Cubic Convolution Interpolation for Digital Image Processing

ROBERT G. KEYS

Abstract—**Cubic convolution interpolation is a new technique for resampling discrete data. It has a number of desirable features which make it useful for image processing. The technique can be performed efficiently on a digital computer. The cubic convolution interpolation function converges uniformly to the function being interpolated as the sampling increment approaches zero. With the appropriate boundary conditions and constraints on the interpolation kernel, it can be shown that the order of accuracy of the cubic convolution method is between that of linear interpolation and that of cubic splines.**

A one-dimensional interpolation function is derived in this paper. A separable extension of this algorithm to two dimensions is applied to image data.

Introduction

INTERPOLATION is the process of estimating the intermediate values of a continuous event from discrete samples. Interpolation is used extensively in digital image processing to magnify or reduce images and to correct spatial distortions. Because of the amount of data associated with digital images, an efficient interpolation algorithm is essential. Cubic convolution interpolation was developed in response to this requirement.

The algorithm discussed in this paper is a modified version of the cubic convolution algorithm developed by Rifman [1] and Bernstein [2]. The objective of this paper is to derive the modified cubic convolution algorithm and to compare it with other interpolation methods.

Two conditions apply throughout this paper. First, the analysis pertains exclusively to the one-dimensional problem; two-dimensional interpolation is easily accomplished by performing one-dimensional interpolation in each dimension. Second, the data samples are assumed to be equally spaced. (In the two-dimensional case, the horizontal and vertical sampling increments do not have to be the same.) With these conditions in mind, the first topic to consider is the derivation of the cubic convolution algorithm.

Basic Concepts Concerning the Cubic Convolution Algorithm

An interpolation function is a special type of approximating function. A fundamental property of interpolation functions is that they must coincide with the sampled data at the interpolation nodes, or sample points. In other words, if f is a sampled function, and if g is the corresponding interpolation function, then $g(x_k) = f(x_k)$ whenever x_k is an interpolation node.

Manuscript received July 29, 1980; revised January 5, 1981 and April 30, 1981.

The author is with the Exploration and Production Research Laboratory, Cities Service Oil Company, Tulsa, OK 74102.

For equally spaced data, many interpolation functions can be written in the form

$$g(x) = \sum_k c_k u\left(\frac{x - x_k}{h}\right). \tag{1}$$

Among the interpolation functions that can be characterized in this manner are cubic splines and linear interpolation functions. (See Hou and Andrews [3].)

In (1), and for the remainder of this paper, h represents the sampling increment, the x_k's are the interpolation nodes, u is the interpolation kernel, and g is the interpolation function. The c_k's are parameters which depend upon the sampled data. They are selected so that the interpolation condition, $g(x_k) = f(x_k)$ for each x_k, is satisfied.

The interpolation kernel in (1) converts discrete data into continuous functions by an operation similar to convolution. Interpolation kernels have a significant impact on the numerical behavior of interpolation functions. Because of their influence on accuracy and efficiency, interpolation kernels can be effectively used to create new interpolation algorithms. The cubic convolution algorithm is derived from a set of conditions imposed on the interpolation kernel which are designed to maximize accuracy for a given level of computational effort.

The Cubic Convolution Interpolation Kernel

The cubic convolution interpolation kernel is composed of piecewise cubic polynomials defined on the subintervals $(-2, -1)$, $(-1, 0)$, $(0, 1)$, and $(1, 2)$. Outside the interval $(-2, 2)$, the interpolation kernel is zero. As a consequence of this condition, the number of data samples used to evaluate the interpolation function in (1) is reduced to four.

The interpolation kernel must be symmetric. Coupled with the previous condition, this means that u must have the form

$$u(s) = \begin{cases} A_1|s|^3 + B_1|s|^2 + C_1|s| + D_1 & 0 < |s| < 1 \\ A_2|s|^3 + B_2|s|^2 + C_2|s| + D_2 & 1 < |s| < 2 \\ 0 & 2 < |s|. \end{cases} \tag{2}$$

The interpolation kernel must assume the values $u(0) = 1$ and $u(n) = 0$ when n is any nonzero integer. This condition has an important computational significance. Since h is the sampling increment, the difference between the interpolation nodes x_j and x_k is $(j - k)h$. Now if x_j is substituted for x in (1), then (1) becomes

$$g(x_j) = \sum_k c_k u(j - k). \tag{3}$$

Because $u(j - k)$ is zero unless $j = k$, the right-hand side of (3)

0096-3518/81/1200-1153$00.75

reduces to c_j. The interpolation conditon requires that $g(x_j) = f(x_j)$. Therefore, $c_j = f(x_j)$. In other words, the c_k's in (1) are simply replaced by the sampled data. This is a substantial computational improvement over interpolation schemes such as the method of cubic splines. The spline interpolation kernel is not zero for nonzero integers. As a result, the c_k's must be determined by solving a matrix problem.

In addition to being 0 or 1 at the interpolation nodes, the interpolation kernel must be continuous and have a continuous first derivative. From these latter conditions, a set of equations can be derived for the coefficients in (2).

The conditions $u(0) = 1$ and $u(1) = u(2) = 0$ provide four equations for these coefficients:

$$1 = u(0) = D_1$$
$$0 = u(1^-) = A_1 + B_1 + C_1 + D_1$$
$$0 = u(1^+) = A_2 + B_2 + C_2 + D_2$$
$$0 = u(2^-) = 8A_2 + 4B_2 + 2C_2 + D_2.$$

Three more equations are obtained from the fact that u' is continuous at the nodes 0, 1, and 2:

$$-C_1 = u'(0^-) = u'(0^+) = C_1$$
$$3A_1 + 2B_1 + C_1 = u'(1^-) = u'(1^+) = 3A_2 + 2B_2 + C_2$$
$$12A_2 + 4B_2 + C_2 = u'(2^-) = u'(2^+) = 0.$$

In all, the constraints imposed on u result in seven equations. But since there are eight unknown coefficients, one more condition is needed to obtain a unique solution. Rifman [1] and Bernstein [2] use the constraint that $A_2 = -1$. In this presentation, however, A_2 will be selected so that the interpolation function g approximates the original function f to as high a degree as possible. In particular, assume that f has several orders of continuous derivatives so that Taylor's theorem, from calculus, applies. The idea will be to choose A_2 so that the cubic convolution interpolation function and the Taylor series expansion for f agree for as many terms as possible.

To accomplish this, let $A_2 = a$. The remaining seven coefficients can be determined, in terms of a, from the previous seven equations. The solution for the interpolation kernel, in terms of a, is

$$u(s) = \begin{cases} (a+2)|s|^3 - (a+3)|s|^2 + 1 & 0 < |s| < 1 \\ a|s|^3 - 5a|s|^2 + 8a|s| - 4a & 1 < |s| < 2 \\ 0 & 2 < |s|. \end{cases} \quad (4)$$

Now suppose that x is any point at which the sampled data is to be interpolated. Then x must be between two consecutive interpolation nodes which can be denoted by x_j and x_{j+1}.

Let $s = (x - x_j)/h$. Since $(x - x_k)/h = (x - x_j + x_j - x_k)/h = s + j - k$, (1) can be written as

$$g(x) = \sum_k c_k u(s + j - k). \quad (5)$$

Furthermore, since u is zero except in the interval $(-2, 2)$, and since $0 < s < 1$, (5) reduces to

$$g(x) = c_{j-1}u(s+1) + c_j u(s) + c_{j+1}u(s-1) + c_{j+2}u(s-2). \quad (6)$$

From (4), it follows that

$$u(s+1) = a(s+1)^3 - 5a(s+1)^2 + 8a(s+1) - 4a = as^3 - 2as^2 + as$$
$$u(s) = (a+2)s^3 - (a+3)s^2 + 1$$
$$u(s-1) = -(a+2)(s-1)^3 - (a+3)(s-1)^2 + 1 = -(a+2)s^3 + (2a+3)s^2 - as$$
$$u(s-2) = -a(s-2)^3 - 5a(s-2)^2 - 8a(s-2) - 4a = -as^3 + as^2.$$

By substituting the above relationships into (6) and collecting powers of s, the cubic convolution resampling function becomes

$$g(x) = -[a(c_{j+2} - c_{j-1}) + (a+2)(c_{j+1} - c_j)]s^3 + [2a(c_{j+1} - c_{j-1}) + 3(c_{j+1} - c_j) + a(c_{j+2} - c_j)]s^2 - a(c_{j+1} - c_{j-1})s + c_j. \quad (7)$$

If f has at least three continuous derivatives in the interval $[x_j, x_{j+1}]$, then according to Taylor's theorem

$$c_{j+1} = f(x_{j+1}) = f(x_j) + f'(x_j)h + f''(x_j)h^2/2 + 0(h^3) \quad (8)$$

where $h = x_{j+1} - x_j$. $0(h^3)$ represents the terms of order h^3; that is, terms which go to zero at a rate proportional to h^3. Similarly,

$$c_{j+2} = f(x_j) + 2hf'(x_j) + 2h^2 f''(x_j) + 0(h^3) \quad (9)$$
$$c_{j-1} = f(x_j) - hf'(x_j) + h^2 f''(x_j)/2 + 0(h^3). \quad (10)$$

When (8)-(10) are substituted into (7), the following equation for the cubic convolution interpolation function is obtained.

$$g(x) = -(2a+1)[2hf'(x_j) + h^2 f''(x_j)]s^3 + [(6a+3)hf'(x_j) + (4a+3)h^2 f''(x_j)/2]s^2 - 2ahf'(x_j)s + f(x_j) + 0(h^3). \quad (11)$$

Since $sh = x - x_j$, the Taylor series expansion for $f(x)$ about x is

$$f(x) = f(x_j) + shf'(x_j) + s^2 h^2 f''(x_j)/2 + 0(h^3). \quad (12)$$

Subtracting (11) and (12)

$$f(x) - g(x) = (2a+1)[2hf'(x_j) + h^2 f''(x_j)]s^3 - (2a+1)[3hf'(x_j) + h^2 f''(x_j)]s^2 + (2a+1)shf'(x_j) + 0(h^3). \quad (13)$$

If the interpolation function $g(x)$ is to agree with the first three terms of the Taylor series expansion for f, then the parameter a must be equal to $-\frac{1}{2}$. This choice provides the final condition for the interpolation kernel: $A_2 = -\frac{1}{2}$. When $A_2 = a = -\frac{1}{2}$, then

$$f(x) - g(x) = 0(h^3). \tag{14}$$

Equation (14) implies that the interpolation error goes to zero uniformly at a rate proportional to h^3, the cube of the sampling increment. In other words, g is a third-order approximation for f. The constraint $A_2 = -\frac{1}{2}$ is the only choice for A_2 that will achieve third-order precision; any other condition will result in at most a first-order approximation.

Using the final condition that $A_2 = -\frac{1}{2}$, the cubic convolution interpolation kernel is

$$u(s) = \begin{cases} \frac{3}{2}|s|^3 - \frac{5}{2}|s|^2 + 1 & 0 < |s| < 1 \\ -\frac{1}{2}|s|^3 + \frac{5}{2}|s|^2 - 4|s| + 2 & 1 < |s| < 2 \\ 0 & 2 < |s|. \end{cases} \tag{15}$$

Boundary Conditions

In the initial discussion, f, the function being sampled, was defined for all real numbers. In practice, however, f can only be observed on a finite interval. Because the domain of f is restricted to a finite interval, say $[a, b]$, boundary conditions are necessary.

First of all, the sample points x_k must be defined to correspond to the new interval of observation, $[a, b]$. Let $x_k = x_{k-1} + h$ for $k = 1, 2, 3, \cdots, N$, where $x_0 = a$, $x_N = b$, and $h = (b - a)/N$ for some large integer N. (The integer N may be chosen from the Nyquist criterion.) The results of the previous section are valid for any set of uniformly spaced sample points and thus are not affected by this new definition for the x_k's.

On the interval $[a, b]$, the cubic convolution interpolation function can be written as

$$g(x) = \sum_{k=-1}^{N+1} c_k u\left(\frac{x - x_k}{h}\right) \tag{16}$$

since to determine g for all x in the interval $[a, b]$, the values of c_k for $k = -1, 0, 1, \cdots, N+1$ are needed. For $k = 0, 1, 2, \cdots, N$, $c_k = f(x_k)$. For $k = -1$ and for $k = N+1$, however, the value of f is unknown, since x_{-1} and x_{N+1} fall outside the interval of observation. The values assigned to c_{-1} and c_{N+1} are boundary conditions.

Boundary conditions must be chosen so that $g(x)$ is an $0(h^3)$ approximation to $f(x)$ for all x contained in the interval $[a, b]$. To find an appropriate condition for the left-hand boundary, suppose that x is a point in the subinterval $[x_0, x_1]$. For this value of x, the interpolation function reduces to

$$g(x) = c_{-1}u(s+1) + c_0 u(s) + c_1 u(s-1) + c_2 u(s-2) \tag{17}$$

where $s = (x - x_0)/h$. By substituting the equations in (15) for u and collecting powers of s, the interpolation function reduces to

$$g(x) = s^3[c_2 - 3c_1 + 3c_0 - c_{-1}]/2 - s^2[c_2 - 4c_1 + 5c_0 - 2c_{-1}]/2 + s[c_1 - c_{-1}]/2 + c_0. \tag{18}$$

If g is an $0(h^3)$ approximation for f, then the s^3-term must be zero. This means that c_{-1} should be chosen so that $c_{-1} = c_2 - 3c_1 + 3c_0$, or

$$c_{-1} = f(x_2) - 3f(x_1) + 3f(x_0). \tag{19}$$

After substituting (19) into (18), the interpolation function becomes

$$g(x) = s^2[f(x_2) - 2f(x_1) + f(x_0)]/2 + s[-f(x_2) + 4f(x_1) - 3f(x_0)]/2 + f(x_0). \tag{20}$$

All that remains is to show that (20) is a third-order approximation for $f(x)$. First expand $f(x_2)$ and $f(x_1)$ in a Taylor series about x_0:

$$f(x_2) = f(x_0) + 2f'(x_0)h + 2f''(x_0)h^2 + 0(h^3) \tag{21}$$

$$f(x_1) = f(x_0) + f'(x_0)h + f''(x_0)h^2/2 + 0(h^3). \tag{22}$$

By replacing $f(x_2)$ and $f(x_1)$ in (20) with (21) and (22), the following result is obtained.

$$g(x) = f(x_0) + f'(x_0)sh + f''(x_0)s^2h^2/2 + 0(h^3). \tag{23}$$

Since $sh = x - x_0$, the Taylor series expansion for $f(x)$ about x_0 is

$$f(x) = f(x_0) + f'(x_0)sh + f''(x_0)s^2h^2/2 + 0(h^3). \tag{24}$$

Subtracting (23) from (24)

$$f(x) - g(x) = 0(h^3).$$

Thus, the boundary condition specified by (19) results in a third-order approximation for $f(x)$ when $x_0 < x < x_1$.

A similar analysis can be used to obtain c_{N+1}. If x is in the interval $[x_{N-1}, x_N]$, the boundary condition

$$c_{N+1} = 3f(x_N) - 3f(x_{N-1}) + f(x_{N-2}) \tag{25}$$

will provide a third-order approximation for f.

Using the interpolation kernel defined by (15) and the boundary conditions (19) and (25), a complete description of the cubic convolution interpolation function can now be given.

When $x_k < x < x_{k+1}$, the cubic convolution interpolation function is

$$g(x) = c_{k-1}(-s^3 + 2s^2 - s)/2 + c_k(3s^3 - 5s^2 + 2)/2 + c_{k+1}(-3s^3 + 4s^2 + s)/2 + c_{k+2}(s^3 - s^2)/2$$

where $s = (x - x_k)/h$ and $c_k = f(x_k)$ for $k = 0, 1, 2, \cdots, N$; $c_{-1} = 3f(x_0) - 3f(x_1) + f(x_2)$; and $c_{N+1} = 3f(x_N) - 3f(x_{N-1}) + f(x_{N-2})$.

One of the basic assumptions used to derive the cubic convolution algorithm was that the sampled function possessed a continuous third derivative. This assumption is not unreasonable for many practical problems. For example, the sampled function is often assumed to be band limited. Since band-limited functions are infinitely differentiable, they easily meet the requirements for cubic convolution interpolation.

Although a continuous third derivative is required for the sampled function, no such restriction is imposed on the interpolation function. In general, the cubic convolution interpolation function will not have a continuous second drivative. Nevertheless, if the sampled function has a continuous third derivative, then from the results of the last two sections, the

interpolation function is a third-order approximation. This fact is not inconsistent with other interpolation methods. For example, linear interpolation gives a second-order approximation, provided the sampled function has a continuous second derivative. However, the linear interpolation function is not everywhere continuously differentiable. Further comparisons with other interpolation methods are the topic of the next section.

Comparison with Other Methods

There are several important considerations in the analysis of an interpolation method. Of major importance is the accuracy of the technique: the exactness with which the interpolation function reconstructs the sampled function. Additionally, some interesting effects can be predicted from the spectral characteristics of the interpolation kernel. In this section, the accuracy, the spectral properties, and the efficiency of the cubic convolution interpolation algorithm will be compared with other methods.

Some indication of the accuracy of the method is given by the type of function which can be exactly reconstructed. The cubic convolution interpolation function exactly reconstructs any second-degree polynomial. This is because the third derivative of any second-degree polynomial is zero and, thus, the approximation error is zero. In contrast, linear interpolation will reproduce at most a first-degree polynomial. A scheme referred to by Rifman [1] and Bernstein [2] as the "nearest-neighbor" algorithm uses the nearest sample as the interpolated value. The nearest-neighbor algorithm is exact only when the sampled function is a constant. By using cubic convolution instead of linear interpolation or nearest-neighbor resampling, the degree of complexity of functions which can be exactly reconstructed is increased.

The relative accuracy of different interpolation methods can be determined from their convergence rates. The convergence rate is a measure of how fast the approximation error goes to zero as the sampling increment decreases. In the derivation of the cubic convolution algorithm, it was found that the approximation error consists of terms proportional to h^3, where h is the sampling increment. In this case, the approximation error goes to zero at least as fast as h^3 goes to zero. Thus, the convergence rate for the cubic convolution interpolation function is $0(h^3)$.

Linear interpolation functions have a $0(h^2)$ convergence rate. To see this, suppose that x is a point between the pair of interpolation nodes x_j and x_{j+1}. Let $s = (x - x_j)/h$. From Taylor's theorem, if f has a continuous second derivative in the interval $[x_j, x_{j+1}]$, then

$$f(x_j) = f(x) - f'(x)\, sh + 0(h^2) \tag{26}$$

where $0(h^2)$ is the remainder term. Since $x_{j+1} - x = (1 - s)\, h$, it also follows (from Taylor's theorem) that

$$f(x_{j+1}) = f(x) + f'(x)\,(1 - s)\, h + 0(h^2). \tag{27}$$

Now if (26) is multiplied by $1 - s$ and if (27) is multiplied by s and the resulting equations are added, then

$$(1 - s) f(x_j) + sf(x_{j+1}) = f(x) + 0(h^2). \tag{28}$$

The left-hand side of (28) is the linear interpolation function for $f(x)$. The right-hand side of (28) shows that the approximation error is proportional to h^2. Thus, the convergence rate for linear interpolation is $0(h^2)$. Since the cubic convolution algorithm has a $0(h^3)$ convergence rate, the cubic convolution interpolation function will generally be a more accurate approximation to the sampled function than the linear interpolation function.

The nearest-neighbor algorithm has a $0(h)$ convergence rate. This is an immediate consequence of the mean value theorem. If f has a continuous derivative on the interval between x and x_j, then there is a point m between x and x_j such that

$$f(x) = f(x_j) + f'(m)\, sh \tag{29}$$

where $s = (x - x_j)/h$. If x_j is the nearest interpolation node to x, then $f(x_j)$ is the value of the nearest-neighbor interpolation function for $f(x)$. The approximation error in (29) is proportional to h which means that the convergence rate is $0(h)$.

Additional insight into the behavior of interpolation functions can be gained from their frequency domain characteristics. All of the interpolation functions mentioned so far can be written in the form

$$g(x) = \sum_k c_k u\left(\frac{x - x_k}{h}\right). \tag{30}$$

Examples of some interpolation kernels which replace u in (30) are shown in Fig. 1.

Taking the Fourier transform of (30),

$$G(\omega) = \sum_k c_k \exp(-i\omega x_k)\, hU(\omega h) \tag{31}$$

where

$$G(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} g(x) \exp(-i\omega x)\, dx$$

and

$$U(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} u(x) \exp(-i\omega x)\, dx.$$

Equation (31) illustrates the "smoothing" effect of interpolation. The summation term in (31) is the discrete Fourier transform of the sampled data, and $U(\omega h)$ acts as a smoothing filter. An analysis of the various interpolation schemes can be made by comparing the Fourier transforms of their interpolation kernels.

The amplitude spectra of the nearest-neighbor, linear interpolation, and cubic convolution interpolation kernels are graphed in Fig. 2 for frequencies from 0 to $4\pi/h$. The response of an ideal interpolation kernel (for band-limited data) is a unit step function which has the value of one for frequencies between $-\pi/h$ and $+\pi/h$, and zero elsewhere. Such an interpolation kernel would pass every frequency component of a band-limited function without change, provided the sampling increment was sufficiently small.

Deviations from the ideal spectrum in the shaded region in Fig. 2 (from 0 to $+\pi/h$) cause a loss of high frequency infor-

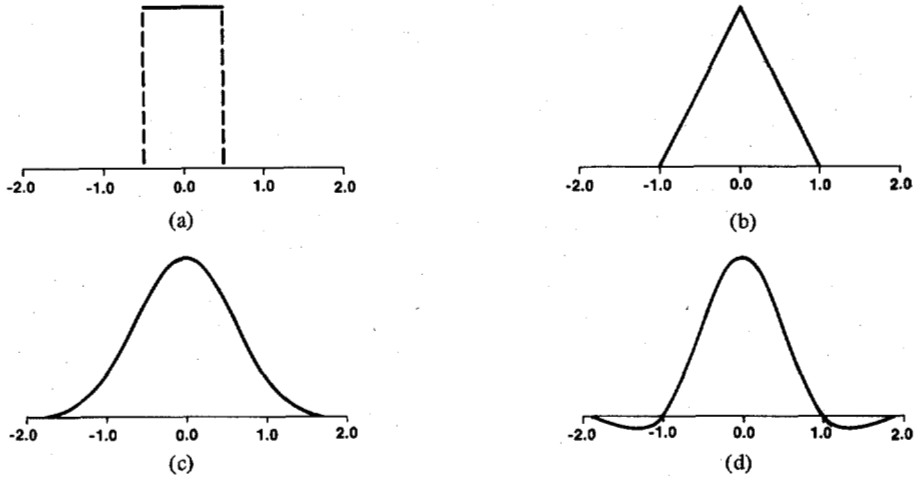

Fig. 1. Interpolation kernels. (a) Nearest-neighbor. (b) Linear interpolation. (c) Cubic spline. (d) Cubic convolution.

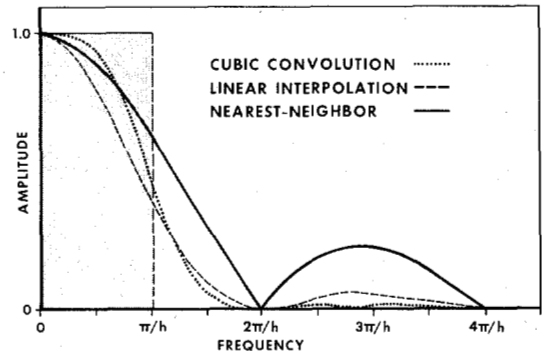

Fig. 2. Amplitude spectra of interpolation kernels.

mation. In image data, the loss of high frequency information causes the image to appear blurred. On the other hand, deviations from the ideal spectrum beyond the shaded area contribute to aliasing. From Fig. 2, it is evident that of the three methods presented, the cubic convolution method provides the best approximation to the ideal spectrum.

Some of the points discussed in this section can be illustrated with a numerical example. Since the cubic convolution algorithm was developed for resampling image data, it is appropriate to use a two-dimensional example. Consider the two-dimensional radially symmetric function, $f(x, y) = \sin(0.5r^2)$, where $r^2 = x^2 + y^2$. Since the Fourier transform of this function is $F(\omega_x, \omega_y) = \cos(0.5\rho^2)$ where $\rho^2 = \omega_x^2 + \omega_y^2$, the function $f(x, y)$ is not band limited. From the sampling theorem, it is known that any attempt to reconstruct f from discrete samples must fail. Nevertheless, with the results derived in the last section, a reasonably accurate approximation for f can be obtained within a bounded region.

For this example, identical sampling increments were used for both the x and y coordinates. The sampling increment h was chosen to be half the length of the interval between the 22nd and 23rd zeros of $f(r)$. This value of h guarantees that there will be at least two samples between each zero crossing of $f(r)$ within the region over which f is being sampled. In this case, the sampling increment will be $h = 0.132119066$. Samples of $f(x, y)$ were obtained on a 64 X 64 element grid with the origin in the upper left-hand corner.

An image was formed from the two-dimensional data by converting amplitude values into light intensities. Maximum intensity (white) was assigned to the maximum amplitude of f, +1. Zero intensity (black) was assigned to the minimum value of f, -1. Intermediate values of $f(x, y)$ were converted to proportional shades of gray.

The physical size of an image is controlled by the number of samples in an image and the sample spacing. The example images in this paper are displayed at a sample spacing of 100 samples/in horizontally and 96 samples/in vertically. Each image fills a 3.5 in square. Therefore, a two-dimensional array of 350 X 336 points is required to represent each image. To obtain a 350 X 336 point array from a 64 X 64 point array, two-dimensional interpolation must be used. Interpolation employed in this manner is equivalent to digital image magnification.

Two-dimensional interpolation is accomplished by one-dimensional interpolation with respect to each coordinate. The two-dimensional cubic convolution interpolation function is a separable extension of the one-dimensional interpolation function. When (x, y) is a point in the rectangular subdivision $[x_j, x_{j+1}] \times [y_k, y_{k+1}]$, the two-dimensional cubic convolution interpolation function is

$$g(x, y) = \sum_{l=-1}^{2} \sum_{m=-1}^{2} c_{j+l,k+m}\, u\left(\frac{x - x_{j+l}}{h_x}\right) u\left(\frac{y - y_{k+m}}{h_y}\right)$$

where u is the interpolation kernel of (15) and h_x and h_y are the x and y coordinate sampling increments. For interior grid points, the c_{jk}'s are given by $c_{jk} = f(x_j, y_k)$. If x_N is the upper

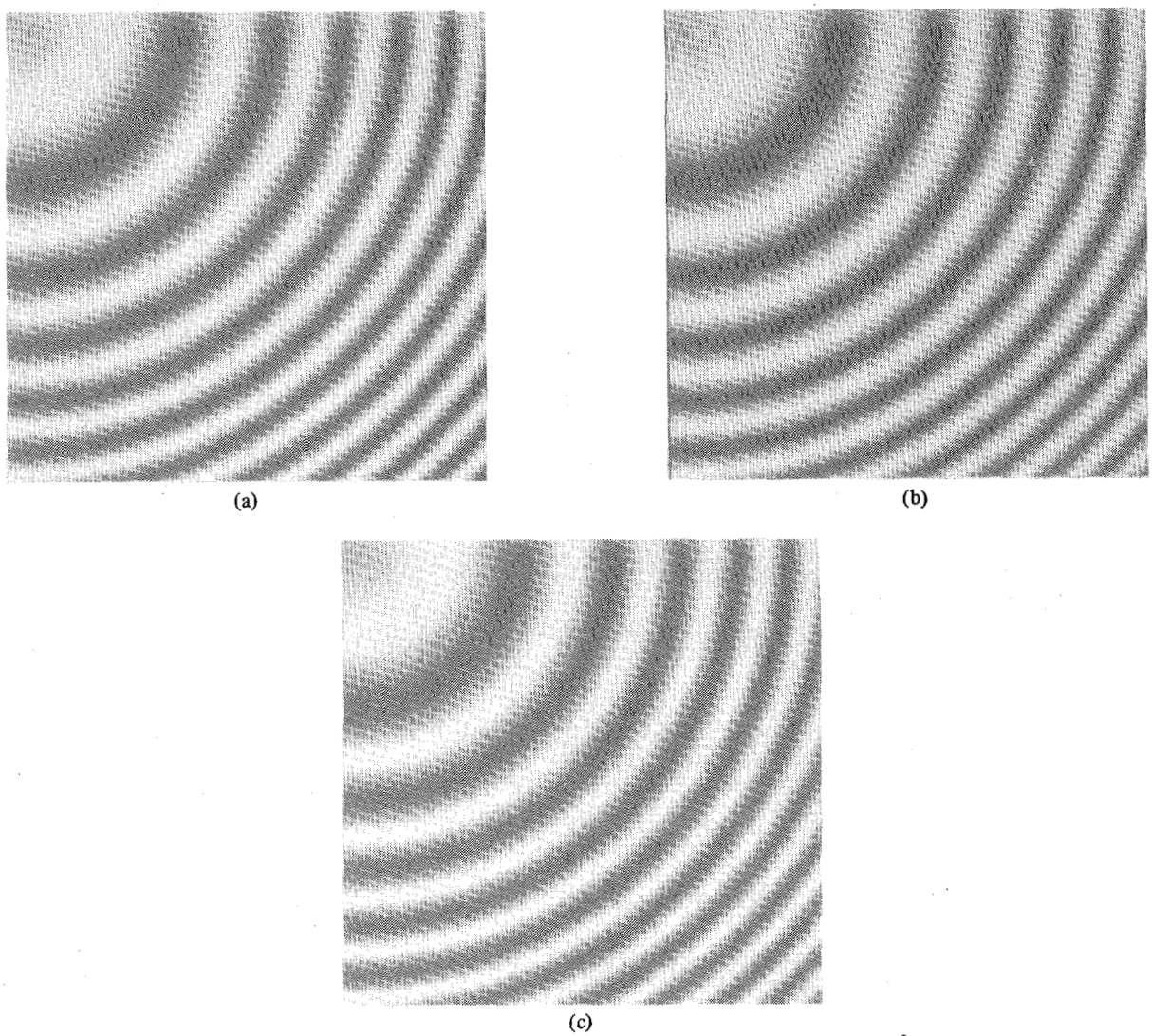

Fig. 3. Interpolation of image data. Image (a) was created by sampling the test function, $\sin(0.5r^2)$, on a 350 × 336 point grid. Image (b) was constructed from a 64 × 64 array of test function samples, using linear interpolation to estimate intermediate values. Image (c) was derived from the same 64 × 64 array of data using cubic convolution.

boundary of the x-partition and y_M is the upper boundary of the y-partition, then the boundary conditions are

$$c_{-1,k} = 3f(x_0, y_k) - 3f(x_1, y_k) + f(x_2, y_k)$$

$$c_{N+1,k} = 3f(x_N, y_k) - 3f(x_{N-1}, y_k) + f(x_{N-2}, y_k)$$

$$\text{for } k = 0, 1, 2, \cdots, M$$

$$c_{j,-1} = 3f(x_j, y_0) - 3f(x_j, y_1) + f(x_j, y_2)$$

$$c_{j,M+1} = 3f(x_j, y_M) - 3f(x_j, y_{M-1}) + f(x_j, y_{M-2})$$

$$\text{for } j = 0, 1, 2, \cdots, N$$

$$c_{-1,-1} = 3c_{0,-1} - 3c_{1,-1} + c_{2,-1}$$

$$c_{N+1,-1} = 3c_{N,-1} - 3c_{N-1,-1} + c_{N-2,-1}$$

$$c_{-1,M+1} = 3c_{0,M+1} - 3c_{1,M+1} + c_{2,M+1}$$

$$c_{N+1,M+1} = 3c_{N,M+1} - 3c_{N-1,M+1} + c_{N-2,M+1}.$$

For comparison, a 350 × 336 point array was created by directly evaluating f at each grid point. The image created in this manner is shown in Fig. 3(a). Next, the 64 × 64 point array was resampled to a 350 × 336 point array with a linear interpolation algorithm. The resulting image is shown in Fig. 3(b). Finally, the cubic convolution algorithm was used to resample 64 × 64 point array. In Fig. 3(c), the image obtained by cubic convolution interpolation is shown.

To compare the accuracy of the two techniques, "error" images were obtained by subtracting the interpolated images from the exact image [Fig. 3(a)] and taking absolute values. The linear interpolation error image is shown in Fig. 4(a) and the cubic convolution error image is displayed in Fig. 4(b). Maximum intensity (white) corresponds to the maximum absolute error in the linear interpolation image. Zero error is represented by zero intensity (black). Error magnitudes between zero and maximum linear interpolation error are represented by corresponding shades of gray. Thus, increasing intensity denotes an increasing magnitude of error in Fig. 4(a) and (b).

The most obvious difference between the two error images is that the cubic convolution error image is uniformly darker. As expected, the cubic convolution interpolation function is a better approximation to the sampled function than the linear interpolation function.

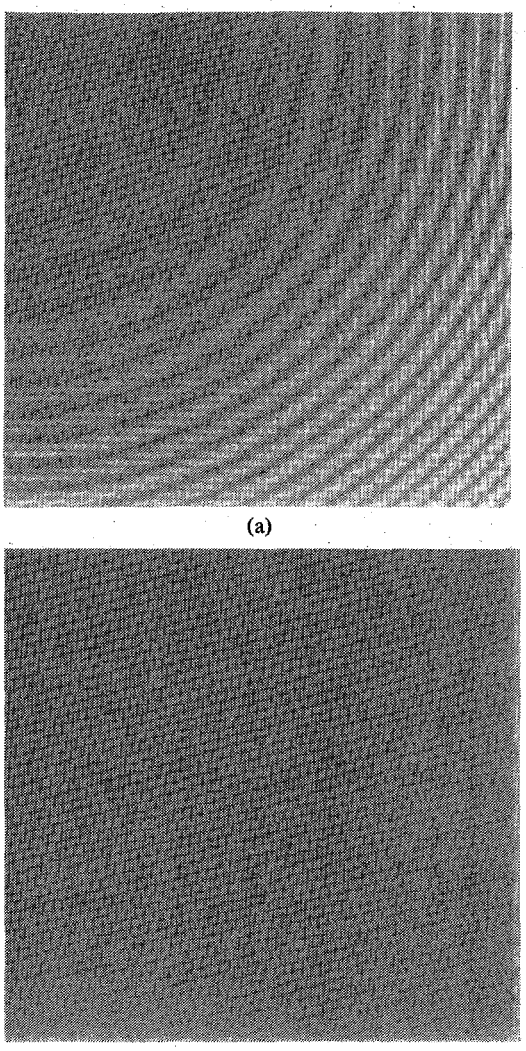

Fig. 4. Interpolation error; the absolute value of the difference between Fig. 3(a) and the interpolated image. In the above display, black denotes zero error; white is the maximum absolute error. (a) Error caused by resampling with linear interpolation. (b) Cubic convolution resampling error.

It is also possible to detect aliasing effects in the linear interpolation error image. Superimposed on the circular ridges in Fig. 4(a) are horizontal and vertical lines. These lines are artifacts created by resampling with a linear interpolation function. Although aliasing effects are present in the error image derived by cubic convolution interpolation, their magnitude is much smaller. These results are predictable from the comparison of the amplitude spectra in Fig. 2.

Another important consideration when comparing numerical procedures is their computational efficiency. Of the methods discussed so far, the nearest-neighbor algorithm is the simplest. For this procedure, no arithmetic operations are necessary. The interpolated sample point is assigned the value of the nearest sample from the original data.

Next in simplicity is linear interpolation. For this procedure, two sample points are involved in the interpolation of each new point. Two additions and one multiplication are needed for each interpolated point.

The cubic convolution method uses four sample points for each interpolation point. Nine additions and eight multiplications are performed for each interpolated point. The example in Fig. 3 provides a practical demonstration of the efficiency of cubic convolution. The computer time required for cubic convolution resampling of the image in Fig. 3 was slightly greater than the computer time needed for linear interpolation (0.29 min versus 0.28 min). However, in applications with larger data sets, the computer time for cubic convolution resampling is about twice that of linear interpolation.

Cubic spline interpolation requires even greater computational exertion. This is because the interpolation coefficients —the c_k's in (1)—must be determined by solving a tridiagonal matrix problem; in two dimensions, the matrix is block tridiagonal. Since there is one c_k for each data sample, the rank of the matrix is equal to the number of data samples. Unless the c_k's can be stored in the same location used for the input data, additional storage, equal to the size of the input data file, must be provided. After the interpolation coefficients have been computed, cubic spline interpolation involves roughly the same amount of work as the cubic convolution method.

Cubic convolution interpolation is much more efficient than the method of cubic splines, in terms of both storage and computation time. Because the data samples are the cubic convolution interpolation coefficients, the efficiency of the cubic convolution method is closer to the efficiency of linear interpolation than the cubic spline method.

The objective up to this point was to derive the cubic convolution algorithm and to compare its performance with other common interpolation algorithms. In the next section, variations of the cubic convolution method will be considered. In particular, a brief outline will be given for the construction of an interpolation function which has fourth-order accuracy.

Fourth-Order Algorithms

The convergence rate of the cubic convolution interpolation function is $O(h^3)$. The cubic convolution interpolation function is also unique. Therefore, any interpolation function whose kernel satisfies the conditions outlined in the first part of this paper will have at most a third-order convergence rate. Furthermore, the convergence rate will be third-order if and only if the interpolation function is the cubic convolution interpolation function. Nevertheless, interpolation functions with higher order convergence rates are possible. By removing or altering the conditions on the interpolation kernel, interpolation functions with fourth-order convergence rates can be derived.

As an example, suppose the condition that the interpolation function must be differentiable is removed. Then a piecewise cubic Lagrange interpolation polynomial has a kernel which satisfies the remaining conditions, and the interpolation function has a $O(h^4)$ convergence rate [4, p. 28]. Because the piecewise cubic Lagrange interpolation function is not continuously differentiable, the interpolation function will have sharp edges at the sample points. Normally, this is not a desirable effect, and therefore it is useful to require the interpolation function to have a continuous derivative.

Cubic splines are twice continuously differentiable, but, as previously mentioned, the interpolation kernel is not zero for nonzero integers. The cubic spline interpolation function is a

$0(h^4)$ approximation to the interpolated function [4, pp. 57-58].

The convergence rate can also be increased to $0(h^4)$ by increasing the length of the interval over which the interpolation kernel is not zero. Consider the interpolation kernel which is composed of piecewise cubic polynomials on the subintervals $(-3,-2)$, $(-2,-1)$, $(-1,0)$, $(0,1)$, $(1,2)$, and $(2,3)$. Outside the interval $(-3,3)$, let the interpolation kernel be zero. If the interpolation kernel is symmetric then it must have the form

$$u(s)=\begin{cases} A_1|s|^3+B_1|s|^2+C_1|s|+D_1 & 0<|s|<1 \\ A_2|s|^3+B_2|s|^2+C_2|s|+D_2 & 1<|s|<2 \\ A_3|s|^3+B_3|s|^2+C_3|s|+D_3 & 2<|s|<3 \\ 0 & 3<|s|. \end{cases} \tag{32}$$

If the interpolation kernel is continuous and differentiable, and if $u(0)=1$ and $u(n)=0$ when n is a nonzero integer, then ten independent equations can be derived for the unknown coefficients in (32). These equations are

$$\begin{aligned} 1&=D_1 \\ 0&=A_1+B_1+C_1+D_1 \\ 0&=A_2+B_2+C_2+D_2 \\ 0&=8A_2+4B_2+2C_2+D_2 \\ 0&=8A_3+4B_3+2C_3+D_3 \\ 0&=27A_3+9B_3+3C_3+D_3 \\ -C_1&=C_1 \\ 3A_1+2B_1+C_1&=3A_2+2B_2+C_2 \\ 12A_2+4B_2+C_2&=12A_3+4B_3+C_3 \\ 27A_3+6B_3+C_3&=0. \end{aligned}$$

Since there are twelve coefficients, two additional constraints must be imposed. If, as in the derivation of the cubic convolution interpolation kernel, the interpolation function is required to match the Taylor series expansion of the sampled function for as many terms as possible, then a unique solution can be obtained. When $A_2=-7/12$ and $A_3=1/12$, the interpolation function is a $0(h^4)$ approximation to the sampled function. The interpolation kernel for this case is

$$u(s)=\begin{cases} 4|s|^3/3-7|s|^2/3+1 & 0<|s|<1 \\ -7|s|^3/12+3|s|^2-59|s|/12+15/6 & 1<|s|<2 \\ |s|^3/12-2|s|^2/3+21|s|/12-3/2 & 2<|s|<3 \\ 0 & 3<|s|. \end{cases}$$

Fourth-order convergence is the highest order of convergence that can be achieved with piecewise cubic polynomials. To match the next higher order terms in the Taylor series expansion, s^4 terms are needed in the interpolation function.

The above interpolation function achieves fourth-order accuracy by increasing the number of sample points used for each interpolation point. Six sample points are needed for each point of interpolation compared to four for cubic convolution. Since fourth-order accuracy is the ultimate accuracy possible with piecewise cubic polynomials, it immediately follows that no further increase in accuracy can be gained by increasing the length of the interpolation kernel; higher order convergence rates require higher order piecewise polynomials.

Summary

The cubic convolution interpolation function is derived from a set of conditions imposed on the interpolation kernel. The cubic convolution interpolation kernel is composed of piecewise cubic polynomials defined on the unit subintervals between -2 and $+2$. The kernel is required to be symmetric, continuous, and have a continuous first derivative. It is further required for the interpolation kernel to be zero for all nonzero integers and one when its argument is zero. This condition has an important computational significance—namely, that the interpolation coefficients become simply the sampled data points. Finally, the cubic convolution interpolation function must agree with the Taylor series expansion of the function being interpolated for as many terms as possible. The interpolation kernel derived from these conditions is unique and results in a third-order approximation.

The cubic convolution interpolation function is more accurate than the nearest-neighbor algorithm or linear interpolation method. Although not as accurate as a cubic spline approximation, cubic convolution interpolation can be performed much more efficiently.

References

[1] S. S. Rifman, "Digital rectification of ERTS multispectral imagery," in *Proc. Symp. Significant Results Obtained from ERTS-1 (NASA SP-327)*, vol. I, sec. B, 1973, pp. 1131-1142.

[2] R. Bernstein, "Digital image processing of earth observation sensor data," *IBM J. Res. Develop.*, vol. 20, pp. 40-57, 1976.

[3] H. S. Hou and H. C. Andrews, "Cubic splines for image interpolation and digital filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, Dec. 1978.

[4] M. H. Schultz, *Spline Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1973.

Robert G. Keys was born in Tulsa, OK, on April 25, 1950. He received the B.S. degree in mathematics in 1972 and the M.S. degree in mathematics in 1974 from the University of Tulsa, Tulsa.

In 1974 he joined Amoco Production Company at their Exploration and Production Research Laboratory, Tulsa, where he was involved in seismic signal processing. In 1978 he joined Cities Service Oil Company, Tulsa, as a Research Geophysicist, where his area of work is the application of image processing techniques to remote sensing. He is currently studying for the Ph.D. degree in geophysics from the University of Tulsa. His research topic is the application of inverse scattering to seismic wave propagation.