Shape Transformation for Cubic Parametric Curves

Wei-Chang Du
Department of Information Engineering, I-Shu University
1, Section 1, Hsueh-Cheng Rd., Ta-Hsu Hsiang, Kaohsiung, Taiwan, ROC
e-mail: wcdu@csa500.isu.edu.tw

Abstract

Techniques that transform one polygon to another have been gained considerable interest in recent years. Extending these techniques, this paper addresses the problem of shape transformation between two given shapes, both of which are represented by piecewise Bézier curves. In order to achieve visual pleasantness, a new transformation method based on Hermite-form representation is proposed to blend two Bézier segments. Unlike the traditional vertex-to-vertex matching strategy, we use the dynamic programming to match the similar segments between two sets of piecewise curves. Some experimental results are given to demonstrate the quality of proposed algorithm.

Keywords. shape transformation, shape blending, metamorphosis, interpolation, computer animation.

1. Introduction

Shape transformation is defined as a geometric transformation from one object to another. This process generates a sequence of in-between shapes, which provides a continuous transformation between objects. Such a technique has been widely used in the fields of geometric modeling and computer animation. In geometric modeling, shape transformation is used to blend original shapes for creating new shapes[4,6]. Similar work is also applied to produce new fonts[10,11]. In computer animation, it is often used to generate in-between shapes from key-frame objects[2,7].

In recent years, many methods are proposed for solving the transformation of polygonal shapes in 2-D[4,11,12,13,15], or polyhedral objects in 3-D[8,16]. These methods usually consist of two processes: construction of vertex/edge/face correspondence and shape interpolation. The former is to establish the matching between two objects. The latter is to blend the source and target entities according to this correspondence. However, the blending of high-order curves is given less emphasis. In this paper, we focus our attention on the 2-D or 3-D shape transformation for piecewise cubic Bézier curves. As known, Bézier curve is widely used in CAD or other drawing packages[1] since it provides a convenient control over its behavior. Constructing a composite curve from several simple Bézier curves can be made G^1 geometric continuity (i.e. a curve with continuous tangent directions) simply[3]. Therefore, it is quite easy to design a smooth curve by using piecewise Bézier curves.

In general, the transformation between two shapes is not unique. However, a natural transformation should have some common characteristics. Hence we give some basic criteria and show our proposed algorithm satisfying these criteria.

identity preserving: When two given curves are identical, the blending is the same as the input curves.

translation invariable: If the original curves have the same shape but different locations, then the blending curves are the translation of the input curves at an uniform speed of movement.

smoothness preserving: Smooth curves are often generated in many computer graphics applications. For the transformation of high-order curves, all blending curves should be smooth if both original

curves are smooth.

feature preserving: If the original shapes have common features, the features should be retained during the transformation.

In this paper, we firstly introduce two blending methods between any two simple Bézier curves based on Bézier-form and Hermite-form representations, and suggest the one to be used for joining composite blending curves smoothly. In order to find the global matching between a pair of piecewise curves, we define a cost function to measure the goodness of blending results, and then use the dynamic programming to find the transformation with the least cost. Finally, experimental results are given to show the quality of proposed transformation.

2. Representation of Parametric Cubic Curves

In this section, the two types of curve formulations, named Bézier form and Hermite form, that are interest here are reviewed. More details can be found in [5].

2.1. Bézier form

A cubic parametric curve in Bézier form is determined from four control points, which is defined as

$$Q(u) = B_u(P_0, P_1, P_2, P_3)$$
$$= (1-u)^3 P_0 + 3u(1-u)^2 P_1 + 3u^2(1-u)P_2 + u^3 P_3$$

$0 \le u \le 1$, where P_0, P_1, P_2 and P_3 are control points. From the definition, the following formulas can be easily derived:

$$Q(0) = P_0,$$
$$Q(1) = P_3,$$
$$Q'(0) = 3(P_1 - P_0),$$
$$Q'(1) = 3(P_3 - P_2),$$

where $Q'(u)$ denotes the first derivative of $Q(u)$ with respect to the parameter u. Thus, this curve starts on $Q(0) = P_0$ and stops on $Q(1) = P_3$, and the tangent vector at P_0 is $Q'(0) = 3(P_1 - P_0)$ and at P_3 is $Q'(1) = 3(P_3 - P_2)$.

2.2. Hermite form

The Hermite form of cubic curve is uniquely defined by two endpoints and their tangent vectors. In a similar way, we denote this form of curve as

$$Q(u) = H_u(P_0, P_3, T_0, T_3) = H_u(P_0, P_3, k_0 V_0, k_3 V_3)$$

$0 \le u \le 1$, where $T_0 = Q'(0) = k_0 V_0$ and $T_3 = Q'(1) = k_3 V_3$, and V_0 and V_3 are unit tangent vectors, k_0 and k_3 are the tangent magnitudes. Note that a curve originally defined in Bézier form can be converted into Hermite form by calculating the tangent vectors in terms of the control points[5]; that is,

$$Q(u) = H_u(P_0, P_3, T_0, T_3) = B_u(P_0, P_0 + \tfrac{1}{3}T_0, P_3 - \tfrac{1}{3}T_3, P_3)$$

$0 \le u \le 1$.It is known that Bézier form has many advantages over Hermite form. For example, the shape of Bézier curve is well controlled and there exists efficient geometric subdivision algorithms for generating Bézier curves.

For an arbitrary object, it may be difficult to devise a single set of parametric equations that completely defines its shape. But any object can be formed by piecing several segments over different parts of the object. Such a piecewise construction must be carefully designed to ensure that there is a smooth transition from one segment to the next.

3. Blending between Bézier Segments

Given two single Bézier curves $Q^0(u)$ and $Q^1(u)$, we will introduce how to find the blending curve $Q^t(u)$, $0 \le t \le 1$, and discuss the continuity of adjacent blending curves in this section.

3.1. Bézier-form blending

Let $Q^0(u) = B_u(P_0^0, P_1^0, P_2^0, P_3^0)$ and $Q^1(u) = B_u(P_0^1, P_1^1, P_2^1, P_3^1)$ denote two Bézier curves. The blending of $Q^0(u)$ and $Q^1(u)$ is defined as

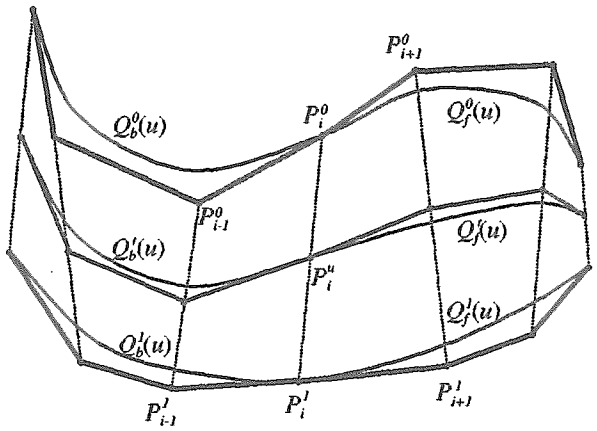

Figure 1. The continuity of joining point.

$$
\begin{aligned}
Q^t(u) &= sQ^0(u) + tQ^1(u) \\
&= sB_u(P_0^0, P_1^0, P_2^0, P_3^0) + tB_u(P_0^1, P_1^1, P_2^1, P_3^1) \\
&= (1-u)^3(sP_0^0 + tP_0^1) + 3(1-u)^2u(sP_1^0 + tP_1^1) \\
&\quad + 3(1-u)u^2(sP_2^0 + tP_2^1) + u^3(sP_3^0 + tP_3^1) \\
&= B_u(sP_0^0 + tP_0^1, sP_1^0 + tP_1^1, sP_2^0 + tP_2^1, sP_3^0 + tP_3^1)
\end{aligned}
$$

where $s = 1 - t$. This implies that we linearly interpolate the coordinate of each corresponding control points, and then find the in-between curves according to these new control points. From the above equations, the endpoints of $Q^t(u)$ are given by

$$
\begin{aligned}
Q^t(0) &= sP_0^0 + tP_0^1 \\
Q^t(1) &= sP_3^0 + tP_3^1
\end{aligned}
$$

and the tangents on the endpoints are given by

$$
\begin{aligned}
Q^t(0) &= 3s(P_1^0 - P_0^0) + 3t(P_1^1 - P_0^1) \\
Q^t(1) &= 3s(P_3^0 - P_2^0) + 3t(P_3^1 - P_2^1).
\end{aligned}
$$

If the two original composite curves are G^1 continuity, then we should observe the continuity of blending curves at joining points carefully. Figure 1 shows an illustration for the join of two adjacent Bézier curves $Q_b^t(u)$ and $Q_f^t(u)$. Since $Q_b^t(1) = sP_i^0 + tP_i^1 = Q_f^t(0)$, the blending curve is continuous. Moreover, the tangents at P_i are

$$
\begin{aligned}
Q_b^t(1) &= 3s(P_i^0 - P_{i-1}^0) + 3t(P_i^1 - P_{i-1}^1) \\
Q_f^t(0) &= 3s(P_{i+1}^0 - P_i^0) + 3t(P_{i+1}^1 - P_i^1)
\end{aligned}
$$

This implies that $Q_b^t(1)$ and $Q_f^t(0)$ have not the same tangent direction except $Q_f^0(0) \times Q_f^1(0) = 0$. So this blending doesn't give any guarantee to be G^1 continuity.

3.2. Hermite-form blending

Next, we consider the Hermite-form case. Let $Q^0(u) = H_u(P_0^0, P_3^0, k_0^0V_0^0, k_3^0V_3^0)$ and $Q^1(u) = H_u(P_0^1, P_3^1, k_0^1V_0^1, k_3^1V_3^1)$. In order to obtain G^1 blending, the blending between $Q^0(u)$ and $Q^1(u)$ is defined as

$$
Q^t(u) = H_u(Q^t(0), Q^t(1), Q^t(0), Q^t(1))
$$

where

$$
\begin{aligned}
Q^t(0) &= sP_0^0 + tP_0^1 \\
Q^t(1) &= sP_3^0 + tP_3^1 \\
Q^t(0) &= \frac{(sk_0^0 + tk_0^1)(sV_0^0 + tV_0^1)}{\left\| sV_0^0 + tV_0^1 \right\|} \\
Q^t(1) &= \frac{(sk_3^0 + tk_3^1)(sV_3^0 + tV_3^1)}{\left\| sV_3^0 + tV_3^1 \right\|}.
\end{aligned}
$$

Since $Q_b^t(1) = sP_i^0 + tP_i^1 = Q_f^t(0)$, the blending curve is continuous. Besides, the blending curve is also G^1

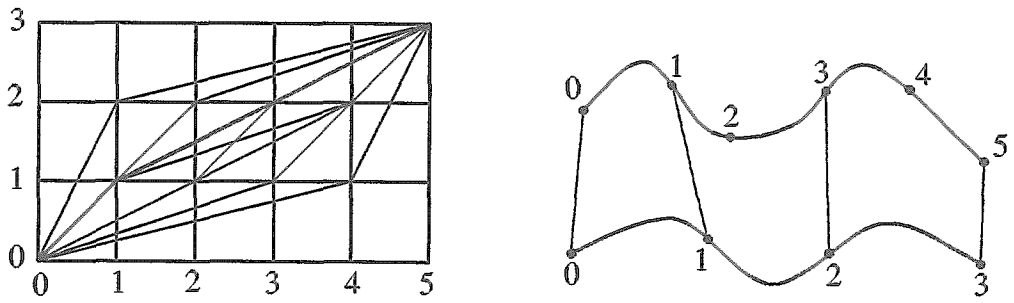

Figure 2. A matching example for m=5 and n=3.

continuous because both unit vectors of $Q_f^t(0)$ and $Q_b^t(1)$ are equal to $(sV_i^0 + tV_i^1)/\left\|sV_i^0 + tV_i^1\right\|$.

4. Blending between Piecewise Bézier Curves

Based on the blending of single curve segments, this section presents a correspondence method for matching two sets of piecewise curves. Because it is possible that two given curves have different number of curve segments, one or more segments will match one segment. Our strategy is to define a cost function for any two Bézier curves firstly, and then to employ a discrete graph to find an optimal matching between a pair of piecewise curves.

4.1. the cost function

Let $Q^0 = Q^0(u) = B_u(P_0^0, P_1^0, P_2^0, P_3^0)$ and $Q^1 = Q^1(u) = B_u(P_0^1, P_1^1, P_2^1, P_3^1)$ be two any Bézier curves. Let $T_i = P_i^1 - P_i^0$. The cost function between Q^0 and Q^1 is define as

$$Cost(Q^0, Q^1) = \left\|T_0 - T_1\right\|^2 + \left\|T_1 - T_2\right\|^2 + \left\|T_2 - T_3\right\|^2.$$

It is obvious that the cost function is zero if $Q^0 = Q^1$, or Q^0 is a translation of Q^1. In other words, the optimal matching will satisfy identity preserving and translation invariant.

4.2. the correspondence

Let $C^0 = (Q_0^0, Q_1^0, \Lambda, Q_{m-1}^0)$ and $C^1 = (Q_0^1, Q_1^1, \Lambda, Q_{n-1}^1)$ denote two piecewise Bézier curves. Suppose that the starting point of Q_i^0 (resp. Q_i^1) is denoted by S_i^0 (resp. S_i^1). Then we define two boundary-point lists as $L^0 = (S_0^0, S_1^0, \Lambda, S_m^0)$ and $L^1 = (S_0^1, S_1^1, \Lambda, S_n^1)$. All curve-segment correspondences can be represented in a $(m+1)\times(n+1)$ rectangular graph. Figure 2 shows a matching example. The column represents boundary points on C^0 and the rows represent boundary points on C^1. Each node (i, j) of the graph represents a possible matching of point S_i^0 of the first list L^0 with point S_j^1 of the second list L^1. So the optimal matching of the two lists is equivalent to define the path with the minimal cost from $(0,0)$ to (m,n). Furthermore, each point on (i, j) can connect to next point $(i+1, j+k)$, $1 \le k \le n-j$, or $(i+k, j+1)$, $1 \le k \le m-i$. The remainder is a typical problem of graph theory. We will use the dynamic programming to find the optimal solution.

Let $Cost(i_0, j_0, i_1, j_1)$ to be the elementary cost of the arc joining nodes (i_0, j_0) and (i_1, j_1). Suppose that $C(i, j)$ is the cumulated cost of the optimal path starting at node $(0,0)$ and ending at (i, j). Then we obtain

$$C(i, j) = MIN\{C_1(i, j), C_2(i, j)\}$$

where

$$C_1(i, j) = MIN\{C(k, j-1) + Cost(k, j-1, i, j)\}, \forall 0 \le k \le i-1$$
$$C_2(i, j) = MIN\{C(i-1, k) + Cost(i-1, k, i, j)\}, \forall 0 \le k \le j-1$$

Starting from the first column, the algorithm proceeds to next column until the last one is reached. Thus the last column is

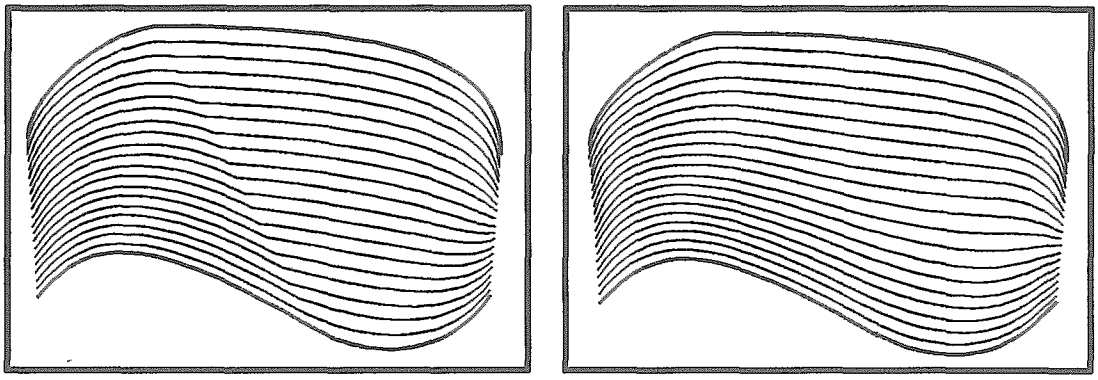

Figure 3. Bézier-form vs. Hermite-form blending.

searched for the node with the minimal cost, and the optimal path is found by tracing back the remembered nodes. According to this optimal path, we uniformly subdivide the Bézier curves such that two composite curves have the same number of segments. Then, applying the Hermite-form blending to interpolate each pair of Bézier curves, we can successfully obtain the final blending results.

5. Experimental Results

In order to demonstrate the validity of our approach, we firstly compare the blending results based on Bézier-form and Hermite-form representations. In Figure 3, two original G^1 curves are denoted as bolder curves. As expected, it can be found that the Bézier-form blending is not smooth at the join points. Therefore, using Hermite-form blending can produce better results at the aspect of smoothness preserving.

Our correspondence strategy can be also applied to the polygonal transformation. Most traditional methods are based on the vertex-to-vertex matching strategy[4,12,15]. However, our method uses the segment-to-segment matching in order to avoid intense bending. In Figure 4, the second example shows this comparison.

The third example is to demonstrate the transformation of piecewise Bézier curves in Figure 5. Although the original curves have different number of curve segments, the experiment shows that the algorithm works well for the cases where the input curves have similar features.

6. Conclusions

In this paper, we have described an algorithm to the problem of blending a pair of cubic Bézier curves. This method

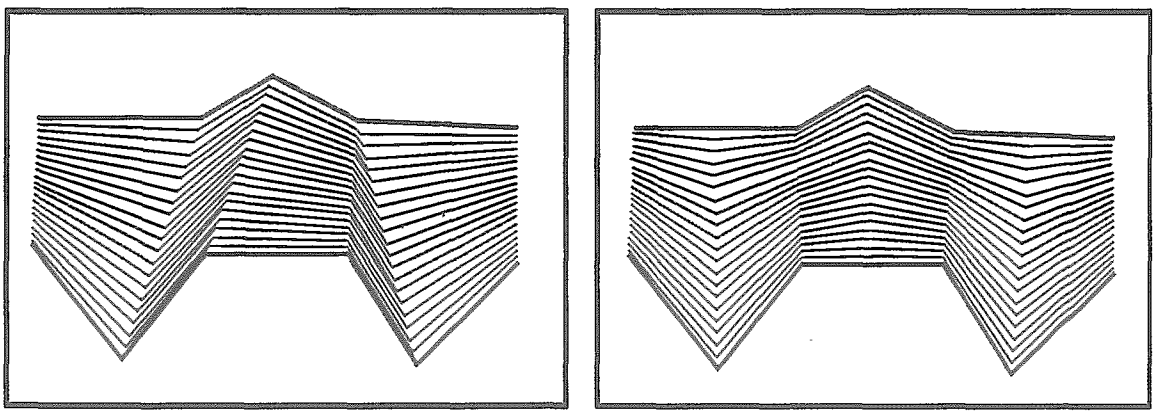

Figure 4. (a) vertex-to-vertex matching, (b) segment-to-segment matching.

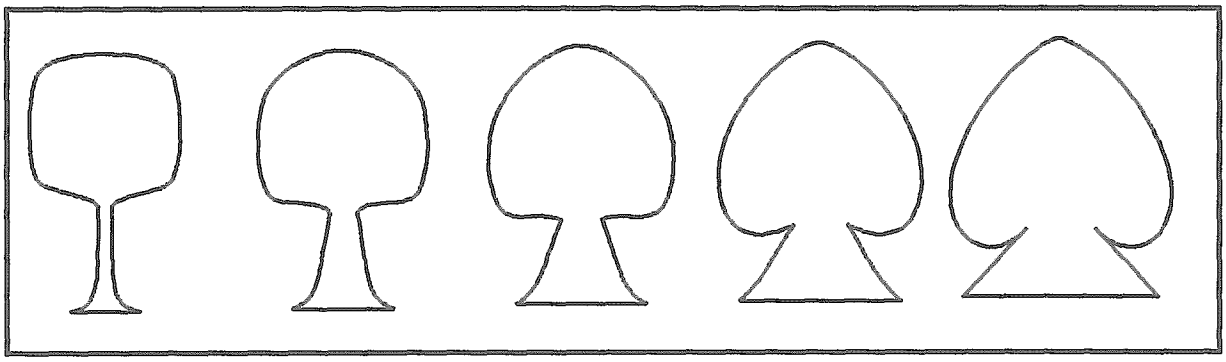

Figure 5. A blending between piecewise Bézier curves.

successfully meets the criteria of identity preserving, translation invariant, smoothness preserving and feature preserving. The algorithm has been implemented and tested for several examples. Experimental results show that the method can produce pleasant results.

7. Acknowledgements

This research work was supposed in part by the National Science Council under contract NSC-87-2218-E-214-012.

References

[1] Richard H. Bartels, *An introduction to splines for use in computer graphics and geometric modeling*, Morgan Kaufmann Publishers, 1987.

[2] E. W. Bethel and S. P. Uselton, "Shape distortion in computer-assisted keyframe animation," *State of the Art in Computer Animation*, pp. 215-224, 1989.

[3] M. D. Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, 1976.

[4] S. E. Chen and R. E. Paren, "Shape averaging and its applications to industrial design," *IEEE Computer Graphics and Applications*, pp. 47-54, 1989.

[5] J. D. Foley, A. V. Dam, S. K. Feiner and J. F. Hugher, *Computer Graphics: Principles and Practice*, Addison-Wesley, 1992.

[6] K. C. Hui and Yadong Li, "A feature-based shape blending technique for industrial design," *Computer-Aided Design*, vol. 30, no. 10, pp. 823-834, 1998.

[7] A. Kaul and J. Rossignac, "Solid-interpolating deformations: construction and animation of PIPs," *Proc. Eurographics '91*, pp. 493-505, 1991.

[8] J. R. Kent, M. E. Carlson and R. E. Parent, "Shape transformation for polyhedral objects," *Computer Graphics*, vol. 26, no. 2, pp. 47-54, 1992.

[9] F. Lazarus, "Smooth interpolation between two polylines in space," *Computer Aided Design*, vol. 29, no. 3, pp. 189-196, 1997.

[10] Z. Pan, X. Ma, M. Zhang and J. Shi, "Chinese font composition method based on algebraic system of geometric shapes," *Computer & Graphics*, vol. 21, no. 3, pp. 321-328, 1997.

[11] Laxmi Parida, "A computational technique for general shape deformations for use in font design," *Computer & Graphics*, vol. 17, no. 4, pp. 349-356, 1993.

[12] T. W. Sederberg and E. Greenwood, "A physically based approach to 2-D shape blending," *Computer Graphics*, vol. 26, no. 2, pp. 25-34, 1992.

[13] T. W. Sederberg, P. Gao, G. Wang and H. Mu, "2-D shape blending: an intrinsic solution to the vertex path problem," Computer Graphics, vol. 27, no. 2, pp. 15-18, 1993.

[14] B. Serra and M. Berthod, "Subpixel contour matching using continuous dynamic programming," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 202-207, 1994.

[15] M. Shapira and A. Rappoport, "Shape blending using

the star-skeleton representation," *IEEE Computer Graphics and Applications*, vol. 15, no. 2, pp. 44-50, 1995.

[16] Y. M. Sun, Wenping Wang and Francis Y. L. Chin, "Interpolating polyhedral models using instrinsic shape parameters," *The Journal of Visualization and Computer Animation*, vol. 8, pp. 81-96, 1997.