中華民國八十四年全國計算機會議論文集
Proceedings of National Computer Symposium 1995

一個產生雙二次保形曲線的幾何演算法
GENERATING BIQUADRATIC SHAPE PRESERVING CURVES BY A GEOMETRIC ALGORITHM

李岳鴻　林基成　李振華
Yueh-Hong Lee, Ji-Cherng Lin and Jenn-Hua Lee

元智工學院電機與資訊工程研究所
Institute of Electrical Engineering and
Computer Engineering and Science
YUAN-ZE Institute of Technology
Chung-Li Taiwan, R.O.C.

摘要

本論文發展了一個能夠自動產生雙二次保形曲線的插值演算法。使用者輸入了一些插值點之後，本演算法將產生一條通過這些插值點的雙二次保形曲線，此曲線是由一些曲線段連接而成，而每一曲線段包含兩段二次伯濟耳曲線，整條曲線的形狀相當接近插值點所定義之折線的形狀。本演算法提供了一些形狀參數，讓使用者能夠輕易的調整曲線的形狀。相對於三次保形曲線演算法，本演算法比較簡單、直接、並且容易實作。

關鍵詞：保形曲線，形狀參數。

Abstract

Interpolating algorithm using piecewise biquadratic Bezier curves is developed. The algorithm can automatically generate shape preserving biquadratic curves which faithfully describe the shape implied by the given data points. The curves obtained satisfy the conditions for 'visual content'. Using this algorithm, the designer is able to control the tension of the interpolating curve at each of the interpolating points. Besides the ease of shape control, the algorithm is conceptually simple and computationally efficient. It can be used to model 2D objects of various shapes. By appropriately adjusting the shape parameters at related interpolating points, the designer can easily change the curve segments' shape without affecting the remaining portions of the curve. From the comparisons of our new interpolating curves with cubic splines, it shows that our method is simple, direct and easy to be implemented.

Keywords: shape preserving curves, shape parameter.

1. Introduction

Curve generation plays an important role in CAGD (Computer Aided Geometric Design) and data interpolation. In a CAD system, the generated curve is supposed to interpolate, or approximate, a given set of points such that the curve can be used to model some desired objects, for example, the aircraft bodies or car bodies. While in the 'data interpolation' field, it is reasonable that the curve must faithfully depict the information contained in the given data points.

Many curve interpolation methods have been proposed since the early sixties, but it is disappointing that the curve generated by some algorithms are less 'eye-pleasing' than the one drawn by a draftsman. The elementary requirement of the curve's shape is the designer's 'visual content'. It is really vague that whether a curve satisfies 'visual content' or not. In this paper, the interpolating curve maintains at least slope continuity (visual smoothness).

Comparing with generating interpolating curves with cubic Bezier curve segments traditionally, using biquadratic Bezier curve segments instead have many benefits:

- The same control points, but with one more degree of freedom.
- The control polygon of quadratic Bezier curve is much closer to the curve than the control polygon of cubic Bezier curve.
- There is no need to classify *convex*, *inflective* and *straight* cases.
- Generating quadratic curve is faster than cubic curve.

From the merits list above, we begin our research on generating shape preserving curves with biquadratic Bezier curve segments. Figure 1.1. and Figure 1.2. show the difference between cubic Bezier curve and biquadratic Bezier curve.

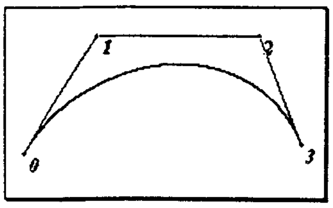


Figure. 1.1 Cubic Bezier curve.

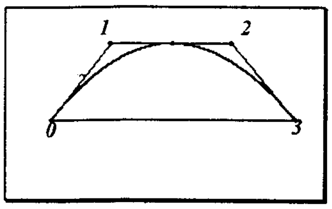


Figure. 1.2. Biquadratic Bezier curve.

Walton and Xu [8] use biquadratic Bezier curve for drawing *turning point preserving curves*, that is, the curvature extremum of the spline is forced to occur at the turning points. Their method ensures that the generated curves have non-self-intersection property (if the polygon of input data points does not intersect itself, the curve does not intersect itself), whereas the generated curves are sometimes too taut. It is also impossible to change the appearance of the curves unless the user changes input data points.

In this paper, we first infer tangent vector and normal vector of each control point. Then a geometric algorithm for locally generating quadratic shape preserving curves is proposed, so that the interpolating curve maintains at least *slope continuity* (visual smoothness) [5, 7], that is, adjacent curve segments have the same tangent direction at the point p_i ($1 < i < n$, p_i is the given control point). It is further assumed that no more than three data points are given collinear to form a straight line. From the given examples, we will show that our method is simple, direct and easy to be implemented, and the interpolating curves possess all the properties of a good interpolation method.

2. The Bezier Curves

2.1 Introduction

In this paper, Bezier curve [3] will be used to interpolate data points. The Bezier formulations for parametric curves possessed the following advantages:

- The Bezier curve passes through the first and the last Bezier points of the Bezier polygon.
- Each Bezier curve segment is computed separately.
- There is an intuitive relationship between the shape of the Bezier polygon and the curve.
- The method is easy of computation and subdivision.
- Bezier curves are invariant under affine transformations.
- The method is simple and efficient than others.

2.2 Bezier curves

A Bezier curve [3] is a parametric curve whose shape approximates the control polygon. Two of the polygon's vertices are the end points of the spline. Since the curve's shape will tend to follow the polygon's shape, it is intuitive that changing the position of the polygon's vertices will change the shape of the curve. Even a new user is able to change the shape of the curve to his intent by reposition the control points.

The mathematical basis of the Bezier curve is a polynomial blending function which interpolates between the first and last vertices. The Bezier basis functions are *Bernstein polynomials*. Thus, the Bezier curve is said to have a Bernstein basis. It is more strict to call the Bezier curves as Bernstein-Bezier curves. Figure 2.2.1. gives two examples of Bezier curves.

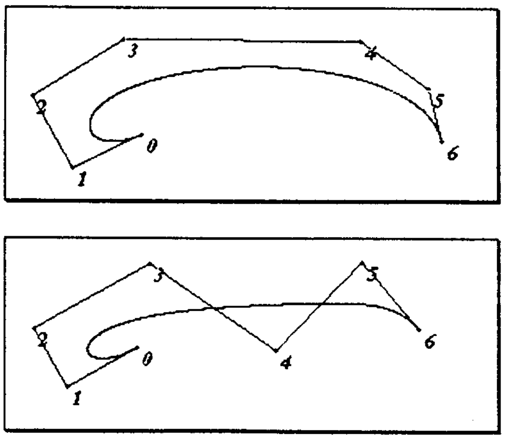


Figure 2.2.1. Bezier curves.

The Bezier curve of degree n can be expressed as:

(2.2.1) $$B(t) = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i} P_i,\ 0 \le t \le 1$$

Where $\frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$ is the so called 'Bernstein polynomials.' From Equation (2.2.1), we can find that:

(2.2.2)
$$B(0) = P_0,$$
$$B(1) = P_n,$$
$$B_t(0) = n(P_1 - P_0),$$
$$B_t(1) = n(P_n - P_{n-1}).$$

Where $B_t(t)$ stands for the first derivative of the segment $B(t)$. Thus we know that the curve depicted by the Bezier's form passes through the end points P_0 and P_n. The tangent vector at P_0 points from P_0 to P_1 and the tangent vector at P_n points from P_{n-1} to P_n. The straight lines P_0P_1, P_1P_2, ..., $P_{n-1}P_n$ form a figure called "characteristic polygon" or "Bezier polygon" of the curve. P_0, P_1, P_2, ..., P_n are called "control points" or "Bezier points" of the Bezier curve.

2.3 Composite Bezier curves

The Bezier curve does not possess the local property, i.e., any positional change of P_i ($i = 0, 1, ..., n$) influences the shape of entire curve. The simplest way to attain local property is to stitch Bezier curve segments to form the whole curve. Suppose that we wish to join segment $B^{(i)}(t_i)$, $0 \le t_i \le 1$, (segment i for short), to segment $B^{(j)}(t_j)$, $0 \le t_j \le 1$. For human's 'visual pleasantness', the basic requirement is to make the curve continuous at the joint, and maintains at least slope continuous there. Thus we must have:

(2.3.1) $$B^{(i)}(1) = B^{(j)}(0)$$

for position continuity, and,

(2.3.2)
$$B_t^{(i)}(1) = \alpha_i T$$
$$B_t^{(j)}(0) = \alpha_j T$$

for slope continuity. Where, T is the unit vector in the common tangent direction at the joint, moreover α_i and α_j are scalar constants which influence the fullness of the curve segments. Figure 2.3.1. illustrates the composition of two cubic Bezier segments. The end point $B^{(i)}(1)$ of segment $B^{(i)}$ joins with the start point $B^{(j)}(0)$ of segment $B^{(j)}$. The tangent vector on $B^{(i)}(1)$ and the tangent vector on $B^{(j)}(0)$ point to the same direction.

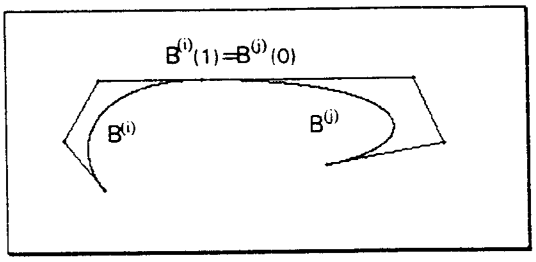


Figure 2.3.1. The composition of two cubic Bezier segments.

If we regard Bezier segments as segments $B^{(i)}(t_i)$ and $B^{(j)}(t_j)$, then Equation (2.3.1) and (2.3.2) become to be:

(2.3.3)
$$P_n^{(i)} = P_0^{(j)},$$
$$\beta_i (P_n^{(i)} - P_{n-1}^{(i)}) = P_1^{(j)} - P_0^{(j)}.$$

where β_i is equal to α_i/α_j.

From Equation (2.3.3), we are able to construct a composite Bezier curve with slope continuity on each joint by defining each β_i.

3. Generating Biquadratic Shape Preserving Curves by a Geometric Algorithm

3.1. Definition of Shape Preserving Curve Interpolation

Before proceeding, we have to introduce some preliminary definitions first. A plane parameterized differentiable curve segment $R(t)$, $0 \le t \le 1$, is said to be *regular* if all of $R_t(t) \neq 0$. The curve $R(t)$ is simple if it has no further self-intersection except possibly at end points. A regular plane curve segment $R(t)$ is *convex* if and only if it is simple and can be oriented so that its curvature is positive or zero. This means that the curvature of each point in the curve segment does not change sign. A differentiable curve segment is an *inflective* one if it has an *inflection point.* Finally, it is trivial that a curve segment is a *straight line segment* if curvature is zero everywhere.

From above discussion, we are ready to define the shape preserving interpolating curves now. Since the *unit tangent continuity* at each joint point is a basic requirement, here we take into account the directions of tangent vectors.

Definition 3.2.1 Let tangent vector T_i at the interpolating point P_i be chosen such that $T_i = V / |V|$, where $V = (1-\delta_i)\overrightarrow{P_{i-1}P_i} + \delta_i\overrightarrow{P_iP_{i+1}}$, then a smooth interpolating curve is shape preserving if we construct each curve segment i with the following rules:

Rule 1: If vectors $\overrightarrow{P_{i-1}P_i} \times \overrightarrow{P_iP_{i+1}}$ and $\overrightarrow{P_iP_{i+1}} \times \overrightarrow{P_{i+1}P_{i+2}}$ have same directions and both are not null vectors, we generate curve segment i to be a convex one.

Rule 2: If vectors $\overrightarrow{P_{i-1}P_i} \times \overrightarrow{P_iP_{i+1}}$ and $\overrightarrow{P_iP_{i+1}} \times \overrightarrow{P_{i+1}P_{i+2}}$ have different directions and both are not null vectors, we generate curve segment i to be an inflective one.

Rule 3: If either $\overrightarrow{P_{i-1}P_i} \times \overrightarrow{P_iP_{i+1}}$ or $\overrightarrow{P_iP_{i+1}} \times \overrightarrow{P_{i+1}P_{i+2}}$ is a null vector, we generate curve segment i to be a straight one.

We note that if $\overrightarrow{P_iP_{i+1}}$ corresponds to a convex curve segment then the curve must be locally convex at both P_i and P_{i+1}, which means that the curve lies on one side of the tangent.

From the above definition, the shape of curve segment i is implied by four consecutive interpolating points. Hence, we can show that the shape preserving condition requires the turning behavior of an interpolating curve to accord with its control polygon.

3.2. Composite Quadratic Bezier Curve Interpolation

Consider a curve segment that is composed of two quadratic Bezier curve segments, $B_R(t)$ and $B_L(t)$. The quadratic Bezier curve segment $B_L(t)$ can be defined by the following equation:

(3.2.1) $B_L(t) = (1-t)^2b_0 + 2t(1-t)b_1 + t^2b_2,\ 0 \le t \le 1$

Here b_0, b_1 and b_2 are Bezier points. We refer b_0 and b_2 as *boundary Bezier points* and b_1 as *interior Bezier point*. The control polygon formed by b_0, b_1 and b_2 is a triangle, and the curve segment $B_L(t)$ is a parabola.

Similarly, $B_R(t)$ is defined by:

(3.2.2) $B_R(t) = (1-t)^2b_2 + 2t(1-t)b_3 + t^2b_4,\ 0 \le t \le 1$

Here b_2, b_3 and b_4 are Bezier points. We refer b_2 and b_4 as boundary Bezier points and b_3 as interior Bezier point.

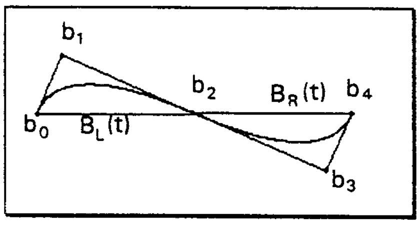


Figure 3.2.1 Inflective curve.

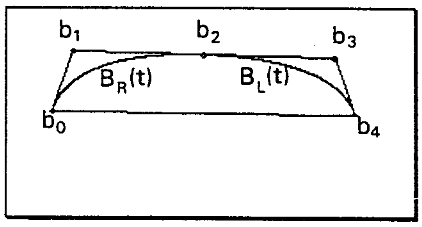


Figure 3.2.2 Convex curve.

$B_R(t)$ and $B_L(t)$ are joined at b_2 such that they have the same unit tangent vector at b_2. Figure 3.2.1 illustrates the composition of an inflective curve and Figure 3.2.2 illustrates the composition of a convex curve.

From Equation (3.2.1), we can derive Equation (3.2.3) and Equation (3.2.4) easily.

(3.2.3) $B_L(0) = b_0,\ B_L(1) = b_2$ and

(3.2.4) $B_L'(0) = 2(b_1 - b_0),\ B_L'(1) = 2(b_2 - b_1)$

According to Equation (3.2.3), the quadratic Bezier curve segment passes through its boundary Bezier points. From Equation (3.2.4), we know that the tangent directions at Bezier points b_0 and b_2 are the same with vector t_0 and t_2, respectively, where t_0 is a vector with initial point b_0 and terminal point b_1, and t_2 is a vector with initial point b_1 and terminal point b_2. The same inference can be used on the quadratic Bezier curve segment passes through, b_2, b_3, and b_4.

For C^1 continuity of slope at the joint b_2, we must have $B_R'(0) = B_L'(1)$, that is, $(b_3 - b_2) = (b_2 - b_1)$. But we know that slope continuity is enough to attain 'visual content'. Thus b_2 can be placed on any position on line segment $b_3\ b_1$.

3.3. Locally Inferring Tangent Vectors and Normal Vectors

The smoothness of a piecewise curve has traditionally been measured by maintaining parametric continuity at joints; that is, a continuity in the parametric derivatives on both sides of the joints. Recent work has shown that parametric continuity is overly restrictive; more relaxed constraints of geometric continuity have recently been proposed [4, 1, 2]. For the visual content of human beings, the interpolating curve maintains at least *slope continuity* at joints [5, 7]. Thus the tangent vector on each data point affects how the generated curve looks like.

Locally inferring tangent vector and normal vector plays an important role in an automatic curve-generating algorithm. Walton and Xu [8] have proposed a method to compute tangent vector on each boundary Bezier point p_i.

Supposed $p_i = (x_i, y_i)$, $i = 0, 1, \ldots, n$, are boundary Bezier points. The interior unit tangent vector t_i on p_i may be defined as:

(3.3.1) $$G_i = \frac{p_i - p_{i-1}}{|p_i - p_{i-1}|} + \frac{p_{i+1} - p_i}{|p_{i+1} - p_i|}$$

(3.3.2) $$t_i = \frac{G_i}{|G_i|}$$

and the unit normal vector n_i as:

(3.3.3) $$n_i = t_i \cdot \begin{bmatrix} \cos 90^\circ & \sin 90^\circ \\ -\sin 90^\circ & \cos 90^\circ \end{bmatrix}$$

This technique for choosing the tangent vectors ensures that the angles from the tangent vector t_i to the vector $p_i - p_{i-1}$ and $p_{i+1} - p_i$ are within $\pi/2$ rad. While generating biquadratic curve segments with our algorithm, this property is useful that ensures the generated curves will not have undesired shapes. The curves generated using tangent vectors derived by this method are satisfactory as far as the shape preserving property is concerned.

3.4. Generating Biquadratic Bezier Curve Segments by a Geometric Algorithm

Let the i-th curve segment be denoted by index i and begins from boundary Bezier point p_i to p_{i+1}. As we have inferred the unit tangent vector t_i and unit normal vector n_i on each boundary Bezier point p_i, we can compute the shape preserving curves easily from the following steps:

(1) Find the intersection point f_i of line T_i and T_{i+1}, where T_i and T_{i+1} are lines which parallel to vector t_i and t_{i+1}, and pass through points p_i and p_{i+1}, respectively.

(2) Find the intersection point g_i of lines N_i and T_{i+1} and the intersection point h_i of lines N_{i+1} and T_i, where N_i and N_{i+1} are lines which parallel to vector n_i and n_{i+1}, and pass through points p_i and p_{i+1}, respectively.

(3) Then we can get the interior Bezier point $b_1^i = p_i + \delta_i \alpha_i t_i$ and $b_3^i = p_{i+1} - \delta_i \beta_{i+1} t_{i+1}$, here $b_0^i = p_i$, $b_4^i = p_{i+1}$, $\alpha_i = \min(\|p_i, f_i\|, \|p_i, h_i\|)$ and $\beta_{i+1} = \min(\|p_{i+1}, f_i\|, \|p_{i+1}, g_i\|)$, $\|c, d\|$ denotes the distance function between c and d. δ_i ($0 \le \delta_i \le 1$) is a shape parameter of the i-th curve segment. For the purpose of keeping the quadratic Bezier curve segments C^1 *continuous* at the joint b_2^i, where b_2^i is supposed to be the mid-point of $\overline{b_1^i b_3^i}$.

Figure 3.4.1. shows the intersection point f_i of lines T_i and T_{i+1}, the intersection point g_i of lines N_i and T_{i+1}, and the intersection point h_i of lines N_{i+1} and T_i.

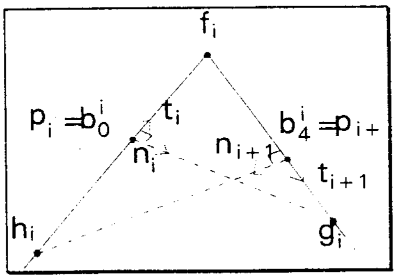


Figure 3.4.1 Find the intersection points f_i, g_i, and h_i.

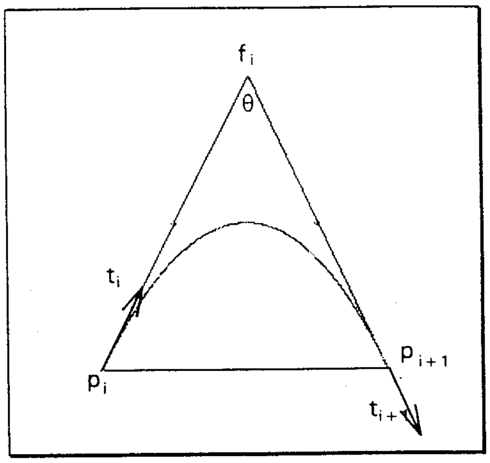


Figure 3.4.2.

In Figure 3.4.2., if we just take into account the interpolation point f_i of lines T_i and T_{i+1}, when the angle q between lines T_i and T_{i+1} become smaller, evidently the distance between f_i and line p_ip_{i+1} will become longer, and the shape of generated curve will tend to bend too much. Thus the use of normal vector is a good solution for this problem. In our algorithm, no matter how the angle q changes, the interpolating curve will be limited in a reasonable area.

Figure 3.4.3. and Figure 3.4.4. illustrate the composite of convex curve and inflective curve. They show that if the tangent vectors t_i at the interpolating points P_i are chosen by the method we have mentioned in 3.3., and the interpolating curve segments are generated from the above steps in this section, then the generated curves satisfy the definition of shape preserving curve interpolation.

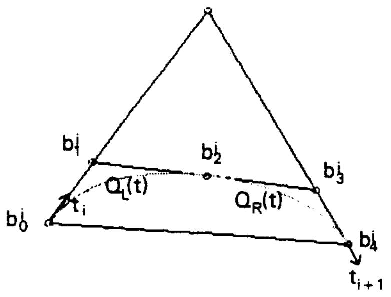


Figure 3.4.3 The composite of convex curve.

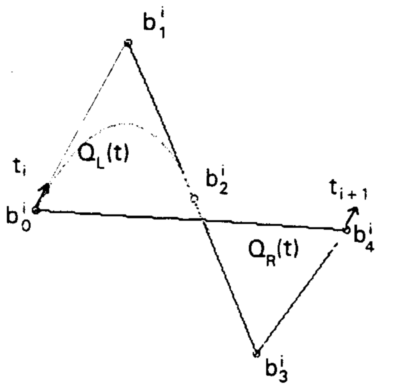


Figure 3.4.4 The composite of inflective curve.

The ***locally non-self-intersection property*** can be illustrated as follows: The shape preserving curve segments of three consecutively input data points does not intersect itself.

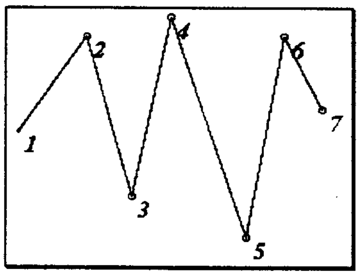
If the tangent vector on each data point is inferred by the above method, it can be shown that the curve has the locally non-self-intersection property.

4. Shape Control with Interpolating Biquadratic Bezier Curves

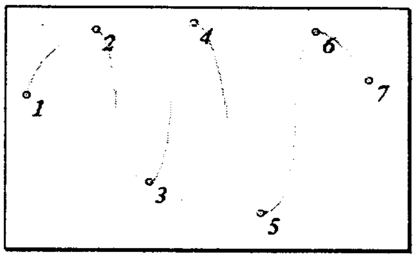
4.1. Shape Parameter

To use interpolation as curve design has become popular. For interpolating curves, shape parameters offer a good way to vary curve's shape. In our shape preserving method, each δ_i can be served as shape parameter which can be adjusted locally.

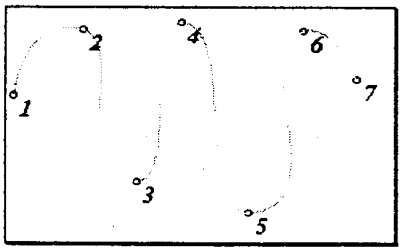
Figure 4.1.1 gives examples of curves with different shape parameter δ_i. We show the control of the curve's shape by using a global shape parameter. If δ_i is increased or decreased, the curve will be loosened or tightened accordingly. This effect is called ***tension***. When the value of δ become smaller, the curve is closer to the linear polygon which interpolates the given points.



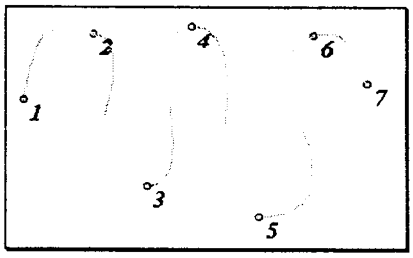
(a) $\delta_i = 0$.



(b) $\delta_i = 0.3$



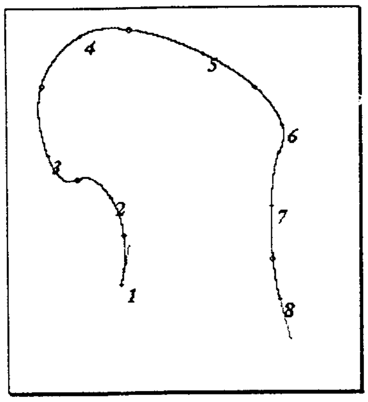
(c) $\delta_i = 0.5$



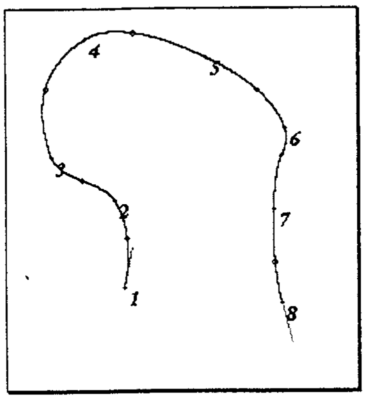
(d) $\delta_i = 0.7$

Figure 4.1.1 Curves with different δ_i.

Since each shape parameter δ_i can be adjusted locally, our method can provide the user with greater convenience in the construction of the curves. By varying the shape parameter, the user can modify a portion of the curve without affecting other portions of the curve, and it is not necessary for the user to re-input new data points for the purpose of modifying the curve's shape. In Figure 4.1.2., we change δ_2 (the shape parameter of curve segment 2) from 0.5 to 0.2. It changes the shape of curve segment 2 without affecting other portions of the entire curve.



(a) δ_2=0.5



(b) δ_2=0.2

Figure 4.1.2. The change of single shape parameter.

4.2. Example

The proposed method was implemented in Visual Basic under Windows environment. Here the example is given to compare our method with other approaches.

Example 1. This example combines convex data, oscillatory data, and data extracted from straight line segments.

The method of Goodman and Unsworth [6] is shown in Figure 4.2.1. This curve has the undesirable feature that the curve segment joining the points labeled 6 and 7 intersects the curve segment joining the points labeled 7 and 8.

The usual C^2 cubic spline is shown in Figure 4.2.2. It also has the undesirable feature that the curve segment joining the points labeled 6 and 7 intersects the curve segment joining the points labeled 7 and 8.

The method of Walton and Xu [8] is shown in Figure 4.2.3. The curve segments joining the points labeled 4 and 5, the points labeled 6 and 7, and the points labeled 7 and 8, appear to be too taut for visual smoothness.

The proposed method is shown in Figure 4.2.4. The curve has the locally non-self-intersection property. The curve appears to be less taut than the curve generated by the method of Walton and Xu.

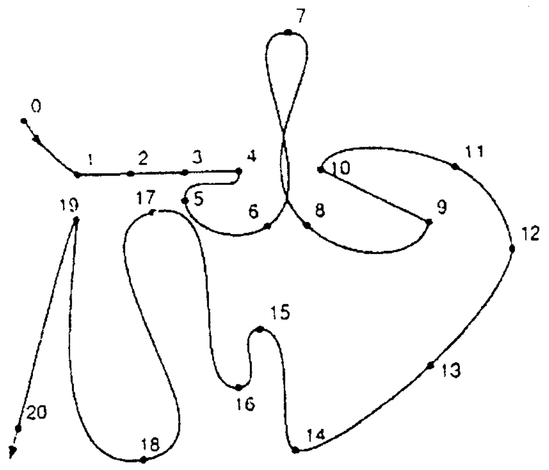


Figure 4.2.1 Method of Goodman for Example 1.

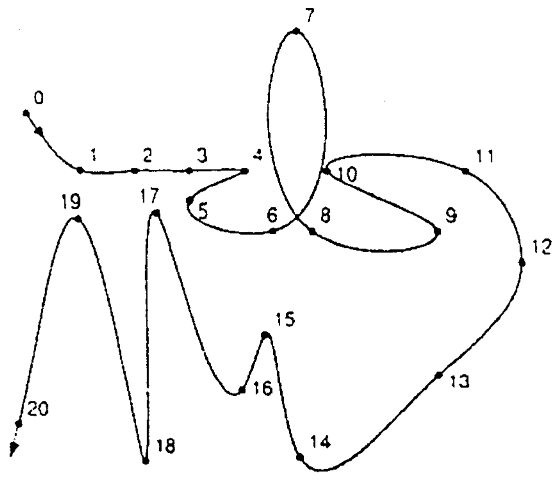


Figure 4.2.2 Usual cubic spline for Example 1.

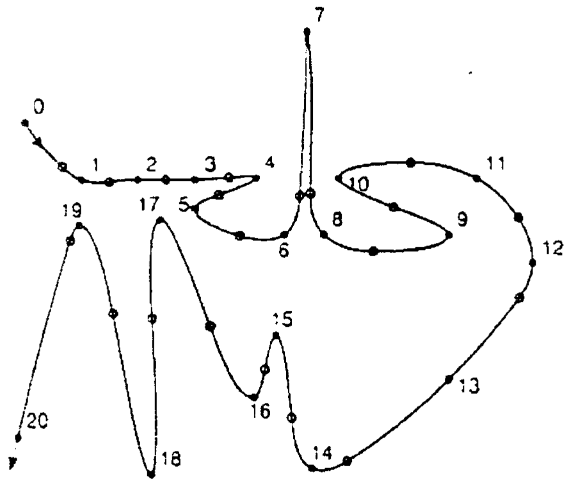


Figure 4.2.3 Method of Walton and Xu for Example 1.

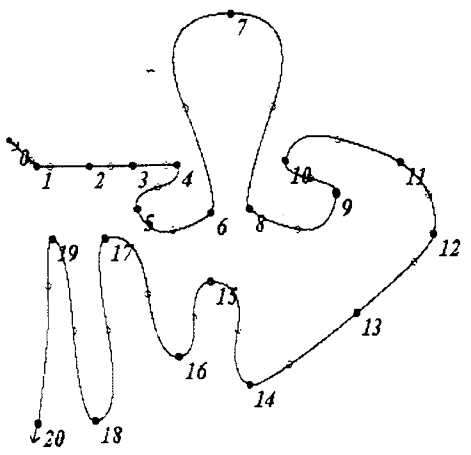


Figure 4.2.4 Our method for Example 1.

5. Conclusion

Our method has the following advantages:

- It is easily computed so that it may be generated fast in an interactive graphics environment.
- The curve has some degree of smoothness (i.e., slope continuity).
- There is no need to classify convex, inflective and straight cases.
- The curve has the locally non-self-intersection property, so that the generated curve does not have the undesirable intersections.
- The shape parameter offers an easy way to vary curve's shape locally under an interactive environment.

References

[1] B. A. Barsky and J. C. Beatty, "Local control of bias and tension in Beta_splines", *Computer Graphics*, Vol. 17, No. 3, pp. 193-218, July 1983.

[2] B. A. Barsky and T. D. De Rose, "The Beta2_ spline : a special case of the Beta_spline curve and surface interpolation", *IEEE Computer Graphics and Applications*, Vol. 5, No. 9, pp. 46-58, 1985.

[3] P. E. Bezier, *Empoi des machines a commande numerique*, Masson et Cie, Paris, 1970. Translated by D. R. Forrest and A. F. Pankhurt as P. E. Bezier, *Numerical control-mathematics and applications*, John Wiley & Sons, Inc., London, 1972.

[4] G. Farin, "Visually C^2 Cubic Splines", *Computer-Aided Design*, Vol. 14, No. 3, pp. 137-139, May 1982.

[5] I. d. Faux and M. J. Pratt, *Computational geometry for design and manufacture*, Ellis Horwood, Chichester, 1979.

[6] T. N. T. Goodman and K. Unsworth, "Shape preserving interpolation by curvature continuous parametric curves", *Computer-Aided Geometric Design*, pp. 323-340, 1988.

[7] J. H. Lee and S. N. Yang, "Shape preserving and shape control with interpolating Bezier curves", *Journal of Computational and Applied Mathematics*, Vol. 28, pp. 269-280, 1989.

[8] D. J. Walton and R. Xu, "Turning point preserving planar interpolation", *ACM Transactions on Graphics*, Vol. 10, pp. 297-311, Jul 1991.