Fast Cubic B-Spline Interpolation

James A. Boatright

Keywords: Fast cubic B-spline interpolation of band-limited uniformly sampled signals, image re-sampling, high-spatial-frequency pre-enhancement.

Background

This paper was originally written as a letter to my Lockheed supervisor and NASA Task Monitor on August 9, 1983, describing a then newly developed interpolating image re-sampling computer algorithm for use in geometrically correcting Landsat Multi-Spectral Scanner (MSS) and Thematic Mapper (TM) image data. I did this original development work while employed as an engineer with Lockheed Engineering and Management Services Company supporting the ERSYS Project of the National Aeronautics and Space Administration's Lyndon B. Johnson Space Center at Houston, Texas. Unfortunately, the project ended before this technique could be implemented and operationally tested, so the paper was never widely published. By the mid-1980's, the NASA budget was being squeezed to the point that they had to make a decision to give up science to keep the Space Shuttle Program running. While originally developed for use with space-borne, cross-track scanner imagery, this interpolative re-sampling technique is generally applicable for the processing of uniform sampling interval, band-limited digital data. Cubic B-spline interpolation is technically superior to other interpolation methods in many ways [Ref. 1].

Professor Michael Unser of EPFL in Lausanne, Switzerland, developed the same techniques independently for working with biomedical images in 1989. Professor Unser and his colleagues have since developed a fundamentally new approach to all of signal processing based on using B-splines. E-mail Michael.Unser@epfl.ch and URL: http://bigwww.epfl.ch/ . I recommend his paper from 1999 entitled: ***Splines: A Perfect Fit for Signal/Image Processing,*** which is available online. They have gone much beyond anything that I could have done had this paper been published earlier.

Introduction

We have considerable experience in ERSYS using several different re-sampling algorithms to interpolate the output brightness value corresponding to each output pixel location mapped back into the input image space. These include "nearest-neighbor" (**NN**), "bi-linear" (**BL**), and several versions of "cubic-convolutional" (**CC**) re-sampling. The nearest neighbor re-sampling operator is the easiest to implement and the fastest re-sampling procedure in operation. This re-sampling operator is also a zero-order basis spline (B-spline) function ($\mathbf{B_0}$). It possesses the outstanding virtue of never changing the histogram of the brightness values used in the output imagery in any systematic way. But, as faithfully as this **NN** operator transfers image brightness statistics, the phase information for features in the original scene is significantly degraded in the output image. The edges of linear features in the input scene pick up one pixel and one line discontinuities at frequently recurring intervals—resulting in a visually obvious version of the familiar "jaggies." **NN** interpolation converges on the original signal only at a first order rate $\mathbf{O(\Delta x)}$ as the sampling interval $\mathbf{\Delta x}$ is reduced [Ref. 3]. The real part of a

Fourier transform of the $\mathbf{B_0}$ operator for random re-sampling offsets **d (0≤d<1)** is the function, **sinc(ν)**, where **ν** is the spatial frequency argument in cycles per sample:

$$\mathbf{sinc(\nu) = [\sin(\pi\nu)]/[\pi\nu].}$$

This **sinc**-function represents the Modulation Transfer Function (MTF) of the **NN** re-sampling operator, while the imaginary part of the transformed operator is its Phase Transfer Function (PTF).

Bi-linear interpolation improves the transfer of phase information in the re-sampling process, but unfortunately, it also applies an unacceptable low-frequency-emphasis filtering to the scene brightness values. The **BL** operator is a first-order B-spline ($\mathbf{B_1}$), and can be formed by the convolution of two $\mathbf{B_0}$ functions:

$$\mathbf{B_1 = B_0 * B_0}$$

While the $\mathbf{B_0}$ operator selects a single input pixel value for output, the $\mathbf{B_1}$ operator (**BL** interpolation) requires the local support of the four pixel values from the two-dimensional input image array surrounding the location in the input scene of the desired output sample. As the convolution theorem would indicate, the real part of the Fourier transform of the $\mathbf{B_1}$ operator is the $\mathbf{sinc^2\nu}$ function. The **BL** interpolation operator converges at a second order $\mathbf{O(\Delta x^2)}$ rate as the sampling interval **Δx** approaches zero [Ref. 3].

The second order basis-spline $\mathbf{B_2}$ is not much used for image re-sampling, but a cubic B-spline $\mathbf{B_3}$ could be very useful indeed [Ref. 1]:

$$\mathbf{B_3 = B_1 * B_1 = B_0 * B_0 * B_0 * B_0}.$$

Predictably enough, the Fourier transform of the $\mathbf{B_3}$ operator is $\mathbf{sinc^4\nu}$. The MTF's for the basis spline functions of orders zero, one and three, together with those for cubic B-spline interpolation and a couple of versions of "cubic convolution," are graphed in Figure 1. This $\mathbf{B_3}$ function is strictly positive, as are the $\mathbf{B_1}$ and $\mathbf{B_0}$ operators, but since this positive function also always spans four consecutive samples with non-zero weights, it cannot stand alone as a proper interpolating operator. The central peak of the normalized $\mathbf{B_3}$ function rises only to a value of **2/3** instead of **unity**, and its value at plus or minus **Δx** is **1/6** instead of **zero**, as would be required for any interpolating kernel. If we were to convolve this $\mathbf{B_3}$ operator directly with the large number **N** of our equally-spaced input image pixels, the resulting output image would be "grayed out" and its higher frequency components would be significantly attenuated. But, we can use *digital filtering* to "pre-enhance" the high frequencies within the input image data array itself so that the combined, effective cubic B-spline re-sampling operator (**CB**) can truly become an *interpolating* function.

Working in just one dimension for the time being, the **(0**, **1/6**, **4/6**, **1/6, 0)** weights of the $\mathbf{B_3}$ operator at its knots indicate that the *inverse* of an **N**x**N** real, symmetric, positive definite, tri-diagonally banded matrix $\mathbf{[E]_{NxN}}$ would yield just the pre-enhancing digital filter weights that we need. Indeed, Hou and Andrews carried the process this far in their seminal paper of 1978 [Ref. 1]. Their equations 27 and 28 give the elements of this symmetric inverse matrix (allowing slight corrections and notation changes) as:

$$[E^{-1}]_{NxN} = 6(e_{j,k})_{NxN}$$

$$e_{j,k} = e_{k,j} = [(-1)^{j+k}\ \mu_j\ \mu_{N+1-k}]/[\mu_{N+1}] \qquad (j \le k) \qquad \text{(Eq. 27)}$$

where $\mu_1 = 1$

$\mu_2 = 4$

…

$$\mu_i = 4\mu_{i-1} - \mu_{i-2} \qquad (i = 3, 4, \ldots, N+1) \qquad \text{(Eq. 28).}$$

New Work

Picking up the ball at this point, we recognize this recursively defined sequence μ_i as a simple generalization of the ancient Fibonacci number sequence, and we know that, as in almost any recursively defined relationship, the variable μ_i should be expressible in some closed form. Indeed, after Niven and Zuckerman [Ref. 2], we can write a generalized version of Binet's equation for the Fibonacci sequence as:

$$\mu_i = [(\alpha_1)^i - (\alpha_2)^i]/(\alpha_1 - \alpha_2)$$

where (α_1, α_2) are the roots of the expression:

$$\alpha^2 - 4\alpha + 1 = 0.$$

Note that $\alpha_1 = 2 + \sqrt{3} = 4 - \alpha_2 = 2^{1.9000} = \mathbf{Lim}[(\mu_{i+1})/(\mu_i)]$ for $5 < \mathbf{i}$,

$\alpha_2 = 2 - \sqrt{3} = 1/\alpha_1 = 2^{-1.9000}$

and $\alpha_1 - \alpha_2 = 2\sqrt{3}.$

Thus, $\mu_i = [(2 + \sqrt{3})^i/(2\sqrt{3})][1 - 2^{(-3.8000)i}] \qquad (0 \le i < N+1)$.

For $\mathbf{i > 5}$, the expression in the right-hand brackets is unity to the significance of the 24-bit mantissa in a typical 32-bit floating-point format number representation.

Thus, we can express the elements of the inverse matrix as

$$e_{j,k} = e_{k,j} = [(-1)^{j+k}\ (2 + \sqrt{3})^{j-k}]/(2\sqrt{3})$$

or $e_{j,k} = e_{k,j} = [(\sqrt{3} - 2)^{k-j}]/(2\sqrt{3}) \qquad (5 < j \le k < N-5).$

Note that the 5-sample "border" requirement in input scene space for 24-bit output accuracy is only a "soft" requirement, and not a hard limit.

Now, if we set $\mathbf{a} = \sqrt{3} - 2 = -(2^{-1.9000}) = -0.267949192\ldots,$ the inverse matrix becomes

$$[E^{-1}]_{NxN} = \sqrt{3}[A]_{NxN}$$

where $[A]_{NxN}$ is diagonally banded in powers of $\mathbf{a}$, starting with $\mathbf{a^0 = 1}$ on the principal diagonal, the values $\mathbf{a^1} = \sqrt{3} - 2$ on the adjacent diagonals, etc. The elements alternate signs within each row and column throughout the matrix **[A]**.

So, the pre-enhanced coefficient value $\mathbf{c_k}$, corresponding to the $\mathbf{k}^{th}$ input pixel value $\mathbf{p_k}$, is given by:

$$[c_k]_{Nx1} = [E^{-1}]_{NxN}\,[p_k]_{Nx1} = \sqrt{3}[A]_{NxN}\,[p_k]_{Nx1} \qquad (1 \le k \le N),$$

or $$c_k = \sqrt{3}(p_k + a(p_{k+1} + p_{k-1} + a(p_{k+2} + p_{k-2} + a(\ldots.)))),$$

in nested, factored form.

If the pixel values p_k represent typical 8-bit brightness data having maximum values of 255, all those pixels located more than 5 places away from **k** do not contribute to c_k because

$$255\,(a^5) = -0.352,$$

which is smaller in magnitude than the quantization error **(**or **+/- 0.5)** in p_k itself, to which this value will be added. Then each c_k value would span up to ***eleven*** input pixel values p_k in each dimension.

Interestingly, the real part of a discrete Fourier transform (DFT) of a row or column of the matrix $[A]_{NxN}$ yields the function:

$$(\sqrt{3})\mathrm{DFT}[0, a^N, \ldots a, 1, a, \ldots a^N, 0, 0] = 3/(2 + \cos 2\pi v),$$

and the DFT of the combined effective cubic B-spline interpolation operator, **CB(d**) is just:

$$\mathrm{DFT}[CB(d)] = (3\mathrm{sinc}^4 v)/(2 + \cos 2\pi v),$$

as indicated by Achilles [Ref. 4].

The effective combined **CB** interpolation kernel **CB(x)** can be defined as the *convolution*:

$$CB(x) = (\sqrt{3})[\ldots, a^5, a^4, a^3, a^2, a, 1, a, a^2, a^3, a^4, a^5, \ldots] * B_3(x),$$

where the expression in square brackets represents the elements of a row or column of the matrix **[A]** weighting a train of Dirac impulses spaced at unit intervals on the x-axis (with $a^0 = 1$ aligned with **x = 0**). **CB(x)** is essentially zero except for **x** within the range of approximately **-5** to **+5**. $B_3(x)$ is strictly positive and continuously defined by piecewise cubic polynomials over the interval from **x = -2** to **x = +2,** and is zero outside that range. The cubic **B**-spline function is normalized to unit power so that $B_3(0) = 2/3$. Since **a** is negative and greater than **-1**, the summation of the alternating double geometric series $(\sqrt{3})(1 + 2\sum a^n)$ converges *very rapidly* to *unity* as a limit as **n** gets large (**n > 5**). This effective combined **CB**-interpolation kernel is also called the cubic ***cardinal spline*** interpolation function. A portion of its waveform is shown in Figure 2 for informational purposes only, as we do not plan to implement this function directly as a re-sampling operator in this form.

We plan actually to implement CB-interpolative re-sampling in two separate stages—first by *computing*

$$[c_k]_{Nx1} = \sqrt{3}[A]_{NxN}\,[p_k]_{Nx1} \qquad (1 \le k \le N),$$

and then by *convolving* these "pre-enhanced" coefficients c_k with the B_3 spline function at its proper offset (**d**) to compute the cubic B-spline interpolated brightness value **CB(k+d)** for output:

$$CB(k+d) = [c_k] * [B_3(d)]$$

where **k+d** = Re-sampling location mapped into input image space,

k is integer, **(5 <k<N-5)**

and **0.0≤d<1.0.**

This **CB-*interpolated*** signal converges on the original band-limited signal at a fourth order $\mathbf{O(\Delta x^4)}$ rate as **Δx** approaches zero [Ref. 1]. No other piecewise cubic polynomial interpolation, including cubic hermite (**CH**) or other piecewise cubic polynomial approximation to a windowed **sinc**-function, converges at better than third order rate [Ref. 3]. While the one-dimensional cubic $\mathbf{B_3}$ spline spans only four successive coefficient values for good local support, the effect of the pre-enhancement filtering effectively spreads out the span to as many as eleven samples for 8-bit data values. This wider span of support is necessary for better high-frequency response and really affects only the size of the unusable border areas in the input image space. By my calculation, the roll-off rate for spatial frequencies approaching the Nyquist limit (at **ν = 0.5** cycles per sample) is about **10 dB** per octave steeper for the full **CB** interpolation operator than with **CH** interpolation, for example. This ***cubic B-spline interpolation*** should provide an *optimum* interpolation technique [Ref. 1] for many applications using band-limited data having good signal-to-noise ratio. The reconstructed estimate of the original signal that had been uniformly sampled follows very smooth cubic polynomial curves through all of the data points without over-fitting the data and becoming "too wiggly" as tends to occur with higher-order polynomial-based interpolations. Both the piecewise cubic polynomial $\mathbf{B_3}$ function itself and the cubic B-spline interpolated reconstruction of the original signal are continuous in value, slope and curvature at the knots [Ref. 1].

Implementation for Fast CB Interpolation

The ordered sequence of add and multiply operations, used in calculating the coefficient array $\mathbf{c_i}$ from a row or column of input data samples, must be carefully controlled to maintain full numerical accuracy, especially since floating-point arithmetic is likely to be employed. Otherwise, repeatedly summing mantissas of diverging magnitudes quickly becomes meaningless. Instead of working inward from the analytically determined outer edges for each coefficient $\mathbf{c_i}$, let me suggest separating the calculations of the coefficients $\mathbf{c_i}$ for the entire row (or column) of **N** input samples $\mathbf{p_i}$ into left-hand-side and right-hand-side moving summations as follows:

$$\mathbf{c_i = \sqrt{3}(CL_i + CR_i)} \qquad \mathbf{(i = 1, 2, ...N)}$$

where $\mathbf{CL_i = p_i + a(p_{i-1}) + a^2(p_{i-2}) +... = \underline{a(CL_{i-1}) + p_i}}$

$\mathbf{CR_i = ap_{i+1} + a^2(p_{i+2}) + a^3(p_{i+3}) +... = \underline{a(CR_{i+1} + p_{i+1})}}$

$\mathbf{a = \sqrt{3} -2 = -0.267949192….}$

$\mathbf{CL_0 = 0}$

$\mathbf{CR_N = a(p_N).}$

Note that the index **i** *increments* from **1** to **N** during the recursive calculation of the $\mathbf{CL_i}$ values, but that the index *decrements* from **N** to **1** during the separate recursive

calculation of the $\mathbf{CR_i}$ values, which are usually negative. By use of these two oppositely moving multiply, add and store sequences, full computational accuracy is maintained for each of the **N** subsequently formed $\mathbf{c_i}$ values, and essentially the *fewest possible* number of arithmetic steps is required. The contribution of the $\mathbf{k^{th}}$ input sample automatically shifts out of significance in the $\mathbf{i^{th}}$ one-sided moving sum (at 1.90 bits per step) as the summation moves farther away from the sample location. The pre-enhanced coefficient values $\mathbf{c_i}$ have three times greater potential dynamic range (per dimension) than do the input data samples $\mathbf{p_i}$.

Two-dimensional image re-sampling can be accomplished by using crossed, completed one-dimensional re-sampling steps, as in a dedicated hardware implementation, for example. However, for geometric corrections more general than affine transforms, separating the reverse mapping function into two single-dimensional relationships can become tedious.

Perhaps a better alternative for computer implementation would be to perform just the coefficient array calculation as crossed one-dimensional operations, and then to follow with a single-pass, two-dimensional re-sampling of the coefficient array. Two-dimensional **CB** re-sampling is accomplished by convolving the "pre-enhanced" coefficient array with a two-dimensional cubic B-spline operator. The strictly positive $\mathbf{B_3}$x$\mathbf{B_3}$ spline function accomplishes simultaneous two-dimensional interpolation when applied in this fashion. Opportunities for parallel processing are evident. After calculating the two-dimensional coefficient array, the subsequently applied two-dimensional 4x4 **CB** convolutional operator might not need to be restricted to having parallelogram-shaped weight contours, so long as its power is normalized to unity. Coarsely rounded, elliptical or cruciform operator contours could also be considered.

Image filtering options abound, as well, with this approach to image re-sampling. The two crossed pass pairs through the input image might not necessarily have to be identical for original scanned images. The **a**-value could be set to some other negative number between zero and **-1**, and the factor of $\sqrt{\mathbf{3}}$ in the calculation of the coefficients could then be replaced with

$$\mathbf{(1 - a)/(1 + a)}$$

if normalization of the modified coefficients is desired. Other options could include forming *linear combinations* of the input pixel values $\mathbf{p_i}$ with their corresponding coefficient values $\mathbf{c_i}$ before re-sampling. Many digital image processing effects are made possible by having available two congruent images containing different spatial frequencies. However, if the input image is to be under-sampled, the spatial frequencies in the input image that will fall above the Nyquist limit in the reduced output image should be filtered-out to prevent aliasing. The dynamic range of the interpolated brightness values may need to be clamped if the interpolated output values are to be re-imaged. On the other hand, high-pass filtering could be used to generate a geometrically corrected "edge" image in a single step.

Summary

Cubic B-spline *interpolation* using the recursively-calculated, pre-enhanced coefficient array as described herein can accomplish a flexible and technically superior re-sampling

of two-dimensional images with little extra cost in processing time compared to cubic hermite or "cubic convolutional" image re-sampling. In terms of the number of multiply and add operations required to form the coefficient array from the input data samples, this approach is ***N/3-times faster per dimension*** than the vector inner-product approach of prior art, where **N** is the number of samples per row or column. Using the proposed oppositely moving sums requires **3N** multiply/add operations to "pre-enhance" each input pixel, versus $\mathbf{N^2}$ operations per data sample in each dimension using a vector/matrix multiply approach. This **CB** interpolation converges on the original band-limited signal at a fourth order rate as the sampling interval is decreased, as opposed to at best a third order rate with other cubic polynomial-based interpolation kernels. And its frequency roll-off rate is about **10 dB** per octave sharper at the Nyquist limit, as well. This composite **CB** re-sampling operation enjoys both the four-by-four-sample *local support* of the crossed $\mathbf{B_3}$ spline function and the pre-enhanced *high-frequency response* of the separately calculated coefficient array, typically requiring an eleven-by-eleven-sample data set for calculating each coefficient value.

Figures:

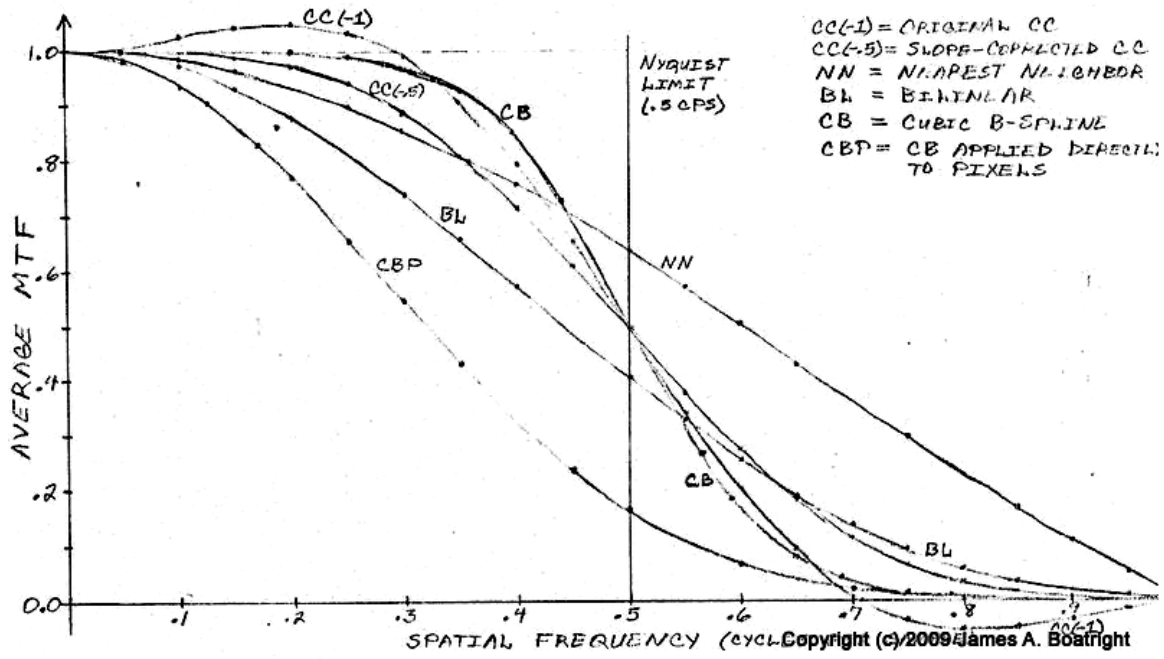

Figure 1. MTF's of Six Possible Re-sampling Functions

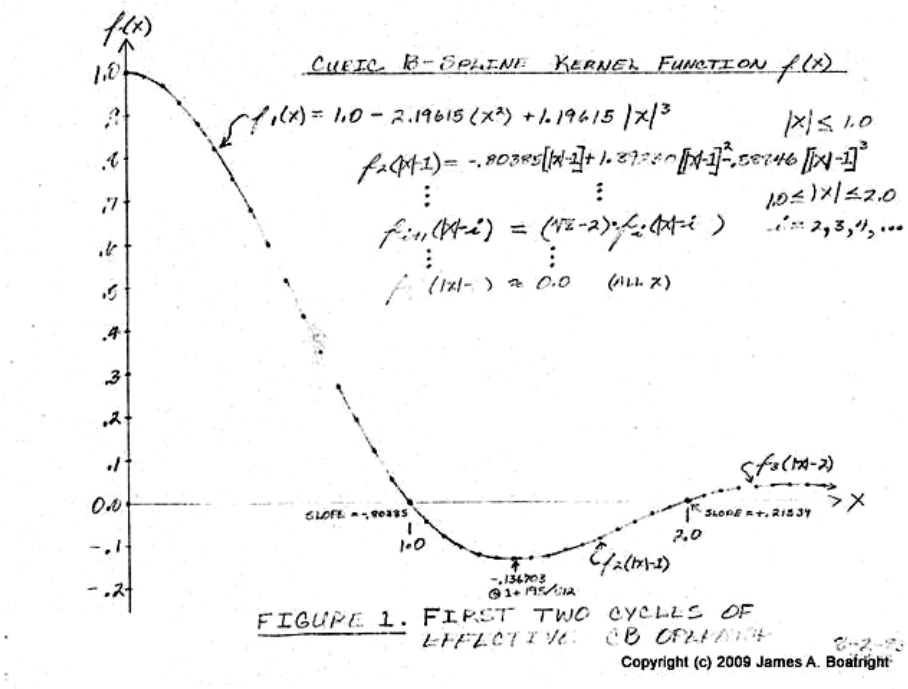

Figure 2. A Portion of Cardinal B-spline Interpolation Kernel

References:

1. H. S. Hou and H. C. Andrews, "Cubic Splines for Image Interpolation and Digital Filtering," IEEE Trans. Accoust., Speech, Signal Processing, vol. ASSP-26, no. 6, Dec. 1978.
2. I. Niven and H. S. Zuckerman, *An Introduction to the Theory of Numbers*, (pp 90-91, Recurrence functions), J. Wiley, New York, 1960.
3. R. G. Keys, "Cubic Convolution Interpolation for Digital Image Processing," IEEE Trans. Accoust., Speech, Signal Processing, vol. ASSP-29, no. 6, Dec. 1981.
4. D. Achilles, "New Algorithms for Fast Convolution Based on Convolution Preserving Spline Signals," IEEE 1979.