A Generalized de Casteljau Approach to 3D Free-form Deformation

Yu-Kuang Chang and Alyn P. Rockwood

Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287-5406

ABSTRACT

This paper briefly presents an efficient and intuitive 3D free-form deformation approach based on iterative affine transformations, a generalized de Casteljau algorithm, whereby the object warps along a Bézier curve as its skeleton.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - Curve, surface, solid, and object representations; Hierarchy and geometric transformations.

Additional Key Words: The de Casteljau algorithm, affine transformation, Bézier curve, B-spline, geometric modeling, deformation.

1. INTRODUCTION

Free-form deformation (FFD) has become important tool in computer-assisted geometric design and animation. Barr first suggested a set of hierarchical transformations for deforming an object including stretching, bending, twisting, and tapering operators [1]. It is a very efficient and useable method if somewhat constrained. Sederberg and Parry [2] proposed a general technique for deformation which is based on trivariate Bernstein polynomials and enables the deformation of objects by manipulating control points (see also [3]). Another successful approach with an initial lattice and a scheme of B-spline control points that approximate the shape of the intended deformation is given by Coquillart [4].

We propose a technique that deforms by repeatedly applying affine transformations in space. The object warps along a user defined curve. Our approach reduces the definition of the free-form deformation from a crowded set of control points to a single Bézier curve and a few affine maps controlled by readily understood "handles." It also generalizes the well-known de Casteljau algorithm for curve evaluation [5].

2. PRELIMINARIES

We assume some familiarity with curves and surfaces (see Farin [5]). The de Casteljau Algorithm for evaluating a Bézier curve of degree n with control points p_i and at a parameter u is

$$p_i^j(u) = (1-u)\cdot p_i^{j-1}(u) + u\cdot p_{i+1}^{j-1}(u),\ 1 \le j \le n,\ 0 \le i \le n - j,$$

$$\text{where } p_i^0(u) = p_i, \quad 0 \le i \le n. \tag{1}$$

The value $p_0^n(u)$ is the point on the curve at u.

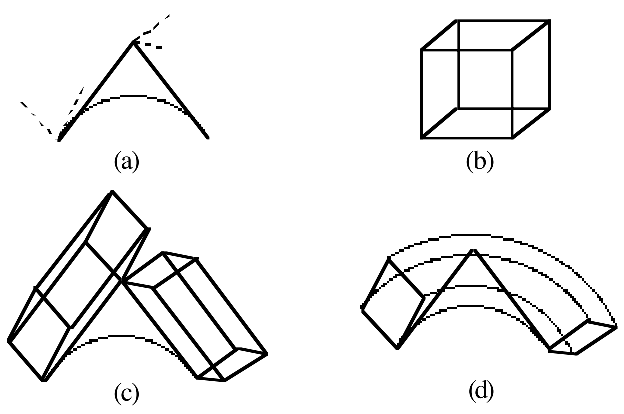

Figure 1: (a) Bézier curve, control polygon and user-specified axes, (b) cube primitive, (c) first-level execution of the generalized de Casteljau algorithm, and (d) second-level execution of the generalized de Casteljau algorithm.

Equation (1) consists of repeated univariate linear interpolation. Our approach generalizes the de Casteljau

1. TEL: (602)965-4154
Internet: ychang@enws120.cagd.eas.asu.edu
2. TEL: (602)965-8267
Internet: rockwood@asu.edu

algorithm to a trivariate scheme. Each segment of the Bézier control polygon is given with two user-specified axes, called *handles*, at one endpoint which defines a local coordinate system [2]. Objects in u-v-w space are mapped along an embedded Bézier curve by iterative affine transformations derived from the handles and control polygon segments. For example, Figure 1(a) shows a quadratic Bézier curve, control polygon, and two local axes on each segment. Figure 1(b) is a cube to be deformed. After executing the first level of the generalized de Casteljau algorithm, the cube is mapped affinely to each segment of the control polygon in Figure 1(c). In Figure 1 (d), the second level of the algorithm warps the original cube along the Bézier curve.

3. THE DEFORMATION ALGORITHM

Observe that in 3-space each iteration of the de Casteljau algorithm is just the execution of a degenerate affine transformation. It maps space to a line. Therefore the de Casteljau algorithm can be generalized to an iterative transformation scheme simply by raising the rank of the transformation matrix.

If vectors $\underline{r}$, $\underline{s}$, and $\underline{t}$ span an affine space, the function $\Phi[\underline{p},\underline{q}]$: $R^3 \rightarrow R^3$ is defined as an affine transformation from parameter space into affine space in homogeneous form as

$$\Phi[\underline{p},\underline{q}]\begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix} = \begin{pmatrix} q_x - p_x & s_x & t_x & p_x \\ q_y - p_y & s_y & t_y & p_y \\ q_y - p_y & s_z & t_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2}$$

where $\underline{r}=(q_x-p_x,q_y-p_y,q_z-p_z)$, $\underline{s}=(s_x,s_y,s_z)$, and $\underline{t}=(t_x,t_y,t_z)$. Note especially that the interval [0,1] on u is mapped to $[\underline{p},\underline{q}]$ on $\underline{r}$. The generalized de Casteljau algorithm is given by

$$\begin{aligned} &\underline{p}_i^0(\underline{u}) = \underline{p}_i, && 0 \le i \le n, \\ &\underline{p}_i^j(\underline{u}) = \Phi\left[\underline{p}_i^{j-1}, \underline{p}_{i+1}^{j-1}\right](\underline{u}), && 1 \le j \le n,\ 0 \le i \le n-j, \end{aligned} \tag{3}$$

where n is the degree and $\underline{p}_0^n(\underline{u})$ is the deformed point on an object at $\underline{u}$=(u,v,w). The deformation algorithm performs iterative affine transformations and as a result the deformation of space includes the Bézier curve.

In the first level of the generalized de Casteljau algorithm, if $\underline{s}$ in the transformation matrix is a zero vector, then a solid will be mapped into just a surface patch because the depth information is lost. The case where elements of vector $\underline{t}$ are all zero is similar except the degeneration occurs on a different axis. The deformation process is reduced to the classic de Casteljau algorithm if and only if both $\underline{s}$ and $\underline{t}$ are zero vectors. Under these circumstances, the solid is mapped to a curve since the parameter v and w are no longer effective.

For the second level or above, the vectors, $\underline{s}$ and $\underline{t}$, could be zero or nonzero. If they are all zero, the function of second-level execution linearly blends the result of the first-level execution and similarly for higher levels. If s_x, s_y, or s_z is nonzero, the deformed object is sheared by an amount proportional to the v value along the direction of $(s_x,0,0)$, $(0,s_y,0)$, or $(0,0,s_z)$. The same applies to vector $\underline{t}$ except that the shearing is proportional to w instead of v. The effects of the shearing operations are hierarchical. For instance, there are three levels of affine transformations in the cubic case. A nonzero vector $\underline{s}$ or $\underline{t}$ in second-level transformation matrix shears only a portion of the object related to the appropriate control polygon segments, while one in the third-level affects the whole object.

4. IMPLEMENTATION

Figure 2 shows examples of stretch, taper, swell, twist, and bend operations applied to a cube primitive (upper left). They mimic Barr's deformations, but are polynomial.

Figure 2: "Barr"-like deformations applied to a cube.

Our deformation algorithm can be applied to any geometric model. Figure 3 and Figure 4 show polygons before and after deformation. A figure similar to Figure 4 could also be generated with the Sederberg and Parry's approach. It would require specification of more, loosely related control points. Moreover, the generalized de Casteljau approach is computationally more efficient than trivariate de Casteljau, because it is an affine transformation plus a univariate interpolation in space vs. one that iterates over three variables. A comparison of computation times is given in Table 1.

Figure 3: Undeformed polygons.

Figure 5: The user interface for the free-form deformation.

Figure 4: Deformed polygons.

Figure 6: The design of a logo.

Deformation method \ Degree	Linear	Quadratic	Cubic
Bézier lattice method (Sederberg and Parry's approach)	42 multiplications 21 additions	90 multiplications 45 additions	144 multiplications 72 additions
Generalized de Casteljau method (Our approach)	9 multiplications 9 additions	27 multiplications 27 additions	54 multiplications 54 additions

Table 1: Comparison of computation complexity.

The design interface supports user two functions: First, specification of a curved spline by Bézier control points and second, definition of two local axes on each control polygon segment. These are intuitive and simple to use (see Figure 5).

Tangent continuity (C^1) between two volumes that share local coordinate systems at the endpoints is guaranteed if adjacent local systems are linearly dependent. Figure 7 shows two pieces of cubic Bézier curves which are joined smoothly (C^2 with B-splines, for instance). The axis triple ($\underline{r}_i$,$\underline{s}_i$,$\underline{t}_i$) is associated with the ith segment of control polygon where $\underline{r}_i = \underline{p}_{i+1} - \underline{p}_i$. At the junction point $\underline{p}_3$ of the consecutive Bézier curves, two volumes defined on those two curves are connected with C^1 continuity if the derivative vectors at the boundary are the same. By calculating partial derivatives (refer to [5]) at a point on the boundary, it induces the following constraints:

$$\underline{p}_3 = \tfrac{1}{2}(\underline{p}_2 + \underline{p}_4)$$

$$\underline{s}_3 = \tfrac{1}{2}(\underline{s}_2 + \underline{s}_4), \qquad (4)$$

$$\underline{t}_3 = \tfrac{1}{2}(\underline{t}_2 + \underline{t}_4).$$

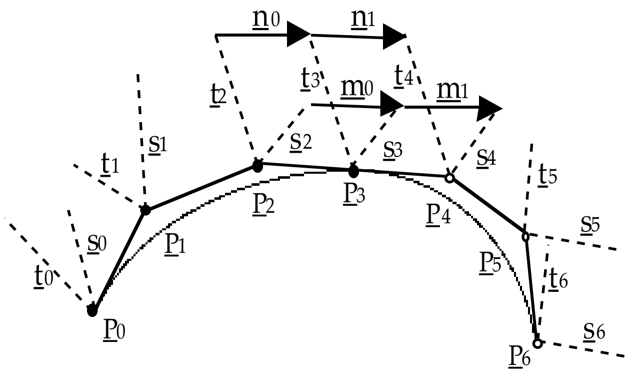

Figure 7: Local coordinate systems defined over piecewise Bézier curves.

Tangent direction continuity (G^1) requires only the collinearity of the cross-boundary derivatives. It results in

$$\underline{p}_4 - \underline{p}_3 = \mu_0(\underline{p}_3 - \underline{p}_2)$$

$$\underline{m}_1 = \mu_1 \underline{m}_0, \qquad (5)$$

$$\underline{n}_1 = \mu_2 \underline{n}_0,$$

where $\mu_i \neq 0$ for i=0 to 2. It is C^1 if $\mu_0=\mu_1=\mu_2=1$. If μ_0, μ_1, and μ_2 are all equal, then $\underline{s}_2$, $\underline{s}_3$ and $\underline{s}_4$ are linearly dependent as are $\underline{t}_2$, $\underline{t}_3$ and $\underline{t}_4$. However, linear dependence is only the sufficient condition for continuity. For example, if μ_0 and μ_1 differ, then $\underline{s}_3$ and $\underline{t}_3$ are still affine transformations of the adjacent handles.

Figures 6 shows the design of a logo which is generated on a piecewise Bézier (B-spline) basis. The primitives are cubes and the deformed shapes are chosen somewhat arbitrarily.

5. CONCLUSIONS

The proposed approach replaces the sometimes overwhelming problem of control point clutter with a simple design scheme for a broadly useful set of deformations, and does so with greater computational efficiency.

We have only used polygons and cube primitives in the paper. Clearly any object defined in R^3 could be warped. Extensions in higher dimensions are also straightforward.

Finally and most importantly, the deformation algorithm is just one example of the generalized de Casteljau idea. Other applications such as interpolation to local affine systems are yet to be investigated. The idea is applicable to any iterative scheme like de Casteljau's. One might consider, for instance, a generalized de Boor scheme directly for B-splines or an Aitken scheme for Lagrange interpolation.

ACKNOWLEDGEMENTS

We thank faculty and students of the Computer Aided Geometric Design (CAGD) group of the Arizona State University for their comments and support.

REFERENCES

[1] Barr, A. H., Global and Local Deformations of Solid Primitives, Proceedings of SIGGRAPH '84, Computer Graphics 18, 3 (July 1984), 21-30.

[2] Sederberg, T. W. and Parry, S. R., Free-Form Deformation of Solid Geometric Models, Proceedings of SIGGRAPH '86, Computer Graphics 20, 4 (August 1986), 151-159.

[3] Bézier, P., Mathematical and Practical Possibilities of UNISURF, In Barnhill, R. E. and Riesenfeld, R. F. (eds), Computer Aided Geometric Design, Academic Press (1974), New York, 127-152.

[4] Coquillart, S., Extended Free-Form Deformation: A sculpturing Tool for 3D Geometric Modeling, Proceedings of SIGGRAPH '90, Computer Graphics 24, 4 (August 1990), 187-196.

[5] Farin, G. E., Curves and Surfaces for Computer Aided Geometric Design, Academic Press (1993), 3rd Edition, Boston.