\delta y/\delta\phi$ lie in the target surface. The desired normal is therefore in the direction of the cross product

$$\mathbf{N} = \frac{\delta y}{\delta\theta} \times \frac{\delta y}{\delta\phi} \tag{8}$$

The components of (8) are easily calculated, e.g.

$$\frac{\delta y_1}{\delta\theta} = \frac{\delta y_1}{\delta x_1}\frac{\delta x_1}{\delta\theta} + \frac{\delta y_1}{\delta x_2}\frac{\delta x_2}{\delta\theta} + \frac{\delta y_1}{\delta x_3}\frac{\delta x_3}{\delta\theta} \tag{9}$$

and

$$\frac{\delta x_1}{\delta\theta} = -\sin(\theta)\,\sin(\phi) \tag{10}$$

and so forth.

Using (5) for our F_i, we can write

$$y_g = \sum_{j=1}^{n} v_{gj}\, e^{q_j\left(\sum_{i=1}^{3} x_i w_{ij} - 1\right)},\quad g = 1, 2, 3 \tag{11}$$

Differentiating with respect to x_i we have

$$\frac{\delta y_g}{\delta x_h} = \sum_{j=1}^{n} v_{gj}\, q_j\, w_{hj}\, e^{q_j\left(\sum_{i=1}^{3} x_i w_{ij} - 1\right)},\quad h = 1, 2, 3 \tag{12}$$

Using (8), (9), (10), (12), etc., we can calculate N for any point on the surface. If we start from the vector x, we can avoid trigonometric functions since

$$\sin(\theta) = \frac{x_2}{\sqrt{x_1^2 + x_2^2}}\ \text{etc.} \tag{13}$$

Notice that in (12), q_j, w_{hj} are constants. This means that $\delta y_g/\delta x_h$ can be calculated from the same network by replacing each v_{jg} by $v_{jg}q_j w_{hj}$.

In many cases, we can avoid an exact calculation of the normal. If, for example, we calculate the normal for the key points ($\mathbf{x} = \mathbf{w}_i$), we can use each of these normal vectors in place of the target vectors, $\mathbf{t}_i$. The modified network will calculate an approximation to the surface normal for any input vector x, using the same algorithm that calculates the shape.

8. Projection onto a View Plane

Projections of a Cao En surface can also be made directly with the original network. Let $\mathbf{z} = (z_1, z_2, \ldots z_p)$ be a vector in the target space. Then the component of y projected onto z is

$$\mathbf{y}.\mathbf{z} = \sum_{j=1}^{p} y_j z_j = \sum_{j=1}^{p}\sum_{i=1}^{n} k_i v_{ji} z_j \tag{14}$$

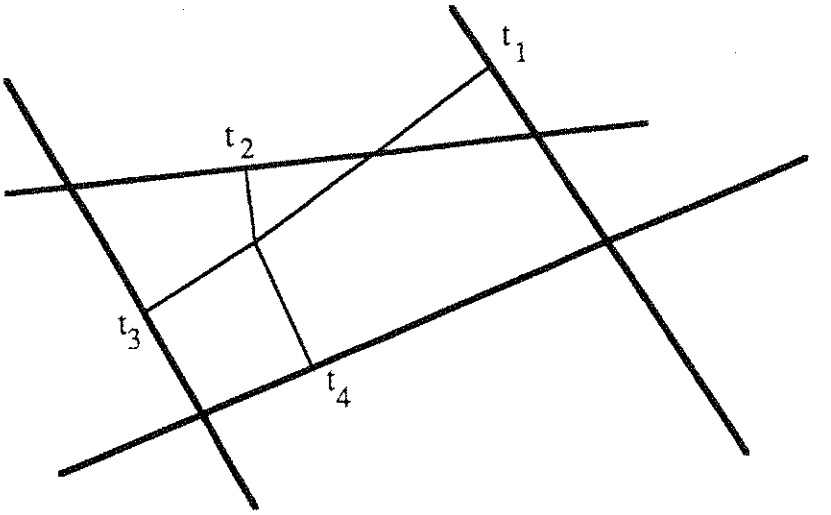

Fig. 5. Implicit surface construction. The thick lines represent the hyperplanes, the thin lines represent the perpendiculars.

Reordering this sum, we have

$$\mathbf{y}.\mathbf{z} = \sum_{i=1}^{n} k_i \sum_{j=1}^{p} v_{ji} z_j \tag{15}$$

and for each i we can pre-calculate

$$v_{iz} = \sum_{j=1}^{p} v_{ji} z_j \tag{16}$$

This amounts to saying that we can construct a new shape generator G_z, whose inputs are v_{iz} and whose output is $\mathbf{y} \cdot \mathbf{z}$, the projection of y onto an arbitrary axis.

Similarly, we can construct a pair of generators G_α, G_β, where α, β are vectors representing the perpendicular axes of a plane coordinate system. The same network calculates the projected coordinates. If we choose the α, β plane as our screen coordinates, then we produce the final view directly from the network.

In the case of the perspective transformation, things are a little more complicated. We can, of course, create a true perspective projection using homogenized coordinates but that requires an extra division when each screen point is eventually displayed or processed. A neater solution is to calculate the screen coordinates for each key point and use these in place of the original t_i vectors. The network then interpolates these values to give an approximation to the true perspective projection of each y.

9. Implicit Form

Like other parametric surfaces, the *general* Cao En surface can intersect itself, turn inside out, and exhibit cusps and other peculiarities. For this reason, it cannot be transformed into an implicit functional form such that every point in space can be classified, uniquely, as inside, outside, or on the surface.

In certain special cases, however, it is possible to construct an implicit form. The following construction works for convex surfaces.

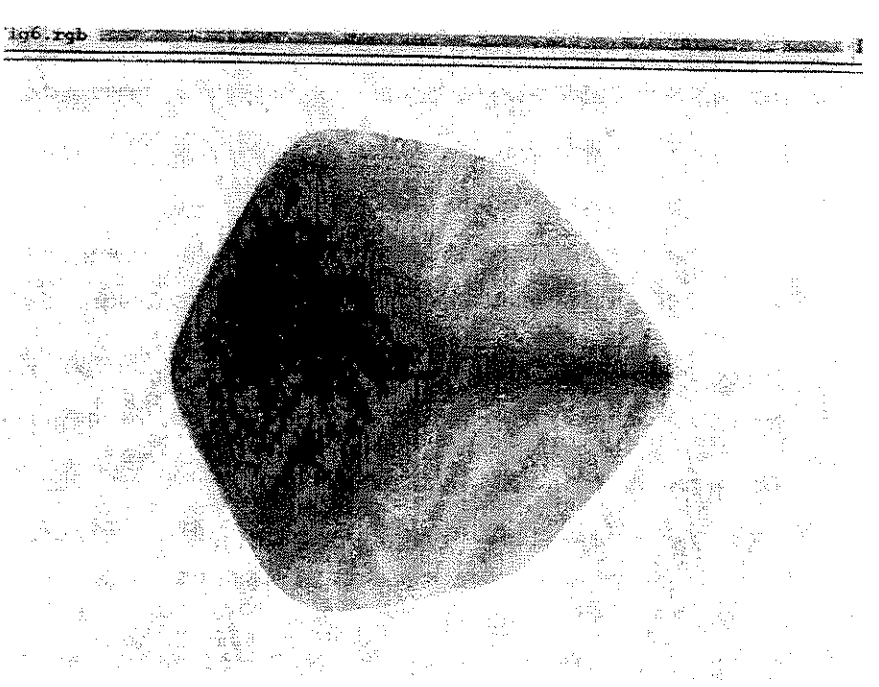

Fig. 6

The same mapping is used as for the ordinary parametric surface, and the same network describes it. But instead of mapping from a unit hypersphere onto a freeform surface, we are mapping from one hyperspace to another. Each source point, $x = (x_1, x_2, ...x_m)$, maps to a target point $y = (y_1, y_2, ...y_p)$, and the point x is considered to be inside the surface if and only if y is inside the hypersphere $y^2 = 1$. We assume that $m \geq p$. For simplicity, p can equal 1 and the inside/outside test is reduced to the comparison $y_1 \leq s$; s is any number greater than 1.0.

Assume that the desired shape is approximated by a convex polyhedron and that the perpendicular from some center point to each bounding hyperplane meets that hyperplane at the target point. Figure 5 shows the construction. Let the points where these perpendiculars intersect the hyperplanes be represented by vectors t_i and define the blending matrix W by

$$w_i = t_i / t_i^2 \tag{17}$$

If the output points are chosen to lie on the unit hypersphere, then the same network will calculate the implicit form of the surface.

A more general implicit form would be very valuable, but we do not yet know how to characterize the subset of Cao En surfaces that can be expressed implicitly.

10. Rendering Methods

To date, we have used three methods of rendering Cao En surfaces in 3D.

10.1. Ray Tracing

For the special case of convex surfaces, the implicit form can be ray traced using ray marching (Perlin 1989). A bounding sphere is placed around the surface and the two intersection points with this sphere are found. Somewhere between these points an intersection may occur. A binary search is made along the ray between the sphere intersection points in an attempt to find a point inside the surface. The search examines the mid-point of the range. If this is outside, it divides the range into two halves and searches these recursively until an inside point is found or the range of search is below some limit. If an inside point is detected, then the intersection points can be found by means of an ordinary bisection algorithm along the way. This finds all intersection points because the implicit surfaces are strictly convex and can have only two intersections. By finding both intersection points we can use the Can En surfaces with CSG (Roth 1982). Figure 6 shows such a ray traced surface.

Fig. 7

10.2. Polygon Mesh

We cover the original sphere with a fine polygon mesh and map each polygon vertex to its corresponding point on the Cao En surface. The result is a polygon mesh that approximates the desired surface, and it can be rendered by standard methods. This approach is not intended for final rendering but it enables us to see models quickly using the graphics pipeline in a Silicon Graphics Personal IRIS computer. The teapot in Fig. 7 was generated with polygons from four Cao En surfaces. The original teapot data contained only 104 target points. It is worth pointing out that the lid and body of the teapot are not surfaces of revolution. They are complete Cao En surfaces and the key points have only sixfold symmetry.

10.3. Depth Buffer

A depth buffer is stored for the screen and a set of sample points is defined on the sphere. For each sample point, the network is applied to create a point on the Cao En surface. The depth buffer is updated if the depth associated with the new point is less than that stored for its pixel. In the case where a pixel is "skipped" between adjacent sample points, an intermediate sample point is generated. This process is not very efficient and further work on a scanline renderer is needed.

11. Alternative Topology

The Cao En surface, as defined in Section 2, might better be called the spherical Cao En surface because it is topologically equivalent to a sphere, or hypersphere. It is also possible to use the same network to define surfaces with other topology. For example, a point on a torus can be described by a pair of angles θ, ϕ as shown in Fig. 8. And if we are avoiding explicit trigonometric functions, we can represent such a point as a vector $\mathbf{x} = (x_1, x_2, x_3, x_4)$, where

$$x_1 = \cos(\theta),\ x_2 = \sin(\theta),\ x_3 = \cos(\phi),\ x_4 = \sin(\phi) \tag{18}$$

The dot product $\mathbf{x} \cdot \mathbf{w}_i$ provides a measure of distance between points $\mathbf{x}$ and $\mathbf{w}_i$ and we can use the same network to generate surfaces topologically equivalent to a torus. The target points would, of course, be defined in 3D. Similarly, we can construct special coordinate systems for other topologies. Because the Cao En surface is equally well defined in any number of directions, this is not difficult.

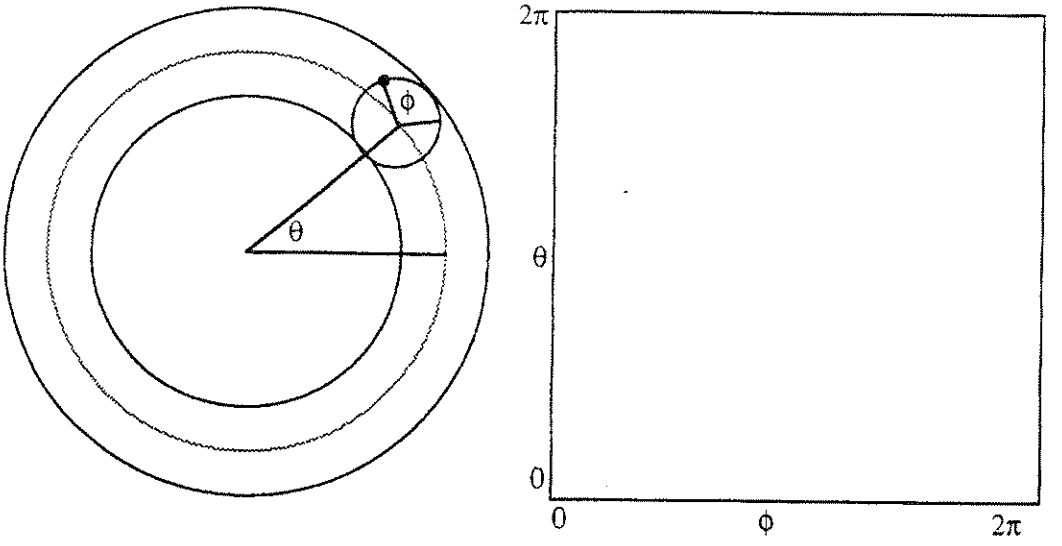

Fig. 8. A point ● on the torus is defined by angles ϕ and θ.

It is also possible to define a mapping that is independent of the surface topology. If you project the surface of a sphere onto a plane, you produce a circle in which each point represents two points on the original surface. Similarly, you can project the surface of a 4D hypersphere onto a 3D space and create a sphere within which every point represents a pair of points on the original hypersurface.

Any surface drawn inside this sphere can be used as an input surface for the network and it can be used to generate a corresponding freeform surface in 3D.

12. Applications

The Cao En surface was conceived as a way to describe freeform shapes in 3D. The network, however, defines a very flexible and general transformation function capable of many other applications. Here we list a few that we are actively studying.

12.1. Metamorphosis

When a metamorphosis is described in an animation sequence, it is necessary and usually difficult to specify the correlation between points on the changing objects.

If, however, two objects are described as Cao En surfaces, they are already related because they are both derived from a sphere. The transformations can be effected simply by allowing the various $\mathbf{w}_i$ and $\mathbf{t}_i$ values to change from the one shape into the other. Even if one model has more key points than the other, this can still be done. Any key point for which the elements of the corresponding $\mathbf{v}_i$ are all zero has no effect on the surface. So these "empty" key points can be changed into the extra ones required.

12.2. Distortion

If the network is used as described in Section 11 to map from the surface of a 4D hypersphere into 3D, it actually describes a distortion function in 3D. Such a function has an arbitrary number of controls.

12.3. Direct Reflection and Refraction

Just as the network will calculate surface normals directly, it can be used to calculate the direction of a reflected or refracted ray directly. The incident ray direction becomes part of the input vector. Similarly, lighting calculations based on the cosine law can be performed by the network so that the output can be a shaded, projected image.

12.4. Texture Mapping

Clearly, any texture that can be applied to the original sphere can be mapped automatically onto the target surface. In some cases, we can make almost the same shape using different key point positions w_i but corresponding to the same shape vectors t_i. This implies that we are mapping different parts of the original sphere to the same parts of the freeform surface. If the sphere is textured, then we have a means to decide how and where the texture will be stretched to fit the target surface.

Another possibility is to use one network to create the shape and another to create a shape within a 3D texture space. Since both are mapped from the sphere, the texture sampled in one space can be applied to the other. This sort of approach could provide a high degree of control of the way a texture stretches.

12.5. Image Compression

The Cao En surface can be used for image compression. A bit map is a collection of scalar values R, G, B. The network can be used to store a collection of R, G, B triplets from the screen and then reconstruct the values between these points.

The target points are specified as (x, y, R, G. B). These are mapped to points on the surface of a 5D hypersphere. Then the network is evaluated for any point on the surface of the hypersphere x, y, and the R, G, B values of the original bit map are calculated.

The choice of (x, y, R, G, B) is very important. The network will interpolate between the key points. If points on color boundaries are given, the network can be used to reconstruct the smooth regions of the image bounded by the key points. In regions of high frequency more target points should be used than in areas of low frequency. The density of target points at any area in the image will depend on the complexity of that area of the image. This density is ruled by the Shannon (1948) sampling theorem.

The target points can be saved to disk as a representation of the bit map. If very few key points are specified, the image reconstruction will be very bad. With more target points the reconstruction will be better. The optimal number of target points is frequency band limited. At some limit, adding more target points will not improve the image. The network can be used to anti-alias a bit mapped image by evaluating the image at sub-pixel positions.

13. Discussion

The Can En surface is built in the simplest way; every point on the surface is expressed as a weighted average of the target shape vectors t_i. Other parametric shapes, such as Bezier patches, are often expressed as a weighted sum of cubic base functions (Foley 1990). Similarly, Soft objects (Wyvill 1986) are blends of spheres. In both cases, the blending and the geometry are defined together. The Cao En surface provides a modelling scheme in which the blending characteristics and the shape of information are encoded in separate layers of a network. It is this separation of blending and geometry that makes it so versatile.

14. Conclusion

A new technique for geometric modelling of freeform surfaces has been developed. We have named it the Cao En surface, after its inventor.

Cao En surfaces provide a simple, potentially parallel algorithm for generating a parametric surface of arbitrary complexity and any topology. They avoid the need for surface patches.

The Cao En surface is generated by an algorithm represented by an artificial neural network. The same network can generate projections and surface normals. Even texture mapping and lighting calculations can be folded into the network process.

For convex surfaces, an implicit form can be created from the same network. This is potentially useful in ray tracing and CSG applications.

15. Acknowledgment

The photographic illustrations were prepared with the help and support of the Computer Graphics Laboratory, EPFL, Lausanne, Switzerland.

LITERATURE CITED

Blinn, J. (1982). "A generalization of algebraic surface drawing," *ACM Trans. Graphics* **1**, 235-256.

Foley, J., et al. (1990). *Computer Graphics Principles and Practice*, Addison-Wesley, Redwood City, California.

Hertz, J., Krogh, A., and Palmer, R. (1991). *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City, California.

Nishimura, H., Hirai, M., Kawai, T., Kawata, T., Shirakawa, I., Omura, K. (1985). "Object modeling by distribution function and a method of image generation," *Papers given at the Electronics Communication Conference '85* J68-D (4) (in Japanese).

Perlin, K. (1989). "Hypertexture," *Comput. Graphics* **23** (3), 253-26.

Roth, S. D. (1982). "Ray casting for modeling solids," *Comput. Graphics Image Process.* **18**, 109-144.

Shannon, C. E. (1948). "A mathematic theory of communication," *Bell Systems Tech. J.* **27**, 379-425.

Wyvill, G., McPheeters, C., and Wyvill, B. L. M. (1986). "Data structure for soft objects," *The Visual Comput.* **2** (4), 227-234.

Wyvill, G. and Trotman, A. (1990). "Ray tracing soft objects," Proceedings of CG International 90, pp. 469-476.

ANIMATED ROTATIONS USING QUATERNIONS AND SPLINES ON A 4D SPHERE

G. M. Nielson and R. W. Heiland

A new method for animating rotations is presented. Given a sequence of orientations of an object, this method allows for the computation of intermediate positions which represent a smooth transition from one to the next. The method uses the quaternion representation of a rotation matrix and a new curve interpolation scheme for points on a four-dimensional sphere.

1. Introduction

Given a sequence of orientations as shown in Fig. 1, it is often desirable to find intermediate orientations which represent smooth transitions from one orientation to the next. The orientation of an object relative to a particular coordinate system can be represented by a rotation matrix

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{1.2}$$

where the orientation is achieved by matrix multiplication. That is,

$$\{\text{oriented object}\} = \{\text{object}\}R.$$

Computer Science, Arizona State University, Tempe, Arizona 85287-5406. Published in Programmirovanie, No. 4, pp. 17-27, July-August, 1992. Original article submitted June 1, 1992.

0361-7688/92/1804-0145$12.50