APPLICATION OF CONTINUOUS, ORTHONORMAL WAVELETS TO IMAGE DATA REDUCTION

George A. Geri

University of Dayton Research Institute
300 College Park Avenue
Dayton, OH 45469

Izidor C. Gertner

Computer Science Department
City College of CUNY
Convent Avenue and 138th Street
New York, NY 10031

HUMAN RESOURCES DIRECTORATE
AIRCREW TRAINING RESEARCH DIVISION
6001 South Power Road, Bldg. 558
Mesa, AZ 85206-0904

April 1994

Final Technical Report for Period October 1992 - December 1993

Approved for public release; distribution is unlimited.

AIR FORCE MATERIEL COMMAND

NOTICES

This technical report is published as received and has not been edited by the technical editing staff of the Armstrong Laboratory.

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Office of Public Affairs has reviewed this paper, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.

Elizabeth L. Martin
ELIZABETH L. MARTIN
Project Scientist

Dee H. Andrews
DEE H. ANDREWS, Technical Director
Aircrew Training Research Division

LYNN A. CARROLL, Colonel, USAF
Chief, Aircrew Training Research Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1994	Final October 1992 - December 1993

4. TITLE AND SUBTITLE

Application of Continuous, Orthonormal Wavelets to Image Data Reduction

5. FUNDING NUMBERS

C - F33615-90-C-0005
PE - 62205F
PR - 1123
TA - 03
WU - 85

6. AUTHOR(S)

George A. Geri
Izidor C. Gertner

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Dayton Research Institute
300 College Park Avenue
Dayton, OH 45469

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Armstrong Laboratory (AFMC)
Human Resources Directorate
Aircrew Training Research Division
6001 South Power Road, Bldg. 558
Mesa, AZ 85206-0904

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AL/HR-TR-1994-0031

11. SUPPLEMENTARY NOTES

Armstrong Laboratory Technical Monitor: Dr. Elizabeth L. Martin, (602) 988-6561.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Images are decomposed and reconstructed by applying recently developed multiresolution techniques to a particular class of orthonormal wavelets. An original image is decomposed into a series of sub-images in which the low- and high-frequency information of the original image has been segregated. The image is then reconstructed after any redundant high-frequency information is discarded. This procedure allows a perceptually equivalent image to be generated using less information. Further, the resulting reduced-data image can be stored, processed, transmitted and displayed more efficiently. The advantages of the technique are that the wavelets used are continuous and well-localized in both space and spatial-frequency, the reconstruction can be performed by simply reversing the multiresolution decomposition process, and the reconstruction is error-free. The C source code required to implement the present technique on a PC-compatible computer is also presented.

14. SUBJECT TERMS

Data compression
Data reduction
Image processing
Multiresolution analysis
Orthonormal bases
Wavelets

15. NUMBER OF PAGES

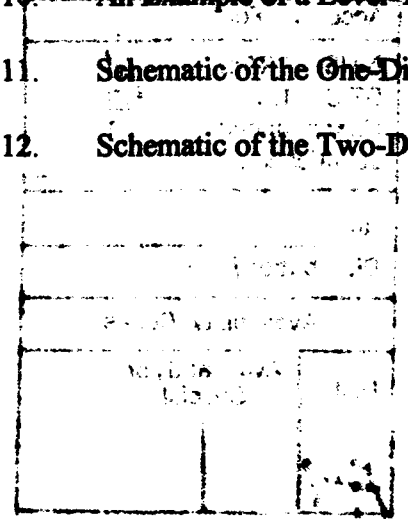
48

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

CONTENTS

List of Figures



PREFACE

The research reported here was conducted in support of the Armstrong Laboratory/Aircrew Training Research Division (AL/HR) under Work Unit 1123-03-85, Flying Training Research Support.

This research was supported by Air Force Contract F33615-90-C-0005 (UDRI). The laboratory contract monitor was Mrs. Patricia Spears.

The authors thank Colonel Lynn Carroll for supporting the Image Generator Project under which the present research was performed. We also thank Drs. Byron Pierce and Elizabeth Martin for their encouragement and for administrative support. Some of the graphics routines implemented in the programs included here were written by Mr. Craig Vrana.

APPLICATION OF CONTINUOUS, ORTHONORMAL WAVELETS TO IMAGE DATA REDUCTION

GENERAL INTRODUCTION

It is often possible to adequately represent a digital image using significantly less information than is required to specify each individual picture element. Such a representation is useful since the image can then be stored, transmitted, and displayed more efficiently (see, e.g., Geri, Zeevi, & Porat, 1990; Gertner & Geri, 1994). The classical techniques of Fourier analysis (FA) have been used to efficiently represent images by a set of spectral functions which, when added together, will adequately reproduce the original image. Fourier analysis however, is not ideal for representing natural images since the latter are generally nonstationary -- that is, their characteristics vary as a function of location within the image. The spectral functions of FA each encompass the entire image, and hence they are not well suited for representing the localized features of natural images.

Wavelet decomposition involves finding image expansion coefficients with respect to a basis derived from those wavelets. The wavelet coefficients, or some subset of them, can then be stored, transmitted, compressed, and used for image generation. The efficient representation of images, therefore, requires that coefficients be discarded such that the image generated from the remaining coefficients is perceptually equivalent to the original. We have decomposed images and reconstructed reduced versions of them by selectively discarding high-frequency components that are known to be less important in conveying visual form information. The reduced, full gray-scale, multiresolution images are suitable for use in visual simulators.

INTRODUCTION TO WAVELETS

Many joint position/spatial-frequency techniques have been developed for overcoming the limitations of FA described above (cf., Jacobson & Wechsler, 1988). The most straightforward way to obtain a joint representation is to simply multiply the signal by an appropriate window

function, compute the Fourier transform of the product, and repeat the procedure for windows translated in position. This procedure defines the *short-time (or windowed) Fourier transform* that allows a signal (or image) to be represented in both position and spatial frequency. Any smooth, spatially-localized function can be used as a window function, and many have been tried. Gabor (1946) has shown, however, that a window function in the form of a gaussian provides the greatest joint localization in position and spatial frequency. The Gabor representation is a special case of what are known, in the physics literature, as coherent states associated with the Weyl-Heisenberg group (Daubechies, 1992).

In the present context, the Weyl-Heisenberg group represents one of two classes of what are now referred to as *wavelets*. Wavelets are sets of functions that are formed by applying dilation and/or translation operators. Thus, we can define a class of *Weyl-Heisenberg group* wavelets, which are generated by a translation in both space and spatial frequency as follows:

$$g_{a,b}(s) = g(s-b) \cdot e^{i2\pi as} \quad , \tag{1}$$

where $g(s-b)$ is a window function that can be translated in space by an amount proportional to b, and the complex exponential represents spectral functions translated in frequency by an amount proportional to a. Examples of Weyl-Heisenberg group wavelets are shown in Figure 1a. These wavelets all have the same window function (a gaussian in this case) that has been translated in space and multiplied by spectral functions of various spatial frequencies.

The second class of wavelets are the *affine group* wavelets, which are generated by dilations and translations in space as follows:

$$\psi_{a,b}(s) = |a|^{-\frac{1}{2}} \psi(\frac{s-b}{a}) \quad , \tag{2}$$

where the parameter a represents dilation, and the parameter b represents translation. Examples of affine group wavelets are shown in Figure 1b. These wavelets were developed to analyze transients in signals (or images), which had previously been dealt with by non-optimal, and largely *ad hoc*, techniques.

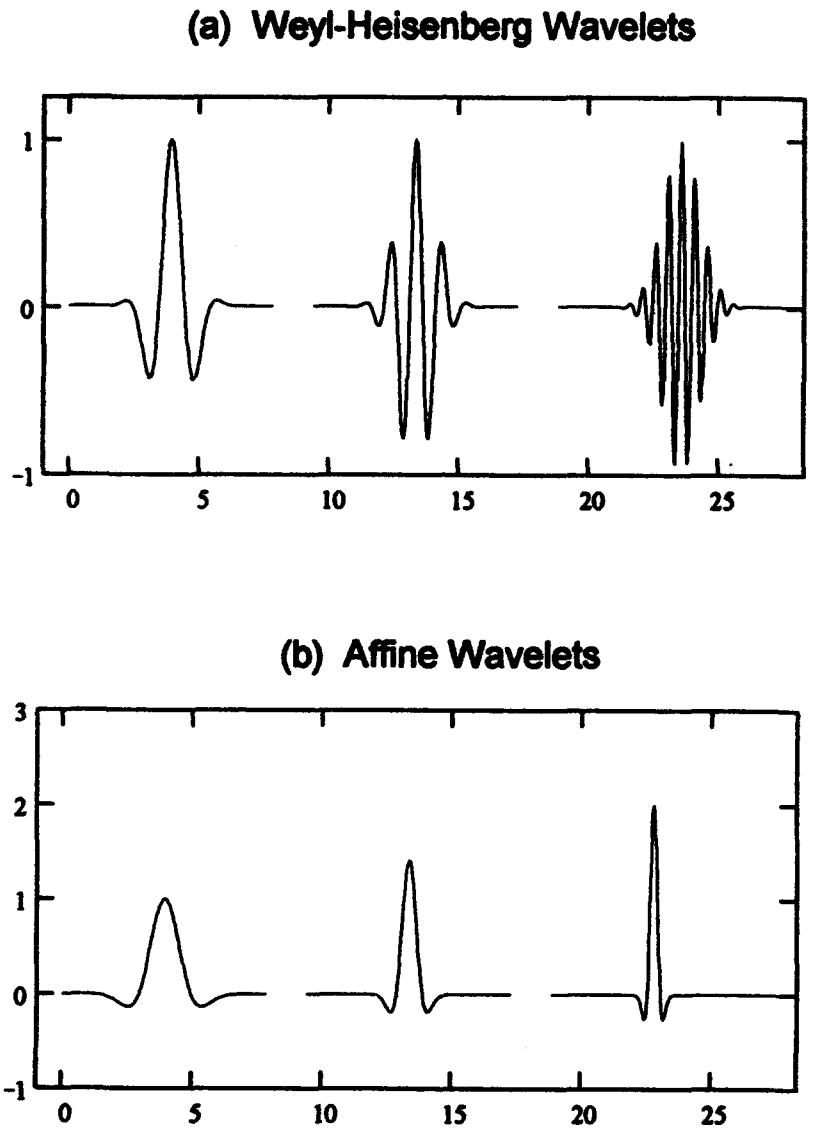


Figure 1
Examples of (a) Weyl-Heisenberg Wavelets and (b) Affine Wavelets.

As noted above, one major motivation for developing wavelet techniques is the requirement for a joint space and spatial frequency analysis to overcome the limitation of FA, which assumes infinite resolution in one domain and no resolution in the other. In the context of image representation, wavelets are used to decompose images, and the decomposition is most efficient when the image information represented by each of the wavelets in the set is nonredundant. Wavelets which allow such a nonredundant representation are called *orthogonal*, and can be used to represent images with a minimum number of coefficients. The affine-group wavelets are orthogonal whereas the Weyl-Heisenberg group wavelets are not. The affine-group wavelets are also better suited for representing transients and other high-frequency image components. Since our major objective is image data-reduction (as a first step toward image compression), we are more concerned with the above-mentioned desirable properties of affine-group wavelets and less concerned with the high joint localization attainable using the Weyl-Heisenberg group wavelets. We will, therefore, limit the remainder of our discussion to affine group wavelets.

CONSTRUCTION OF AFFINE-GROUP WAVELETS

The affine-group wavelets to be described here fall into two categories that will be referred to as *scaling functions* and *wavelets*. Scaling functions are obtained as solutions to a two-scale difference equation of the form:

$$\phi(x) = \sum_k c_k \, \phi(2x - k) \quad , \tag{3}$$

where the c_k are a set of coefficients.

Wavelets are obtained as solutions to a difference equation of the form:

$$\psi(x) = \sum_k (-1)^{k+1} c_{N-(k+1)} \, \phi(2x - k) \quad , \tag{4}$$

where N is the number of coefficients. Unique solutions to Equations 3 and 4 are guaranteed if the sum of the coefficients, c_k, is exactly two (cf., Strang, 1989). The solutions can be obtained starting from any of a number of initial conditions, denoted $\phi^{(0)}(x)$, for $\phi(x)$. The simplest initial condition is the "box" function, $\phi^{(0)}_B(x)$, shown in Figure 2a. Using the box function, various interesting and useful scaling functions and wavelet pairs can be obtained by using different sets of coefficients. For instance, when $c_0 = c_1 = 1$, a scaling function can be obtained as a solution to Equation 3:

$$\phi^{(1)}_H(x) = \phi^{(0)}_B(2x) + \phi^{(0)}_B(2x-1) \quad , \tag{5}$$

which is simply the box function again (i.e., $\phi^{(j+1)}=\phi^{(j)}$). This construction is shown graphically in Figure 2. The associated wavelet, for $c_0 = c_1 = 1$, is the solution to Equation 4:

$$\psi^{(1)}_H(x) = -\phi^{(0)}_B(2x) + \phi^{(0)}_B(2x-1) \quad , \tag{6}$$

which is illustrated graphically in Figure 2. The function, $\psi_H(x)$, known as Haar's wavelet, is simple in form and was the first function to be used to generate an orthogonal basis. However, whereas $\psi_H(x)$ is well localized in space, it has a discontinuity, and so does not provide good frequency localization. It is, therefore, not ideal for image representation.

Another scaling function and wavelet can be obtained again using the box function, but with the coefficients $c_0 = ½$, $c_1 = 1$, and $c_2 = ½$. The scaling function is obtained using Equation 3 as:

$$\phi^{(1)}_T(x) = \tfrac{1}{2}\phi^{(0)}_B(2x) + \phi^{(0)}_B(2x-1) + \tfrac{1}{2}\phi^{(0)}_B(2x-2) \quad , \tag{7}$$

and the corresponding wavelet is obtained from Eqn. (4) as:

$$\psi^{(1)}_T(x) = -\tfrac{1}{2}\phi^{(0)}_B(2x) + \phi^{(0)}_B(2x-1) - \tfrac{1}{2}\phi^{(0)}_B(2x-2) \quad . \tag{8}$$

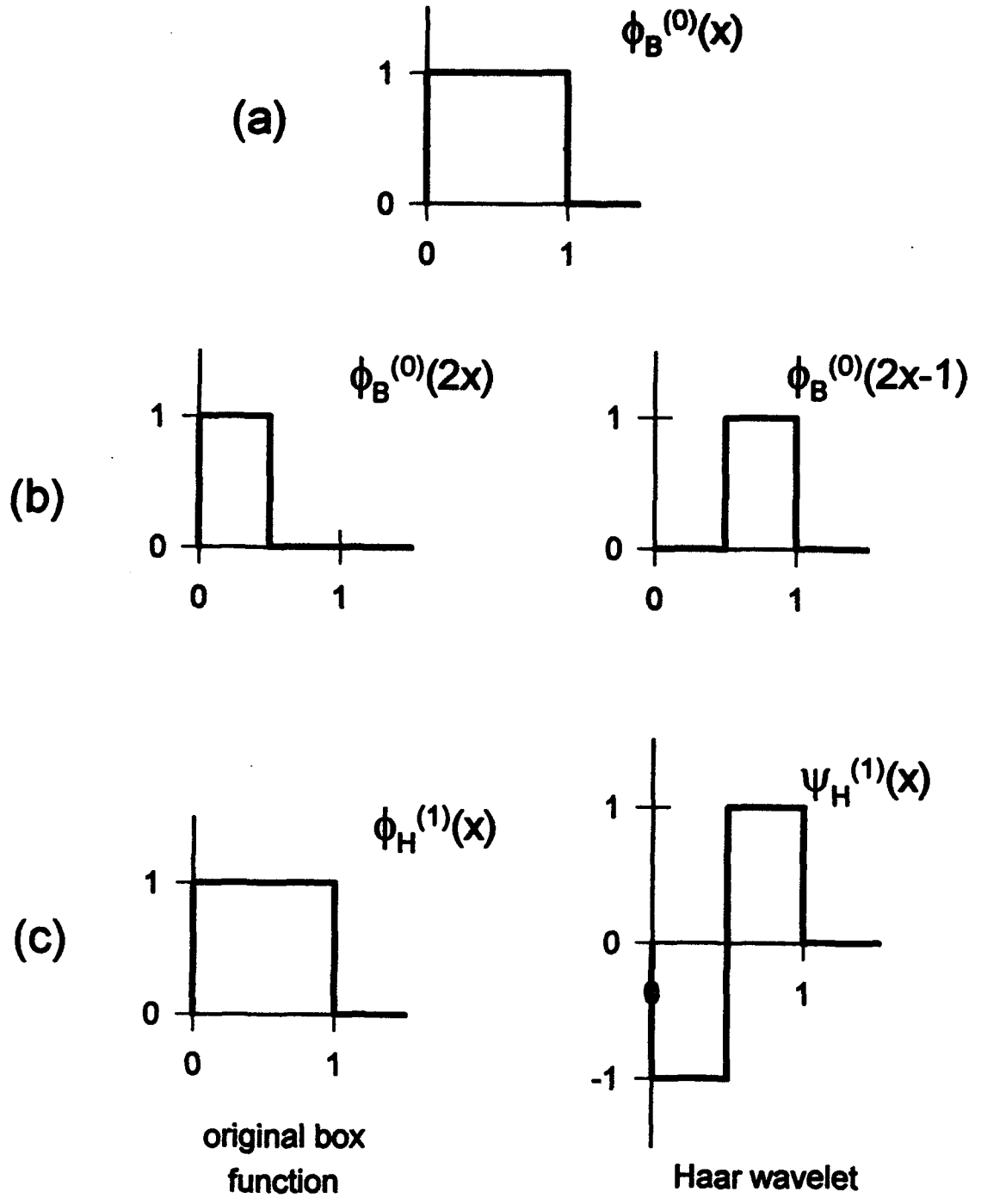


Figure 2
Generation of a Scaling Function and Haar Wavelet (c) from the Scaled and Translated "Box" Functions shown in (a) and (b).

The results of four iterations using Equation 7 and 8 are depicted graphically in Figures 3a and 3b, respectively. The scaling function shown in Figure 3a is triangular function, while the wavelet of Figure 3b is the so-called "hat" function. Although the hat function is smoother than the Haar wavelet, it is not orthogonal to its translations and dilations and thus yields a nonorthogonal basis.

A final example of a useful scaling function and wavelet pair is the cubic B-spline obtained using the coefficients $c_0 = 1/8$; $c_1 = 4/8$; $c_2 = 6/8$; $c_3 = 4/8$; and $c_4 = 1/8$. The first through fourth iterations of these functions are shown in Figures 4a and 4b, respectively. The scaling function is smooth and may represent a good approximation to a gaussian low-pass filter. The wavelet, on the other hand, has a triphasic form which is similar to that of both measured and theoretical visual receptive field profiles (cf., Young, 1987).

A Continuous, Localized, Orthonormal Wavelet

A family of wavelets (actually scaling functions and wavelets) of both practical and theoretical importance was devised by Daubechies (1988). The scaling functions and wavelets shown in Figures 2 through 4 were generated using rational coefficients. Daubechies found a set of irrational coefficients, which resulted in scaling functions and wavelets that are *orthonormal* to their integer translations and dilations. These functions are particularly useful because both they and their Fourier transforms have compact support (i.e., are of finite extent) and are continuous. Since we will be using this scaling function and wavelet to decompose and reconstruct images, the technique for generating them will now be described in detail.

Since all initial conditions converge to the same function, for simplicity we will again use the box function, $\phi^{(0)}_B(x)$ to generate the scaling function and wavelet. The set of four coefficients suggested by Daubechies (1988) are:

$$c_0 = \tfrac{1}{4}\left(1 + \sqrt{3}\right) = 0.683013$$

$$c_1 = \tfrac{1}{4}\left(3 + \sqrt{3}\right) = 1.18301$$

$$c_2 = \tfrac{1}{4}\left(3 - \sqrt{3}\right) = 0.316987$$

$$c_3 = \tfrac{1}{4}\left(1 - \sqrt{3}\right) = -0.183013$$

Using Equation 3 and following the procedures described earlier, the first-iteration scaling function is obtained as:

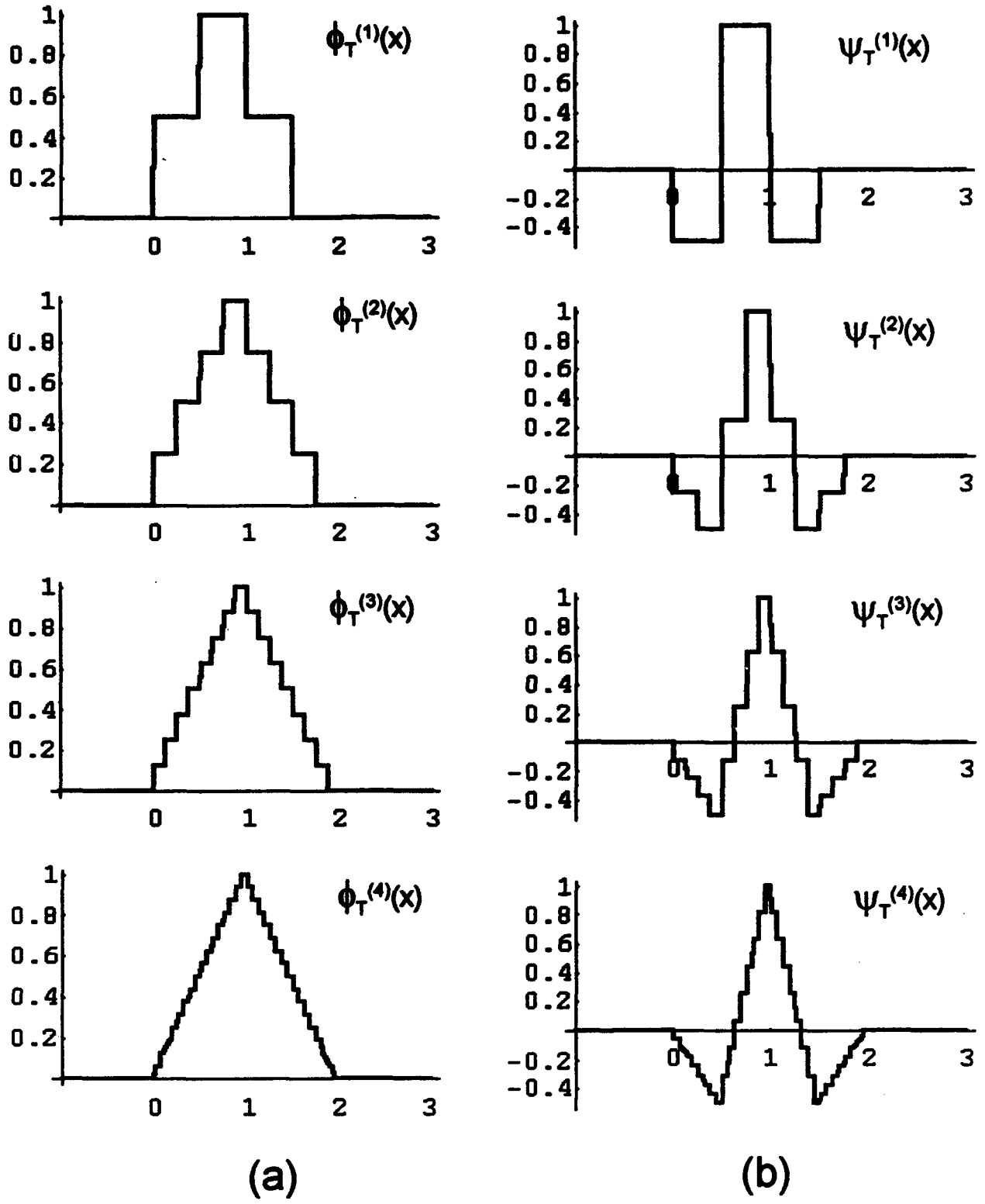


Figure 3
Four Iterations in the Generation of (a) a Triangular Scaling Function and (b) a "Hat" Wavelet

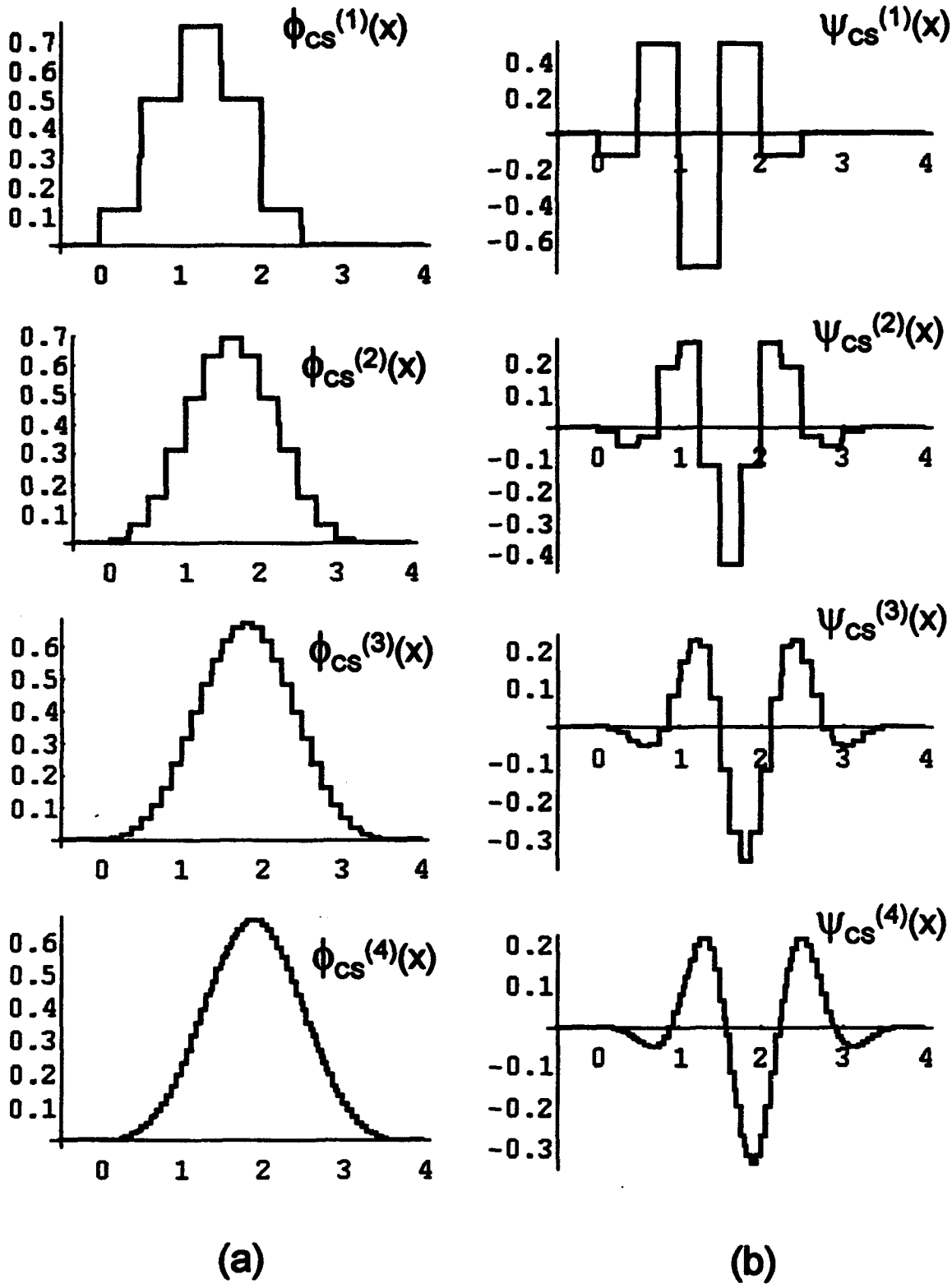


Figure 4
Four Iterations in the Generation of (a) a Cubic-Spline Scaling Function and (b) a Cubic-Spline Wavelet

$$\phi_D^{(1)}(x) = 0.683\,\phi_B^{(0)}(2x) + 1.183\,\phi_B^{(0)}(2x-1) + 0.317\,\phi_B^{(0)}(2x-2) + (-0.183)\,\phi_B^{(0)}(2x-3) \qquad (9)$$

The function $\phi^{(1)}{}_D(x)$ is shown graphically at the top of Figure 5a. The second-iteration scaling function is then obtained from $\phi^{(1)}{}_D(x)$ as follows:

$$\phi_D^{(2)}(x) = 0.683\,\phi_D^{(1)}(2x) + 1.183\,\phi_D^{(1)}(2x-1) + 0.317\,\phi_D^{(1)}(2x-2) + (-0.183)\,\phi_D^{(1)}(2x-3) \qquad (10)$$

This procedure is repeated to obtain all higher-iteration scaling functions as required. The second through fourth-iteration Daubechies scaling functions are also shown in Figure 5a. As a practical matter, no significant changes occur in this function after the seventh iteration.

In analogous fashion, but using Equation 4 rather than Equation 3, the first-iteration Daubechies wavelet can be obtained as follows :

$$\psi_D^{(1)}(x) = 0.183\,\phi_D^{(0)}(2x) + 0.317\,\phi_D^{(0)}(2x-1) - 1.183\,\phi_D^{(1)}(2x-2) + 0.683\,\phi_D^{(1)}(2x-3) \; . \qquad (11)$$

The first-iteration wavelet is shown in Figure 5b along with the second- through fourth-iteration wavelets. In the next section we will describe a technique for constructing an orthonormal basis from the scaling function and wavelet of Figure 5. Such a basis yields a better joint representation than does that associated with Haar's wavelet, but at the expense of basis-function regularity.

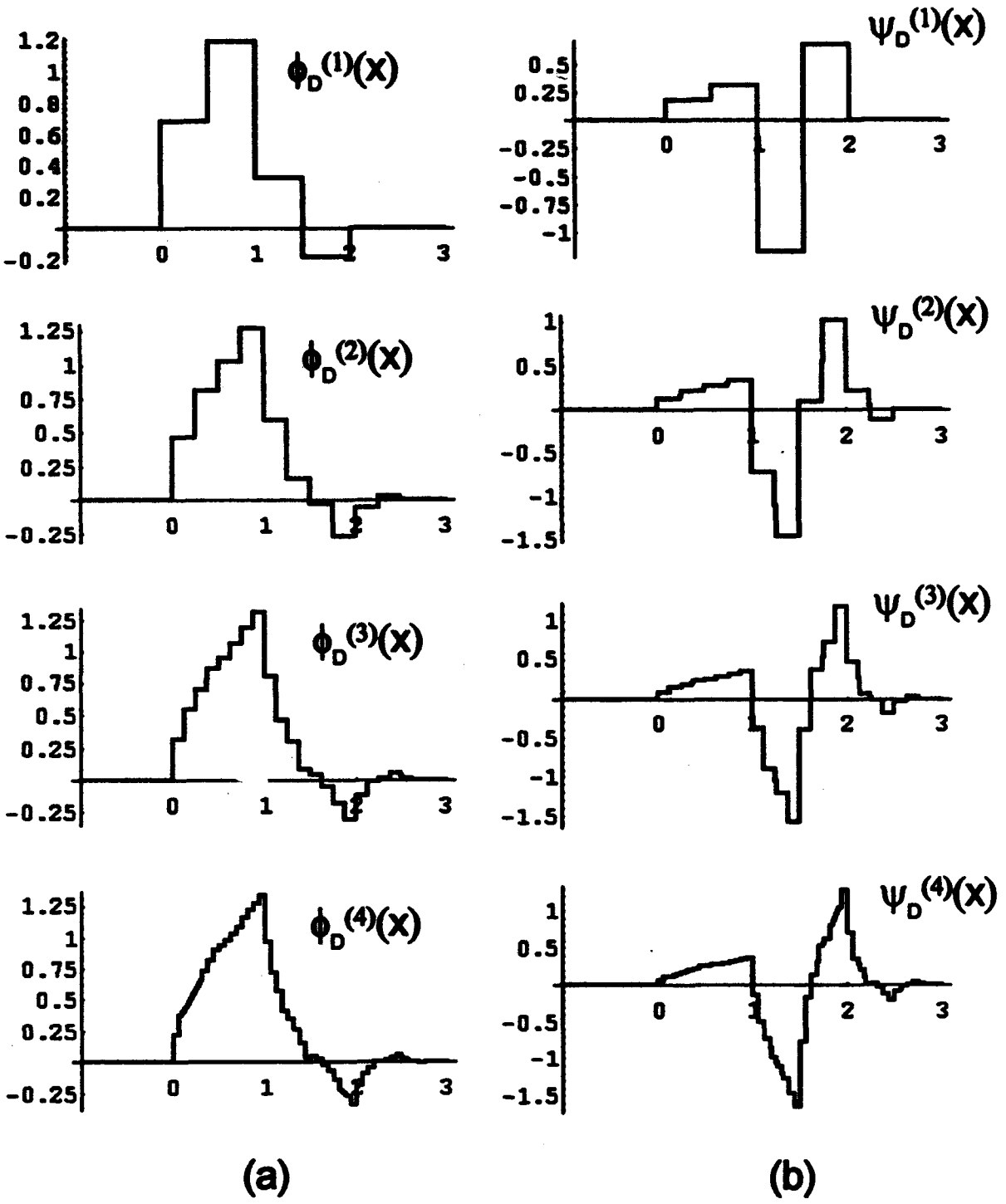


Figure 5
Four Iterations in the Generation of a Continuous,
Orthonormal Scaling Function (a) and Wavelet (b).

IMAGE REPRESENTATION USING WAVELETS

Introduction

Image decomposition is a procedure for obtaining an alternative representation (i.e., a transformation) of an image. The procedure results in a set of numbers, called *expansion coefficients*, which represent the amplitudes of each of a set of specialized functions (known collectively as a *basis*) that can then be summated to produce the original image. The usefulness of image decomposition is that certain image components, which may be more important than others in a particular context, can be identified and isolated. Once isolated, these components can be selectively subjected to further processing. In the context of images, the decomposition procedure partitions information into two domains, namely *space* (or position) and *spatial-frequency*. Since the space domain is the image itself, most of the novel information resulting from a decomposition is in the spatial-frequency domain. Further, it is often the case that components of interest are segregated in the spatial-frequency domain. This observation is the major motivation for the development of *multiresolution (or pyramid)* decomposition techniques.

In multiresolution decomposition (cf., Akansu & Haddad, 1992), an image is filtered into low- and high-frequency components and subsampled, and this procedure is repeated as required. Multiresolution analysis was originally done (Burt & Adelson, 1983) using second derivatives of gaussians (i.e., Laplacian functions). Mallat (1989) formalized the procedure and extended it by using symmetric, non-orthogonal wavelets, since orthonormal wavelets were not available at that time (see Figure 1 in his Appendix A). As noted above, Daubechies (1988) has derived orthonormal wavelets, which while not symmetric, do have compact support and reasonably good joint localization in space and spatial-frequency. It was not immediately clear that orthonormal wavelets were suitable for multiresolution image decomposition and reconstruction because they were not symmetric, and image processing was usually performed with symmetric kernels. However, it is now well established that Daubechies' wavelets are appropriate for this purpose (Akansu & Haddad, 1992). Further, they are computationally efficient in that decomposition and reconstruction can be done using filters constructed from as few as four coefficients. In the next section we will outline a method for decomposing and reconstructing an image using the

computational procedures described by Mallat (1989), and the wavelet basis derived by Daubechies (1988).

Image Decomposition

In order to perform a multiresolution image decomposition using the scaling functions and wavelets described earlier, we use $\phi_D(x)$ and $\psi_D(x)$ (see Figure 5), and define the following *orthonormal wavelet basis* (Daubechies, 1988):

$$\phi_{jn}(x) = 2^{-j/2}\phi_D(2^{-j}x - n) \tag{12}$$

$$\psi_{jn}(x) = 2^{-j/2}\psi_D(2^{-j}x - n) \quad . \tag{13}$$

We can further define (cf., Daubechies, 1988; Mallat, 1989) filters h and g as the inner product of particular scaling functions and wavelets as follows:

$$h(n-2k) = 2^{-1/2} \int \phi(\tfrac{x}{2}) \cdot \phi(x-(n-2k))\, dx\,, \tag{14}$$

and

$$g(n-2k) = 2^{-1/2} \int \psi(\tfrac{x}{2}) \cdot \phi(x-(n-2k)\, dx\,. \tag{15}$$

The functions $h(n-2k)$ and $g(n-2k)$ can be considered the pulse responses of low-pass and high-pass filters, respectively. Thus, using the multiresolution technique developed by Mallat (1989) for symmetrical wavelets, we can expand a one-dimensional signal, S (here denoted S_{L0}), into two components, S_{L1} and S_{H1} as follows:

$$S_{L1} = \sum_k h(n-2k) \cdot S_{L0} \tag{16}$$

and

$$S_{H1} = \sum_k g(n-2k) \cdot S_{L0} \quad . \tag{17}$$

Thus, S_{L1} and S_{H1}, which are the low-pass and high-pass versions, respectively, of the image, S_{L0}, are obtained by convolving S_{L0} with the low-pass and high-pass filters $h(n)$ and $g(n)$. The factor of two in the argument of both Equations 16 and 17 indicates that the filter (convolution) output is subsampled by a factor of two.

The decomposition process of Equations 16 and 17 can be represented as:

$$S \triangleq S_{L0} \rightarrow S_{L1}, S_{H1} \quad ,$$

indicating that S_{L0} is decomposed into S_{L1} and S_{H1}, where S_{L1} represents a lower resolution (smoother) version of S_{L0}, and S_{H1} represents the details of S_{L0} that are missing from S_{L1}. The decomposition process can be continued by further decomposing S_{L1} such that:

$$S_{L1} \rightarrow S_{L2} \ , \ S_{H2} \quad ,$$

and more generally,

$$S_{Lp} \rightarrow S_{Lp+1} \ , \ S_{Hp+1} \tag{18}$$

where, again, S_{Lp+1} is the low-pass filtered version of S_{Lp}, and S_{Hp+1} is the high-pass filtered version of S_{Lp}. Note that if $h(n)$ is designed to be symmetric [i.e., $h(n) = h(-n)$], then Equations 16 and 17 can be written as:

$$S_{Lp+1}(k) = \sum_n h(n) \cdot S_{Lp}(n-2k) \tag{19}$$

and

$$S_{Hp+1}(k) = \sum_n g(n) \cdot S_{Lp}(n-2k) \quad , \tag{20}$$

respectively. The operations of Equations 19 and 20 are equivalent to taking every other sample from the result of convolving S_{Lp} with h and g.

In summary, the decomposition of a signal, S_{Lp-1}, into its components, S_{Lp} and S_{Hp} is achieved by convolving that signal with the filters h and g, respectively, followed by decimating the resultant signals by a factor of two (i.e., taking every other sample). A level-one decomposition of the signal S_{Lp-1}, whose length is N, results in the components, S_{Lp} and S_{Hp} each of length $N/2$. This decomposition process is depicted in Figure 6. Linearly convolving S_{Lp-1}, which is of length N, with either h or g, which are of length L, results in a signal of length $N+L-1$. To deal with this problem we used circular convolution. As Equations 16 and 17 indicate, if the signal S_{Lp} is of length N then its components, S_{Lp+q} and S_{Hp+q} will each be of length $N/2^q$.

Shown in Figure 7 is a schematic for extending the decomposition process to two-dimensional signals such as images. Figure 8 gives a more detailed graphical description of the two-dimensional decomposition process. Signals corresponding to S_{L1} and S_{H1} are obtained by filtering each row of the original image, and are denoted LP and HP in Figure 8. These signals form the intermediate image. The filtering process is then repeated for the columns of the intermediate image thereby producing the level-one decomposition. An example of the level-one decomposition applied to a real image is shown in Figure 9. In this figure (as well as in Figure 10 below), the pixel values in each of the three cells containing high-pass (HP) information have been artificially increased to make them more easily visible. An example of a higher-order (level-three) decomposition is shown in Figure 10.

Image Reconstruction

In the present context, image reconstruction is the reverse operation of image decomposition. The higher resolution signal, S_{Lp-1}, can be reconstructed from its components, S_{Lp} and S_{Hp}, as follows:

$$S_{Lp-1} = \sum_k h(n-k) \cdot S'_{Lp} + \sum_k g(n-k) \cdot S'_{Hp}$$

$$= S'_{Lp} * h(n) + S'_{Hp} * g(n) \quad , \tag{21}$$

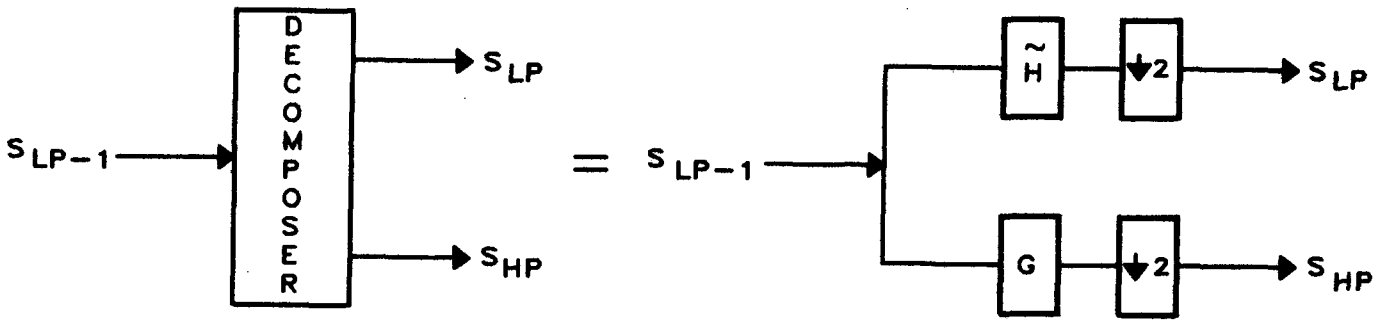


Figure 6
Schematic of the One-Dimensional Decomposition Process

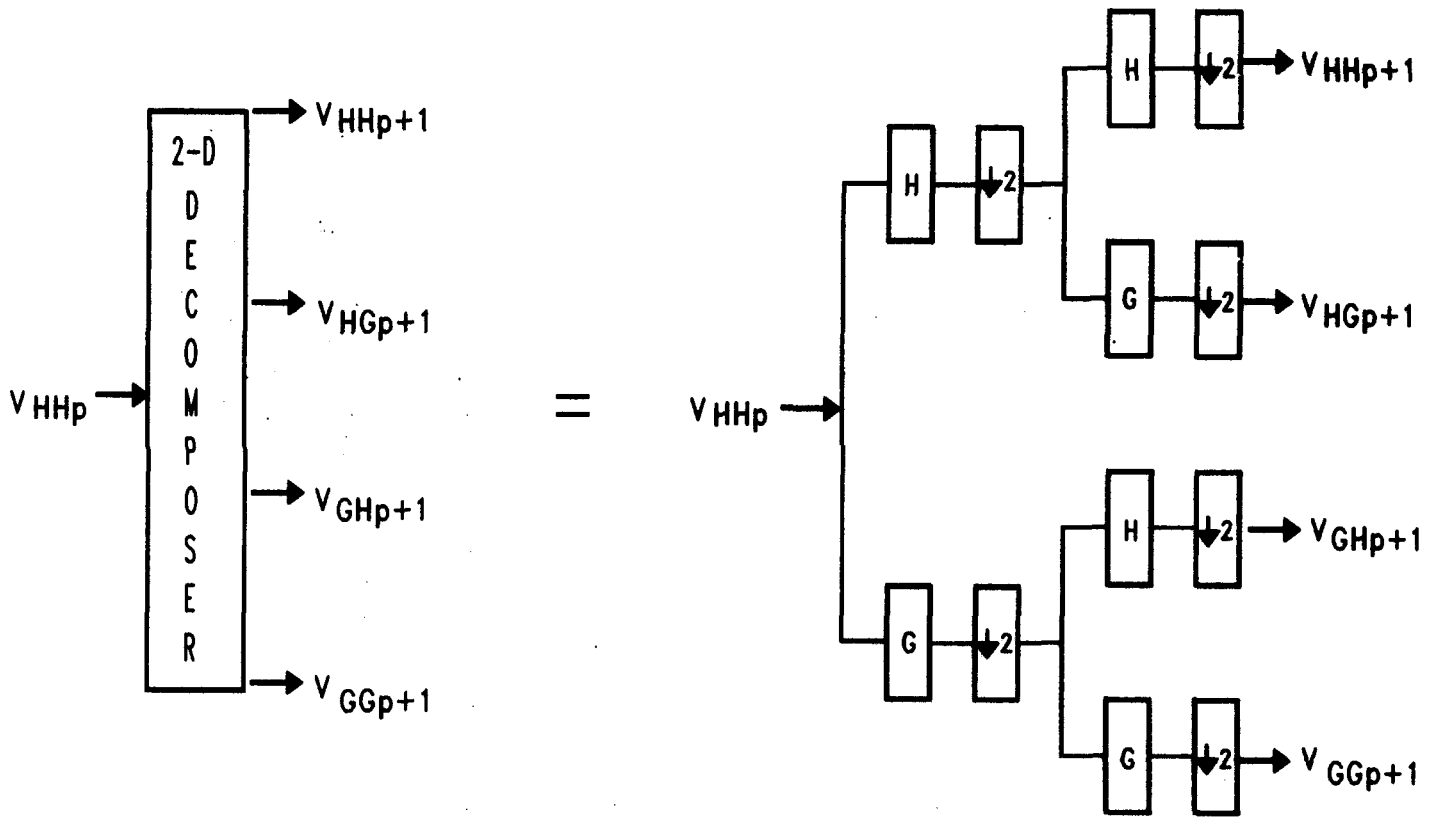


Figure 7
Schematic of the Two-Dimensional Decomposition Process

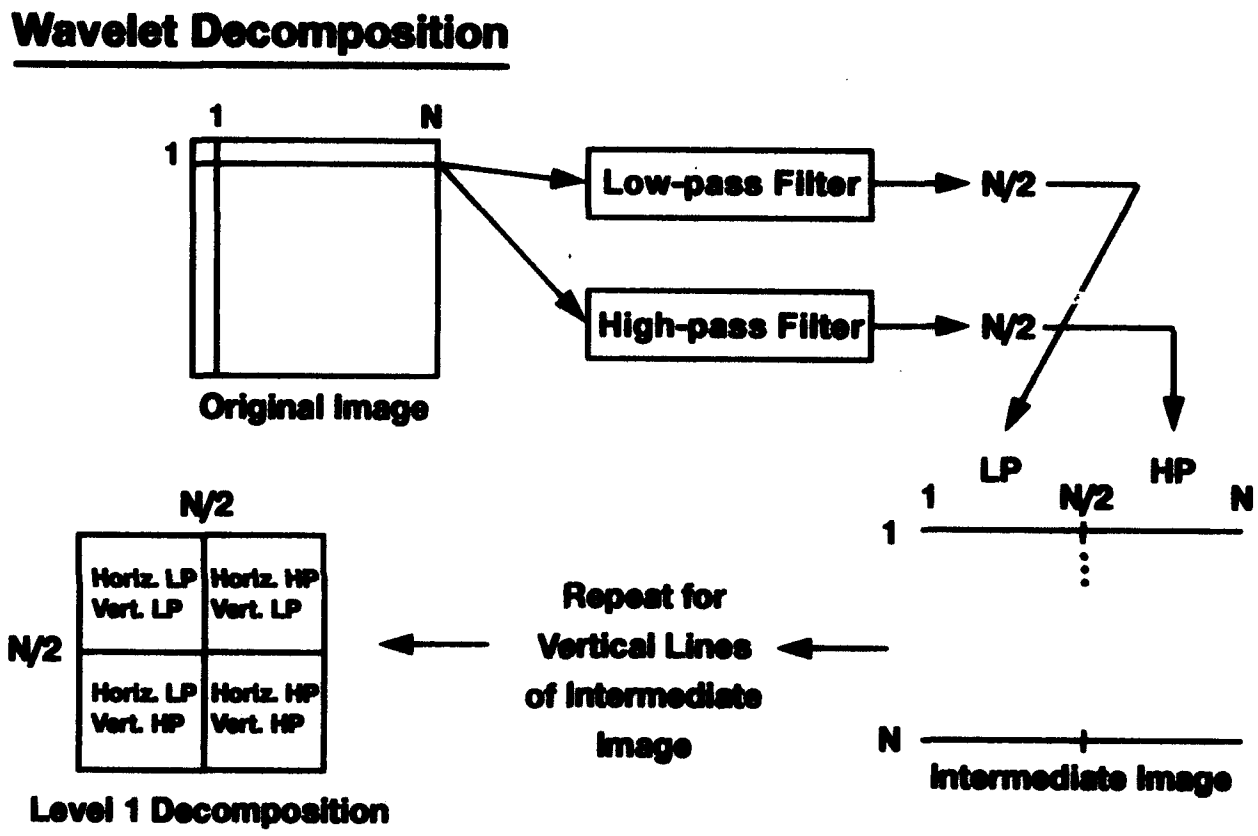


Figure 8
A More Detailed Schematic of the Two-Dimensional Decomposition Process to Level One

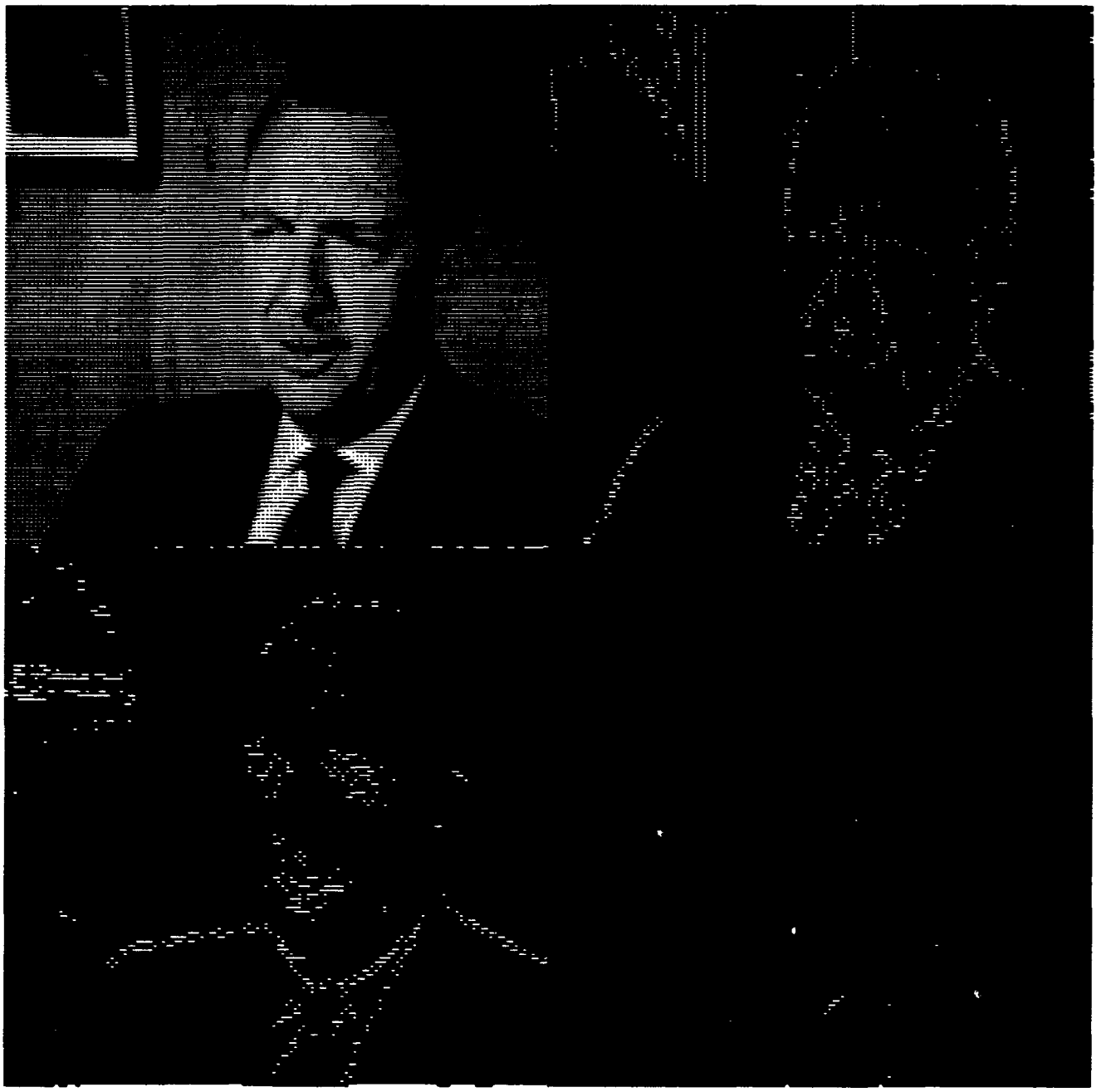


Figure 9
An Example of a Level-One Decomposition

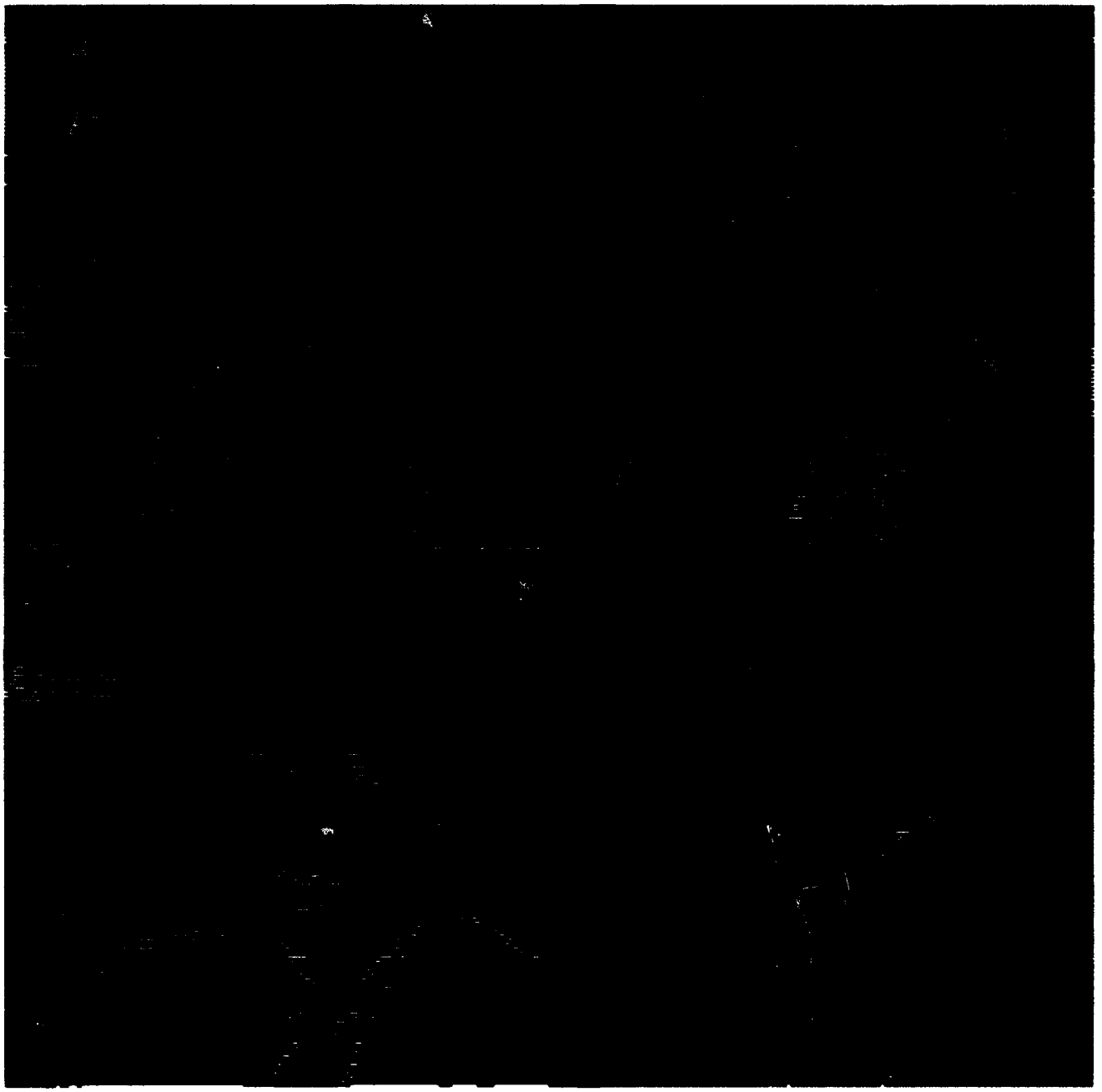


Figure 10
An Example of a Level-Three Decomposition

where (*) is the convolution operator and S'_{Lp} is an interpolated version of S_{Lp} (i.e., with a mean luminance value inserted after every sample of S_{Lp}). Schematics for the one- and two-dimensional reconstruction process are shown in Figures 11 and 12, respectively.

Implementation

It follows from Daubechies (1988) that *g(n)* and *h(n)* are related as:

$$g(n) = (-1)^n h(1-n)$$

where *h(1-n)* is the mirror filter of *h(n)*. Thus, as a practical matter, if *g(n)* is known, *h(n)* can be obtained directly. We will now show, further, that *h(n)* and *g(n)* are equivalent to the c_k's of Equations 3 and 4. The significance of this equivalence is that it allows the required expansion coefficients to be computed from the scaling-function coefficients, c_k. Thus, it is not necessary to generate the full wavelet waveform or to use that waveform in the computation of the inner products from which the expansion coefficients are obtained.

As implied by Equations 16, 17, and 18, we can decompose an image, H, at resolution level 2^{j-1} with respect to the functions ϕ and ψ. Those equations can be rewritten as:

$$S_{Lp} = H(x)_{\text{at level } 2^{j-1}} = \sum_k \langle H , \phi_{jk} \rangle \phi_{jk} + \sum_k \langle H , \psi_{jk} \rangle \psi_{jk} \qquad (22)$$

where ϕ_{jk} and ψ_{jk} are the low- and high-pass filter coefficients (or basis functions), and so the summations represent low- and high-pass filtering. The inner products of Equation 22 give the wavelet (or decomposition) coefficients. For instance, when extended to two-dimensions, the first inner product is simply the upper left corner of Figure 9. We will now show that the inner products in the summations are equivalent to the c_k's of Equation 1.

We begin by using Equations 4 and 13. Specifically, from Equation 4 we have:

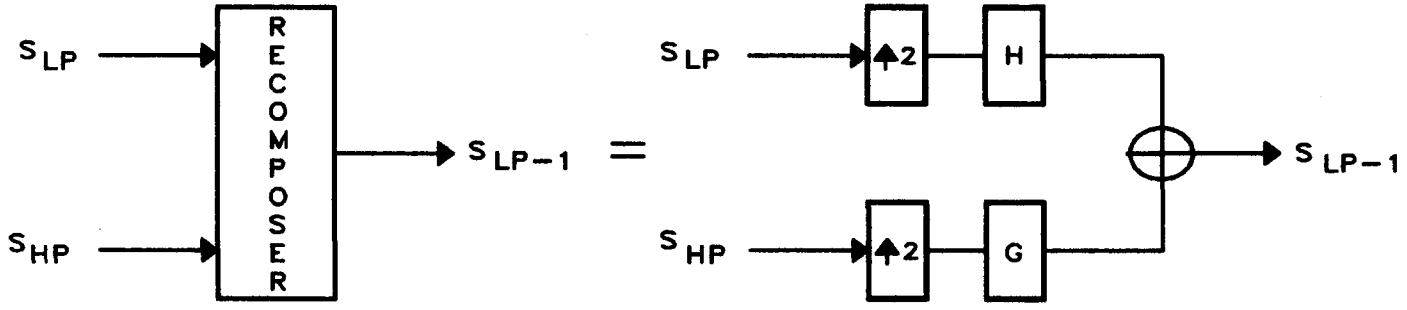


Figure 11
Schematic of the One-Dimensional Reconstruction Process

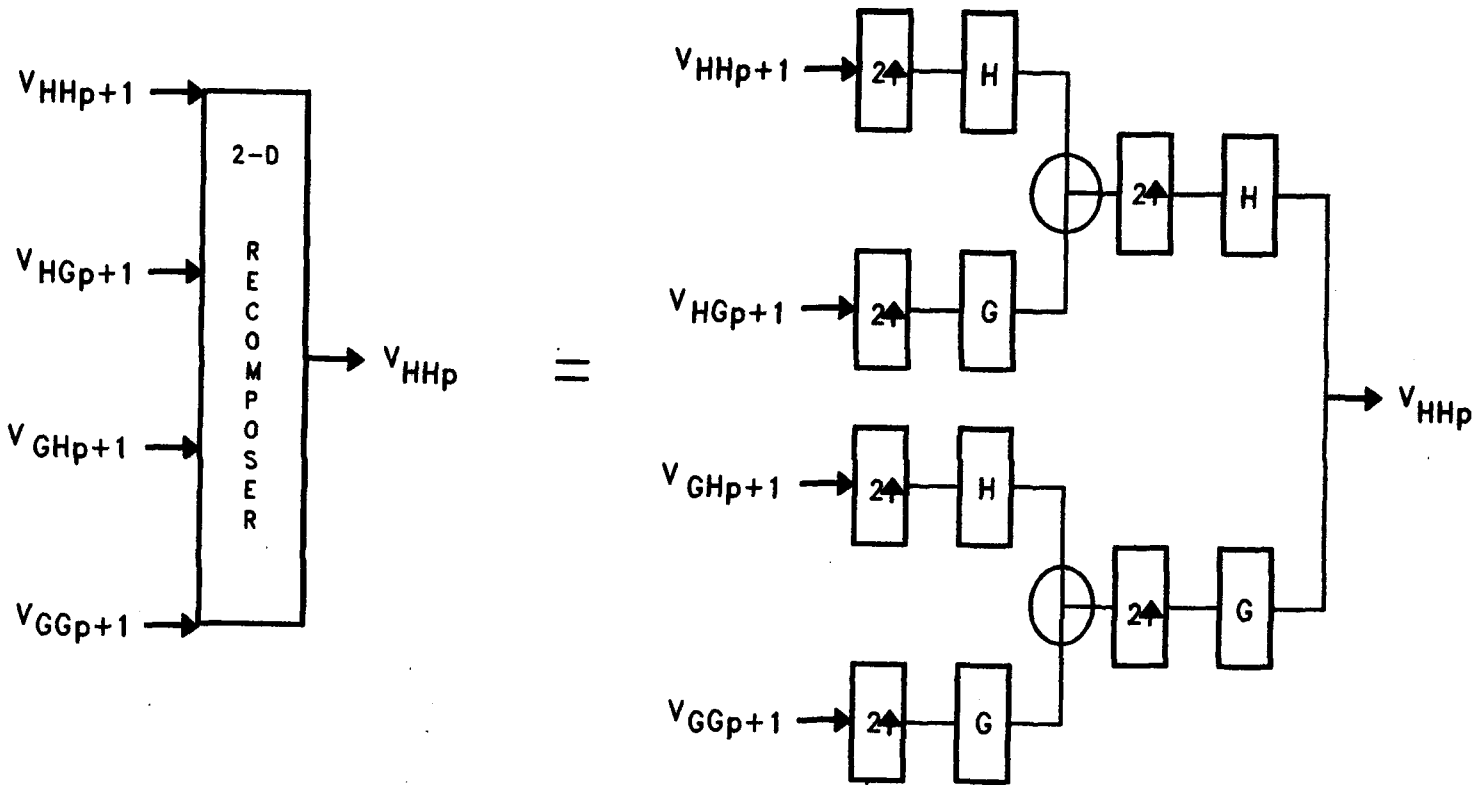


Figure 12
Schematic of the Two-Dimensional Reconstruction Process

$$\psi_{jn}(2^{-\frac{j}{2}}x - n) = \sum_k (-1)^{k+1} c_{N-(k+1)}\, \phi(2 \cdot (2^{-j}x - n) - k)$$

$$= \sum_k (-1)^{k+1} c_{N-(k+1)} \cdot \phi(2^{-(j-1)} \cdot x - (2n + k)) \quad . \tag{23}$$

Substituting Equation 23 into Equation 13 gives:

$$\psi_{jn}(x) = \sum_k (-1)^{k+1} c_{N-(k+1)} \cdot 2^{-\frac{j}{2}} \cdot \phi(2^{-(j-1)} \cdot x - (2n + k))$$

$$= \sum_k (-1)^{k+1} c_{N-(k+1)} \cdot 2^{-\frac{1}{2}} \cdot 2^{\frac{-(j-1)}{2}}\, \phi(2^{-(j-1)} \cdot x - (2n + k)) \quad ,$$

and using the notation of Equation 12 we get:

$$\psi_{jn}(x) = \sum_k (-1)^{k+1} c_{N-(k+1)} \cdot 2^{-\frac{1}{2}}\, \phi_{(j-1),(2n+k)}(x) \quad . \tag{24}$$

Denoting $l = 2n+k$ and $k+1 = (l-2n+1)$ gives:

$$\psi_{jn}(x) = \sum_l (-1)^{l-2n+1} c_{N-(l-2n+1)} \cdot 2^{-\frac{1}{2}}\, \phi_{(j-1),l} \quad ,$$

and denoting,

$$g(l - 2n) = (-1)^{l-2n+1} c_{N-(l-2n+1)} \cdot 2^{-\frac{1}{2}} \tag{25}$$

gives,

$$\psi_{jn}(x) = \sum_l g(l - 2n) \cdot \phi_{(j-1),l} \quad . \tag{26}$$

The high-pass wavelet coefficients at a given resolution level can be obtained by taking the inner product of the above equation with the image, H :

$$\langle H, \psi_{jn} \rangle = \sum_l g(l-2n) \cdot \langle H, \phi_{(j-1)l} \rangle \quad , \tag{27}$$

where g is a high-pass filter. The inner product (< ~ >) on the right side of Equation 27 represents the wavelet coefficients at one resolution level, whereas the inner product on the left side of the equation represents the wavelet coefficients at the next higher level.

Following the same procedure as indicated in Equations 23 through 26, the low-pass wavelets coefficients can be obtained as:

$$\langle H, \phi_{jn} \rangle = \sum_l h(l-2n) \cdot \langle H, \phi_{(j-1)l} \rangle \quad , \tag{28}$$

where, in analogy with Equation 25, $h_l = c_l / \sqrt{2}$.

Equations 14, 15, and 22 show how to compute a coarser approximation from a finer one, as well as how to compute the difference in information between the successive approximations. To obtain a finer representation from a coarser one, we proceed similarly obtaining:

$$\langle H, \phi_{(j-1)m} \rangle = \sum_l h(m-2k) \cdot \langle H, \phi_{jk} \rangle + \sum_l g(m-2k) \cdot \langle H, \psi_{jk} \rangle \quad . \tag{29}$$

SUMMARY

The essence of multiresolution image decomposition and reconstruction is the use of a pair of filters--one low-pass and one high-pass. The filters are constructed from orthonormal wavelets by taking the inner product of the dilated wavelet and a shifted version of the undilated wavelet. Image decomposition is performed by first passing each image row and column through both a low-pass filter and a high-pass filter. Since this filtering operation reduces the image bandwidth, fewer pixels can now be used to represent each line with no loss of information. Therefore, the output from each filter is subsampled by eliminating every other pixel. The

procedure is repeated for each row and column producing four images each one-quarter the size of the original. The procedure is applied recursively to obtain additional levels of decomposition.

Decomposition is, as a practical matter, always performed down to the 2 x 2 pixel level. After decomposition, redundant information can be removed up to any chosen level. Reconstructions can then be performed up to various levels in order to determine the lowest level which gives an acceptable reconstruction. We start the reconstruction from the lowest decomposition level (the 2 x 2 level). The subimage corresponding to that level is first processed through the same high-pass and low-pass filters used in the decomposition. The resulting filtered images are then doubled in size by inserting the mean luminance value between all pixels, and corresponding pixel values are added together to produce a single image. This procedure is then repeated on this image and all subsequent images until the original image size is achieved. The original image can be reconstructed *exactly* by this procedure. The point of the multiresolution analysis, in the present context, is to identify information that can be removed without significantly affecting the appearance of the reconstructed image. The advantage of the orthonormal wavelet technique is that it provides a formal method for reconstructing an original image from the multiresolution decomposition. Prior to the development of this technique, multiresolution reconstruction could not be performed error free and not so simply as in the orthonormal wavelet case wherein reconstruction is simply the reverse process of decomposition.

REFERENCES

Akansu, A.N., & Haddad, R.A. (1992). *Multiresolution signal decomposition*. San Diego: Academic Press.

Burt, P.J., & Adelson, E.H. (1983). The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31, 532-540.

Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41, 909-996.

Daubechies, I. (1992). *Ten lectures on wavelets*. Philadelphia: SIAM, pp. 53-105.

Gabor, D. (1946). Theory of communication. *Journal of the IEE*, 93, 429-459.

Geri, G.A., Zeevi, Y.Y., & Porat, M. (1990). Efficient image generation using localized frequency components matched to human vision. *(AFHRL-TR-90-25*, AD A224 903). Williams Air Force Base AZ: Air Force Human Resources Laboratory, Operations Training Division.

Gertner, I.C., & Geri, G.A. (1994). Image representation using Hermite functions. *Biological Cybernetics* (in press).

Jacobson, L.D., & Wechsler, H. (1988). Joint spatial/spatial-frequency representation. *Signal Processing*, 14, 37-68.

Mallat, S.G. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11, 674-693.

Strang, G. (1989). Wavelets and dilation equations: A brief introduction. *Numerical Analysis Report 89-9*, Cambridge, MA: MIT, Department of Mathematics.

Young, R.A. (1987). The Gaussian derivative model for spatial vision: I. Retinal mechanisms. *Spatial Vision*, 2, 273-293.

APPENDIX

Source code for the program *wavelts.c* which performs image decomposition and reconstruction with respect to a continuous, orthonormal wavelet basis.

```
  1: #define BLACK          0
  2: #define BLUE           1
  3: #define GREEN          2
  4: #define CYAN           3
  5: #define RED            4
  6: #define MAGENTA        5
  7: #define BROWN          6
  8: #define LIGHTGRAY      7    /* Color Definition */
  9: #define DARKGRAY       8
 10: #define LIGHTBLUE      9
 11: #define LIGHTGREEN    10
 12: #define LIGHTCYAN     11
 13: #define LIGHTRED      12
 14: #define LIGHTMAGENTA  13
 15: #define YELLOW        14
 16: #define WHITE         15
 17:
 18: #define OFF          0
 19: #define ON_BLOCK     1
 20: #define ON_NORM      2
 21: #define ENTER       13
 22:
 23: #include <stdio.h>
 24: #include <stdlib.h>
 25: #include <os.h>
 26: #include <malloc.h>
 27: #include <math.h>
 28: #include <errno.h>
 29: #include <grex.h>
 30: #include "nrutil.h"
 31:
 32: void DECOMPOSE();
 33: void RECONSTRUCT();
 34:
 35: void Wrt_Str(int,int,char*,int,int);
 36: void put_window(int,int,int,int,int,int,char*,int,int);
 37: char *int2ch(int);
 38: void error_message(int,int,int,int,char *);
 39: void cursoronof(int);
 40: short *save_scr(int,int,int,int);
 41: void rest_scr(int,int,short*);
 42: char Wrt_Ch(int,int,int);
 43:
 44:
 45: void wavelts(infile,outfile1,outfile2,n,level)
 46: char infile[13];
 47: char outfile1[13];
 48: char outfile2[13];
 49: int n;
 50: int level;
 51: {
 52:     float                   *x, **pic;
 53:     int                     i, j, l, ll, row, col, nn, **pic_out;
 54:     int                     ans, begin_row, begin_col, end_row, end_col, band, mul;
 55:     char                    response;
```

```
 56:     int                     count;
 58:     unsigned char           *ftemp;
 59:     FILE                    *t, *t1, *t2, *t3;
 60:     char                    temp[80];
 61:     int                     num_read, num_written;
 62:     int                     mode, max_row, max_col;
 63:
 64:     char                    *mess_info[15];
 65:     char                    *error_info[15];
 66:     char                    *strtemp;
 67:     int                     resp;
 68:     short                   *savwin;
 69:     float                   bp;
 70:     int                     dcpt, sign;
 71:     char                    *tempstr, *bpstr;
 72:
 73:     error_info[0] = " Error Opening Image File for Reading...";
 74:     error_info[1] = " Error Opening Decomposed Image File for Writing...";
 75:     error_info[2] = " Error Opening Reconstructed Image File for Writing...";
 76:     error_info[3] = " Error Opening Data File for Writing...";
 77:     error_info[4] = " Error in Input Image Size detected...";
 78:     error_info[5] = " Error in Writing Image File detected...";
 79:
 80:     mess_info[0]  = " Reading Image Input File...";
 81:     mess_info[1]  = " Decomposing Image Data into ";
 82:     mess_info[2]  = " levels..";
 83:     mess_info[3]  = " End of Image Decomposition...";
 84:     mess_info[4]  = " Writing Decomposed Image into File...";
 85:     mess_info[5]  = " Number of Logarithmic Frequency-Bands = ";
 86:     mess_info[6]  = " Thresholding ...";
 87:     mess_info[7]  = " Consider BAND # ";
 88:     mess_info[8]  = " for Reconstruction (y/n) ? ";
 89:     mess_info[9]  = " BITS per PIXEL = ";
 90:     mess_info[10] = " Reconstructing decomposed Image...";
 91:     mess_info[11] = " Writing Reconstructed Image into File...";
 92:     mess_info[12] = " End of Image Processing, all Files are Saved...";
 93:     mess_info[13] = " Push any Key to continue...";
 94:
 95:     t = fopen( infile, "rb" );
 96:     if (t == NULL )
 97:     {
 98:         cursoronof(OFF);
 99:         error_message(6,6,40,3,error_info[0]);
100:         return;
101:     }
102:
103:     t1 = fopen( outfile1, "wb" );
104:     if (t1 == NULL )
105:     {
106:         cursoronof(OFF);
107:         error_message(6,6,50,3,error_info[1]);
108:         return;
109:     }
110:
111:     t2 = fopen( outfile2, "wb" );
```

```
112:      if (t2 == NULL )
114:      {
115:          cursoronof(OFF);
116:          error_message(6,6,60,3,error_info[2]);
117:          return;
118:      }
119:
120:      t3 = fopen( "temp.dat", "wt" );
121:      if (t3 == NULL )
122:      {
123:          cursoronof(OFF);
124:          error_message(6,6,40,3,error_info[3]);
125:          return;
126:      }
127:
128:      pic = matrix( 0, n-1, 0, n-1 );          /* Allocate float array */
129:      ftemp = unsigned_vector( 0, n-1 );
130:      x = vector( 0, n-1 );
131:
132:      cursoronof(OFF);
133:
134:      put_window(5,2,60,15,0x1f,3," Executing Wavelet Decomposition ",RED,YELLOW);
135:
136:      Wrt_Str(7,4,mess_info[0],WHITE,BLUE);
137:
138:      for ( i = 0; i < n; i++ )
139:      {
140:          num_read = fread( ftemp, 1, n, t );
141:
142:          if( num_read != n )
143:          {
144:              cursoronof(OFF);
145:              error_message(6,6,45,6,error_info[4]);
146:              return;
147:          } else
148:              {
149:                  for ( j = 0; j < n; j++ )
150:                      pic[i][j] = (float) ftemp[j];
151:
152:              }
153:      }
154:
155:      Wrt_Str(7,5,mess_info[1],WHITE,BLUE);
156:      Wrt_Str(36,5,int2ch(level),WHITE,RED);
157:      Wrt_Str(42,5,mess_info[2],WHITE,BLUE);
158:
159:      nn = n;
160:      for( L = 1; L <= level; L++ )
161:      {
162:          for( row = 0; row < nn; row++ )
163:          {
164:              for( col = 0; col < nn; col++ )
165:              {
166:                  x[col] = pic[row][col];
167:              }
```

```
168:                DECOMPOSE( x, nn );
170:                for( j = 0; j < nn; j++ )
171:                    pic[row][j] = x[j];
172:            }
173:
174:            for( col = 0; col < nn; col++ )
175:            {
176:                for( row = 0; row < nn; row++ )
177:                {
178:                    x[row] = pic[row][col];
179:                }
180:
181:                DECOMPOSE( x, nn );
182:                for( j = 0; j < nn; j++ )
183:                {
184:                    pic[j][col] = x[j];
185:                }
186:            }
187:            nn = nn/2;
188:
189:        }
190:        Wrt_Str(7,6,mess_info[3],WHITE,BLUE);
191:        Wrt_Str(7,7,mess_info[4],WHITE,BLUE);
192:
193:        for ( i = 0; i < n; i++ )
194:        {
195:            for ( j = 0; j < n; j++ )
196:            {
197:                ftemp [j] = (unsigned char)fabs( pic[i][j] );
198:            }
199:            num_written = fwrite( ftemp, 1, n, t2 );
200:            if( num_written != n )
201:            {
202:                cursoronof(OFF);
203:                error_message(6,6,40,6,error_info[6]);
204:                return;
205:            }
206:        }
207:        fclose(t2);
208:        free( ftemp );
209:
210:        pic_out = imatrix( 0, n-1, 0, n-1 );      /* Allocate int array */
211:
212:        Wrt_Str(7,8,mess_info[5],WHITE,BLUE);
213:
214:        band = 3 * level + 1;
215:
216:        Wrt_Str(48,8,int2ch(band),WHITE,RED);
217:        Wrt_Str(7,9,mess_info[6],WHITE,BLUE);
218:
219:        savwin = save_scr(4,10,65,5);
220:
221:        put_window(4,10,65,5,0x1f,3," BAND Input for Reconstruction ",RED,YELLOW);
222:
223:        Wrt_Str(6,12,mess_info[7],WHITE,BLUE);
```

```
224:     Wrt_Str(22,12,"1",WHITE,RED);
226:     itoa(nn, strtemp, 10);
227:     Wrt_Str(26,12,strtemp,WHITE,RED);
228:     Wrt_Str(30,12,"x",WHITE,BLUE);
229:     Wrt_Str(32,12,strtemp,WHITE,RED);
230:     Wrt_Str(35,12,mess_info[8],WHITE,BLUE);
231:     Wrt_Str(63,12,"  ",WHITE,LIGHTGRAY);
232:     resp = Wrt_Ch(64,12,WHITE);
233:
234:     if ((resp == 'y') || (resp == 'Y'))
235:             ans = 1;
236:     else
237:             ans = 0;
238:
239:     resp = 'n';
240:     count = nn*nn*ans;
241:     for( i = 0; i < nn; i++ )
242:     {
243:         for( j = 0; j < nn; j++ )
244:         {
245:             pic_out[i][j] = ans*(int)(pic[i][j]+0.5);
246:         }
247:     }
248:
249:     LL = 1;
250:     for( L = 1; L <= level; L++ )
251:     {
252:         for( band = 0; band < 3; band++ )
253:         {
254:             resp = 'n';
255:             Wrt_Str(6,12,mess_info[7],WHITE,BLUE);
256:             itoa((2+band+3*(L-1)), strtemp, 10);
257:             Wrt_Str(22,12,strtemp,WHITE,RED);
258:             itoa((LL*nn), strtemp, 10);
259:             Wrt_Str(26,12,strtemp,WHITE,RED);
260:             Wrt_Str(30,12,"x",WHITE,BLUE);
261:             Wrt_Str(32,12,strtemp,WHITE,RED);
262:             Wrt_Str(35,12,mess_info[8],WHITE,BLUE);
263:             Wrt_Str(63,12,"  ",WHITE,LIGHTGRAY);
264:             resp = Wrt_Ch(64,12,WHITE);
265:
266:             if ((resp == 'y') || (resp == 'Y'))
267:                     ans = 1;
268:             else
269:                     ans = 0;
270:
271:             if( band == 0 )
272:             {
273:                 begin_row = 0;
274:                 end_row = nn*LL;
275:                 begin_col = nn*LL;
276:                 end_col = 2*nn*LL;
277:             }
278:             if( band == 1 )
279:             {
```

```
280:                    begin_row = nn*LL;
282:                    end_row = 2*nn*LL;
283:                    begin_col = 0;
284:                    end_col = nn*LL;
285:                }
286:
287:                if( band == 2 )
288:                {
289:                    begin_row = nn*LL;
290:                    end_row = 2*nn*LL;
291:                    begin_col = nn*LL;
292:                    end_col = 2*nn*LL;
293:                }
294:
295:                count += nn*LL*nn*LL*ans;
296:                for( i= begin_row; i < end_row; i++ )
297:                {
298:                    for( j = begin_col; j < end_col; j++ )
299:                    {
300:                        pic_out[i][j] = ans*(int)(pic[i][j]+0.5);
301:                    }
302:                }
303:
304:            }
305:            LL *= 2;
306:        }
307:        rest_scr(4,10,savwin);
308:        free_text_window(savwin);
309:
310:        bp = 8*(float)count/(float)(n*n);
311:        tempstr = fcvt(bp,5,&dcpt,&sign);
312:
313:        j=0;
314:        for (i=0;i<strlen(tempstr)-1;i++)
315:        {
316:            if(i == 1)
317:            {
318:                j++;
319:                bpstr[j] = '.';
320:            }
321:            bpstr[j+i] = tempstr[i];
322:        }
323:        bpstr[6] = '\0';
324:
325:        Wrt_Str(7,10,mess_info[9],WHITE,BLUE);
326:        Wrt_Str(25,10,bpstr,WHITE,RED);
327:
328:        free_matrix(pic,0,n,0,n);
329:
330:        nn = 2*nn;
331:
332:        Wrt_Str(7,11,mess_info[10],WHITE,BLUE);
333:
334:        for( L = 1; L <= level; L++ )
335:        {
```

```
336:          for( row = 0; row < nn; row++ )
338:          {
339:              for( col = 0; col < nn; col++ )
340:              {
341:                  x[col] = pic_out[row][col];
342:              }
343:
344:              RECONSTRUCT( x, nn );
345:              for( j = 0; j < nn; j++ )
346:              {
347:                  pic_out[row][j]=(int)(x[j]+0.5);
348:              }
349:          }
350:
351:          for( col = 0; col < nn; col++ )
352:          {
353:              for( row = 0; row < nn; row++ )
354:              {
355:                  x[row] = pic_out[row][col];
356:              }
357:
358:              RECONSTRUCT( x, nn );
359:              for( j = 0; j < nn; j++ )
360:              {
361:                  pic_out[j][col] = (int)(x[j]+0.5);
362:                  if( pic_out[j][col] > 255 )
363:                  {
364:                      pic_out[j][col] = 255;
365:                  }
366:              }
367:          }
368:          nn = nn*2;
369:
370:      }
371:      ftemp = unsigned_vector( 0, n-1 );
372:
373:      Wrt_Str(7,12,mess_info[11],WHITE,BLUE);
374:
375:      for ( i = 0; i < n; i++ )
376:      {
377:          for ( j = 0; j < n; j++ )
378:          {
379:              ftemp [j] = (unsigned char)abs( pic_out[i][j] );
380:          }
381:          num_written = fwrite( ftemp, 1, n, t1 );
382:          if( num_written != n )
383:          {
384:              perror( "Error writing image w_rec.img" );
385:              pxerror( "Extended error" );
386:          }
387:      }
388:
389:      fprintf( t3, "3\n" );
390:      fprintf( t3, "%d\n", n );
391:      fprintf( t3, "%s\n", infile );

FILE=wavelts.c    Tue Jan 25 12:48:07 1994    PAGE=7
```

```
392:      fprintf( t3, "%s\n", outfile1 );
394:      fprintf( t3, "%s\n", outfile2 );
395:
396:      free(ftemp);
397:      free_matrix(pic_out,0,n,0,n);
398:      free_vector(x, 0, n);
399:
400:      Wrt_Str(7,13,mess_info[12],WHITE,BLUE);
401:
402: fclose( t1 );
403: fclose( t3 );
404:
405: Wrt_Str(20,15,mess_info[13],YELLOW,RED);
406: pauseb();
407:
408: }
```