TRANSPLINE—A system for representing curves using transformations among four spline formulations

Brian A. Barsky† and Spencer W. Thomas

Department of Computer Science, University of Utah, Salt Lake City, Utah 84112, USA

Various techniques of curve representation have been developed and each has properties that make it more appropriate in certain situations than in others. Thus, it is very beneficial to be able to transform between formulations in order to invoke the one which is best suited to any particular instance. This concept has been applied to a set of four parametric spline curve formulations and is demonstrated in the TRANSPLINE interactive curve representation system. The available formulations are the interpolatory spline, uniform B-spline, spline under tension, and ν-spline. The mathematical theory of each formulation is briefly reviewed, the transformations between them are derived, and the system implementation is described.

(Received August 1980)

1. Introduction

Various mathematical techniques of curve representation have been proposed, and each one has advantages and drawbacks which make it more suitable for certain purposes than for others. For this reason it would be very desirable to have the capability of transforming between formulations so as to be able to exploit the particular advantages of one formulation when they are most needed (Barsky, 1979).

The TRANSPLINE interactive curve representation system demonstrates the application of this concept to a set of four parametric spline curve formulations, each of which was selected on the basis of its appropriateness to a different phase of design. The *interpolatory spline* is ideal for initial data entry since it is much easier to establish a rough approximation of a desired curve shape by specifying points which lie *on* the curve, rather than points which lie *near* the curve. The *uniform B-spline* is well suited to the modification of already entered data because the local nature of a B-spline allows a change to be made to one portion of the curve without altering the remainder of the curve, as would be the case if an interpolatory spline formulation were modified. The *spline under tension* and *ν-spline* are appropriate for the final shape definition because of the capability of the former to *flatten* spline segments which may have too great a curvature, and of the latter to *sharpen* corners while maintaining the desired continuity properties of the spline.

Complete explanations of the parametric representation and the mathematical theory of the formulations can be found in Barsky (1981a; 1981b); a brief review will now be provided and the transformations among the formulations will be derived. Finally, the system implementation will be described; a more detailed presentation is given in Barsky and Thomas (1980).

2. The formulations

2.1 *Interpolatory spline*

2.1.1 *Explanation.* A *cubic interpolatory spline* (Rogers and Adams, 1976; Schultz, 1973) is a *piecewise* function, composed of polynomials of degree not exceeding three, called *spline curve segments.* It *interpolates* a set of points and satisfies the constraints of positional *continuity* along with continuity of the first and second parametric derivative vectors.

A spline segment can be defined by specifying the position of each endpoint along with the corresponding value of the first derivative vector. This specifies *cubic Hermite interpolation* (Schultz, 1973) which has continuity of position and the first derivative vector trivially guaranteed by their specification at each interior point. In cubic spline interpolation, continuity of the second derivative vector is achieved at the expense of the freedom of choosing the value of the first derivative vector at each interior point.

There exists a special set of values of the first derivative vector with the property that the resulting piecewise cubic Hermite curve has a continuous second derivative vector. Thus, the cubic interpolatory spline can be constructed by first determining this set of values and then invoking cubic Hermite interpolation using these values of the first derivative vector at the interior points.

2.1.2 *Hermite interpolation.* As explained above, each curve segment in cubic Hermite interpolation is specified by defining the position of each endpoint as well as the corresponding value of the first derivative vector. Specifically, let $\{\mathbf{P}_0, \mathbf{P}_1, \ldots \mathbf{P}_m\}$ be a set of $m + 1$ points to be interpolated and $\{\mathbf{P}_0^1, \mathbf{P}_1^1, \ldots \mathbf{P}_m^1\}$ be the set of corresponding values of the first derivative vector. Parametrically, the ith curve segment is described as the parameter u varies from a value of u_{i-1} at the beginning of the segment to a value of u_i at the end. In particular, this curve segment can be written as

$$\mathbf{Q}_i(u) = \sum_{j=0}^{1} \sum_{k=0}^{1} g_{j,k}(u_{i-1},u_i;u)\,\mathbf{P}_{i-1+k}^{j} \tag{1}$$

for

$$u_{i-1} \leqslant u < u_i \qquad i = 1, \ldots m$$

The functions $g_{j,k}(u_{i-1}, u_i;u)$ are the *generalised cubic Hermite basis functions* and can be written in matrix form as

$$[g_{0,0}(u_{i-1},u_i;u)\ \ g_{0,1}(u_{i-1},u_i;u)\ \ g_{1,0}(u_{i-1},u_i;u)\ \ g_{1,1}(u_{i-1},u_i;u)]$$

$$= [w^3 w^2 w 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \Delta u \\ \Delta u \end{bmatrix} \tag{2}$$

where

$$w = \frac{u - u_{i-1}}{\Delta u_i} \qquad \Delta u_i = u_i - u_{i-1}$$

2.1.3 *Determining the set of values of the first derivative vector.* The process of determining the set of values of the first derivative vector at the interior points that guarantees continuity of the second derivative vector requires the formulation and sub-

†Present address: Computer Science Division, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720, USA.

CCC-0010-4620/81/0024-0271 $03.50

sequent solution of a system of simultaneous linear equations. Differentiating the expression for the *i*th cubic Hermite curve segment given in Eqns (1) and (2) twice with respect to *u* results in

$$\mathbf{Q}_i^{(2)}(u) = \sum_{j=0}^{1} \sum_{k=0}^{1} g_{j,k}^{(2)}(u_{i-1}, u_i; u)\, \mathbf{P}_{i-1+k}^{j} \tag{3}$$

where

$$\mathbf{Q}^{(a)}(c) = \frac{d^a}{du^a} \mathbf{Q}(u) \Big|_{u=c}$$

for

$$u_{i-1} \leqslant u < u_i \qquad i = 1, \ldots m$$

where the functions $g_{j}^{(2)}{}_{,k}(u_{i-1}, u_i; u)$ can be written in matrix form as

$[g_{0,0}^{(2)}(u_{i-1}, u_i; u)\ g_{0,1}^{(2)}(u_{i-1}, u_i; u)\ g_{1,0}^{(2)}(u_{i-1}, u_i; u)$
$g_{1,1}^{(2)}(u_{i-1}, u_i; u)]$

$$= \begin{bmatrix} \dfrac{u - u_{i-1}}{\Delta u_i} & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 3 & 3 \\ -1 & 1 & -2 & -1 \end{bmatrix} \begin{bmatrix} 6/\Delta^2 u_i \\ 6/\Delta^2 u_i \\ 2/\Delta u_i \\ 2/\Delta u_i \end{bmatrix} \tag{4}$$

and

$$\Delta u_i = u_i - u_{i-1}$$

Using Eqns (3) and (4), the continuity of the second derivative vector, at each of the $m - 1$ interior points, can be represented by the equation:

$$\Delta u_{i+1} \mathbf{P}_{i-1}^1 + 2(\Delta u_i + \Delta u_{i+1}) \mathbf{P}_i^1 + \Delta u_i \mathbf{P}_{i+1}^1$$
$$= 3 \left\{ (\mathbf{P}_i - \mathbf{P}_{i-1}) \frac{\Delta u_{i+1}}{\Delta u_i} + (\mathbf{P}_{i+1} - \mathbf{P}_i) \frac{\Delta u_i}{\Delta u_{i+1}} \right\} \tag{5}$$

for

$$i = 1, \ldots m - 1$$

This is a system of $m - 1$ linear equations for the $m + 1$ unknown values of the first derivative vector. In order to obtain a unique solution, two more equations are needed. This can be fulfilled by specifying an end condition at both $\mathbf{P}_0$ and $\mathbf{P}_m$; various such end conditions are discussed in (Barsky, 1981b).

2.2 *B-spline curve*

2.2.1 *Explanation.* An overview of the B-spline curve representation will now be presented; a more complete explanation can be found in Barsky (1981a) and Riesenfeld (1973). A B-spline curve approximates, but does not interpolate, an ordered sequence of points, called *control vertices*, which are connected in succession to form a *control polygon*. The B-spline curve smooths out or mimics the shape of this control polygon.

The major advantage of the B-spline formulation is that it is a *local* representation. Each cubic B-spline curve segment is controlled by only four of the control vertices and is completely unaffected by all other control vertices. Equivalently, a given control vertex only influences four B-spline curve segments and has no effect whatsoever on the remaining segments. This means that the effects of moving a control vertex are confined to four segments (**Fig. 1**).

Since each cubic B-spline curve segment is completely controlled by only four of the control vertices, a point on this segment can be regarded as a weighted average of these four control vertices. Associated with each control vertex is a weighting factor which is a scalar function evaluated at some parametric value. For a uniform B-spline curve segment, this parameter indicates the location in the segment as it varies from a value of zero at the beginning of the segment to a value of unity at the end.

2.2.2 *Mathematical expression.* In particular, let the control polygon be composed of the set of control vertices

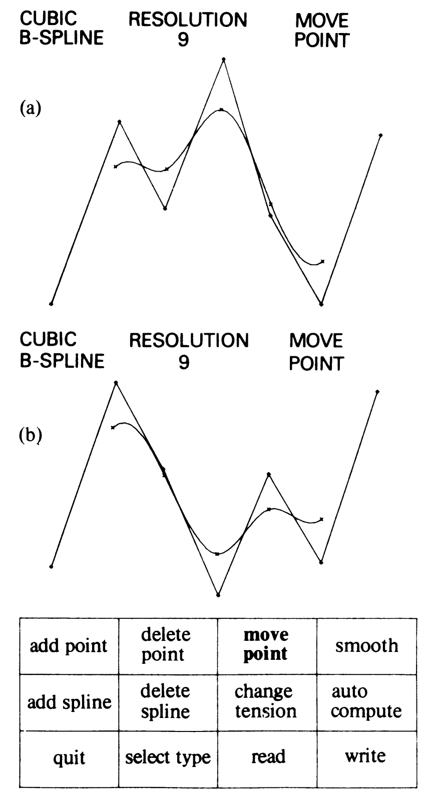


Fig. 1 Local control of B-splines. The menu is shown at the foot

$\{\mathbf{V}_{-1}, \mathbf{V}_0, \ldots \mathbf{V}_{m+1}\}$. The coordinates of the point $\mathbf{Q}_i(u)$ on the *i*th uniform cubic B-spline curve segment are then given by

$$\mathbf{Q}_i(u) = \sum_{r=-2}^{1} b_r(u)\, \mathbf{V}_{i+r} \qquad 0 \leqslant u < 1 \tag{6}$$

The weighting factors are the four scalar functions $b_{-2}(u)$, $b_{-1}(u)$, $b_0(u)$, and $b_1(u)$, evaluated at some value of the parameter u. A detailed derivation of these *univariate uniform cubic B-spline basis functions* and efficient algorithms for their evaluation are presented in Barsky (1981a). They can be written in matrix form as

$$[b_{-2}(u)\ b_{-1}(u)\ b_0(u)\ b_1(u)]$$
$$= [u^3 u^2 u 1]\ (1/6) \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \tag{7}$$

2.3 *Spline under tension*

2.3.1 *Explanation.* The cubic spline sometimes exhibits unnecessary oscillations due to 'extraneous' inflection points. In order to eliminate them, it is desirable to intuitively 'pull out' these points by increasing tension. This concept was first analytically modelled by Schweikert (1966) and an alternative development was given in Cline (1974) and generalised in Pilcher (1973). A detailed derivation of the generalised form based on a variational principle is given in Barsky (1981b).

Continuity of position and the second derivative vector are trivially achieved by their specification at each interior point and continuity of the first derivative vector is achieved by solving a system of equations for the set of values of the second derivative vector. In particular, let $\{\mathbf{P}_0^2, \mathbf{P}_1^2, \ldots \mathbf{P}_m^2\}$ be the set of values of the second derivative vector corresponding to the set of points given in Section 2.1.2. Denoting the tension over the ith curve segment as s_i, the ith curve segment can be written as

$$\mathbf{Q}_i(u) = \frac{\mathbf{P}^2_{i-1}\sinh[s_i(u_i - u)] + \mathbf{P}^2_i\sinh[s_i(u - u_{i-1})]}{s_i^2\sinh(s_i\Delta u_i)} + \left(\frac{\mathbf{P}_{i-1} - \mathbf{P}^2_{i-1}}{s_i^2}\right)\frac{u_i - u}{\Delta u_i} + \left(\frac{\mathbf{P}_i - \mathbf{P}^2_i}{s_i^2}\right)\frac{u - u_{i-1}}{\Delta u_i} \tag{8}$$

where

$$\Delta u_i = u_i - u_{i-1}$$

for

$$u_{i-1} \leqslant u < u_i \qquad i = 1, \ldots m$$

2.3.2 *Determining the set of values of the second derivative vector.* The continuity requirement of the first derivative vector at the interior point $\mathbf{P}_i$ is represented by

$$l_i\mathbf{P}^2_{i-1} + (n_i + n_{i+1})\,\mathbf{P}^2_i + l_{i+1}\,\mathbf{P}^2_{i+1} = \mathbf{R}_{i+1} - \mathbf{R}_i \tag{9}$$

where

$$l_i = \frac{v_i^{-1} - \operatorname{cosech} v_i}{s_i}$$

$$n_i = \frac{\coth v_i - v_i^{-1}}{s_i}$$

$$\mathbf{R}_i = \frac{\mathbf{P}_i - \mathbf{P}_{i-1}}{\Delta u_i}$$

$$v_i = s_i\Delta u_i \qquad i = 1, \ldots m - 1$$

This is a system of $m - 1$ linear equations for the $m + 1$ unknown values of the second derivative vector. To find a unique solution, two more equations are needed, and this can be fulfilled by specifying an end condition at both $\mathbf{P}_0$ and $\mathbf{P}_m$. The complete system of equations for both the first derivative vector specification end condition and the natural spline end condition are given in Barsky (1981b).

2.4 *ν-spline*

2.4.1 *Explanation.* An objection to the spline under tension is that it is expressed in terms of exponential functions rather than polynomials, which is a major impediment to efficient evaluation. To circumvent this problem, a polynomial alternative to the spline under tension was developed by Nielson (1974a; 1974b) which he called the ν-spline. It is derived in detail in Barsky (1981b) using the cubic Hermite basis functions approach, thereby emphasising its relation to the conventional cubic interpolatory spline.

It is important to note that each tension value for a ν-spline is associated with a *point* to be interpolated, not a spline curve *segment* as is the case with a spline under tension. A curve segment does, however, converge to a straight line segment as the tension values at both endpoints are increased. In addition, the number of tension values for a ν-spline is therefore one more than that for a spline under tension.

Unlike the spline under tension, the ν-spline does not have continuity of the second parametric derivative vector when the tension values are nonzero. The discontinuity in the second parametric derivative vector at an interior point to be interpolated is parallel to the corresponding first derivative vector, and the ratio of the respective magnitudes is the tension value there. Specifically,

$$\mathbf{Q}^{(2)}_{i+1}(u_i) - \mathbf{Q}^{(2)}_i(u_i) = t_i\,\mathbf{Q}^{(1)}_{i+1}(u_i) = t_i\,\mathbf{Q}^{(1)}_i(u_i) \tag{10}$$

where t_i is the tension value on the point $\mathbf{P}_i$ for $i = 1, \ldots m - 1$. Note that the magnitude of this discontinuity reduces to zero for a zero tension value, corresponding to the continuous second parametric derivative vector of the cubic interpolatory spline. However, the ν-spline does have geometric second derivative continuity; that is, $\mathrm{d}^2y_\mathrm{Q}/\mathrm{d}x^2_\mathrm{Q}$ is continuous where $x^{(1)}_\mathrm{Q}(u)$ is nonzero, and similarly $\mathrm{d}^2x_\mathrm{Q}/\mathrm{d}y^2_\mathrm{Q}$ is continuous where $y^{(1)}_\mathrm{Q}(u)$ is nonzero. This implies that the ***curvature*** of a ν-spline is also continuous (Barsky, 1981c; Nielson, 1974b).

2.4.2 *Determining the set of values of the first derivative vector.* Since each ν-spline curve segment is a cubic polynomial, the spline can be represented by cubic Hermite interpolation (see Section 2.1.2). Now the set of values of the first derivative vector are determined such that the resulting cubic Hermite curve is a ν-spline.

Recall that for the interpolatory spline, the values of the first derivative vector at the interior points were determined by solving a system of equations representing the continuity of the second derivative vector. Although the ν-spline does not generally have continuity of the second parametric derivative vector, the amount of discontinuity at an interior point to be interpolated is given by Eqn (10); therefore, this equation can be used to determine the values of the first derivative at the interior points. Using Eqns (3) and (4) to evaluate both terms on the left hand side of Eqn (10), and performing some algebraic simplification results in:

$$\Delta u_{i+1}\mathbf{P}^1_{i-1} + \left[2(\Delta u_i + \Delta u_{i+1}) + \frac{t_i}{2}\right]\mathbf{P}^1_i + \Delta u_i\mathbf{P}^1_{i+1} = 3\left\{(\mathbf{P}_i - \mathbf{P}_{i-1})\frac{\Delta u_{i+1}}{\Delta u_i} + (\mathbf{P}_{i+1} - \mathbf{P}_i)\frac{\Delta u_i}{\Delta u_{i+1}}\right\} \tag{11}$$

$$i = 1, \ldots m - 1$$

This is a system of $m - 1$ equations for the $m + 1$ unknown values of the first derivative vector. To find a unique solution, two more equations are needed, and this can be fulfilled by specifying an end condition at both $\mathbf{P}_0$ and $\mathbf{P}_m$. The complete system of equations for two different end conditions are provided in Barsky (1981b).

3. Transformations

3.1 *B-spline to interpolatory spline*

The transformation from a B-spline to interpolatory spline formulation can be accomplished by determining a set of points on the curve and the value of the first derivative vector at the beginning and end of the B-spline.

In order to find the identical curve, the set of points to be input to the interpolatory spline formulation must be the endpoints of the B-spline curve segments. Specifically,

$$\begin{aligned}\mathbf{P}_0 &= \mathbf{Q}_1(0) \qquad \text{for } i = 0\\ \mathbf{P}_i &= \mathbf{Q}_i(1) \qquad \text{for } i = 1, \ldots m\end{aligned} \tag{12}$$

Explicit expressions for these points in terms of the known control vertices can be derived by evaluating the right hand side of each equation using Eqn (6) at extreme values of the parameter u. In particular,

$$\mathbf{P}_i = (\mathbf{V}_{i-1} + 4\mathbf{V}_i + \mathbf{V}_{i+1})/6 \qquad \text{for } i = 0, \ldots m \tag{13}$$

The value of the first derivative vector at the beginning and end of the B-spline is

$$\mathbf{P}^1_0 = \mathbf{Q}^{(1)}_0(0) \qquad \mathbf{P}^1_m = \mathbf{Q}^{(1)}_m(1) \tag{14}$$

Differentiating Eqn (6) with respect to u, evaluating at the beginning and end of the B-spline, and substituting into the right hand side of each equation yields the following explicit expressions for the first derivative vector:

$$\mathbf{P}^1_0 = (\mathbf{V}_1 - \mathbf{V}_{-1})/2 \qquad \mathbf{P}^1_m = (\mathbf{V}_{m+1} - \mathbf{V}_{m-1})/2 \tag{15}$$

3.2 *Interpolatory spline to B-spline*

This transformation reduces to determining an appropriate set of B-spline control vertices which must be supplied to the B-spline formulation to generate an interpolating curve. The theory for accomplishing this for a surface was originally developed by Barsky (1979) and Barsky and Greenberg

(1980). The identical notation is used in the following results which can be derived as the simplified case of a curve.

The interpolation conditions can be represented by the following $m + 1$ equations:

$$\begin{aligned} \mathbf{Q}_1(0) &= \mathbf{P}_0 \quad \text{for } i = 0 \\ \mathbf{Q}_i(1) &= \mathbf{P}_i \quad \text{for } i = 1, \ldots m \end{aligned} \tag{16}$$

Evaluating Eqn (6) at extreme values of the parameter u, substituting the resulting expression into the left hand sides of Eqn (16), and multiplying by 6 yields:

$$\mathbf{V}_{i-1} + 4\mathbf{V}_i + \mathbf{V}_{i+1} = 6\mathbf{P}_i \quad \text{for } i = 0, \ldots m \tag{17}$$

This is a system of $m + 1$ equations for the $m + 3$ unknown control vertices in terms of the known points to be interpolated. In order to completely determine the vertices, an additional two equations are required. The derivation of these equations is dependent upon the choice of end conditions.

3.2.1 *First derivative vector specification end condition.* The value of the first derivative vector at the beginning and end of the B-spline is now specified. That is,

$$\mathbf{Q}_1^{(1)}(0) = \mathbf{P}_0^1 \qquad \mathbf{Q}_m^{(1)}(1) = \mathbf{P}_m^1 \tag{18}$$

Evaluating the left hand side of each equation using Eqn (6) and multiplying by 2 results in:

$$\mathbf{V}_1 - \mathbf{V}_{-1} = 2\mathbf{P}_0^1 \qquad \mathbf{V}_{m+1} - \mathbf{V}_{m-1} = 2\mathbf{P}_m^1 \tag{19}$$

This pair of equations with Eqn (17) form a system of $m + 3$ equations for the $m + 3$ unknown control vertices. Although the matrix for this system is not precisely tridiagonal, it can be made symmetric, tridiagonal, strictly diagonally dominant, and smaller in rank by matrix manipulation, resulting in:

$$\begin{bmatrix} 2 & 1 & & & \\ 1 & 4 & 1 & & \\ & \cdot & \cdot & \cdot & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & 1 & 4 & 1 \\ & & & 1 & 2 \end{bmatrix} \begin{bmatrix} \mathbf{V}_0 \\ \mathbf{V}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{V}_{m-1} \\ \mathbf{V}_m \end{bmatrix} = \begin{bmatrix} 3\mathbf{P}_0 + \mathbf{P}_0^1 \\ 6\mathbf{P}_1 \\ \cdot \\ \cdot \\ \cdot \\ 6\mathbf{P}_{m-1} \\ 3\mathbf{P}_m - \mathbf{P}_m^1 \end{bmatrix} \tag{20}$$

and

$$\begin{aligned} \mathbf{V}_{-1} &= \mathbf{V}_1 - 2\mathbf{P}_0^1 \\ \mathbf{V}_{m+1} &= \mathbf{V}_{m-1} + 2\mathbf{P}_m^1 \end{aligned}$$

3.2.2 *Natural spline end condition.* The value of the second derivative vector at the beginning and end of the B-spline is now set to zero.

$$\mathbf{Q}_1^{(2)}(0) = 0 \qquad \mathbf{Q}_m^{(2)}(1) = 0 \tag{21}$$

Evaluating the left hand side of each equation using Eqn (6) yields:

$$\mathbf{V}_{-1} - 2\mathbf{V}_0 + \mathbf{V}_1 = 0 \qquad \mathbf{V}_{m-1} - 2\mathbf{V}_m + \mathbf{V}_{m+1} = 0$$

Combining these two equations with Eqn (17) yields another system of $m + 3$ equations for the $m + 3$ unknown control vertices. This system can be written with a matrix which is not only symmetric, tridiagonal, and strictly diagonally dominant, but has the desirable 1–4–1 structure and is of even smaller rank.

$$\begin{aligned} \mathbf{V}_0 &= \mathbf{P}_0 \\ \mathbf{V}_m &= \mathbf{P}_m \end{aligned}$$

$$\begin{bmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \cdot & \cdot & \cdot & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \cdot \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{V}_{m-2} \\ \mathbf{V}_{m-1} \end{bmatrix} = \begin{bmatrix} 6\mathbf{P}_1 - \mathbf{V}_0 \\ 6\mathbf{P}_2 \\ \cdot \\ \cdot \\ \cdot \\ 6\mathbf{P}_{m-2} \\ 6\mathbf{P}_{m-1} - \mathbf{V}_m \end{bmatrix} \tag{23}$$

and

$$\begin{aligned} \mathbf{V}_{-1} &= 2\mathbf{V}_0 - \mathbf{V}_1 \\ \mathbf{V}_{m+1} &= 2\mathbf{V}_m - \mathbf{V}_{m-1} \end{aligned}$$

3.3. *Interpolatory spline to tensed spline*

The transformation from an interpolatory spline to a tensed spline is trivial since the former is a special case of the latter with all the tension values set to zero.

3.4 *Tensed spline to interpolatory spline*

If the tension values are zero, this transformation is trivial since the tensed spline and interpolatory spline are identical in this case. Otherwise, this transformation is nonsensical since an interpolatory spline is simply a tensed spline with all the tension values equal to zero; thus, it is not possible to perform this transformation and maintain the same curve shape. In this case the interpolatory spline can be generated from the same information which defines the tensed spline with tension values ignored, although this will not define the same curve shape.

3.5 *Tensed splines*

Since the two forms of tensed splines, the spline under tension and the ν-spline, are expressed in terms of exponential functions and polynomials, respectively, it is not possible to transform between them and retain the identical curve shape in the case where the tension values are nonzero. If the tension values are zero, both splines are identical to the cubic interpolatory spline and hence to each other. Furthermore, for a spline under tension, each tension value corresponds to a spline segment, whereas it corresponds to a point to be interpolated for a ν-spline, and the numerical range of the tension values is different for each of the two types of tensed splines.

4. The implementation

4.1 *Introduction*

The four types of spline curve formulations described in the first part of this paper have been implemented by the authors in a single system utilising an Evans and Sutherland Picture System. The initial implementation was under the RT-11 operating system on a DEC PDP-11/05 minicomputer using a Picture System I. The program has since been transported to a DEC PDP-11/34, and a Norsk Data NORD-10 under the SINTRAN III operating system using a Picture System II. It is currently in the process of being implemented under the UNIX operating system. The implementation provides for the interactive entry and modification of the four types of spline curves, as well as for the transformations among them, as explained in Section 3. The system implementation will now be described; further details of the TRANSPLINE system can be found in Barsky and Thomas (1980).

The system consists of two major divisions: spline type selection and transformation, and spline entry and modification. Interaction with each division is through a digitising tablet and a corresponding menu shown on the display. A menu button may be selected or 'picked' by positioning the cursor within the box and depressing the tablet pen. Certain buttons (for example, **quit**) cause an immediate action to occur; while others (for example, **add spline**) merely determine a mode, in which case the desired action is subsequently initiated by depressing the pen with the cursor outside the menu area (within the picture area). A 'mode line' is displayed at the top of the screen at all times, informing the user of the exact state of the system. The first (leftmost) field of the mode line contains the name of the type of spline currently selected. The second (centre) field displays the 'resolution' value; that is, the number of line segments to be drawn for each splint segment; this is set by a 'graphical potentiometer' utilising the tablet pen. The third (rightmost) field always describes the action which would occur if the tablet pen were depressed with the cursor outside the menu area.

4.2 *Sample design session*

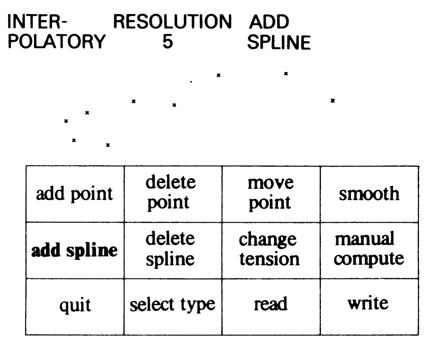
This section presents a sample design session using the TRANSPLINE system. The sequence followed here is initial curve entry using interpolatory splines, curve modification to more closely approximate the desired shape using B-splines, and final shape definition using splines under tension and ν-splines. The selection of a new type of spline automatically transforms previously entered splines to the selected type. This is performed with as little loss of information as possible. For example, when transforming from a tensed spline (spline under tension or ν-spline) to an untensed spline, tension information is ignored, although it will be regained if a tensed spline type is again selected.

Points are entered by placing the cursor at the desired location and depressing the pen which causes a cross to appear marking the position of the point (**Fig. 2**). The curve can now be displayed by picking the **manual compute** button, causing the system to compute the interpolatory spline passing through the points which have been entered (**Fig. 3**).

In order to avoid continually selecting the **manual compute** button to see the effect of each change to the splines, the system can be put into **auto compute** mode. Any subsequent changes to the curves will be immediately reflected in the display by automatically recomputing the splines after each action which modifies one of them. This is generally the preferred mode of operation, but the **manual compute** option is offered for cases where the recomputation time becomes unreasonably long. This choice of modes is unnecessary for B-splines which are always in **auto compute** mode, since they can be recomputed rapidly.

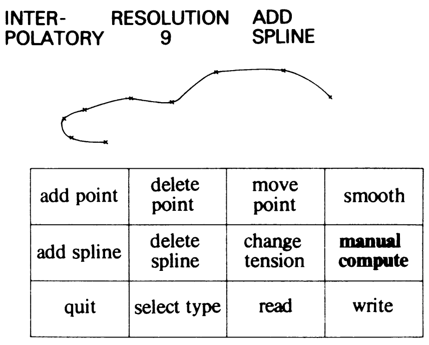
After entering this spline, it is observed that it does not yet offer a sufficiently close approximation to the desired shape; thus, it is modified by moving some points (**Fig· 4**). The **move point** function can be used to modify the position of an interpolated point or B-spline control vertex, or the value of the first derivative vector at either endpoint. The point or vertex which is to be moved is picked by depressing the pen near it, and it then tracks the motion of the cursor until the pen is depressed again. The action is thus that of picking up the point, moving it, and setting it down again. A first derivative vector handle is picked by depressing the pen near either of its endpoints, and it is then moved in the same manner as was described for points.

The shape modification procedure is continued by transforming to a B-spline formulation and moving some of the B-spline control vertices (**Fig. 5**). This process is facilitated by two properties of the B-spline formulation. First, the local control inherent in B-splines restricts the effects of modifying the position of a single control vertex to a small, predeter-



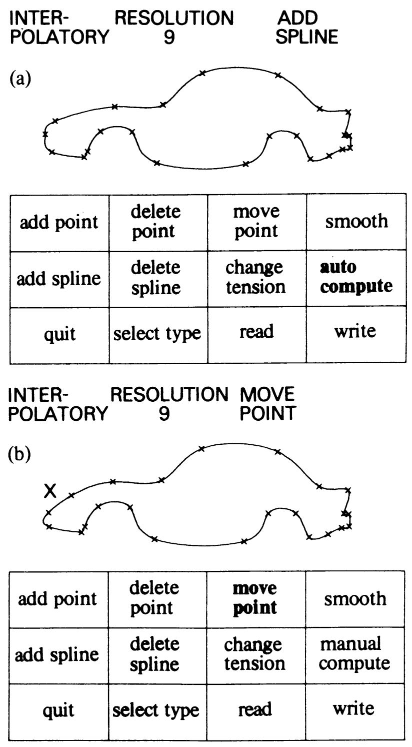
add point	delete point	move point	smooth
add spline	delete spline	change tension	manual compute
quit	select type	read	write

Fig. 2 After entering a few points



add point	delete point	move point	smooth
add spline	delete spline	change tension	**manual compute**
quit	select type	read	write

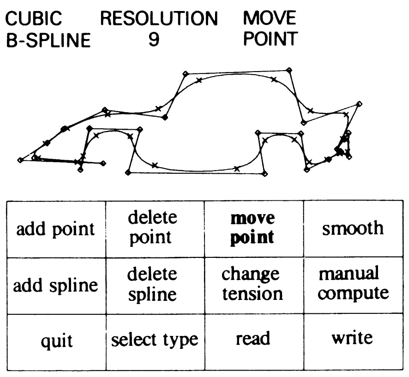
Fig. 3 A computed spline



add point	delete point	move point	smooth
add spline	delete spline	change tension	**auto compute**
quit	select type	read	write

add point	delete point	**move point**	smooth
add spline	delete spline	change tension	manual compute
quit	select type	read	write

Fig. 4 Modifying the spline by moving some points



add point	delete point	**move point**	smooth
add spline	delete spline	change tension	manual compute
quit	select type	read	write

Fig. 5 The spline as a B-spline

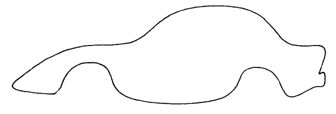
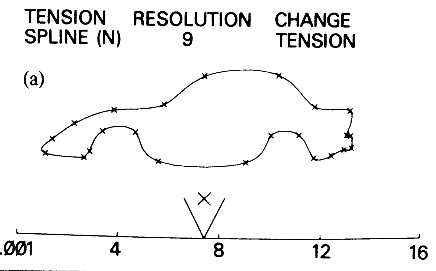
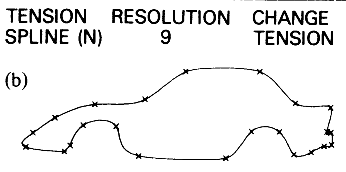


Fig. 6 A smoothly displayed B-spline curve



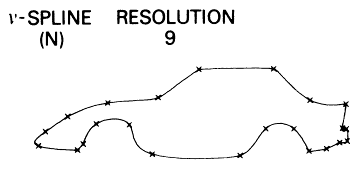


add point	delete point	move point	smooth
add spline	delete spline	**change tension**	manual compute
quit	select type	read	write

Fig. 7 Changing the tension of a spline under tension: straightening the front of the rear wheel well. (a) Changing the tension; (b) the resulting spline

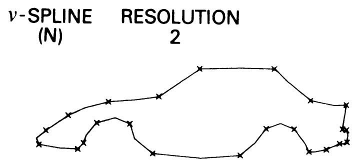
mined portion of the curve (**Fig. 1**). Second, the B-splines are dynamically modified as the control vertex is moved, whereas all the other types of splines are not recomputed until at least the conclusion of the move operation. B-spline modification in realtime is achieved by exploiting several properties of the B-spline formulation so as to avoid a complete recomputation of the curve. The new curve resulting from the movement of a single control vertex can be computed using a 'perturbation' algorithm designed by Barsky (1981a).

To establish a better perception of the exact shape of the spline curve, it can be displayed as a smooth curve using a recursive subdivision algorithm (**Fig. 6**). The subdivision is performed by the Lane and Riesenfeld (1980) B-spline subdivision algorithm applied to each spline segment. An interpolatory spline can also be subdivided by first transforming to the B-spline formulation, which is done automatically.



zoom	resolution	multiple tension	main menu
inter-polatory	cubic B-spline	tension spline	**ν-spline**

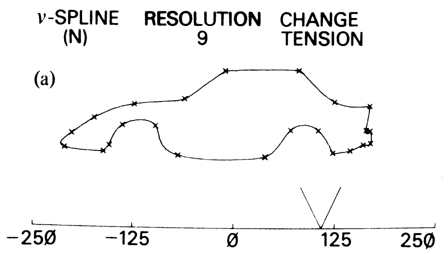
Fig. 8 Changing to a ν-spline

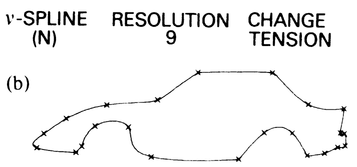


zoom	**resolution**	multiple tension	main menu
inter-polatory	cubic B-spline	tension spline	ν-spline

Fig. 9 Changing the resolution value

Transforming to a spline under tension, the spline segment whose tension value is to be modified is picked. This is done by placing the cursor on the desired segment and depressing the pen. The selected segment is illuminated for identification while a scale with a sliding pointer is displayed at the bottom of the screen **Fig.** 7(a). The system sets the pointer to the current value of the tension, and it can then be moved by positioning the cursor within the 'V' of the pointer. When the pen is depressed, the tension value is set and the menu





add point	delete point	move point	smooth
add spline	delete spline	**change tension**	manual compute
quit	select type	read	write

Fig. 10 Changing the tension of a ν-spline: sharpening the lower front corner of the rear wheel well. (a) changing the tension; (b) the result of the change

reappears [**Fig. 7** (**b**)]. If the system had been in single tension mode (a single tension value over the entire curve), then the entire spline would have been picked and modified, rather than only one segment.

Since each tension value in a spline under tension is applied over a spline *segment*, increasing a tension value tends to 'flatten' the segment. This differs from a ν-spline, for which an increased tension has the effect of 'sharpening' a corner.

The spline is now transformed to a ν-spline with zero tension values [**Fig. 8**]. The number of line segments used to display each curve segment can be adjusted in order to provide either a faster display update, by using a smaller number of line segments, or a more accurate display, with more line segments. The resolution is changed by moving the cursor from left to right across the screen, until the desired resolution value is shown in the mode line (**Fig. 9**). Since the tension values for the ν-spline are associated with the interpolated points rather than the spline segments, the point at which the tension is to be changed is selected, and it is illuminated while the tension modification is in progress (**Fig. 10**).

A ν-spline is allowed to have negative tension values, which can generate some interesting curves [**Fig. 11**(a)].

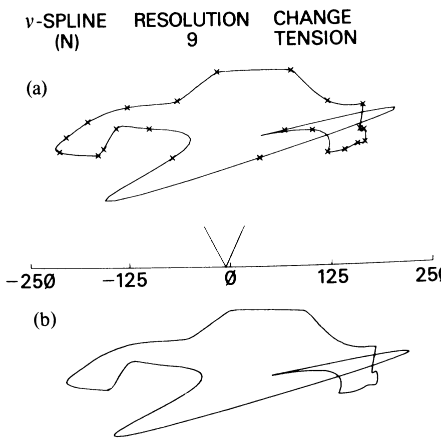


Fig. 11 (a) Negative tension on a ν-spline; (b) smoothly displayed ν-spline

Although the numerical ranges of the tensions are quite different for the two tensed spline formulations, the effects of the maximum values of each are quite similar. In general, a ν-spline does not have the second derivative continuity which is required for transformation to a B-spline for the subdivision process. However, it can still be displayed by this technique by applying the subdivision to one segment at a time, and each ν-spline segment can be expressed as a B-spline because it is a cubic polynomial [**Fig. 11**(**b**)].

The system has the capability of displaying and modifying several splines at once; thus, it can be used to form images of greater complexity than would be possible with a single spline. In **Fig. 12** several splines have been added to the original automobile profile to create the final image.

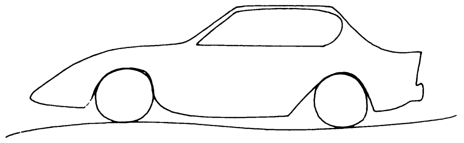


Fig. 12 Final automobile profile consisting of several splines

5. Conclusion

Each of the various techniques of curve representation which have been developed in the field of computer aided geometric design has properties which are more useful in certain phases of the design process than in others. It is therefore very useful to be able to transform, with as little loss of shape information as possible, between the different formulations in order to select the one most appropriate to the situation at hand.

The TRANSPLINE system is an interactive curve representation system which implements this concept with a set of four parametric spline curve formulations. Each member of this set was selected on the basis of its appropriateness to a different design phase; thus, some are better suited to initial data entry, while others are ideal for modification of already entered data, and others are able to supply a particular curve property which the remaining formulations lack. The result is a system which is quite versatile and capable of representing complex shapes, while remaining fairly simple and easy to use.

Acknowledgement
This work was supported in part by the National Science Foundation under grant number MCS-78-01966.

References

Barsky, B. A. (1979). A method for describing curved surfaces by transforming between interpolatory spline and B-spline representations. Master's Thesis, Cornell University, Ithaca, NY.

Barsky, B. A. (1981a). A study of the parametric uniform B-spline curve and surface representations. Submitted.

Barsky, B. A. (1981b). Exponential and polynomial methods for applying tension to an interpolating spline curve. Submitted.

Barsky, B. A. (1981c). The β-spline: A local representation based on shape parameters and fundamental geometric measures, PhD Thesis, University of Utah, Salt Lake City, Utah, to appear.

Barsky, B. A. and Greenberg, D. P. (1980). Determining a set of B-spline control vertices to generate an interpolating surface, *Computer Graphics and Image Processing*, Vol. 14 No. 3, pp. 203-226.

Barsky, B. A. and Thomas, S. W. (1980). TRANSPLINE curve representation system, Tech. Report No. UUCS-80-104, Department of Computer Science, University of Utah.

Cline, A. K. (1974). Scalar- and planar-valued curve fitting using splines under tension, *Communications of the ACM*, Vol. 17 No. 4, pp. 218–220.

Lane, J. M. and Riesenfeld, R. F. (1980). A theoretical development for the computer generation of piecewise polynomial surfaces, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2 No. 1, pp. 35–46.

Nielson, G. M. (1974a). Computation of ν-splines, Tech. Report No. 044-433-11, Department of Mathematics, Arizona State University.

Nielson, G. M. (1974b). Some piecewise polynomial alternatives to splines under tension, in *Computer Aided Geometric Design*, edited by R. E. Barnhill and R. F. Riesenfeld, pp. 209–235. Academic Press, New York.

Pilcher, D. T. (1973). Smooth approximation of parametric curves and surfaces, PhD Thesis, University of Utah, Salt Lake City, Utah.

Riesenfeld, R. F. (1973). Applications of B-spline approximation to geometric problems of computer-aided design, PhD Thesis, Syracuse University, Syracuse, NY. Available as Tech. Report No. UTEC-Csc-73-126, Department of Computer Science, University of Utah.

Rogers, D. F. and Adams, J. A. (1976). *Mathematical Elements for Computer Graphics*, McGraw-Hill, New York.

Schultz, M. H. (1973). *Spline Analysis*, Series in Automatic Computation, Prentice-Hall, Englewood Cliffs, NJ.

Schweikert, D. G. (1966). An interpolation curve using a spline in tension, *Journal of Mathematics and Physics*, Vol. 45, pp. 312–317.