FILTERING BY REPEATED INTEGRATION

Paul S. Heckbert †

Pacific Data Images
Sunnyvale, CA

ABSTRACT

Many applications of digital filtering require a space variant filter - one whose shape or size varies with position. The usual algorithm for such filters, direct convolution, is very costly for wide kernels. Image prefiltering provides an efficient alternative. We explore one prefiltering technique, *repeated integration*, which is a generalization of Crow's *summed area table*.

We find that convolution of a signal with any piecewise polynomial kernel of degree $n-1$ can be computed by integrating the signal n times and point sampling it several times for each output sample. The use of second or higher order integration permits relatively high quality filtering. The advantage over direct convolution is that the cost of repeated integration filtering does not increase with filter width. Generalization to two-dimensional image filtering is straightforward. Implementations of the simple technique are presented in both preprocessing and stream processing styles.

CR Categories: I.3.3 **[Computer Graphics]**: Picture/Image Generation – *display algorithms;* I.3.7 **[Computer Graphics]**: Three-Dimensional Graphics and Realism – *color, shading, shadowing, and texture;* I.4.3 **[Image Processing]**: Enhancement – *filtering.*

General Terms: algorithms, theory.

Additional Key Words and Phrases: space variant filter, convolution, texture mapping, blur.

INTRODUCTION

Filtering is one of the most common operations in signal processing, image processing, and raster image synthesis. Its applications include image enhancement, low pass filtering (blurring), antialiasing for image warping and texture mapping, and special effects. The fundamental filtering operation is convolution, in which a weighting function or *kernel* is passed over the input signal and a weighted average is computed for each output sample.

---

† Current address: Pixar, P.O. Box 13719, San Rafael, CA 94913

Appeared in Proceedings of SIGGRAPH '86, Dallas, Aug. 1986, pp. 317-321.
This version is missing figures.

The most straightforward filtering method is *direct convolution*, in which a weighted average is computed anew for each output sample. For a discrete signal of length M and a filter of width W, the number of operations required by direct convolution is $O(MW)$. The method is thus very expensive for wide kernels.

When the kernel's shape is independent of position the filter is called *space invariant*. Under these conditions we can use Fourier convolution: the signal and kernel are transformed to the frequency domain using an FFT, these are multiplied together, and an inverse FFT is computed [Brigham74]. The cost of this algorithm is $O(M \log M)$. Since the cost is independent of kernel width, Fourier convolution is the preferred method for wide, space invariant kernels.

Many applications demand a *space variant* filter, however, one for which the kernel varies with position. Such applications include nonlinear image warps, texture mapping, and depth of field filtering. Since Fourier convolution is inapplicable, direct convolution is usually used for space variant filtering [Feibush80]. This can be extremely slow for wide kernels, however.

An example from image synthesis will illustrate the problem. In texture mapping, texture images are stored in parametric surface coordinates and projected to the screen by modeling and viewing transformations [Heckbert86]. For proper antialiasing it is necessary to filter the texture area corresponding to each screen pixel [Feibush80]. But the transformations can cause these texture areas to be arbitrarily large (imagine a pixel containing the horizon of an infinite plane). Using direct convolution, accurate filtering of such pixels can be prohibitively expensive. In order to make high quality texture filtering practical we must find alternative techniques to direct convolution.

We are interested in wide kernels for another reason: in general, high quality low pass filters are wide. The ideal low pass filter, a sinc, has infinite width, but we must use finite width approximations to it in practice [Bracewell78].

The Problem

We would like to find efficient algorithms for performing space variant filtering with the criteria (a) cost independent of kernel width, and (b) enough kernel shape control to allow high quality filtering.

A general approach to this problem is signal prefiltering, in which the signal (or image) is preprocessed so that filtering with arbitrary width kernels can be performed in constant time [Heckbert86]. Two data structures have been proposed for this purpose: pyramids and integrated arrays.

Pyramid methods are common in image processing for a variety of purposes [Rosenfeld84], and in image synthesis for texture filtering [Dungan78], [Williams83]. Pyramids are ideal when the access kernel closely matches the kernel used in creating the pyramid, but can be awkward otherwise. Filtering of rectangular areas is inconvenient on pyramids created with square filters.

We will explore an alternative prefiltering technique, the integrated array, which was proposed by Perlin and Crow [Perlin85], [Crow84]. While they focused on applications of the technique for texture mapping and image blurring, our emphasis is on the theory behind the technique.

We begin with filtering of 1-D signals and later generalize to 2-D images.

ONE-DIMENSIONAL FILTERING

An unweighted average is equivalent to convolution with a box. The average value of a signal f between a and b is:

$$\int_a^b f(x)dx = \frac{F(b)-F(a)}{b-a}$$

where F is the indefinite integral of f, evaluated either symbolically or numerically. This formula can be used to convolve f with a box of width W if we choose $a = x - W/2$ and $b = x + W/2$. Computation with this formula is very attractive: for a discrete sequence f we precompute its partial sum F and do merely two table lookups, one subtraction, and one divide per output value. The cost is independent of kernel width, and the width can vary with position. This formula is thus a constant cost space variant filter, albeit a low quality one.

Can we generalize the above formula to higher quality filters? First let's look at some of those filters.

Repeated Box Filters

Consider a box kernel of width W:

$$\mathrm{box}_W(x) = \begin{cases} 1/W & |x| < W/2 \\ 0 & |x| \ge W/2 \end{cases}$$

When a box is convolved with itself n times, we call the result a *repeated box filter*:

$$\mathrm{box}^{*n} \; = \; \mathrm{box} * \mathrm{box} * \cdots * \mathrm{box} \quad \textit{(n times)}$$

where '*' denotes convolution and the superscript $*n$ denotes n-fold convolution.

The repeated box filters are commonly used in both signal processing and computer aided geometric design, although there has been little cross-pollination to date [Hou78], [Goldman83]. In the field of splines these functions are known as the uniform b-spline basis functions [Gordon74]. Figure 1 and the table below summarize the shapes, properties, and common names for low order box filters / b-splines.

order	degree	filter name	spline name	continuity
n=1	0	box, Fourier	nearest neighbor	none
n=2	1	triangle, Bartlett	linear interp.	C^0 (value)
n=3	2	parabolic	quadratic b-spline	C^1 (slope)
n=4	3	Parzen	cubic b-spline	C^2 (accel.)

The filters we have mentioned can be ranked in quality as follows:
$\mathrm{box} < \mathrm{box}^{*2} < \mathrm{box}^{*3} < \ldots <$ gaussian $<$ sinc. By the Central Limit Theorem,

as n increases the repeated box filters broaden and approach a gaussian in shape [Bracewell78].

Integration and Differentiation

Perlin showed how unweighted averaging can be generalized to repeated box filtering [Perlin84].

We'll need a few convolution identities:

$$
\begin{aligned}
f*g &= g*f \\
(f*g)*h &= f*(g*h) \\
\int (f*g) &= (\int f)*g \\
(f*g)' &= (f')*g
\end{aligned}
$$

Repeated application of the latter two identities implies that the convolution of f with g is equivalent to convolution of the nth integral of f with the nth derivative of g:

$$f*g = (\int^n f(x)dx)*(\frac{d^n g}{dx^n}) = f_n * g_{-n} \qquad (1)$$

where a subscript n on a function denotes n-fold integration and a subscript $-n$ denotes n-fold differentiation, a notation we will use henceforth. We will usually regard f as the signal and g as the filter kernel. This formula is especially useful when g_{-n} is a simple kernel such as a train of impulses.

An elegant proof of this identity employing Fourier transforms has been suggested by Perlin [Perlin84]. We write a Fourier transform pair as: $f(x) \longleftrightarrow F(\omega)$ [Bracewell78]. Noting that integration in the spatial domain corresponds to division by $i\omega$ in the frequency domain and differentiation in the spatial domain corresponds to multiplication by $i\omega$ in the frequency domain,

$$
\begin{aligned}
f_1(x) &\longleftrightarrow F(\omega)/(i\omega) \\
f_{-1}(x) &\longleftrightarrow i\omega F(\omega)
\end{aligned}
$$

we can write:

$$f(x)*g(x) \quad \longleftrightarrow \quad F(\omega)G(\omega)$$

$$f_n(x)*g_{-n}(x) \quad \longleftrightarrow \quad \frac{F(\omega)}{(i\omega)^n}(i\omega)^n G(\omega)$$

Reading these equations in clockwise order, we take the Fourier transform of the upper left to arrive at the upper right. Multiplying and dividing by $(i\omega)^n$ in the lower right changes nothing, but when we take its inverse Fourier transform we get the result at lower left. Equality of the spectra (on the right) implies equality of the signals (on the left). Q.E.D.

Repeated Box Filtering by Repeated Integration

By taking the nth derivative of an nth order box filter we will simplify it significantly:

$$g_{-n} \;=\; (\mathrm{box}^{*n})_{-n} \;=\; (\mathrm{box}')^{*n}$$

The derivative of a box is a train of two impulses with opposite signs (figure 2, top), a very simple filter for convolution purposes.

To proceed we switch to the discrete domain. We define the kth integral (partial sum) of f to be:

$$f_k(x) \;=\; \sum_{t=0}^{x} f_{k-1}(t)$$

where $f_0 = f$. We can write the derivative of a box as:

$$\mathrm{box}_W'(x) \;=\; [\delta(x+W/2) - \delta(x-W/2)]/W$$

where $\delta(x)$ is a discrete impulse function, defined to equal 1 for $x = 0$ but zero elsewhere.

When convolved with itself n times, box_W' creates a sequence of $n+1$ impulses each spaced W units apart (figure 2):

$$(\mathrm{box}_W')^{*n}(x) \;=\; W^{-n} \sum_{i=0}^{n} (-1)^i \binom{n}{i} \delta(x + (n/2 - i)W) \qquad (2)$$

Convolving this with the integrated array:

$$f * \mathrm{box}_W^{*n} \;=\; f_n * (\mathrm{box}_W')^{*n} \;=\; W^{-n} \sum_{i=0}^{n} (-1)^i \binom{n}{i} f_n(x + (n/2 - i)W)$$

since convolution with an impulse selects a single sample point: $f(x)*\delta(x-a) = f(x-a)$. In summary, we can convolve with an nth order box using $n+1$ samples of the integrated array f_n.

We have therefore succeeded in generalizing the box filtering algorithm to repeated box filters. Perlin called this a *selective image filter*, because it allows blurring by a different amount at each sample. Below is a basic implementation:

Implementation of Repeated Box Filtering

Setup requires integration of f n times:

```
for k := 1 to n
   for x := 1 to M-1
      f[x] := f[x-1] + f[x]
```

After setup, computation of each filtered sample requires $n+1$ accesses to the integrated array:

```
sum := 0
for i := 0 to n
   sum := sum + -1^i * comb(n,i) * f[x+(n/2-i)*W-n/2]
result := sum / W^n
```

In most implementations the order n of the filter will be fixed and the above loop can be unrolled. In order to allow x and W to have non-integer values all accesses to the integrated array **f** should be replaced by a linear interpolation between adjacent array entries, but Crow points out that this refinement is only needed for small kernels [Crow84].

Integration and Convolution

The techniques developed so far allow efficient convolution with repeated box filters, but none others. By studying the filter equivalent to repeated integration we can gain a deeper understanding of our new tool and thereby learn how to design other efficient filters.

Convolution with a step function is equivalent to integration, so n-fold convolution with a step is equivalent to n-fold integration. If we define δ_n, the nth order *ramp function*, as a step function δ_1 convolved with itself n times [Papoulis62]:

$$\delta_n = \delta_1 * \delta_1 * \cdots * \delta_1 = \delta_1^{*n}$$

then the relation between convolution and integration can be written:

$$f * \delta_n = f_n * \delta_0 = f_n \qquad (3)$$

An equivalent definition of δ_n is the nth integral of an impulse δ_0. Equation (3) is a special case of the Riemann-Liouville integral [Lavoie76].

Graphs of the ramp functions are shown in figure 3. Their formulas are:

$$\delta_n(x) = \begin{cases} x^{n-1}/(n-1)! & x \geq 0 \\ 0 & x < 0 \end{cases}$$

The ramp functions allow us to relate convolution with a polynomial kernel to convolution with an impulse, and thereby generalize repeated integration filtering to a wide variety of kernels.

Equations (1) and (3) suggest that we can choose the kernel g to be a superposition of translated ramp functions:

$$g(x) = \sum_i c_i \delta_{k_i}(x - x_i)$$

Note that we allow different orders of ramp functions to be mixed. The kernel can be designed at unit width and then scaled to width W using the formula $g(x/W)/W$. To scale ramp functions use the rule $\delta_n(ax) = a^{n-1}\delta_n(x)$.

For a kernel expressed as a sum of ramp functions, repeated integration filtering is performed by:

$$f * g = f_k * g_{-k} = \sum_i c_i f_{k_i}(x - x_i) \qquad (4)$$

Once we have expressed a filter as a sum of ramp functions the repeated integration formula can be read off simply by substituting f_k for δ_k.

The repeated box filters are easily described in terms of ramp functions:

$$\begin{aligned} \text{box}_W^{*1} &= W^{-1}[\delta_1(x + W/2) - \delta_1(x - W/2)] \\ \text{box}_W^{*2} &= W^{-2}[\delta_2(x + W) - 2\delta_2(x) + \delta_2(x - W)] \end{aligned}$$

as shown in figure 4, and, in general,

$$\mathrm{box}_W^{*n} \;=\; W^{-n} \sum_{i=0}^{n} (-1)^i \binom{n}{i} \delta_n(x + (n/2 - i)W)$$

The latter equation can be proven either by induction or by integrating equation (2) n times. The original definition of b-splines used an equivalent formula [Schoenberg46, p.70].

Arbitrary Polynomial Kernels

Equation (4) allows us to convolve with filters other than repeated boxes efficiently. In fact, any piecewise polynomial kernel can be expressed as a sum of ramp functions, and thereby convolved using repeated integration.

For example, the Overhauser or Catmull-Rom cubic spline is often used for filtering in image processing (where it is known as parametric cubic convolution) [Catmull74], [Park83]. It can be expressed using a combination of ramp functions of orders 3 and 4 (quadratic and cubic):

$$\begin{aligned}&3\delta_4(x+2) - 12\delta_4(x+1) + 18\delta_4(x) - 12\delta_4(x-1) + 3\delta_4(x-2)\\ &- \delta_3(x+2) \;+\; 2\delta_3(x+1) \qquad\qquad - \;2\delta_3(x-1) \;+\; \delta_3(x-2)\end{aligned}$$

Repeated integration filtering with this kernel requires retention of both 3rd and 4th order integrated arrays in memory.

Filtering On-The-Fly

The algorithm above can be summarized as:

(1) integrate signal as a preprocess
(2) convolve with impulse train with spacing W

Under some circumstances the signal is too large to hold in memory at one time, or it is desirable to filter in real-time as a stream process. This can be accomplished by reversing the order of operations [Perlin85]:

(1) convolve signal with impulse train
(2) integrate

This swap is analogous to the identity $f_n * g_{-n} \;=\; (f * g_{-n})_n$.

An implementation of filtering on-the-fly for repeated box filters is given below:

```
initialize diff array
for x
   input f[x+n*W/2]
   sum := 0
   for i := 0 to n
      sum := sum + -1^i * comb(n,i) * f[x+(n/2-i)*W-n/2]
   diff[n] := sum
   for i := n-1 to 0 step -1
      diff[i] := diff[i] + diff[i+1]
   output diff[0] / W^n
```

This is a generalization of the well-known add-incoming, subtract-outgoing box blur algorithm [Schowengerdt83]. The **f** array above can be stored as a circular buffer of length nW, so the memory requirements are minimal.

This method of filtering on-the-fly is limited, however, to fixed values of W, so it is space *in*variant.

Cost Analysis

A cost comparison demonstrates the efficiency of filtering by repeated integration. Consider a signal of length M convolved n times with a box filter of width W. Using direct convolution n times the cost would be $O(nMW)$. Using repeated integration filtering, the setup time and convolution time are both $O(nM)$. For more general kernels the time cost is proportional to the number of terms in its ramp representation. The maximum number of terms for a piecewise polynomial of degree $n-1$ with K singularities is nK.

One of the disadvantages of repeated integration filtering is the storage cost of integrated arrays. For a signal of length $M = 2^r$ with b bits per sample, integration n times requires a dynamic range of $nr + b$ bits. For example, an 8-bit signal of length 1024 has $b = 8$ and $r = 10$, so the number of bits needed for n=1, 2, 3, and 4 is 18, 28, 38, and 48, respectively.

Such large dynamic ranges are not required when the filter width is limited, however. The variable **sum** in the implementations above has a dynamic range of $n \log_2 W + b$ bits, so if we use modulo arithmetic with at least this number of bits the standard evaluation formula will work perfectly. This trick is helpful if we know an upper bound on W a priori. Imposing a limit on width weakens one of the principal advantages of repeated integration filtering, however.

The best storage technique is probably a hybrid of full $(nr + b)$ and limited $(n \log_2 W + b)$ dynamic range arrays. We can store most of the data at a low dynamic range (say 16 bits) and record changes in higher bits of the integrated array in a sparse array of full dynamic range (say 32 or 64 bits). With this scheme, only very wide kernels require the full range array to be consulted.

TWO-DIMENSIONAL FILTERING

The methods above generalize quite simply to two dimensions. The signal f must be integrated along each dimension:

$$f_{k,l}(x, y) \;=\; \sum_{u=0}^{x} f_{k-1,l}(u, y) \;=\; \sum_{v=0}^{y} f_{k,l-1}(x, v) \quad \text{where} \quad f_{0,0} = f$$

and evaluated at a grid of sample points. For a separable kernel

$$g(x)h(y) \;=\; \sum_i c_i \delta_{k_i}(x - x_i) \;\; \sum_j d_j \delta_{l_j}(y - y_j)$$

the convolution is

$$f * gh \;=\; \sum_i \sum_j c_i d_j f_{k_i, l_j}(x - x_i, y - y_j)$$

Note that the orders of integration in x and y can be different.

Convolution with an nth order box requires $(n+1)^2$ array accesses. First order box filtering is equivalent to Crow's *summed area table* [Crow84]. Perlin has used second order filtering for a number of applications

[Perlin85]. The grids of coefficients for low order filters are shown below:

$$
1 \qquad \begin{matrix} -1 & 1 \\ 1 & -1 \end{matrix} \qquad \begin{matrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{matrix} \qquad \begin{matrix} -1 & 3 & -3 & 1 \\ 3 & -9 & 9 & -3 \\ -3 & 9 & -9 & 3 \\ 1 & -3 & 3 & -1 \end{matrix}
$$

It is desirable to use higher order integration, $n \geq 2$, in order to minimize artifacts. The first order filter, a box, generates fairly strong horizontal and vertical mach bands, as shown in figure 6. As the order n is increased, the x and y cross sections converge toward a gaussian, and because of the gaussian's separability, contours of the two-dimensional filter converge toward circularity. In figure 7, second order filtering has eliminated the mach bands. Third and higher order box filters are nearly the same.

Figure 8 demonstrates Catmull-Rom filtering of text. New filter kernels are easy to test: figure 9 illustrates the use of biased kernels to simulate a common artistic stylization of motion blur. The space variant application of texture mapping is shown in figures 10-12. Note that the improvement from zeroth order (point sampling) to first order (box filtering) is much greater than the improvement from first order to second order (triangle filtering). These images were computed at far lower than normal resolution in order to clearly depict the filtering effects.

Repeated integration filtering in multiple dimensions is not limited to separable kernels. One can simply superpose several kernels to construct new ones. For texture mapping one would like to filter arbitrarily oriented ellipses [Greene86]. This is possible with repeated integration filtering, but it appears that for non-rectilinearly oriented areas the number of array samples required increases with the ellipse eccentricity.

For high resolution images it is sometimes impossible to store an entire image in memory at once. In such cases the stream oriented filtering algorithm is helpful, since it can perform a high quality blur using a buffer of only nW scan lines. Of course, if the kernel is separable, the convolution could also be done in two passes, one horizontal and one vertical: $f * g * h$.

Storage Cost in 2-D

The dynamic range of 2-D integrated arrays is quite substantial. For a $2^r \times 2^r$ monochrome image with b bits per pixel, 2-D integration n times yields $2nr + b$ bits per array element. For example, an 8-bit 512x512 image has $r = 9$ and $b = 8$, so 26 bits are required for $n = 1$ (box filtering, a.k.a. summed area table), 44 bits are needed for $n = 2$ (triangle filtering), 62 bits for $n = 3$ (parabolic filtering), and 80 bits for $n = 4$ (cubic filtering).

As mentioned earlier, however, we need not store the entire integrated array at full dynamic range. To compact the summed area table, Crow suggested that the image be divided into blocks, say 16x16 in size, and that a mini 16-bit limited range integrated array be created for each [Crow84]. In addition, the totals for each block are integrated to form a sparse 32-bit full range array. Unfortunately, this makes integration of a rectangular area rather cumbersome, as it is no longer simply a matter of sampling the four corners (see Crow's comments on wraparound). For large kernels the number of samples required increases in proportion to the side length of the rectangle, but for kernels of limited size the full range array is unnecessary and the number of samples is constant. In either case the asymptotic cost is less than that of direct convolution, which grows in proportion to the *area* of the rectangle.

Using the hybrid storage technique, therefore, the storage cost for integrated arrays is about 2 times that of the original image. Another prefiltering technique, the image pyramid, has a storage factor of only 1.33 [Heckbert86].

CONCLUSIONS

By relating convolution and integration we have generalized definite integration to weighted averaging and thereby developed efficient algorithms for space variant filtering with any piecewise polynomial kernel. Since its cost is independent of filter width, the repeated integration filter is clearly more efficient than direct convolution for wide kernels. The method generalizes easily to the two dimensional domain. Its possible applications are numerous, including audio signal processing, image blurring, nonlinear image warps, motion blur, antialiasing, texture mapping, and depth of field filtering.

Further work is needed on storage economization and two-dimensional kernel shape control. In addition, alternatives to unweighted preintegration and independent integration in x and y should be explored.

Compared to standard pyramid techniques, repeated integration filtering gives greater filter shape control and hence higher quality but at higher storage cost [Crow84], [Heckbert86]. The use of arbitrarily oriented elliptical filters on image pyramids promises to improve pyramidal filtering quality, however [Greene86]. A definitive comparison between various space variant filtering methods has yet to be made.

ACKNOWLEDGEMENTS

The original idea of filtering by repeated integration is due to Ken Perlin. A fertile environment for discussing such topics was provided by Pat Hanrahan and the 3-D Systems Group at NYIT's Computer Graphics Lab. Thanks also to the enthusiastic bunch at PDI, and their addictive Marvin Gaye theme song.

REFERENCES

Bracewell, Ronald N., *The Fourier Transform and Its Applications,* McGraw-Hill, New York, 1978.

Brigham, E. Oran, *The Fast Fourier Transform,* Prentice-Hall, Englewood Cliffs, NJ, 1974.

Catmull, Edwin E., and Raphael Rom, "A Class of Local Interpolating Splines", *Computer Aided Geometric Design,* Robert Barnhill and Richard Riesenfeld, eds., Academic Press, New York, 1974, pp. 317-326.

Crow, Franklin C., "Summed-Area Tables for Texture Mapping", *Computer Graphics,* (SIGGRAPH '84 Proceedings), vol. 18, no. 3, July 1984, pp. 207-212.

Dungan, William, Jr., Anthony Stenger, and George Sutty, "Texture Tile Considerations for Raster Graphics", *Computer Graphics,* (SIGGRAPH '78 Proceedings), vol. 12, no. 3, Aug. 1978, pp. 130-134.

Feibush, Eliot A., Marc Levoy, and Robert L. Cook, "Synthetic Texturing Using Digital Filters", *Computer Graphics,* (SIGGRAPH '80 Proceedings), vol. 14, no. 3, July 1980, pp. 294-301.

Goldman, Ronald N., "An Urnful of Blending Functions", *IEEE Computer Graphics and Applications,* vol. 3, Oct. 1983, pp. 49-54.

Gordon, William J., and Richard F. Riesenfeld, "B-spline Curves and Surfaces", *Computer Aided Geometric Design,* Robert Barnhill and Richard Riesenfeld, eds., Academic Press, New York, 1974, pp. 95-126.

Greene, Ned, and Paul S. Heckbert, "Creating Raster Omnimax Images from Multiple Perspective Views Using The Elliptical Weighted Average Filter", *IEEE Computer Graphics and Applications,* June 1986.

Heckbert, Paul S., "Survey of Texture Mapping", *Graphics Interface '86,* May 1986.

Hou, Hsieh S., and Harry C. Andrews, "Cubic Splines for Image Interpolation and Digital Filtering", *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. ASSP-26, no. 6, Dec. 1978, pp. 508-517.

Lavoie, J. L., Thomas J. Osler, and R. Tremblay, "Fractional Derivatives of Special Functions", *SIAM Review,* vol. 18, 1976, pp. 240-268.

Papoulis, R., *The Fourier Integral and its Applications,* McGraw-Hill, New York, 1962.

Park, Stephen K., and Robert A. Schowengerdt, "Image Reconstruction by Parametric Cubic Convolution", *Computer Vision, Graphics, and Image Processing,* vol. 23, no. 3, Sept. 1983, pp. 258-272.

Perlin, Kenneth, personal communication, Jan. 1984.

Perlin, Kenneth, "Course Notes", *SIGGRAPH '85 State of the Art in Image Synthesis seminar notes,* July 1985.

Rosenfeld, A., ed., *Multiresolution Image Processing and Analysis,* Springer, Berlin, 1984.

Schoenberg, I. J., "Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions", *Quart. Applied Math,* vol. 4, 1946, pp. 45-99, 112-141.

Schowengerdt, Robert A., *Techniques for Image Processing and Classification in Remote Sensing,* Academic Press, New York, 1983.

Williams, Lance, ''Pyramidal Parametrics'', *Computer Graphics,* (SIGGRAPH '83 proceedings), vol. 17, no. 3, July 1983, pp. 1-11.