Vol. 5, No. 2: 00–00

Three-Dimensional Deformation Using Directional Polar Coordinates

Xiaogang Jin
State Key Lab of CAD & CG, Zhejiang University

Y. F. Li
City University of Hong Kong

Abstract. Space deformation is an important tool in both computer animation and shape design. In this paper, we present a new three-dimensional deformation method using directional polar coordinates. The user specifies the source control object and the destination control object which act as the embedded spaces. The source and the destination control objects determine a three-dimensional volume morphing which maps the space enclosed in the source control object to that of the destination control object. By embedding the object to be deformed into the source control object, the three-dimensional volume morphing determines the deformed object automatically without the cumbersome moving of control points. Experiments show that this deformation model is efficient and intuitive, and it can achieve some deformation effects which are difficult with traditional methods.

1. Introduction

Space deformation plays an active role in both geometric modeling and computer animation [Sederberg, Parry 86], [Coquillart 90], [MacCracken 96], [Singh, Fiume 98], [Jin et al. 00]. Free-form Deformations (FFDs) deform objects by embedding them in a control mesh, then moving the points of the control mesh [Sederberg, Parry 86]. Essentially, changing the mesh defines a morph of a volume of space, and the vertices of the model get carried along with it. Although FFD-based methods are flexible, they require the user to define and move control points, which can be cumbersome.

1086-7651/00 $0.50 per page

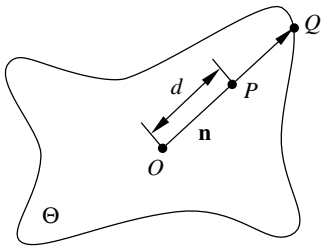


Figure 1. The definition of directional polar coordinates for a star-shaped object.

In this paper, we describe an alternative way to define the spatial morph. The user embeds the model in any star-shaped source object, and specifies any other star-shaped target object. We use *directional polar coordinates* to define a morph between any two star-shaped control objects. This morph is similar to the work of Kent et al. in deforming polygonal objects [Kent et al. 92].

2. Volume Morph for Star-Shaped Objects

An object Θ is *star-shaped* (also known as *star-convex*) if it contains a "center point" o such that any ray from the center point intersects the surface exactly once. For convenience, we represent a star-shaped object as $< O, \Theta >$. So, for any point P inside the object, we can represent the point by $(\mathbf{n}, d)$: the direction $\mathbf{n}$ of the ray to it from the origin, and the fraction d of its distance along the ray to the surface (see Figure 1). We call $(\mathbf{n}, d)$ the directional polar coordinates of point P. To transform the point P, we simply evaluate $(\mathbf{n}, d)$ with respect to the target object; that is, we construct the point along the direction $\mathbf{n}$ from the target center, and a fraction d along the ray to the target surface.

Let $< O_S, \Theta_S >$ and $< O_D, \Theta_D >$ be the source and the destination control star-shaped objects with center points O_S and O_D respectively (see Figure 2), then for a given point P, the deformation algorithm is:

Compute direction $\mathbf{n}$ *from* O_S *to* P*;*
Compute distance l_s *from* O_S *to* P*;*
Intersect ray from O_S *in direction* $\mathbf{n}$ *with source control surface,*
to get a point Q_S*;*
Compute distance L_S *from* O_S *to* Q_S*;*
Compute $d = l_S / L_S$*;*
Intersect ray from O_D *in direction* $\mathbf{n}$ *with destination control surface,*
to get a point Q_D*;*

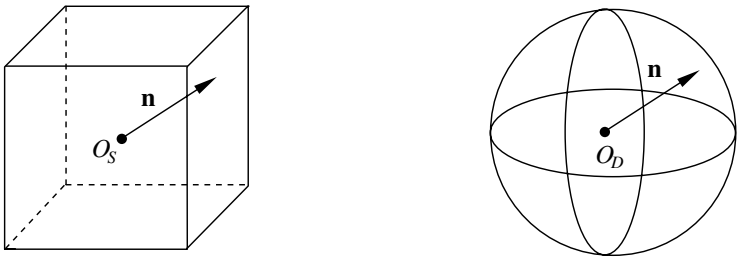


Figure 2. (a) Source control star-shaped object. (b) Destination control star-shaped object.

Compute distance L_D *from* O_D *to* Q_D*;*
Transform point P *to* $P' = O_D + \mathbf{n} * d * L_D$ *;*

A cylindrical star-shaped object Θ has an axis line CL, and any perpendicular cut along the line intersects the surface in a two-dimensional star-shaped object. A cylindrical star-shaped object is represented as $< \Theta, CL >$. For cylindrical star-shaped objects, we can represent a point using a triple $(r, \mathbf{n}, d)$, where r is the fraction along the axis, $\mathbf{n}$ is the two-dimensional direction, and d is the fraction of distance to surface (see Figure 3). The deformation algorithm for cylindrical star-shaped objects is almost the same as above. To transform the point P, we similarly evaluate $(r, \mathbf{n}, d)$ with respect to the target object. That is, we construct the point along the direction $\mathbf{n}$ from the center point which is a fraction r along the target axis line, and a fraction d along the ray to the target surface.

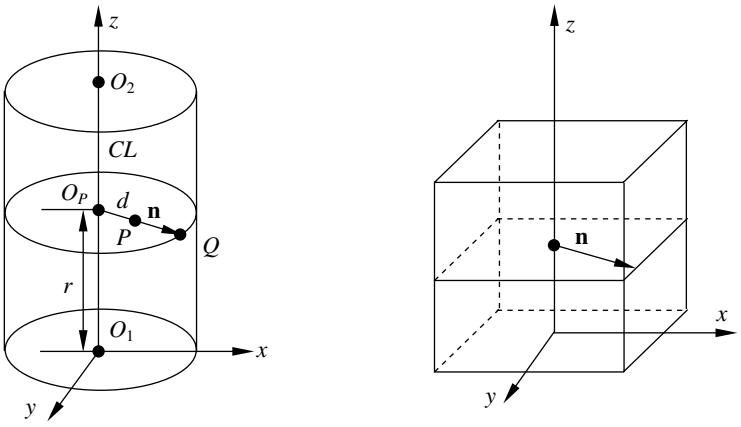


Figure 3. (a) Source cylindrical star-shaped object. (b) Destination cylindrical star-shaped object.

To animate the deformation from source to target shape, we interpolate the representation between the source and target spaces. That is, for the spherical objects,

$$P' = ((1-t) * O_S + t * O_D) + \mathbf{n} * d * ((1-t) * L_S + t * L_D)$$

and similarly for the cylindrical objects.

3. Computing the Intersection of the Ray with Each Surface

The key to the deformation model lies in the calculation of the ray–object intersection. If the control object is a polyhedral object which is commonly adopted in our implementation, the ray–object intersection can be accelerated by the light buffer technique [Arvo, Kirk 89], [Haines, Greenberg 86]. Quadric, parametric, and implicit surfaces can be broken down into polygons for unified processing. The light buffer, introduced by Haines et al. tries to accelerate the calculation of shadows with respect to point light sources. In our deformation model, the center point of the star-shaped object replaces the position of the point light source in the light buffer technique. We first associate a uniformly subdivided direction cube with the control object (see Figure 4), and generate the complete list of candidate polygons with each of the direction cells. Each candidate list contains every polygon which can be seen through the corresponding direction cell. The candidate lists are retrieved by finding the direction cell pierced by each semi- infinite ray originated from the center point of the control object. The polygons in the list are the only ones which will intersect the semi-infinite ray. According to the definition of the star-shaped object, a semi-infinite ray cast from the center point of the control object will intersect one and only one polygon of the control object. By using this fact, it is easy to know that the length of the list in the light buffer is usually very short, and the intersection calculation between a semi-infinite ray and the candidate polygons in the candidate list can be terminated as

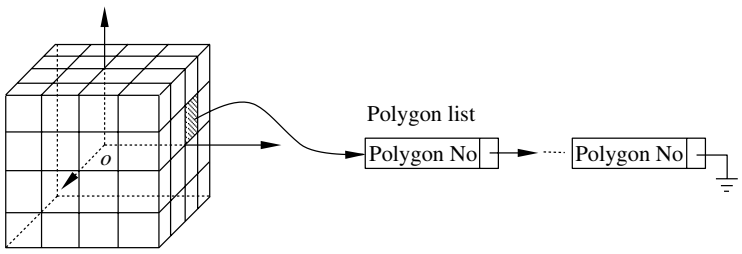
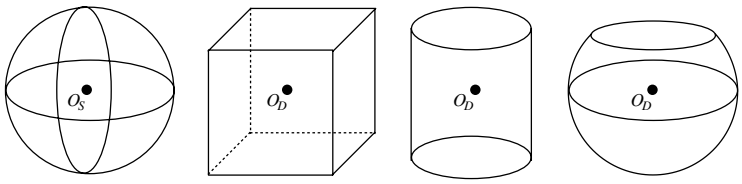


Figure 4. Accelerate ray–object intersection by direction cube with uniform subdivision.

soon as one intersection has been found. For the ray–polygon intersection, we adopt the intersection algorithm introduced by Badouel [Badouel 90], [Moller, Trumbore 97]. The direction cubes are constructed as a preprocessing step. The candidate lists are created by projecting each polygon of the control object onto the six faces of the direction cube, adding them to the candidate lists of those direction cells which are partially or totally covered by the projection. This projection can be performed by a modified scan-line algorithm to the projected edges. By adopting the direction cube, ray–object intersection can be performed efficiently.

4. Examples

We implemented our algorithm on a PII-300 PC under OpenGL environment. All the embedded objects are polygon meshes represented in Wavefront format. Figure 6 shows the deformation of a soccer ball using the control star-shaped objects shown in Figure 5. By animating the deformation from source to target shape, we obtain Figure 8. Figure 10 and Figure 11 are deformation examples using control cylindrical star-shaped objects shown in Figure 9.



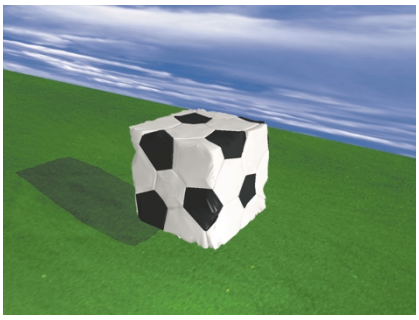
(a) Sphere (b) Cube (c) Cylinder (d) Revolution object

Figure 5. Source control star-shaped object (a) and destination ones (b), (c), (d).

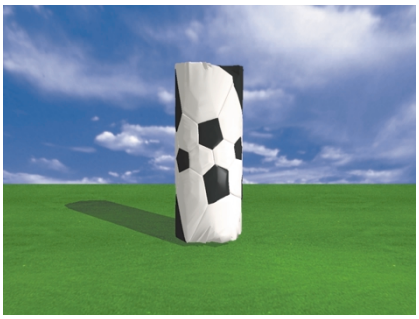
5. Discussion

The control objects can be any star-shaped objects: polyhedral, swept surfaces, surfaces of revolution, quadrics, blobs, parametric surfaces, and so forth. For simple objects like spheres, cylinders, and cones, we use the algebraic representation directly to perform the intersection [Hanrahan 89]. Many star-shaped objects have more than one center point (or line). For example, any interior point of a sphere can be used as the center. The user can control the deformation by specifying the center point. As with other deformation techniques, the user can of course apply the deformation to the entire object, or to just a local region to generate bumps and dents, or select a collection of

(a) Original soccer ball without deformation.



(b) Deformed soccer ball by sphere-to-cube morphing.



(c) Deformed soccer ball by sphere-to-cylinder morphing.

(d) Deformed soccer ball by sphere-to-revolution object morphing.

Figure 6. The deformation of a soccer ball.

points to define a cluster and apply the deformation only to the points in the cluster.

This technique has an advantage over control-mesh-based FFD techniques in that it lets the user specify the control shapes directly and intuitively, instead of by moving control vertices and can achieve some deformations that are difficult to express using other techniques. The disadvantage of the technique is that it is of course limited to control objects that are star-shaped. It also does not provide local control over the deformation, so it may be hard for users to achieve a specific target model shape that they have in mind. Thus, this technique complements rather than replaces other deformation techniques. Finally, as with other techniques, for extreme deformations, such as control shapes with sharp edges, the model will need to be adaptively subdivided to avoid aliasing.

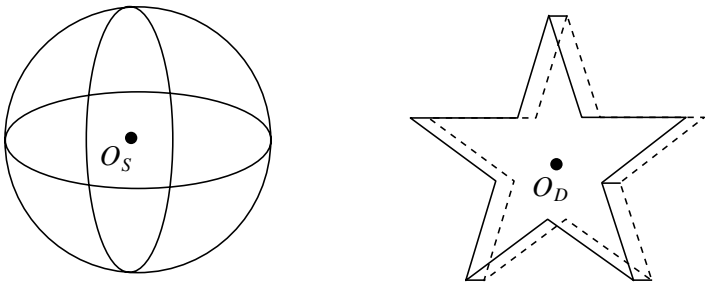


Figure 7. Source and destination control star-shaped objects.

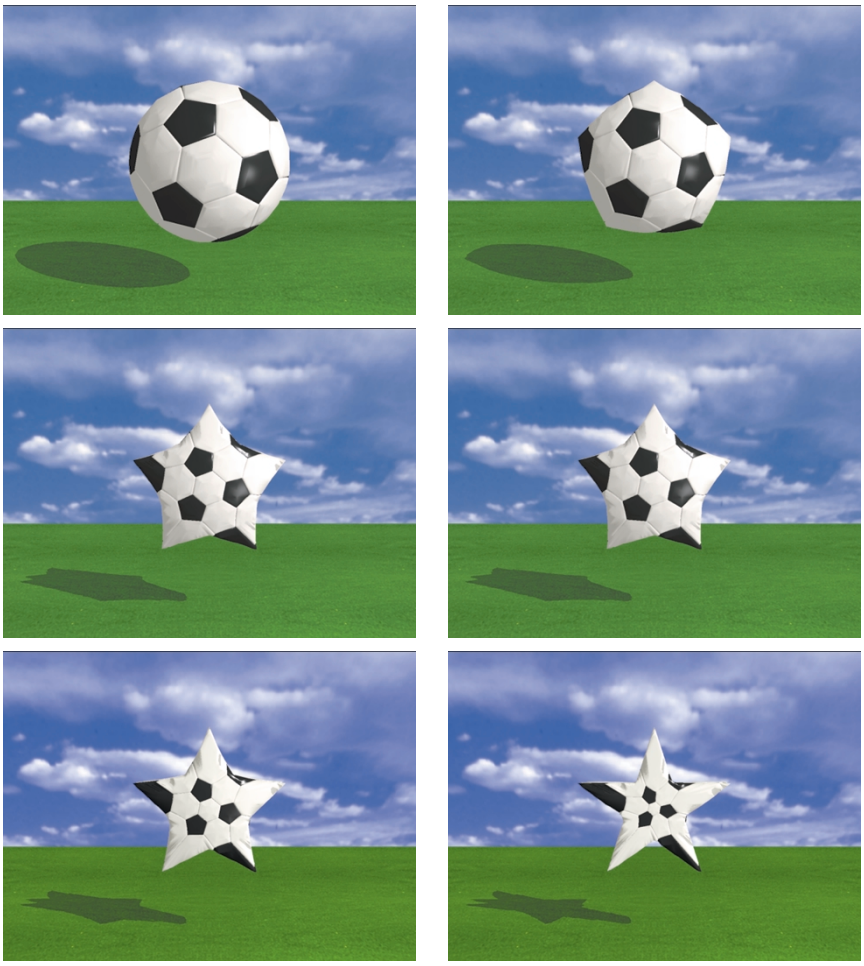


Figure 8. By applying the control objects shown in Figure 7 and animating the deformation model, we obtain this soccer-morphing sequence.

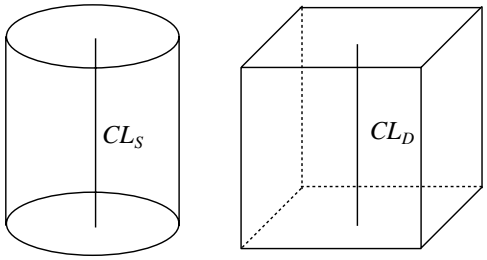
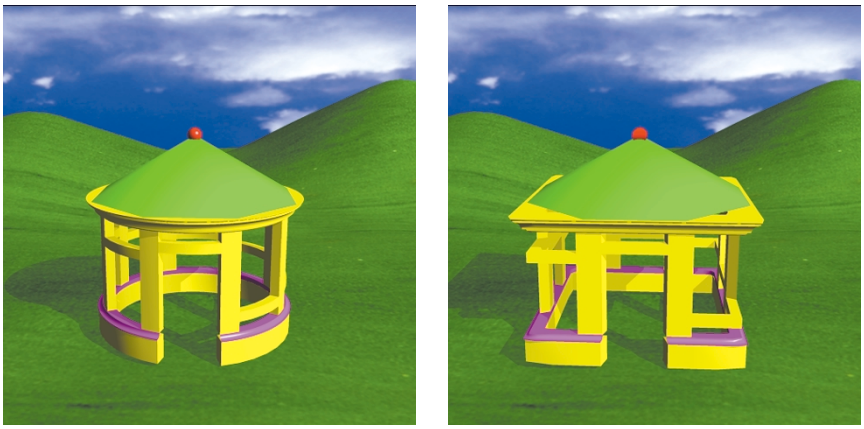


Figure 9. Control cylindrical star-shaped objects.

Figure 10. Deformed soccer ball using cylinder-to-cube morphing.



(a) Original rotunda. (b) Deformed rotunda.

Figure 11. The deformation of a rotunda using cylinder-to-cube morphing.

Acknowledgements This work received support from the National Natural Science Foundation of China (Grant No. 69973040), Zhejiang Provincial Natural Science Foundation (Grant No. 698022), and the Research Grants Council of Hong Kong (Project No. 9040309). We are grateful to Dr. Huagen Wan and Dr. Jieqing Feng for their help on this paper. Special thanks go to Dr. Ronen Barzel for his constructive suggestions on how to improve the presentation of the paper.

References

[Arvo, Kirk 89] J. Arvo and D. Kirk. "A Survey of Ray-Tracing Acceleration Techniques." In *An Introduction to Ray Tracing*, edited by A.S. Glassner, pp. 201–262, London: Academic Press Limited, 1989.

[Badouel 90] D. Badouel. "An Efficient Ray-Polygon Intersection." In *Graphics Gems*, edited by A.S. Glassner, pp. 390–393, Boston: Academic Press Inc., 1990.

[Coquillart 90] S. Coquillart. "Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling," *Computer Graphics (Proc. SIGGRAPH '90)*, 24(4): 187–196 (1990).

[Haines, Greenberg 86] E. Haines and D. Greenberg. "The Light Buffer: A Ray Tracer Shadow Testing Accelerator," *IEEE Computer Graphics & Applications*, 6(9): 6–16 (1986).

[Hanrahan 89] P. Hanrahan. "A Survey of Ray-Surface Intersection Algorithms." In *An Introduction to Ray Tracing*, edited by A.S. Glassner, pp. 79–119, London: Academic Press Limited, 1989.

[Jin et al. 00] X. Jin, Y. Li and Q. Peng. "General Constrained Deformations Based on Generalized Metaballs." *Computers & Graphics*, 24(2): 219–231 (2000).

[Kent et al. 92] J. Kent, W. Carlson and R. Parent. "Shape Transformation for Polyhedral Objects." *Computer Graphics (Proc. SIGGRAPH '92)*, 26(2): 47–54 (1992).

[MacCracken 96] R. MacCracken and K. Joy. "Free-Form Deformations with Lattices of Arbitrary Topology", In *Proceedings of SIGGRAPH 96, Computer Graphics Proceedings, Annual Conference Series*, edited by Holly Rushmeier, pp. 181–188, Reading, MA: Addison Wesley, 1996.

[Moller, Trumbore 97] T. Moller and B. Trumbore. "Fast, Minimum Storage Ray-Triangle Intersection," *journal of graphics tools*, 2(1): 21–28 (1997).

[Sederberg, Parry 86] T. Sederberg and S. Parry. "Free-Form Deformation of Solid Geometric Models," *Computer Graphics (Proc. SIGGRAPH '86)*, 20(4): 151–160 (1986).

[Singh, Fiume 98] K. Singh and E. Fiume. "Wires: A Geometric Deformation Technique. In *Proceedings of SIGGRAPH 98, Computer Graphics Proceedings, Annual Conference Series*, edited by Michael Cohen, pp. 405–414, Reading, MA: Addison Wesley, 1998.

Web Information:

http://www.acm.org/jgt/papers/JinLi00

Xiaogang Jin, State Key Laboratory of CAD &CG, Zhejiang University, Hangzhou, 310027, P.R. China. (jin@cad.zju.edu.cn)

Y. F. Li, Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Kowloon, HongKong. (MEYFLI@cityu.edu.hk)

Received November 11, 1999; accepted in revised form June 23, 2000.