The (2-5-2) Spline Function

Jae H. Park and Leonard A. Ferrari

Abstract. Splines have been used extensively in the interpolation of multidimensional data sets. Linear interpolation utilizes second order splines (first degree piecewise polynomials) and has widespread popularity because of its ease of implementation. Cubic splines are often used when higher degrees of smoothness are required of the interpolation process. Linear interpolation has the advantages of not requiring the solution of an inverse problem (the data points are themselves the coefficients of the triangular basis functions) and extremely efficient generation of the output sample points. Unfortunately, the linear-interpolating function has only C^0 continuity (the function is continuous but its derivatives are discontinuous) and therefore lacks the required smoothness for many applications. We provide a new algorithm in this paper based on the efficient derivative summation approach to spline rendering. Cubic B-spline interpolation for uniformly spaced data points provides C^2 continuity. The interpolation function can be rendered quite efficiently from the basis coefficients and the basis function, using a cascade of four running average filters. Unser *et al.* have shown a digital filter solution for the inverse problem of obtaining the spline coefficients from the data points. A matrix inversion solution is also commonly used. Both solutions require the use of floating point multiplication and addition, while the forward problem can be implemented utilizing only fixed-point additions. In this paper, we develop a class of spline basis functions which solve the interpolation problem using only simple arithmetic shifts and fixed point additions for solutions to both the forward and inverse problems. The system impulse response for the new interpolators appears to be closer to the ideal interpolator than the B-spline interpolator. We refer to the new splines as (2-5-2) splines

§1. Introduction

Splines are well-accepted in Computer Graphics[1,2,3,4]. The high computational requirements of cubic splines and the large amounts of data make them difficult to use in multi-dimensional applications. In many applications, bilinear interpolation is used instead of cubic splines because of its simplicity in implementation. However, bilinear interpolation cannot produce images of sufficient quality for many application because of its C^0 continuity. The

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 307–314.

ISBN 0-8265-1356-5.

(2-5-2) spline approach can be computed at nearly the same efficiency as the bilinear interpolator, but provides much higher quality results.

In general, spline computation can be divided into two parts. The first part requires the solution of an inverse problem to obtain the coefficients (vertices) of the basis functions, and the second part is a forward computation to generate the interpolating spline from its coefficients.

§2. Inverse Problem

Assume we are given $(m-r+3)$ data values P_i, $i = r-1, \ldots, m+1$. Further assume that we wish to find a continuous spline curve such that on a specified set of equally spaced points, $\bar{u}_i$ $i = r-1, \ldots, m+1$, the curve attains the values P_i. Let $B_{i,r}(\bar{u})$ denote an rth order set of basis splines defined on the knot set $i = 0, \ldots, m$. Then, for the case of a curve in one-dimension we can write

$$Q(\bar{u}_p) = \sum_{i=0}^{m} V_i B_{i,r}(\bar{u}_p) = P_p, \qquad p = k-1, \ldots, m+1. \tag{1}$$

This represents $m-r+3$ equations in $m+1$ unknowns. In the case of a cubic spline, $r = 4$ and we are short two equations. We are free to augment the data set P_i at both ends with additional data values. However, these augmented data values effect the shape of the interpolating function $Q(\bar{u})$ non-locally.

In matrix form, we obtain

$$\begin{bmatrix} B_{0,r}(\bar{u}_{r-2}) & \cdots & B_{m,r}(\bar{u}_{r-2}) \\ B_{0,r}(\bar{u}_{r-1}) & & B_{m,r}(\bar{u}_{r-1}) \\ \vdots & & \vdots \\ B_{0,r}(\bar{u}_{m+2}) & \cdots & B_{m,r}(\bar{u}_{m+2}) \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_m \end{bmatrix} = \begin{bmatrix} P_{r-2} \\ P_{r-1} \\ \vdots \\ P_{m+2} \end{bmatrix} \tag{2}$$

or

$$B\underline{V} = \underline{P},$$

where $\underline{V}$ denotes the vector of unknown coefficients, B represents the matrix of basis spline values at the knot locations and $\underline{P}$ represents the given set of data points including the augmented values. The solution to (2) exists whenever the matrix B has an inverse. The solution to (2) is efficient whenever $B^{-1}\underline{P}$ or its equivalent is easy to compute.

For the cubic B-spline defined on uniform knots, it is easy to show that the interpolation problem utilizing simple, uniform knot B-splines leads to the inversion problem

$$\frac{1}{6}\begin{bmatrix} 4 & 1 & 0 & . & . & 0 \\ 1 & 4 & 1 & 0 & . & 0 \\ 0 & 1 & 4 & 1 & . & 0 \\ . & . & . & . & . & . \\ 0 & . & 0 & 1 & 4 & 1 \\ 0 & . & . & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} V_0 \\ . \\ . \\ . \\ . \\ V_m \end{bmatrix} = \begin{bmatrix} P_2 \\ . \\ . \\ . \\ . \\ P_{m+2} \end{bmatrix}. \tag{3}$$

In this case the matrix B can be inverted using forward elimination and backwards substitution as in [1]. While the B-spline matrix is banded and can therefore be inverted reasonably efficiently, the process requires full floating point multiplication and addition. Even after the matrix inversion, to obtain the vertices, full floating point multiplication and addition must be used.

The first attempt at reducing the number of the calculations was made by Unser, *et al* with a digital filtering solution to the problem by noting that the inverse of the FIR filter given by $B(z) = z + 4 + z^{-1}$ is the filter with impulse response [3]

$$b^{-1}(n) = \frac{-6\alpha}{(1-\alpha^2)}\alpha^{|n|}, \tag{4}$$

where, $\alpha = \sqrt{3} - 2$ is the smallest root in absolute value of the polynomial $z^2 + 4z + 1$.

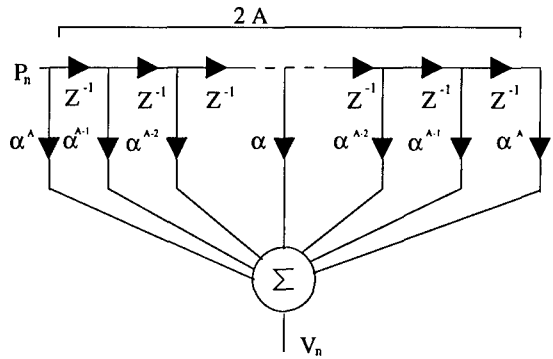
As they implemented it, the above IIR filter should be split into causal and non-causal parts, and applied twice to obtain the vertices: the non-causal sequences and anti-causal sequences. Since $b^{-1}(n)$ becomes smaller as $|n|$ gets larger, we can assume the filter has only several non-zero elements from the center of the filter sequences. Thus, we can approximate the filter with an FIR filter as below, and call it the Inversion FIR filter:

$$b^{-1}(n) = \begin{cases} \frac{-6\alpha}{(1-\alpha^2)}\alpha^{|n|}, & \text{if } |n| \leq m, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Even with the Inversion FIR filter, we still need full floating point multiplication and addition to get the vertices. The (1-4-1) spline does not provide a simple solution to the coefficient inverse problem because its characteristic polynomial, $z^2 + 4z + 1$, has irrational roots. The polynomials $2z^2 + 5z + 2$ has roots which are negative powers of two. We refer to these splines as (2-5-2) splines which have roots which are of similar magnitude to the roots of the (1-4-1) spline.

We assume the spline is defined on the uniform set $\{-2, -1, 0, 1, 2\}$, and that it takes on the set of values $\{0, \frac{2}{9}, \frac{5}{9}, \frac{2}{9}, 0\}$ at the knots. In Sect. 6, we set up sixteen equations to solve for the 16 polynomial coefficients defining the (2-5-2) spline. We note that it has C^2 continuity at knots 1, 2 and 3 and C^1 continuity at the knots at 0 and 4 (from the polynomials). Hence, the (2-5-2) spline is a multiple knot spline defined on seven knots with the knots at 0 and 4 having multiplicity two. We also note that the normalization property holds, that is $\sum_{i=0}^{3} P_i(\bar{u}) = 1$, for $u \in [0, 1]$.

If, as in [3], we assume periodic boundary conditions, equation (2) takes the following circulant form:



A : # of bits in the fixed point number

V_n: Vertices

P_n: Sampled data

Fig. 1. The inverse filter for a (2-5-2) spline: $\alpha = -1/2$.

$$\frac{1}{9}\begin{bmatrix} 5 & 2 & 0 & . & . & 0 & 2 \\ 2 & 5 & 2 & 0 & . & . & 0 \\ 0 & 2 & 5 & 2 & . & . & 0 \\ 0 & . & . & . & . & . & . \\ . & . & . & . & 2 & 5 & 2 \\ 2 & 0 & . & . & 0 & 2 & 5 \end{bmatrix} \begin{bmatrix} V_0 \\ . \\ . \\ . \\ . \\ V_m \end{bmatrix} = \begin{bmatrix} P_2 \\ . \\ . \\ . \\ . \\ P_{m+2} \end{bmatrix} \tag{6}$$

or

$$\frac{1}{9} B_5 \underline{V} = \underline{P}. \tag{7}$$

With the approach Unser, *et al* applied to (1-4-1) splines, and the FIR filter approximation, we can obtain a Inversion FIR filter for (2-5-2) splines and the FIR filter will be

$$b^{-1}(n) = \begin{cases} 3\left(-\frac{1}{2}\right)^{|n|}, & \text{if } |n| \le m, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The filter can be implemented as shown in Fig. 1 with delays, mulipliers and adders. Although the filter shown in Fig. 1 can be used for a (1-4-1) spline with change of $\alpha = \sqrt{3} - 2$, the FIR filter for a (2-5-2) spline can be implemented with an integer processer because all its coefficients are powers of 1/2, and the power of two multiplication can be realized with shifts.

§3. Forward Problem

Ferrari, *et al.*, provided an efficient algorithm (the Fast Spline Transform, (FST)) for computing a spline by Derivative/Summation [5]. Once we obtain

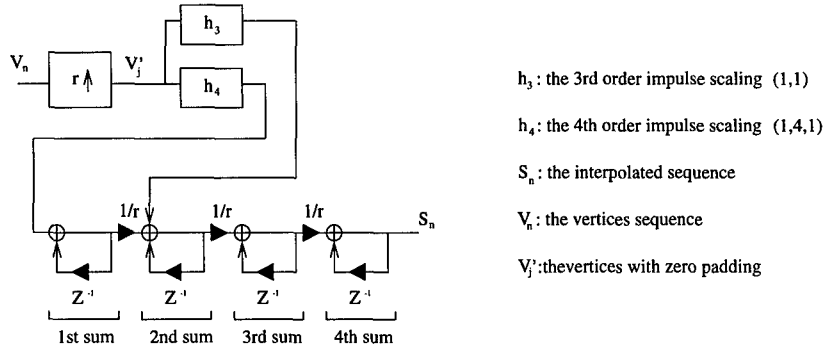


Fig. 2. The FST filter for (2-5-2) spline calculation.

the vertices, the spline is calculated by the FST with appropriate Impulse Scaling. The FST is as follows, where r is the order of the basis function and m is the interpolation ratio:

Algorithm FST

1) Find the Dirac functions corresponding to all orders of the rth order spline's derivatives $(r, r-1, \cdots, 1)$ [4].
2) Create an array of zeros with $m-1$ elements between the knot locations; initialize $k = r$.
3) Scale the appropriate Dirac functions by amplitude V_n, and sum the resulting sequence into the array at the knot locations corresponding to the Dirac functions(requires shift and add for the (2-5-2) spline).
4) "Integrate" the resulting sequence once using repeated summation. Set $k = k - 1$.
5) Return to step 3. Until $k = 0$.
6) The array contains the values of the spline at the specified locations.

The FST algorithm can be implemented as a digital filter for any spline. However, to implement this with only fixed point shifts and additions, every Impulse Scaling element(coefficient) of the spline must be powers of two. The suggested filter (the FST filter) for the (2-5-2) spline is shown in Fig. 2. Because the (2-5-2) spline has double knots, both 4th and 3rd order impulse sets exist. h_4 and h_3 in Fig. 2 correspond to 4th order impulse scaling and 3rd order impulse scaling. Fig. 2 shows clearly that the forward computation of the spline is computed at the input data rate.

§4. Cosine Examples

Since the Impulse Scaling approach for the forward problem always generates any (2-5-2) spline curves on the defined grids, the Impulse Scaling will not effect the accuracy of the generated curves by each Inverse Problem scheme

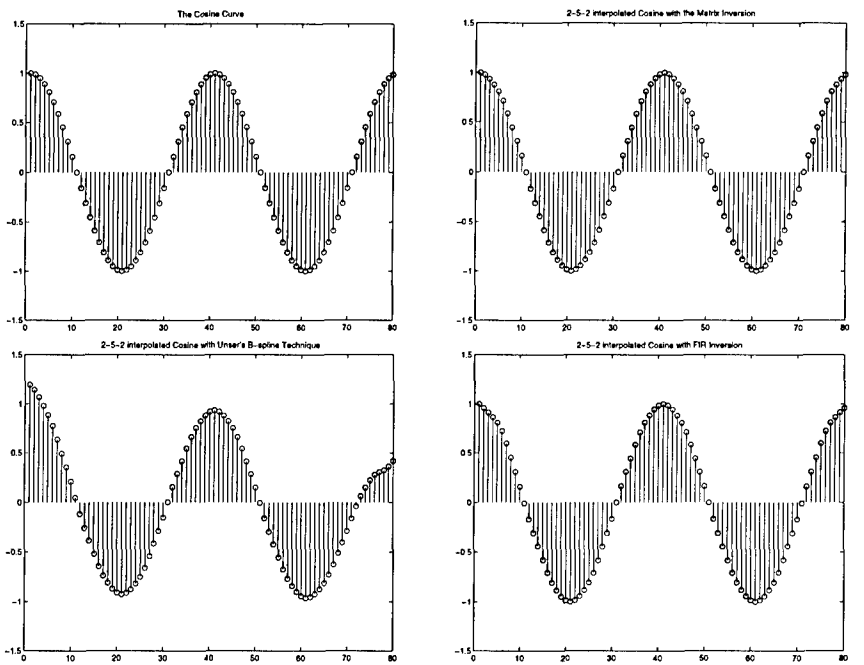


Fig. 3. The cosine interpolated examples: Top Left: True Cosine Curve, Top Right: by Matrix Inversion, Bottom Left: by Unser's Inversion and by FIR inversion.

(i.e., Matrix Inversion, Unser's IIR Filter Inversion and FIR Filter Inversion). The inversion schemes affect it. Unser's IIR filter Inversion and FIR filter Inversion can be considered as an approximation operation of matrix inversion. As shown in Fig. 3, the cosine curve generated by matrix inversion is closest to the actual cosine curves because (2) guarantees that the (2-5-2) spline curve passes through every data point (P_n's). Although the two filter approaches are not guaranteed to pass through every data point, they all produce fairly accurate curves in the middle section. For the (2-5-2) spline, they can be implemented with an integer process because the Filter coefficients of IIR filter and FIR filter's are powers of two. It appears that FIR filter inversion generates more accurate cosine than Unser's IIR filter Invesion in Fig. 3, while the FIR filter is nothing but the approximation of the IIR filter. The initial value estimation for the anti-causal realization of the IIR filter results in less accuracy. It can be confirmed that significant distortion appears at the right end portion of the IIR filter inversion cosine interpolated curve in Fig. 3.

§5. Discussion

With either FIR Filter Inversion or IIR Filter Inversion, the (2-5-2) spline interpolation will generates more accurate curves or surfaces than the (1-4-1)

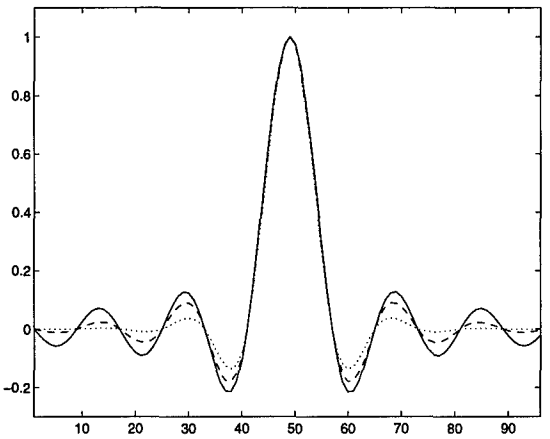


Fig. 4. The impulse responses of a (1-4-1) spline(the dotted line), a (2-5-2) spline(the dashed line) and an ideal interpolator(the solid line).

spline interpolation does. This can be easily confirmed by the fact that the (2-5-2) spline's impulse response is much closer to the sinc function than that of the (1-4-1) spline (refer to Fig. 4). Because the sinc function is the base of the perfect interpolation by the sampling therom, the (2-5-2) spline produces more accurate interpolated results than the B-spline. In addition, the (2-5-2) can be implemented with only fixed point shifts and additions. Therefore, the (2-5-2) spline is at least as accurate as the B-spline, and can be computed much more efficiently.

§6. Polynomial Coefficients for the (2-5-2) Spline

The spline is defined on five uniformly spaced knots $\{\bar{u_0}, \bar{u_1}, \bar{u_2}, \bar{u_3}, \bar{u_4}\}$. For each interval, we define $u \in [0,1]$ as the polynomial variable. Then for each interval, the coefficients $\{a_i, b_i, c_i, d_i\}$ represent the polynomial $a_i + b_i u + c_i u^2 + d_i u^3$. We denote the four polynomials by $p_0(u), p_1(u), p_2(u)$, and $p_3(u)$.

We impose the following sixteen constraints:

i. $P_0(0) = 0$

ii. $P_3(1) = a_3 + b_3 + c_3 + d_4 = 0$

iii. $P_0'(0) = b_0 = 0$

iv. $P_3'(0) = b_3 + 2c_3 + 3d_3 = 0$

v. $P_0(1) = a_0 + b_0 + c_0 + d_0 = \frac{2}{9}$

vi. $P_1(0) = a_1 = \frac{2}{9}$

vii. $P_2(1) = a_2 + b_2 + c_2 + d_2 = \frac{2}{9}$

viii. $P_3(0) = a_3 = \frac{2}{9}$

ix. $P_1(1) = a_1 + b_1 + c_1 + d_1 = \frac{5}{9}$

x. $P_2(0) = a_2 = \frac{5}{9}$

xi. $P_0'(1) = b_0 + 2c_0 + 3d_0 = P_1'(0) = b_1$

xii. $P_1'(1) = b_1 + 2c_1 + 3d_1 = P_2'(0) = b_2$

xiii. $P_2'(1) = b_2 + 2c_2 + 3d_2 = P_3'(0) = b_3$

xiv. $P_0''(1) = 2c_0 + 6d_0 = P_1''(0) = 2c_1$

xv. $P_1''(1) = 2c_1 + 6d_1 = P_2''(0) = 2c_2$

xvi. $P_2''(1) = 2c_2 + 6d_2 = P_3''(0) = 2c_3$

References

1. Bartels, R., J. Beatty, and B. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.*, Morgan Kaufman publishers, 1987, Los Altos, California.
2. de Boor, C., *A Practical Guide to Splines*, Applied Mathematical Sciences, Vol. 27, Springer-Verlag, New York, 1978.
3. M. Unser, A. Aldroubi, and M. Eden, Fast B-Spline transforms for continuous image representation and interpolation, IEEE Transactions on Pattern Analysis and Machine Intelligence, **13**, No.3, March (1991), 277–285.
4. Sankar, P., M. Silbermann, and L. Ferrari, Curve and surface generation and refinement based on a high speed derivative algorithm, CVGIP: Graphical Models and Image Processing, **56**, No. 1, (1994), 94–101.
5. Ferrari, L. A., J. H. Park, A. Healey and S. Leeman, Interpolation using fast spline transform (FST), IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, **46**, (1999), 891–906

Jae H. Park and Leonard A. Ferrari
The Bradley Department of Electrical Computer Engineering
340 Whittemore Hall
Virginia Tech
Blacksburg, VA 24061-0111
ferrari@vt.edu