PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
27 Jan 94

3. REPORT TYPE AND DATES COVERED
Final Technical Report: 1 June 90-31 July 93

4. TITLE AND SUBTITLE
Wavelet Transforms in Parallel Image Processing (U)

5. FUNDING NUMBERS
G: AFOSR-90-0310
PR/TA = 9806/00
(2)

6. AUTHOR(S)
Ching-Chung Li and Richard W. Hall

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Pittsburgh
Department of Electrical Engineering
348 BEH
Pittsburgh, PA 15261

8. PERFORMING ORGANIZATION REPORT NUMBER
TR-CV-93-07
AFOSR-TR- 94 0158

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DARPA-AFOSR/NM
Building 410
Bolling AFB DC 20332-6448

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR-90-0310

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE:
DISTRIBUTION UNLIMITED

12b. DISTRIBUTION CODE
UL

13. ABSTRACT *(Maximum 200 words)*

This project consists of two parts: applications of wavelet transforms in multiscale image processing, and parallel algorithms and architectures. We have studied issues in wavelet-based edge detection: antisymmetry of wavelet filters and their support size with respect to edge localization, and a non-othogonal four-coefficient wavelet edge detector. Texture segmentation using a modulated Daubechies' wavelet has been studied, providing both spatial frequency and orientation selectivity. Object segmentation has been studied by examining silhouettes at multiple resolutions in piecewise linear approximations, and has been explored on LADAR data for use in target recognition. Applications to biomedical image compression, image halftoning and artificial neural network structure have also been investigated. In parallel processing, we have studied embeddings of wavelet transform algorithms as well as other algorithms for 2D and 3D mesh architectures and systolic arrays, and their time complexities have been evaluated. A variety of issues are addressed in utilization of 3D meshes in parallel image processing on 2D and 3D images including: the embedding of 2D images into 3D meshes, 3D shrinking incorporating subfields methodology, 3D connected component labeling, and segmentation in magnetic resonance imaging. Some fundamental issues for parallel reduction and reduction-augmentation operators in 2D and 3D image spaces have been examined. The potential for using graph compound networks for enhancing communication capability of mesh architectures has also been explored.

14. SUBJECT TERMS
Wavelet Transform, Parallel Image Processing, Edge Detection, Object Segmentation, Texture Segmentation, Image Compression, Image Halftoning, Neural Network, Parallel Algorithms, 2D and 3D Mesh Architectures, Topology Preservation, 3D Images, Embeddings

15. NUMBER OF PAGES
137

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	SAR

Table of Contents

A. Executive Summary and Introduction

This project covers two areas of research: one is on applications of wavelet transforms to multiscale image processing, and the other is on parallel algorithms and architectures. We will address first the wavelet transform applications.

In computer vision, the features contained in an image need to be characterized, extracted and recognized in order to provide a final interpretation of the image. Various types of features over a range of sizes appear with different contrast in an image which is usually contaminated with noise and distortion during the imaging process. This often requires processing in different scales, from coarse to fine, in order to obtain both global and local information. One often uses a Gaussian filter to smooth an image prior to any extraction process, with different standard derivations of the Gaussian taken as the scale values. This was advocated by Marr and Hildreth, for example, using zero-crossings of the Laplacian of a Gaussian under various values of the standard derivation(scale parameter) for edge detection. This approach allows for a scale space with a continuous spectrum of the scale parameter; but a problem in using the Gaussian filter is that it is difficult to locate edges at large scales. Work has been done following the introduction of this approach to minimize the delocalization and integrate the information obtained at different scales.

Wavelet theory provides a new multiscale approach for representation and characterization of signals and images. One can select a suitable or an optimal wavelet and its associated scaling function; their translations and dilations form two sets of basis functions which can be used to represent any signal with finite energy, providing both time(space) and frequency localization. An image can be decomposed into the so-called low-frequency components and high-frequency components at successive scales, from which the original image can be reconstructed. Novel applications can be and have been developed using this notion, e.g., image compression, noise suppression, image enhancement, edge detection, segmentation, and texture analysis. This is a rapidly emerging field as evidenced by the topical conferences held and the books published during the past five years.

We have studied the localization property of wavelet-based edge detectors. It has been proven that wavelet filters must be antisymmetric with repect to a non-zero axis of symmetry and scaling filters are symmetric with repect to this axis which is 0.5 for the best edge localization. In order to reduce the interaction between neighboring edges at larger scales, the scaling filter should have the shortest possible support and yet still provide adequate smoothing. Toward this goal, a non-orthogonal wavelet edge detector with four filter-coefficients has been studied where the coefficient values were selected from the parameter space of the solutions to the four-coefficient, two-scale dilation equation. Its experimental performance showed an improvement over what were obtained by the refined regularization method and the first order regularization filter method which we developed earlier. It is comparable to the Mallat-Zhong edge detector with a quadratic spline wavelet, it slightly reduces the edge interaction effect at the expense of noise sensitivity.

The uncertainty constants, in reference to the time(space)-frequency localization, of various Daubechies' and B-spline wavelets and their associated filters have been computed and compared. Among the Daubechies' orthogonal wavelets, D_{12} has the smallest uncertainty constant, however, the uncertainty constant of the discrete filter increases almost linearly with its order. For B-spline wavelets as well as their discrete filters, their uncertainty constants are smaller and approach to 0.5 (the Gaussian case). Hence, if orthogonality is not required, a B-spline wavelet may be prefered in signal decomposition for its very good time-frequency localization.

A modulated Daubechies' wavelet (D_6) has been used in image texture segmentation, providing both spatial frequency and orientation selectivity. The modulation is refered to the scaling function modulation. In the 1-dimensional case, the modulated scaling function is equivalent to a wavelet only if the modulation frequency is equal to π where the modulated scaling filter is, in fact, a wavelet filter. Extent to the 2-dimensional case, the modulated filter is a high-(or band-)pass filter centered at a pair of modulation frequencies with a specific orientation in the 2-dimensional frequency plane. Four orientations, $90°, 45°, 0°$ and $-45°$, and two scale levels were used; these eight channels provide eight texture features for segmentation. Our experiment on a cloud image showed a well localized texture border between the cyclone cloud street and its background.

For recognition and identification of 2-dimensional or 3-dimensional objects, the object of interest should be well segmented and efficiently represented such that the task of matching the segmented object with those stored in the database can be greatly facilitated. We have studied the object segmentation problem by examining its silhouette(or surface) at multiple resolutions in piecewise linear(or planar) approximations by the use of wavelet transform. The Harr wavelet is used in this study to generate a multiresolution surface decomposition. At each resolution, silhouette pixels are classified into four edge categoties: vertical, horizotal, diagonal and compound; compound edges are made up of edges from a combination of the first three categories and hence will be assigned more than one particular orientation. In the 3-dimensinal case, surface voxels are classified into nine orientations. The pixels thus labeled are combined for all resolutions. The neighboring silhouette pixels with the identical label(sharing a common orientation) are then grouped together for reducing the number of representation pixels termed "dominant points". These dominant points can be used for object matching and recognition. This method has been experimented on LADAR images of a military vehicle for possible use in the automatic target recognition.

We have considered a biomedical image compression problem by using the variable rate vector quantization for encoding Daubechies' wavelet transform coefficients. For archiving purpose, the information preserving compression is required, the achievable compression ratio is currently limited to about 3-to-1. For communication purpose, however, some information loss may be acceptable if the decompressed image is visually indistinguishable from its original by expert radiologists. Although the original image can be completely reconstructed from its wavelet transform, any quantization of wavelet transform coeffients constitutes a lossy scheme. It is important to examine how much compression can be obtained while maintaining high fidelity of the decompressed image for diagnostic applications. We experimented on a small set of angiograms and NMR head images by using Daubechies' orthogonal wavelet transform D_4, 4×4 bolcks in vector quantization and a tree structured vector quantizer with varying depth. Our preliminary result showed a 8.8-to-1 compression ratio with high fidelity decompression. This can be improved by using a more regular wavelet and a larger set of training images.

Another application has been directed to the adaptive image halftoning. The gray information at a pixel, including its gray value and gradient, is represented by the number, density and distribution patterns of the set-on dots in a screen. Embedded within an equilateral quadrangular support, an elliptic screen function is designed such that its orientation, elongation and size are adaptive to the wavelet transform modulus, angle and the pixel gray level respectively. Our simulation experiments demonstrated its potential for future development.

Our work on systolic architecture for pipelined processing of discrete wavelet transform has led us to study the wavelet-based artificial neural network structures for dynamic system identification. We have developed a method of utilizing Daubechies' orthogonal wavelet transform to construct a three-layered artificial neural network for identification of stable linear dynamic systems, time invariant or time varying. Special neurons in the middle layer have nonlinearities characterized by Daubechies' orthogonal wavelets at successive scales and the scaling function at the coarsest level. In our simulation study, the training time and accuracy were improved by one order of magnitude when compared to that obtained by the conventional artificial neural networks. By cascading this network to a special three-layered network for nonlinear function approxiamtion where scaled cubic spline functions are used as radial basis functions in the hidden neurons, a special four-layered feedforward network is constructed which is potentially useful for on-line nonlinear system identification.

As images become larger and processing demands greater the role of parallel computing becomes more important in image processing. Many important issues arise in this context and we will summarize a few here. Given desirable or available parallel computing architectures we need to determine how we should place images and image representations into the architecture and in concert with this how we should implement in parallel the required image processing algorithms. The development of parallel algorithms can be complicated by the simultaneity inherent in multiple applications of operators over the whole image which can lead to greater difficulty in algorithm verification. Thus, we need to develop verification methodologies to aid algorithm designers in their search for acceptable parallel algorithms. Based on communication constraints typical in most practical parallel architectures we usually try to identify algorithms based as much as possible on operators with small local supports. Thus, research which helps to identify such operators and limits on such operators can be of substantial aid to algorithm designers. The work on parallel image processing reported here addresses questions bearing on these areas of research interest. The target architectures of primary interest in our studies have been 2D and 3D mesh architectures either with or without reconfigurability. These architectures are in some sense the most practical for massive parallelism.

In order to utilize wavelet transform representations in parallel architectures, we need to decide how we should place the various coefficients in the architectures. It is likely that wavelet notions will find their place typically in low to medium level image processing applications where mesh architectures are particularly well suited. So we have focussed primarily on the question of how to embed wavelet transforms of 2D images into 2D meshes. We identified two key types of embeddings: one which tends to keep particular detail images of the transform in contiguous "blocks" of the mesh and the other which tends to distribute the different components of the detail images (and the remaining lower resolution version of the image) such that all information about given image region is as close together as possible. These embeddings are envisioned for fine-grain realizations in massively parallel architectures where we have one wavelet coefficient per processing element. We compare the time complexity achievable for the two embeddings for a set of algorithms and algorithm classes including: wavelet decomposition, wavelet reconstruction, a canonical class of operators with local support, and a canonical planning task. For each case we consider a target 2D mesh target architecture with and without reconfigurability. In almost all evaluations the distributed form of embedding produced superior time performance and for the reconfigurable mesh architectures the distributed embedding produced essentially ideal results. Further, we derived some evidence for an increased advantage for the distributed embedding in the more coarse-grain realizations where more than one wavelet coefficient is placed at each processing element. We have also considered the case where a 3D transform is embedded into a 3D mesh using extensions of the two proposed embeddings and find that the distributed embedding appears superiour for this case also. Thus, we believe we have identified an excellent (possibly ideal) candidate for the embedding of 2D (3D) wavelet transforms into 2D (3D) meshes.

We have substantial interest in the 3D mesh architecture for both 2D and 3D image processing as it appears to offer the most practical interconnection scheme for packaging massively parallel architectures in 3-space. We have studied the embedding of 2D images into 3D meshes and demonstrate a 2D-3D embedding (into a 3D mesh with certain reconfigurability capabilities) which has dilation of one. This implies that any 2D algorithm can be run on the 3D mesh without a time penalty. Of course, if we are processing 3D images the 3D mesh can offer obvious advantages when we want to realize 3D parallel operators. We demonstrate the superiority of the 3D mesh for the transpose operation which requires a large amount of inter-processor communication and is fundamental for solutions to the sorting problem. Finally, we report a study on expected speedup for 3D meshes as compared to 2D mesh and binary hypercube which indicates effective advantages for 3D mesh over 2D mesh for all cases considered and illustrates for what size parallel machines the hypercube surpasses 3D mesh in expected performance. We conclude that 3D mesh offers some substantial promise in parallel image processing of 2D and 3D images.

We have investigated particular 3D image processing algorithms in an effort to identify potential applications for 3D mesh. We have investigated the apparently low level problem of 3D shrinking to a residue. This problem in its 2D instance is fundamental in component counting and connected component labeling. We developed the first correct (i.e., connectivity preserving) approach to this problem using subfield notions. Unfortunately, in this study we had trouble achieving convergence for all images which led us to a conjecture that one cannot in general find a parallel 3D shrinking to a residue algorithm based on local operators. Although we have not found a formal proof for our conjecture, the kinds of images (i.e., "chains") which form the basis of our conjecture offer daunting challenges for anyone hoping to design a practical algorithm. These observations have impacted on our concurrent study of parallel 3D connected component labeling algorithms. Here we are studying algorithms built from extensions of successful 2D approaches: including radix-based approaches; shrink-expand approaches (based on shrinking); and mixtures of divide and conquer and propagation labeling. Our extensions of radix-based methods to various 3D image spaces have far superior time complexity in reconfigurable 3D meshes when the bus time delay can be modeled as much less than O(bus length). Several of our other methods become competitive (or superior) when bus delay is treated as O(bus length). Finally, we have considered the magnetic resonance imaging problem as a typical area where 3D images are produced. We have begun to address the problem of brain region segmentation and are developing methodologies for comparing the efficacy of different segmentation results.

In image processing we often are concerned with identifying the connected components of a partially processed image as a fundamental step in region segmentation. Thus, there are needs for classes of connectivity preserving (topology preserving) algorithms. Parallel implementations of such algorithms offer sometimes difficult verification challenges and there is a related challenge to identify the smallest possible supports for parallel operators as such small supports can reduce communication overhead in parallel implementations. We have identified optimally small supports for certain 2D connectivity preserving reduction operators, i.e., thinning and shrinking to a residue. We have developed some approaches and tests for easing verification of parallel 2D reduction operators; parallel 2D reduction-augmentation operators; and 3D subfield-based parallel reduction operators. This work offers algorithm designers relatively efficient tests for proving connectivity preservation for these fundamental classes of parallel operators.

Finally, we have considered some alternative architectures. We demonstrate a systolic realization of a 1D wavelet transform in a systolic array which appears promising. Earlier in the project we have investigated the application of compound graph interconnection networks as augmentations of 2D mesh architectures. Certain of our networks offer time performance close to hypercube but at a substantially reduced interconnection cost. One of our networks is fundamentally more powerful than hypercube (while maintaining lower interconnection cost) as it performs sorting and PRAM emulation in O(log(problem size)). Although these theoretical results are quite promising, we feel that massively parallel architectures with fine grain embeddings of images are most likely to be realized in practice with mesh forms of interconnection. Thus, this work on highly interconnected architectures is most likely relevant to networks with smaller numbers of more powerful processors and medium to coarse grain realizations.

B. Research Results

B.1 Applications of Wavelet Transform to Image Processing

We have studied the following problems on applications of wavelet transforms to multiscale image processing: wavelet-based edge detection and localization, texture segmentation using a modulated Daubechies' wavelet, and object segmentation with multiple resolutions in piecewise linear approximations; they are discussed in Section B.1.a, Section B.1.c and Section B.1.d respectively. Uncertainty constants of Daubechies' wavelets and B-spline wavelets and of their filters are discussed in Section B.1.b. Biomedical image compression by using the variable rate vector quantization of Daubechies' wavelet transform coefficients is discussed in Section B.1.e, and the use of spline wavelet transform in the adaptive image halftoning is studied in Section B.1.f. Furthermore, wavelet-based artificial neural networks developed for system identification is described in Section B.1.g.

B.1.a Wavelet-Based Edge Detection and Localization

Detection of edges in noisy images requires an appropriate smoothing over various regions in an image but preserving and extracting the edge information with high accuracy. The criteria consist of reliable detection and accurate localization. The multiscale analysis provides a natural approach to deal with the problems of various feature size and feature contrast. There are generally two ways to study the multiscale edge detection. One is to reconstruct an adequate intensity surface f from the noisy data d, using different scales over different spatial regions, and then to extract the locations of its sharp changes. The other is to extract edges at multiple scales and then integrate them in the finest resolution. We have investigated a space varying regularization method for edge detection by minimizing a membrane energy functional

$$E(f) = \int\int_{\Omega}(f-d)^2 dxdy + \int\int_{\Omega}\lambda(x,y)(f_x^2+f_y^2)dxdy$$

where f_x and f_y denote the gradients of f along x and y directions, and $\lambda(x,y)$ is a regularization parameter controlled spatially by a function of the smoothed error signal between the regularized f and the data d. The control scheme is based on the idea that $\lambda(x,y)$ should be large in continuous regions of f in order to filter out the noise, and should be small at discontinuities in order to prevent significant smoothing across discontinuities. An iteratively refined regularization algorithm has been developed to solve for the variational problem and to determine the edge locations [GOLI93]. We have also investigated an approach of multiscale edge detection by using a set of first order regularization filters with different parameters λ_i and a simple integration scheme of taking union of edge pixels that exist at least for two successive scales [GOLI92]. The 1-D regularization filter $h_r(x;\lambda_i) = \frac{1}{2\lambda}e^{-|x|/\lambda_i}$ is derived from minimizing the 1-D string energy functional where the regularization parameter $\lambda = \lambda_i^2$ has a fixed value. 2-D filtering is obtained by performing the tensor product of two 1-D filtering. Edges at each scale are determined by thresholding local maxima of the gradient of the filtered image. This method performs well if filter parameters λ_i's are appropriately chosen, but the problem is how to make such a choice.

Wavelet theory provides a multiscale approach for representation and characterization of signals and images [MAZH92, MAHW92]. Mallat and Zhong developed a multiscale edge detection process by determining wavelet modulus local maxima. They initially used a cubic spline wavelet in their discrete dyadic wavelet transform [MALL89a]. More recently, they constructed a compactly supported antisymmetric quadratic spline wavelet, which is the derivative of a cubic spline smoothing function, and a symmetric quadratic scaling function, leading to a four-coefficient scaling filter and two-coefficient wavelet filter in their multiscale processing [MAZH92]. In this context, their multiscale edge detector and Canny's edge detector [CANN86] in dyadic scales are functionally equivalent. From these multiscale edges, the original image can be very closely reconstructed. By selecting important edges across the scale, they developed a compact edge-based image compression method. From a pure edge detection point of view, one would like to ensure not only a good performance of detection but also accurate localization of edges which is very important to some high precision industrial inspection problems as well as some medical image analysis problems. It is known that the detected edge locations at large scales may be shifted away from the true edge location. We have studied the localization property of wavelet-based edge detectors both analytically and experimentally [SULI93a, KIKL92, KILI93]. Antisymmetry of wavelet filters and their support size have been investigated with respect to the edge localization. Neighboring edges in close proximity interact one another in the detector may cause displacements of detected edges at larger scales. We have studied another non-orthogonal wavelet edge detector with four filter coefficients, as derived from solutions to the four-coefficient dilation equation, for reducing the degree of edge interactions.

Axial Symmetry for Wavelet Filters:

Let $\psi(x)$ be a wavelet and $\phi(x)$ be its corresponding scaling function, and let $g(n)$ and $h(n)$ be their associated high-pass filter and low-pass filter respectively. Furthermore, let $\Psi(\omega), \Phi(\omega), G(\omega)$ and $H(\omega)$ denote the Fourier transforms of $\psi(x), \phi(x), g(n)$ and $h(n)$ respectively. Then $\Phi(\omega) = H(\omega/2)\Phi(\omega/2)$, $\Psi(\omega) = G(\omega/2)\Phi(\omega/2)$, $|H(\omega)|^2 + |G(\omega)|^2 = 1$, $H(0) = 1$ and $H(\pi) = 0$. The 1-D discrete wavelet transform can be computed recursively by

$$W^{j+1}(n) = S^j(n) * g_j(n)$$
$$S^{j+1}(n) = S^j(n) * h_j(n)$$

where "*" denotes convolution, $W^j(n)$ and $S^j(n)$ represent the coefficients of the dyadic wavelet transform at scale 2^j, $(j = 0, 1, 2, \cdots, J)$, and $g_j(n)$ and $h_j(n)$ are generated by inserting $2^j - 1$ zeros between every pair of successive samples of $g(n)$ and $h(n)$ respectively. $S^0(n)$ is set to be the original signal (image). The inverse wavelet transform is given by

$$S^{j-1}(n) = W^j(n) * g_{j-1}(-n) + S^j(n) * h_{j-1}(-n).$$

Let us discuss the types of symmetry that may be assigned to filters $g(n)$ and $h(n)$. As $h(n)$ is a low-pass filter, it can not be antisymmetric with respect to any axis since otherwise the sum over index n would be zero contracting the requirment $H(0) = 1$. We consider only even-symmetry of $h(n)$ and antisymmetry of $g(n)$, the latter is the case for edge detectors where local mixima of wavelet transform modulus are used.

Let $g(n)$ be antisymmetric with respect to α, i.e., $g(n - \alpha) = -g(\alpha - n)$. The Fourier transform of $g(n - \alpha)$ must be a pure image odd function, i.e.,

$$e^{-i\alpha\omega} G(\omega) = iX(\omega)$$

with

$$X(\omega) = -X(-\omega).$$

$G(\omega)$ must be a 2π-periodic function satisfying the condition $|H(\omega)|^2 + |G(\omega)|^2 = 1$. For real-valued $g(n)$ and $h(n)$, $G(\omega)$ may be written in the following form,

$$G(\omega) = ie^{i\alpha\omega} X(\omega) = -ie^{i\alpha\omega} Sgn(\omega) E(\omega), \qquad for - \pi < \omega \leq \pi$$

where

$$E(\omega) = \sqrt{1 - |H(\omega)|^2}$$

$$\text{Sgn}(\omega) = \begin{cases} 1, & for \quad 0 < \omega \leq \pi \\ -1, & for \quad -\pi < \omega \leq 0 \end{cases}$$

For a good edge detector, $|g(n)|$ and $|G(\omega)|$ must be highly localized. Since $|G(\omega)|^2 = |E(\omega)|^2$, the frequency domain localization of $g(n)$ is not related to α. We need only to be concerned with the spatial domain localization of $g(n)$. In the general case, $g(n)$ may have an infinite support but the value of $|g(n - \alpha)|$ must be sufficiently small when n is large. For a given small number $\epsilon > 0$, there must exist a smallest integer $K > 0$ such that, for $|n - \alpha| \geq K$, $|g(n - \alpha)| < \epsilon$. Then the effective support of $g(n)$ may be taken as $2K$. This concept leads us to consider the asymptotic convergence of $|g(n - \alpha)|$ with respect to α. It is clear that the shape of $g(n)$ does not change if $g(n)$ is shifted by an integer value. Without loss of generality, we may consider only $0 < \alpha < 1$. Let it be assumed that $G(\omega)$ be differentiable in $[0, \pi]$ where the differentiablity at each end point is specified in one direction only. Taking the inverse Fourier transform of $G(\omega)$, we obtain

$$\begin{aligned} g(n) &= \frac{1}{\pi}\int_0^\pi E(\omega)sin(n+\alpha)\omega d\omega \\ &= \frac{1}{\pi(n+\alpha)}\left[2\sum_{k=0}^{n-1}(-1)^k E(\xi_k) + (-1)^n E(\xi_n)(1 - cos\alpha\pi)\right] \end{aligned}$$

where $E(\xi_k)$ denotes the mean value of $E(\omega)$ in the interval $\omega_k \leq \xi_k \leq \omega_{k+1}$, $(k = 0, 1, \cdots, n-1)$, and $E(\xi_n)$ in the interval $\omega_n \leq \xi_n \leq \pi$. It can be shown [SULI93a] that as $n \rightarrow \infty$, regardless n be even or odd,

$$\lim_{n\rightarrow\infty} g(n) = \frac{cos\alpha\pi}{\pi(n+\alpha)}$$

It is clear that the convergence of $|g(n-\alpha)|$ is faster than $\frac{1}{n}$ when α = 0.5 or α = -0.5.

As to the even-symmetry of $h(n)$ with respect to γ, i.e., $h(n-\gamma) = h(\gamma - n)$, $h(n)$ must be a linear phase filter with $H(\omega) = \bar{H}(\omega)e^{i\gamma\omega}$ for $-\pi < \omega \leq \pi$, where $\bar{H}(\omega) = \bar{H}(-\omega)$ is a real-valued function. It has been shown that the order of convergence of $|h(n-\gamma)|$ is at least 3 regardless the choice of γ, if the derivative of $\bar{H}(\omega)$ is 0 at $\omega = 0$ and $\omega = \pi$. Hence, the axis of symmetry of $h(n)$ may be placed arbitrarily.

Based upon the above discussion, it is shown that the filter $g(n)$ must be antisymmetric with respect to either $\alpha = 0.5$ or $\alpha = -0.5$ in order to achieve the optimal edge localization. If the filter $h(n)$ is even-symmetric with respect to γ = 0.5 or γ = -0.5 corresponding, a displacement α_j between $W^j(n)$ and $S^{j-1}(n)$, or a displacement γ_j between $S^j(n)$ and $S^{j-1}(n)$, is resulted in the recursive computation of the discrete dyadic wavelet transform,

$$\alpha_j = 2^{j-1}\alpha = 0.5 \cdot 2^{j-1}, \quad \gamma_j = 2^{j-1}\gamma = 0.5 \cdot 2^{j-1}.$$

This amount of shift should be compensated for in the localization of local maxima of wavelet transform modulus. The detail discussion of the above-discussed symmetry properties can be found in a submitted paper [SULI93b].

A Non-orthogonal Wavelet Edge Detector with Four Filter Coefficients:

The shift compensation discussed above is based on the model of an isolated step edge. In real situations, interaction of neighboring edges also introduce displacements of detected edges at large scales. Edge interactions are mainly caused by the low-passed filter $h(n)$. It is desirable to limit this interaction by choosing a shortest possible support for $h(n)$. In this case, a four-coefficient filter gives the shortest support. We have considered a wavelet-based edge detector with four filter-coefficients whose values are selected from solutions to the four-coefficient, two-scale dilation equation with an attempt to examine and reduce the degree of its edge interaction [KILI93].

A ***two-scale dilation equation*** [STRA89], in particular, a ***lattice*** two-scale dilation equation, is a functional equation of the form

$$\phi(x) = \sum_{n=0}^{N} c_n \phi(kx - n)$$

where $\phi : \mathbf{R} \rightarrow \mathbf{R}$, c_n's a.e real-valued parameters, n and k are integers, and $k \geq 2$. When $k = 2$, the solution of such a dilation equation yields a square integrable scaling ιunction which can be used to construct a dyadic wavelet. A basic assumption on the coefficients is

$$\sum c_{2n} = \sum c_{2n+1} = 1$$

The existence, uniqueness, integrability and smoothness of such solutions under various parameter values have been investigated [COLE92, DAUB91a, DAUB92a, LAWT91, LAWA92]. The lattice dilation equation may be solved for a compactly supported solution by applying an iterative method. The coefficients c_n's are related to the low-pass filter coefficients $\{h(n)\}$ where $h(n) = c_n/2$. A wavelet $\psi(x)$ can be constructed through the equation

$$\psi(x) = \sum 2g(n)\phi(2x - n)$$

where

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

and $\{g(n)\}$ are the filter coefficients of a high-pass filter.

Our goal is to select a wavelet and a scaling function such that the associated filter $h(n)$ provides an adequate smoothing but introduces minimal edge interaction. Hence, the four-coefficient lattice dilation equation

$$\phi(x) = \sum_{n=0}^{3} c_n \phi(2x - n)$$

with

$$c_0 + c_2 = 1, \quad c_1 + c_3 = 1$$

is of particular interest. Colella and Heil discussed the characterization of continuous four-coefficient scaling functions and wavelets, using (c_0, c_3) as two independent parameters [COLE92]. Integrable solutions of $\phi(x)$ exist for (c_0, c_3) to be on or inside the circle centered at $(0.5, 0.5)$ in the (c_0, c_3)-plane with a radius of $1/\sqrt{2}$:

$$(c_0 - 0.5)^2 + (c_3 - 0.5)^2 = \frac{1}{\sqrt{2}}$$

Along the diagonal line $c_3 = c_0$, the resulting scaling functions are symmetric. All points on the circle, except the point $(1, 1)$, yield orthogonal scaling functions and thus give rise to orthogonal wavelets, so the circle is called the circle of orthogonality. Furthermore, $c_0 \leq 1$ and $c_3 \leq 1$ inside the circular region result in continuous scaling functions. The intersections of the circle and the line passing through $(0.5, 0)$ and $(0, 0.5)$ gives the Daubechies scaling function D_4. The line segment between $(0.5, 0)$ and $(0, 0.5)$ indicates differentiable solutions for $\phi(x)$. The point $(1, 0)$ gives the Haar scaling function and wavelet, and the points $(0, 0)$ and $(0, 1)$ produce translates of the Haar wavelet.

We have chosen $c_0 = c_3 = 0.1$, thus $c_1 = c_2 = 0.9$, for obtaining a symmetric scaling function supported in $[0, 3]$. The low-pass filter coefficients are given by $\{h(n)\} = \{0.05, 0.45, 0.45, 0.05\}$, which provide a certain degree of four-point smoothing but dominated mainly by the middle two filter coefficients, thus resulting in a minimal degree of edge interaction at larger scales. Let the above-selected non-orthogonal scaling function be translated to the left by one unit so that it is symmetric with respect to $x = 0.5$; the support region of $\phi(x)$ becomes $[-1, 2]$. The constructed wavelet $\psi(x)$ is antisymmetric about $x = 0.5$ and is also compactly supported in $[-1, 2]$. The scaling function and wavelet are shown in Figure 1. Filters $\{h(n)\}$ and $\{g(n)\}$ are all supported in $[-1, 2]$ and are symmetric and antisymmetric respectively about $x = 0.5$. They are listed in Table 1, and the magnitude of their frequency responses $H(\omega)$ and $G(\omega)$ are shown in Figure 2.

Table 1: Filter coefficients for the chosen wavelet transform.

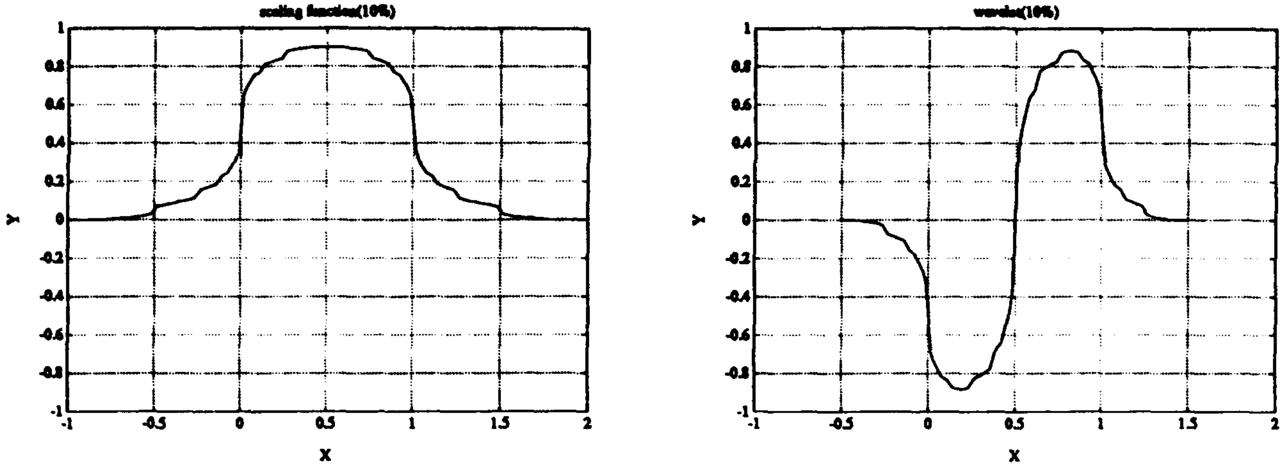
	h(n)	g(n)	l(n)
-3			0.00125
-2			0.0225
-1	0.05		0.12375
0	0.45	-0.5	0.705
1	0.45	0.5	0.12375
2	0.05		0.0225
3			0.00125

For edge detection, no subsampling is used in the wavelet transform analysis. The discrete wavelet transform of a function $f(x)$ at scale 2^j, denoted by $W^j(k)$, is defined as the inner product of the function with ψ_{jk},

$$W^j(k) = < f, \psi_{jk} >;$$

in this case, $\psi_{jk} = 2^{-j}\psi(2^{-j}(x - k))$. It represents the gradient information at scale 2^j. The smoothed signal at scale 2^j is given by

$$S^j(k) = < f, \phi_{jk} >$$



(a) $\phi(x)$ (b) $\psi(x)$

Figure 1: The chosen scaling function and wavelet supported in [-1,2].

where $\phi_{jk} = 2^{-j}\phi(2^{-j}(x-k))$. For a discrete signal of N points, the wavelet transform at the original sampling instants can be recursively computed by

$$W^j(k) = \sum_{n=-1}^{2} g(n) S^{j-1}(k + 2^{j-1}n)$$

$$S^j(k) = \sum_{n=-1}^{2} h(n) S^{j-1}(k + 2^{j-1}n)$$

$$(k = 0, 1, \ldots, N-1; \quad j = 1, 2, \ldots, m; \quad 2^m = N)$$

The original discrete signal is taken as $S^0(k)$.

Wavelet modulus maxima are determined to detect multiscale edges. For a single step edge, the shift in the computed edge location due to the non-zero symmetry axes of both filters can be compensated for at each scale by shifting back $0.5 \cdot 2^{j-1}$ units in position for outputs of the respective filters. Furthermore, the original step edge will become a ramp edge at larger scales. In order to locate the multiscale edge at the middle of the ramp, instead of at a change point, we modified slightly the wavelet maxima detection scheme as follows. (1) Compute the successive differences in wavelet values, $|W^j(k)| - |W^j(k-1)|$; (2) keep track the positive change exceeding a threshold; (3) assign a starting index for the position where the change (either positive or negative) is within a small threshold; (4) assign an ending index for the position where the difference takes a large negative value; and (5) locate the wavelet maximum at the mid point between the starting index and the ending index.

A two-dimensional wavelet transform is obtained by using the tensor product of two one-dimensional wavelet transforms, one in the horizontal direction and the other in the vertical

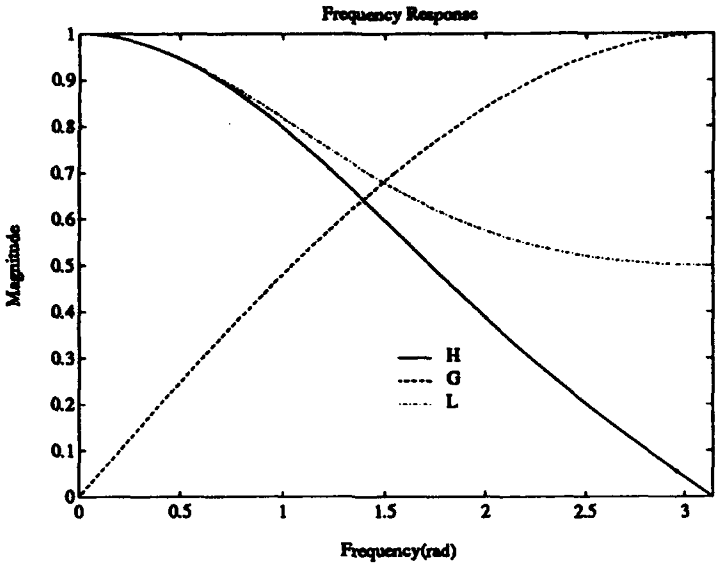


Figure 2: Frequency responses of H(ω)(solid line), G(ω)(dashed line) and L(ω)(dotted line).

direction. Let an all-pass filter $L(\omega)$ be defined as

$$L(\omega) = \frac{1 + |H(\omega)|^2}{2}$$

The filter coefficients $\{l(n)\}$ and the frequency response $L(\omega)$ are also given in Table 1 and Figure 2 respectively. Let the rows of the smoothed image $S^{j-1}(k,l)$ at scale 2^{j-1} be first filtered by $G(\omega)$ filter, the columns of the resulting image be then filtered by $L(\omega)$, and let this operation be designated as filtering by LG to produce a wavelet image $H^j(k,l)$. Similarly, filtering by GL results in another wavelet image $V^j(k,l)$. They are combined to give wavelet modulus $M^j(k,l)$ and angle $A^j(k,l)$ at scale 2^j,

$$\begin{aligned} M^j(k,l) &= \sqrt{|H^j(k,l)|^2 + |V^j(k,l)|^2} \\ A^j(k,l) &= \angle\left(H^j(k,l), V^j(k,l)\right) \end{aligned}$$

the latter is quantized into 8 directional numbers, $(0, 1, \ldots, 7)$, each encoding a 45° sector. Since the edge direction is perpendicular to the wavelet angular direction, both $M^j(k,l)$ and $A^j(k,l)$ are used for edge detection. $M^j(k,l)$ is scanned row-wise and column-wise to locate wavelet modulus maxima $H^j_m(k,l)$ and $V^j_m(k,l)$ respectively. At each of such locations, $H^j_m(k,l)$ is accepted if its corresponding angle code is either 0 or 4 (horizontal direction), $V^j_m(k,l)$ is accepted if its angle code is either 2 or 6 (vertical direction), and either $H^j_m(k,l)$ or $V^j_m(k,l)$ is accepted if its angle code is 1, 3, 5 or 7 (diagonal direction). These wavelet modulus maxima are determined for edge detection in multiple scales.

This edge detector has been experimented on two sets of images: one is a set of test patterns under various signal-to-noise ratios(SNR), and the other is a set of three natural images. The test pattern is a bar image of 256 × 256 pixels containing four vertical bars of 5-, 10-, 15-, and 20-pixel widths respectively and 150-pixel length, as shown in Figure 3. The spacing between bars and to the borders range from 35 pixels to 45 pixels. The background intensity is 75 and the bar intensity is 175. A zero-mean Gaussian noise with standard deviation values of 10, 20, 32, and 50 was added to the synthetic image to result in SNR of 10 dB, 7 dB, 5 dB, and 3 dB respectively. Multiscale edges detected after wavelet modulus maxima were thresholded are shown in Figure 3. We measured (1) $P(AE/IE)$ which is the conditional probability of *detected* (or *assigned*) *edge*, AE, given the *true* (or *ideal*) *edge*, IE, (2) $P(IE/AE)$ which is conditional probability of the true edge given the detected edge, and (3) *Error distance*, which is the root mean square distance between the detected edge pixel and the true edge pixel, i.e.,

$$Error\ distance = \sqrt{\frac{1}{n_a}\sum_{i=1}^{n_a} d_i^2}$$

where n_a is the total number of detected edge pixels and d_i is the minimum distance between the ith detected edge pixel and the correspoding true edge pixel. The measurements were limited to the vertical edges of four bars, excluding corners and the horizontal edges. Also, only those detected edges lying within a band of 16-pixel width around each true edge line were taken into consideration. These performance measures at different scale levels are given in Table 2. For the noise free case, edge localization was accurate except, at the scale level 4, the vertical edges of the narrowest bar (of 5-pixel width) were detected at locations which are two pixels away from their true positions. This was caused by edge interactions as the effective filter support at this level became 32 pixels. For the noisy bar images with SNR of 5dB and up, the edge detector performed well at scale levels 2 and 3; for the case of 3dB SNR, it performed well only at scale level 3 where sufficient smoothing was available. Many noisy edge pixels were obtained at scale level 1 for insufficient smoothing, hence, $P(IE/AE)$ was low. At scale level 4, noises compounded the problem of edge interactions, consequently, edges of the narrowest bar were poorly detected and so were the corners. For comparisons, the performance measures of the Mallat-Zhong quadratic spline wavelet edge detector are given in Table 3. In either case, the performance shows an improvement over what were obtained by the refined regularization method and the first order regularization filter method which we developed earlier.

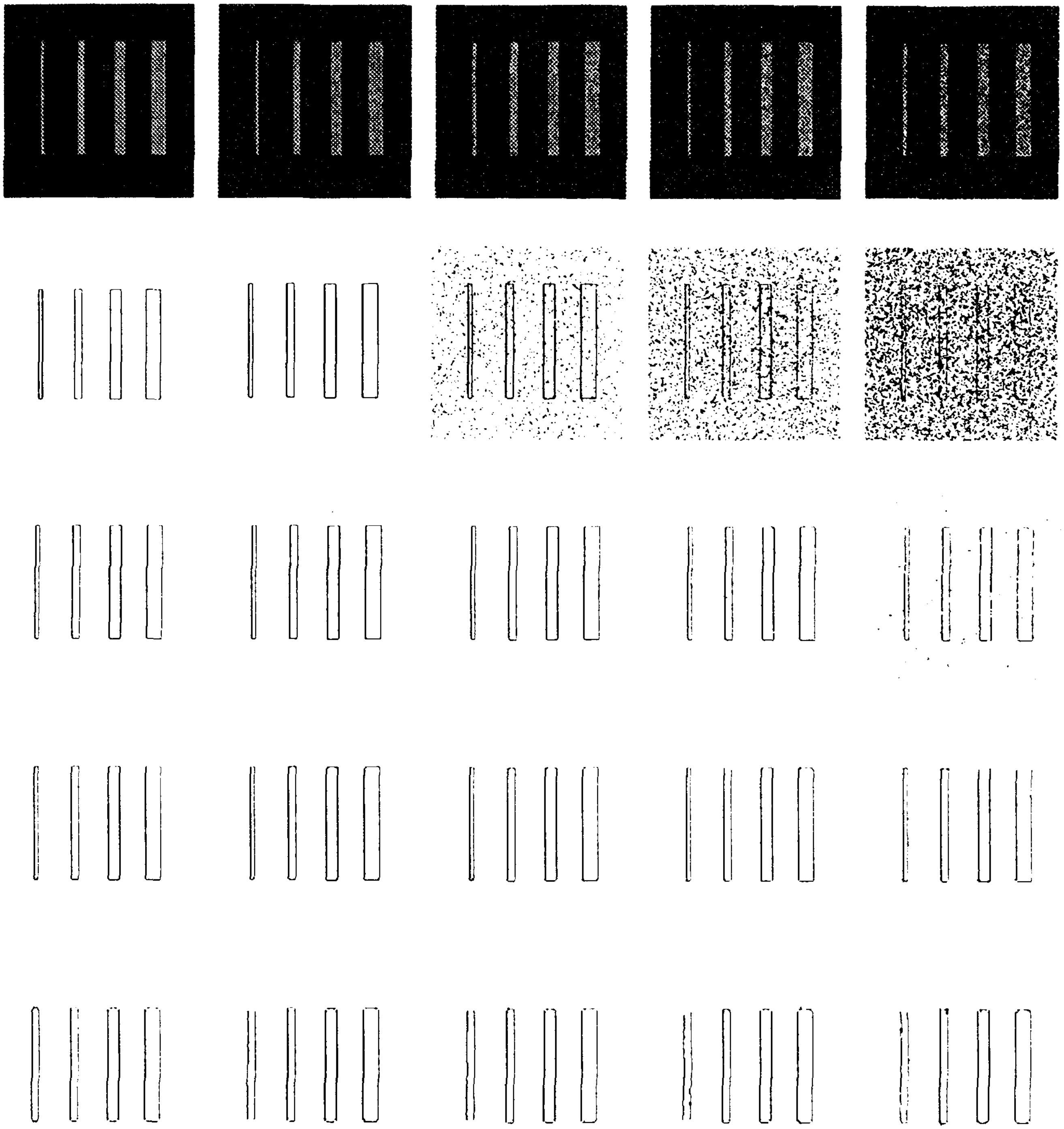


Figure 3: Edge maps of bar images(256 × 256) at four different scales; from left: noisefree, 10dB, 7dB, 5dB, and 3dB; from top: the original images, 1st, 2nd, 3rd, and 4th scale levels.

Table 2: Performance evaluation of the non-orthogonal wavelet edge detector on bar images in Figure 3.

		Noisefree	10dB	7dB	5dB	3dB
Level 1	P(AE/IE)	1.0000	1.0000	0.9898	0.8741	0.7259
	P(IE/AE)	1.0000	1.0000	0.6098	0.3685	0.2132
	Err. Dist.	0.0000	0.0000	3.0012	3.8499	4.2346
Level 2	P(AE/IE)	1.0000	1.0000	1.0000	0.9944	0.8843
	P(IE/AE)	1.0000	1.0000	0.9927	0.9315	0.8052
	Err. Dist.	0.0000	0.0000	0.0858	0.5050	1.1502
Level 3	P(AE/IE)	1.0000	1.0000	1.0000	1.0000	0.9861
	P(IE/AE)	1.0000	1.0000	1.0000	0.9945	0.9744
	Err. Dist.	0.0000	0.0000	0.0000	0.1391	0.2549
Level 4	P(AE/IE)	0.7500	0.7500	0.7500	0.7500	0.7500
	P(IE/AE)	0.7500	0.7500	0.7500	0.7479	0.7438
	Err. Dist.	1.0000	1.0801	1.1627	1.1655	1.1113

Table 3: Performance evaluation of the Mallat-Zhong quadratic spline wavelet edge detector on bar images in Figure 3.

		Noisefree	10dB	7dB	5dB	3dB
Level 1	P(AE/IE)	1.0000	1.0000	0.9944	0.8954	0.7278
	P(IE/AE)	1.0000	1.0000	0.6956	0.4300	0.2416
	Err. Dist.	0.0000	0.0000	2.6085	3.6538	4.1590
Level 2	P(AE/IE)	1.0000	1.0000	1.0000	0.9944	0.9213
	P(IE/AE)	1.0000	1.0000	1.0000	0.9844	0.8813
	Err. Dist.	0.0000	0.0000	0.0000	0.1248	0.6222
Level 3	P(AE/IE)	1.0000	1.0000	0.9778	0.9685	0.8917
	P(IE/AE)	1.0000	1.0000	0.9778	0.9685	0.8884
	Err. Dist.	0.0000	0.0000	0.1491	0.1774	0.3632
Level 4	P(AE/IE)	0.5000	0.5000	0.5000	0.5000	0.5093
	P(IE/AE)	0.5000	0.5000	0.5000	0.5000	0.5079
	Err. Dist.	1.5811	1.5811	1.5811	1.5811	1.5545

The edge detector was also experimented on three natural images: Lenna image, a house image and an airport runway image. The detected edge maps were shown in Figure 4. Lenna's facial features were more faithfully extracted in the edge map at scale level 2, as shown in the left column of Figure 4. For the house image shown in the middle column of Figure 4, the prominent linear edges were all detected at scale levels 2 and 3. For the runway image, the performance on runway edge localization at scale level 2 was very good, in spite of the fact that each runway has only 4 to 5 pixels in width, as shown in the right column in Figure 4.

The experimental results demonstrated that the edge localization performance is very good at scale levels 2 and 3 for images contaminated with a moderate amount of noise. At scale level 4, the detected edge location was shifted for thin objects of 5-pixel width due to the interaction between neighboring edges, and the detection capability deteriorated under severe noise corruptions. Note that the well-known Mallat-Zhong multiscale edge detector, based on a quadratic spline wavelet, may also be interpreted from solutions of the four-coefficient dilation equation. Their symmetric scaling function corresponds to a solution obtained at $(c_0, c_3) = (0.25, 0.25)$ which lies at the intersection of the diagonal line and a line segment (between $(0.5, 0)$ and $(0, 0.5)$) for differentiability. Thus, it provides a differentiable wavelet and scaling function with an optimum smoothing property for the given filter support size. Position shifts at large scales due to edge interactions, however, are unavoidable. Our non-orthogonal wavelet with four filter coefficients has been explored for its performance of reducing localization shift due to edge interactions at the expense of noise sensitivity.

Additional processing is needed to integrate the multiscale edge information for obtaining a further improved edge map; this remains to be a challenging problem. It would be desirable to select edges at appropriate scales adaptive to different spatial regions, analogous to the refined regularization but in the wavelet domain. The recent work on edge detection through nonlinear diffusion ([PEMA90, SHAH91, CATT92, FOBA92, WHPI93]) suggests a plausible alternative to formulate a space varying diffusion process in wavelet domain for detection of edges in multiple scales. This work is currently in progress.

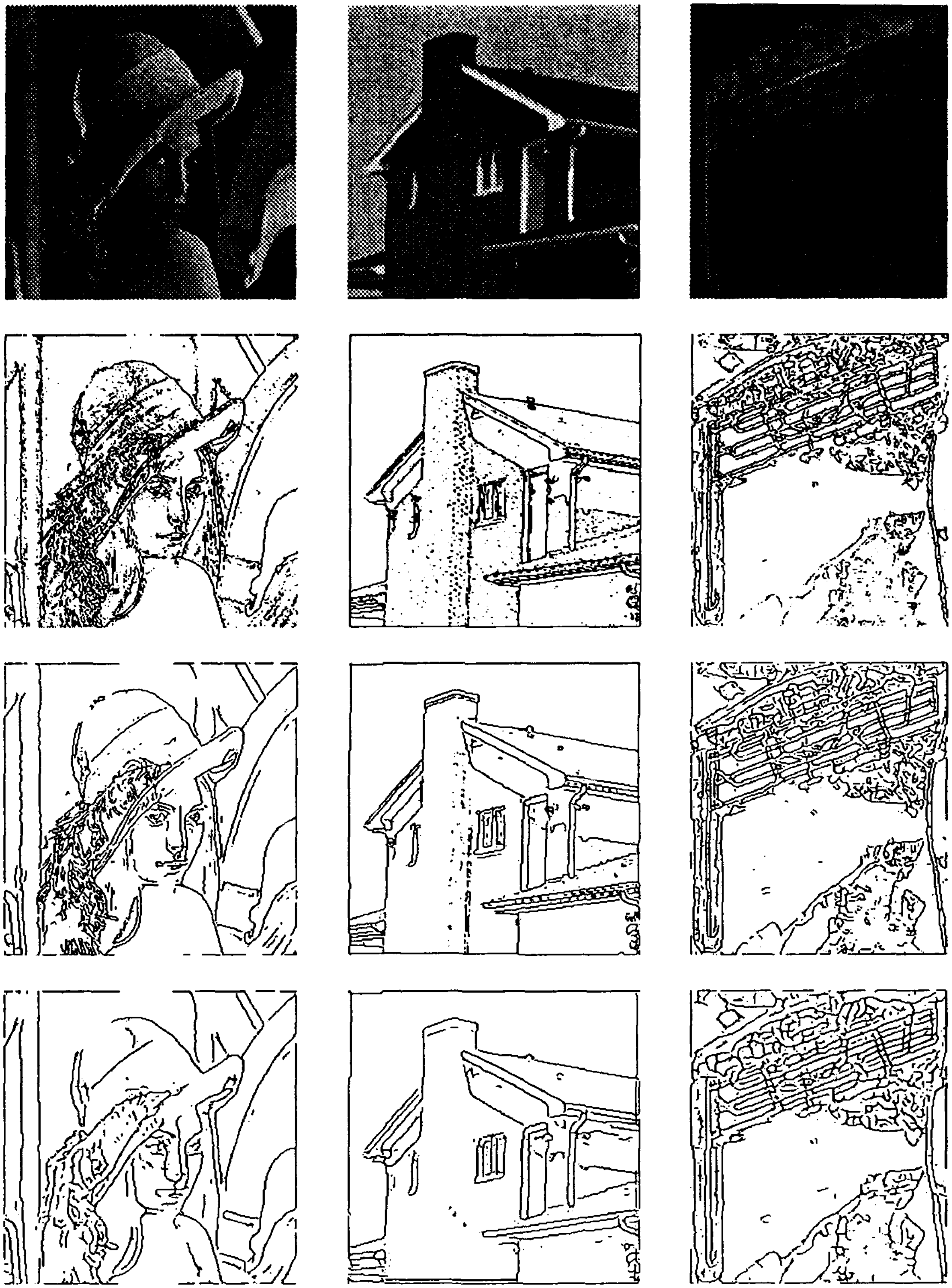


Figure 4: Detected edge maps of three natural images at three different scales; from left: Lenna(256 × 256), House(256 × 256), Runway(222 × 252); from top: the original images, 1st, 2nd, and 3rd scale levels.

B.1.b Uncertainty Constants of Daubechies' and B-spline Wavelets

In the time-frequency analysis of signals, the ideal analyzing (window) function should have the best time-frequency localization. It is well-known that we cannot obtain the most accurate selectivity in both time and frequency at the same time, which is known as the Heisenberg *Uncertainty Principle* [CHUI92, MEYE93]. Let a function $w(x) \in L^2(\mathbf{R})$ be a window function such that

$$xw(x) \in L^2(\mathbf{R}).$$

This condition implies that $w(x)$ is also in $L^1(\mathbf{R})$, and thus its Fourier transform $\hat{w}(\omega)$ is continuous. Moreover, $\hat{w}$ is in $L^2(\mathbf{R})$ by the Parseval identity [CHUI92]. Let the center $\overline{x}$ and the radius Δ_w of the window function $w(x)$ be defined by

$$\overline{x} := \frac{1}{||w||_2^2} \int_{-\infty}^{\infty} x|w(x)|^2 dx$$

and

$$\Delta_w := \frac{1}{||w||_2} \left(\int_{-\infty}^{\infty} (x - \overline{x})^2 |w(x)|^2 dx \right)^{1/2}$$

respectively, where $||w||_2$ denotes the norm induced by the inner product. The center $\overline{\omega}$ and radius $\Delta_{\hat{w}}$ of the Fourier transform $\hat{w}(\omega)$ can be similarly defined. The *Uncertainty Principle* states that

$$\Delta_w \Delta_{\hat{w}} \geq \tfrac{1}{2},$$

where equality is attained if and only if $w(x)$ is a Gaussian or a modulated Gaussian by a complex exponential function. The *uncertainty constant* of a window function is defined as the value of $\Delta_w \Delta_{\hat{w}}$, which shows the degree of time-frequency localization. A modulated Gaussian function, however, is not adequate to be used for wavelet decomposition; its integration over the whole real line is not equal to zero, and thus it does not satisfy the *admissibility condition* of a wavelet [DAUB92b, CHUI92]. Also a Gaussian function cannot be used as a scaling function. Furthermore, by the Balian-Low theorem, there is no coherent type basis having good time-frequency localization, regardless of orthogonality [DAUB91b]. The wavelet analysis should provide a basis for $L^2(\mathbf{R})$, either orthonormal or nonorthogonal, which must be of non-Gaussian or non-Gabor type of functions having fast decay at infinity in both time and frequency domains.

As discrete filters are used in dyadic wavelet decomposition of discrete-time signals, characteristics of the discrete filters are important. The center $\overline{n}$ and the radius Δ_g of a discrete filter $g = (g_n)_{n \in \mathbf{Z}}$ are defined by

$$\overline{n} := \frac{1}{||g||_2^2} \sum_{n \in \mathbf{Z}} n|g_n|^2$$

and

$$\Delta_g := \frac{1}{||g||_2}\left(\sum_{n\in\mathbf{Z}}(n-\overline{n})^2|g_n|^2\right)^{1/2}$$

respectively, where $||g||_2 = (\sum_{n\in\mathbf{Z}}|g_n|^2)^{1/2}$. The transfer function (Fourier representation) $G(\omega)$ of the discrete filter g_n is given by

$$G(\omega) := \sum_{n\in\mathbf{Z}} g_n e^{-in\omega}.$$

The center $\overline{\omega}$ and the radius Δ_G of the filter $G(\omega)$ are determined by

$$\overline{\omega} := \tfrac{1}{||G||_2^2}\textstyle\int_0^{2\pi}\omega|G(\omega)|^2 d\omega$$

and

$$\Delta_G := \tfrac{1}{||G||_2}\left(\textstyle\int_0^{2\pi}(\omega-\overline{\omega})^2|G(\omega)|^2 d\omega\right)^{1/2}$$

respectively, where $||G||_2 := \left(\int_0^{2\pi}|G(\omega)|^2 d\omega\right)^{1/2}$. The uncertainty constants of a wavelet filter g is given by $\Delta_g\Delta_G$. We have examined uncertainty constant of Daubechies wavelets, B-spline wavelets and their corresponding discrete filters [KIML93]. Some interesting characteristics are presented in the following.

Uncertainty Constants of Daubechies' Wavelets:

The Daubechies' wavelets are compactly supported and fully orthogonal, and thus they provide effective decomposition and reconstruction algorithms. They also have a number of vanishing moments, depending upon the support length and regularity of the wavelet. We have computed the uncertainty constants for wavelets and scaling functions of Daubechies as shown in Table 4 and Figure 5 (a) and (b), where D_m stands for the Daubechies' wavelet (or scaling function) of index (order) $m/2$ corresponding to m scaling coefficients. It is noted that D_{12} has the minimum uncertainty constant among the Daubechies' wavelets. On the other hand, the minimum of uncertainty constants for the scaling functions occurs at D_6 instead of D_{12}. This is consistent with the fact that a wavelet $\psi(x)$ and its corresponding scaling function $\phi(x)$ are related by the two-scale equation

$$\hat{\psi}(2\omega) = G(\omega)\hat{\phi}(\omega).$$

Table 5 lists the radii and uncertainty constants of the Daubechies' wavelet filter g_n, which are also shown in Figure 5(c). The uncertainty constant increases almost linearly as the filter size increases.

Table 4: Radii and uncertainty constants of Daubechies' wavelets.

(a) wavelets

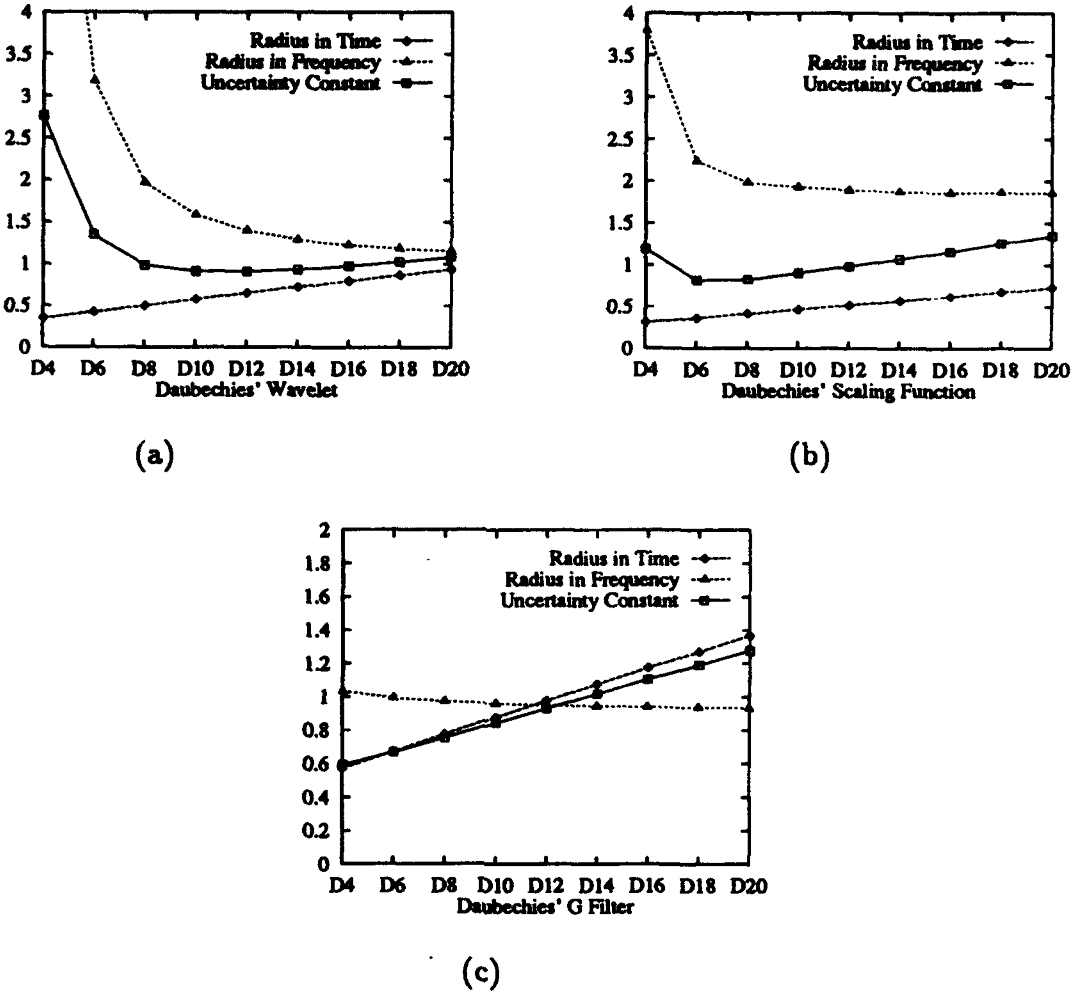
ψ	Δ_ψ	$\Delta_{\hat\psi}$	$\Delta_\psi \cdot \Delta_{\hat\psi}$
D_4	0.3487	7.9234	2.7631
D_6	0.4241	3.1789	1.3481
D_8	0.5005	1.9722	0.9871
D_{10}	0.5761	1.5813	0.9110
D_{12}	0.6506	1.3925	0.9059
D_{14}	0.7237	1.2881	0.9322
D_{16}	0.7956	1.2249	0.9746
D_{18}	0.8664	1.1831	1.0251
D_{20}	0.9362	1.1532	1.0796

(b) scaling functions

ϕ	Δ_ϕ	$\Delta_{\hat\phi}$	$\Delta_\phi \cdot \Delta_{\hat\phi}$
D_4	0.3130	3.7925	1.1872
D_6	0.3634	2.2372	0.8130
D_8	0.4158	1.9796	0.8232
D_{10}	0.4682	1.9284	0.9028
D_{12}	0.5203	1.8906	0.9836
D_{14}	0.5721	1.8696	1.0695
D_{16}	0.6235	1.8563	1.1574
D_{18}	0.6746	1.8614	1.2556
D_{20}	0.7253	1.8546	1.3450

Table 5: Radii and uncertainty constants of Daubechies' discrete filters.

Filter	Δ_g	Δ_G	$\Delta_g \cdot \Delta_G$
D_4	0.5728	1.0333	0.5919
D_6	0.6735	0.9938	0.6694
D_8	0.7762	0.9731	0.7553
D_{10}	0.8781	0.9603	0.8432
D_{12}	0.9786	0.9516	0.9313
D_{14}	1.0778	0.9454	1.0189
D_{16}	1.1757	0.9407	1.1059
D_{18}	1.2723	0.9370	1.1922
D_{20}	1.3679	0.9340	1.2777



(a)

(b)

(c)

Figure 5: Uncertainty constants for (a) Daubechies' wavelets, (b) Daubechies' scaling functions, and (c) their corresponding highpass G filters. D_{12} has the minimum value in (a).

Uncertainty Constants of B-Spline Wavelets:

The uncertainty constant of a Gaussian function is exactly equal to 0.5 which indicates the best possible time-frequency localization. By computations, we found that the uncertainty constants of B-spline functions are very close to the Gaussian case and converge to 0.5 very rapidly; the uncertainty constant is 0.5038 for the quadratic B-spline scaling function and 0.5012 for the cubic B-spline scaling function. The uncertainty constants of B-spline wavelets are also quite close to the Gaussian case, being 0.5302 for the quadratic B-spline wavelet and 0.5064 for the cubic B-spline wavelet; the latter is already close to 0.5. As the order of B-spline is increased, its uncertainty constant gradually approaches to 0.5.

Now let us consider the discrete filters for B-spline wavelets and the discrete Gaussian filters. Their radii and uncertainty constants have been computed as shown in Table 6 and Figure 6 for comparisons. The discrete Gaussian filters are obtained by discretizing the Gaussian function: for a positive integer N,

$$g_n := \frac{3\sigma}{N} f(3\sigma \frac{n}{N}), \quad \forall n \in \mathbf{Z}$$

where f is a normalized (area=1) Gaussian function, i.e., $f(x) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{x^2}{2\sigma^2})$, and $(g_n)_{n \in \mathbf{Z}}$ is truncated so that $\text{supp}(g_n) = [-N, N]$, giving a total of $2N + 1$ filter coefficients. Note that this support is equivalent to $[-3\sigma, +3\sigma]$ for $f(x)$ and covers most of the energy in a Gaussian. Figure 6(b) shows some fluctuations of uncertainty constant of the discrete Gaussian filter of small N, reflecting errors caused by insufficient sampling rate at low orders.

In contrast, B-spline wavelet filters do not show such fluctuations even with fewer number of coefficients. This stability comes from the elegant two-scale relation of $\hat{\phi}(2\omega) = H(\omega)\hat{\phi}(\omega)$ and the construction of the wavelet filter G from the low-pass filter H. Unlike in Daubechies' wavelet filters, the uncertainty constants of B-spline wavelet filters are almost fixed at around 0.5 which is equivalent to the Gaussian case. Here the B-spline wavelet filters are constructed by $g_n := (-1)^{n-1} h_{1-n}$ where h_n is from the scaling coefficients of the two-scale dilation equation of a B-spline function. Then $G(\omega) = \overline{H(\omega + \pi)} e^{-i\omega}$, hence the uncertainty constants (and also radii) are the same for both filters H and G.

If we take another g filter, for example, g = {-0.5, 0.5}, for B-spline wavelets, then the uncertainty constants of the wavelets are around 0.5744 for all orders with very small($< 1\%$) variations that can only be seen for low order B-splines. This is not of our interests since the time-frequency characteristics of wavelet filters are fixed and irrelevant to the B-spline order. For this reason, we choose $g_n := (-1)^{n-1} h_{1-n}$ and observe the time-frequency localization effect as the order of the filter is changed.

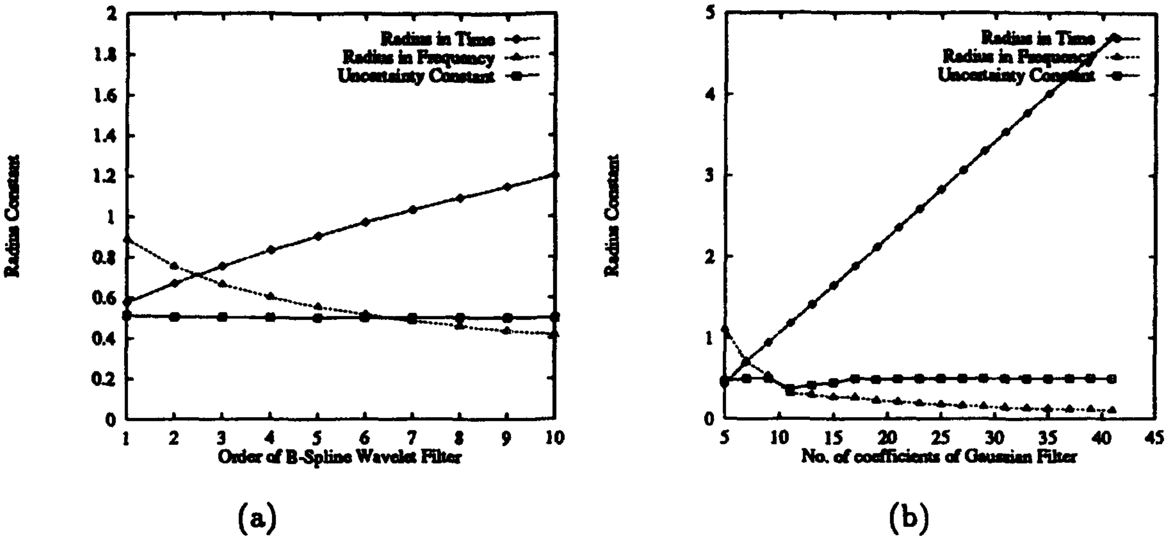
Table 6: Radii and uncertainty constants of B-spline wavelet filters and discrete Gaussian filters. (A B-spline of order 2 means a quadratic B-spline, and the filter length is equal to the order number plus 2 for a B-spline wavelet filter and equal to $2N+1$ for a discrete Gaussian filter.)

(a) B-spline wavelet filters

Order	Δ_g	Δ_G	$\Delta_g \cdot \Delta_G$
1	0.5774	0.8887	0.5131
2	0.6708	0.7534	0.5054
3	0.7559	0.6653	0.5029
4	0.8333	0.6022	0.5018
5	0.9045	0.5542	0.5013
6	0.9707	0.5160	0.5009
7	1.0328	0.4848	0.5007
8	1.0914	0.4586	0.5005
9	1.1471	0.4363	0.5004
10	1.2002	0.4169	0.5004

(b) Discrete Gaussian filters

N	Δ_g	Δ_G	$\Delta_g \cdot \Delta_G$
2	0.4182	1.1064	0.4627
3	0.7064	0.7076	0.4998
4	0.9428	0.5303	0.5000
5	1.1785	0.3125	0.3683
6	1.4142	0.2926	0.4138
7	1.6498	0.2712	0.4474
8	1.8855	0.2652	0.5001
9	2.1212	0.2284	0.4845
10	2.3568	0.2090	0.4925
11	2.5925	0.1916	0.4967
12	2.8282	0.1763	0.4987
13	3.0638	0.1630	0.4995
14	3.2995	0.1515	0.4999
15	3.5351	0.1414	0.5000
16	3.7708	0.1326	0.5001
17	4.0065	0.1248	0.5001
18	4.2421	0.1179	0.5001
19	4.4778	0.1117	0.5001
20	4.7134	0.1061	0.5001



(a) (b)

Figure 6: Uncertainty constants for (a)B-spline wavelet filters, (b)Discrete Gaussian filters. (The uncertainty constant of a function is the multiplication of the radius in time and the radius in frequency.)

In summary, we have observed that the Daubechies' wavelet D_{12} has the best time-frequency localization among Daubechies' wavelets of all orders. This may give a criterion in choosing a Daubechies' wavelet for the continuous wavelet transform. However, the corresponding discrete filters for the Daubechies' wavelets do not show such characteristics. Because the 2π-periodic nature of the transfer function of a discrete filter, its frequency radius does not change much when the filter order is changed and, thus, its uncertainty constant increases almost linearly as the filter order increases. We need to consider a trade-off between regularity (smoothness), filter length, and time-frequency localization.

On the other hand, in the B-spline case, the uncertainty constants of wavelet filters are very close to that of the ideal Gaussian filter for almost all orders; there is no need to be concerned with this factor when we choose a best B-spline wavelet. B-spline wavelets provide very good time-frequency localization. If orthonormality is not required, a B-spline wavelet may be prefered for wavelet decomposition since it is symmetric, very regular, and compactly supported.

B.1.c Texture Segmentation by Modulated Wavelets

Texture segmentation based on the multiresolution imformation is an improtant problem in the low level image processing [UNSE89]. In human vision, there exists a preattentive visual system that can easily detect different textures which have some obvious features in the frequency domain [JULE83]. Gabor filter has been recognized as one that may exist in the 2-dimensional visual cortical filtering process providing the optimun joint space/frequency resolution [DAUG85]. For practical applications, multichannel Gabor filters have been used in texture analysis and segmentation providing localized spatial changes in frequency and orientation [BOVI90, FARR91]. The tuning of their center frequencies and bandwidths requires, however, substantial prior analysis and experimental decision. Wavelet theory provides an efficient decomposition of signals and images giving both time(space) and frequency localization [MALL89c, MALL89d, STRA89, DAUB88]. The standard wavelet transform focuses on the successive decompositions in the low frequency region. But the dominant spatial frequencies of textures are usually in the middle frequency region which is not necessarily focused on by the standard wavelet decomposition. A tree-structured wavelet transform has been developed [CHAN92] to extend the decomposition into the high frequency region. This approach is similar to the wavelet packet method [WICK92a, COWI92], which has recently been applied to texture classification [LAIN92]. Both approaches are restricted in their orientation selectivity in the first quadrant of the frequency plane and do not discriminate the textures of $+\theta$ and $-\theta$ orientations. We have considered a multi-resolution decomposition approach based on a modulated "wavelet", providing multichannel filters to cover both frequency and orientation regions in full [HSLI93].

Multiresolution Analysis:

Wavelet Transform provides a multiresolution decomposition of any signal with finite energy into a set of frequency channels which have the same bandwidth on a logarithmic scale. This property results in high resolution in the time domain for high frequency components and low resolution in the time domain for low frequency components. This characteristic is similar to what has been observed in the humman visual process [NACH75].

Let $L^2(R)$ be the function space of all finite energy signals $f(x)$, A_j be an operator which approximates a signal at resolution 2^j, and V_j be the subspace of $L^2(R)$ which consists of all possible signals at resolution 2^j, where j $\in Z$ (a set of all integers). A multiresolution analysis is a set of vector spaces $(V_j)_{j \in Z}$ that satisfy the following properties:

(1) A_j is an orthogonal projection operator, which approximates a signal with minimum distance error, on the space V_j;

(2) $A_j \circ A_j f(x) = A_j f(x)$;

(3) causality property $V_{j+1} \subset V_j, \forall j \in Z$, which means that the approximation at a higher resolution contains all the information in a lower resolution;

(4) The subspace V_j can be derived from any other subspace V_k by scaling all functions in V_k by the ratio of their resolutions,
$f(x) \in V_j \leftrightarrow f(2^{(k-j)}x) \in V_k$;

(5) $\overline{\bigcup_{j\in Z} V_j} \; = \; L^2(R), \; \bigcap_{j\in Z} V_j \; = \; \oslash.$

Daubechies constructed her famous orthonormal bases of compactly supported wavelets. Let $\phi(x)$ be a scaling function, $\phi_{jn} = 2^{-\frac{j}{2}}\phi(2^{-j}x - n)$; $\psi(x)$ be a wavelet and $\psi_{jn} = 2^{-\frac{j}{2}}\psi(2^{-j}x - n)$. $\{\phi_{jn}\}_{n\in Z}$ forms an orthonormal basis for V_j, $\{\psi_{jn}\}_{n\in Z}$ is the orthonormal compliment tc V_j in V_{j-1}. The wavelet transform of $f(x)$ at resolution 2^j is given by $W_j(k) =< f(x), \psi_{jk}(x) >= \sum g(n)S_{j-1}(n+2k)$ which is the high frequency component of $f(x)$ at scale 2^j. The low frequency component is given by $S_j(k) =< f(x), \phi_{jk}(x) >= \sum h(n)S_{j-1}(n+2k)$, where $h(n) =< \phi(x), \phi(2x-n) >$ is a low-pass filter and $g(n) =< \psi(x), \phi(2x-n) >$ is a high-pass filter. For applications to 2-dimensional images, a tensor product of two one-dimensional functions; $\Phi(x,y) = \phi(x)\phi(y)$, $\Psi^1(x,y) = \psi(x)\phi(y)$, $\Psi^2(x,y) = \phi(x)\psi(y)$, $\Psi^3(x,y) = \psi(x)\psi(y)$, where $\Phi(x,y)$ is the 2-dimensional orthonormal scaling function, $\Psi^1(x,y)$, $\Psi^2(x,y)$ and $\Psi^3(x,y)$ are three 2-dimensional wavelets. This is the standard pyramidal wavelet transform analysis. Wavelet packet provides further decomposition of high frequency components. Both approaches take care of orientations in the first quadrant of the frequency plane (w_x, w_y). They do not, however, discirminate textures in orientations $+\theta$ and $-\theta$.

Modulated "Wavelets" and Multichannels:

For analysis of textured images, it is desirable to have oriented wavelets covering characteristics in one half of the frequency plane (w_x, w_y). Let us consider first the modulated scaling function in 1-dimensional case, $\phi(x)e^{jUx}$, where U is a modulation frequency. Since $h(n) =< \phi(x), \phi(2x-n) >$, the modulated filter becomes

$$\begin{aligned} h(n)e^{jUn} &= e^{jUn} < \phi(x), \phi(2x-n) > \\ &= < \phi(x), \phi(2x-n)e^{-jUn} > \\ &= < \phi(x)e^{j2Ux}, \phi(2x-n)e^{jU(2x-n)} > \end{aligned}$$

Modulating $h(n)$ by U is equivalent to modulating $\phi(x)$ by $2U$ in subspace V_0 and modulating $\phi(2x)$ by U in subspace V_{-1}. If $U = \pi$, the modulated $h(n)$ is, in fact, the high-pass filter $g(n)$ and $H(w-\pi) = G(w)$; in this case, the modulated $\phi(x)$ is close to a wavelet. For other values of U, the modulated $h(n)$ is equivalent to a band-pass filter with center frequency at U. The modulated $\phi(x)$ will be a numerically acceptable "wavelet" if its infinite integral is sufficiently small. Extend to the 2-dimensional case, modulation of a scaling function $\Phi(x,y) = \phi(x)\phi(y)$ induces a corresponding modulated filter

$$g^{(k)}(n,m) = h(n)h(m)e^{j(U_k n + V_k m)}, \text{ k=1,2...}$$

which is a high-(or band-) pass filter with the kth orientation. Without modulation,

$$g^{(0)}(n,m) = h(n)h(m)$$

which gives a low-pass filter. Modulating $\Phi(x,y)$ with frequency pair (U_k, V_k) gives a modulated "wavelet" (depending on the acceptable values of U_k and V_k) with the kth orientations,

$$\Psi^k(x,y) = \phi(x)\phi(y)e^{j(U_k x+V_k y)}$$

The "wavelet" transform with the kth orientation at resolution 2^j is given by

$$W_j^k f(x,y) =< f, \Psi_j^k >= f * \tilde{\Psi}_j^k(x,y)$$

$$\Psi_j^k(x,y) = 2^{-j}\Psi^k(2^{-j}x, 2^{-j}y) \quad \text{and} \quad \tilde{\Psi}_j^k(x,y) = \Psi_j^k(-x,-y).$$

The transform may be interpreted as filtering the image $f(x,y)$ with the band-pass filter $g^{(k)}(n,m)$ having an orientation selectivity,

$$\begin{aligned} W_j^k f(x,y) &= \sum g^{(k)}(n,m) S_{j-1}(n+2x, m+2y) \\ S_j f(x,y) &= \sum g^{(0)}(n,m) S_{j-1}(n+2x, m+2y) \end{aligned}$$

We selected four orientations: $90^o, 45^o, 0^o$, and -45^o, corresponding to $k = 1,2,3$ and 4. For the first scale, the four center frequency pairs (U_k, V_k) are located at $(0,\pi), (\frac{\pi}{\sqrt{2}}, \frac{\pi}{\sqrt{2}}), (\pi, 0)$ and $(\frac{\pi}{\sqrt{2}}, -\frac{\pi}{\sqrt{2}})$ along a circle of radius equal to π in the plane (w_x, w_y). Their frequency responses are shown in Figure 7(b) as compared to those of Gabor filters shown in Figure 7(a). For $|U_k| = |V_k| = \frac{\pi}{\sqrt{2}}$; the wavelet condition is not well satisfied but we adopted the notion nevertheless. For the successive scales, the filter outputs are down-sampled; thus, each filter provides a frequency channel whose output captures the high frequency information centered arround $(\frac{U_k}{2^{j-1}}, \frac{V_k}{2^{j-1}})$.

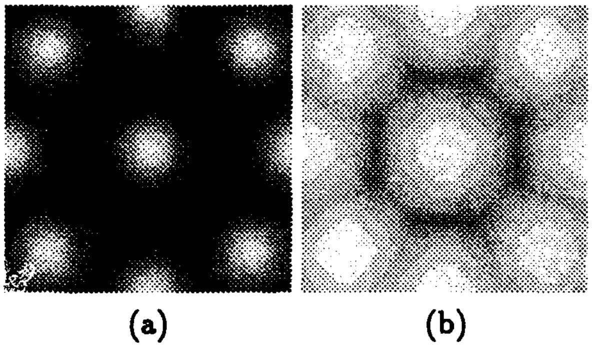


Figure 7: (a) Frequency responses of 2-dimensional Gabor filters modulated by π with 8 orientations; (b) frequency responses of modulated filters derived from modulated Daubechies scaling function D_6.

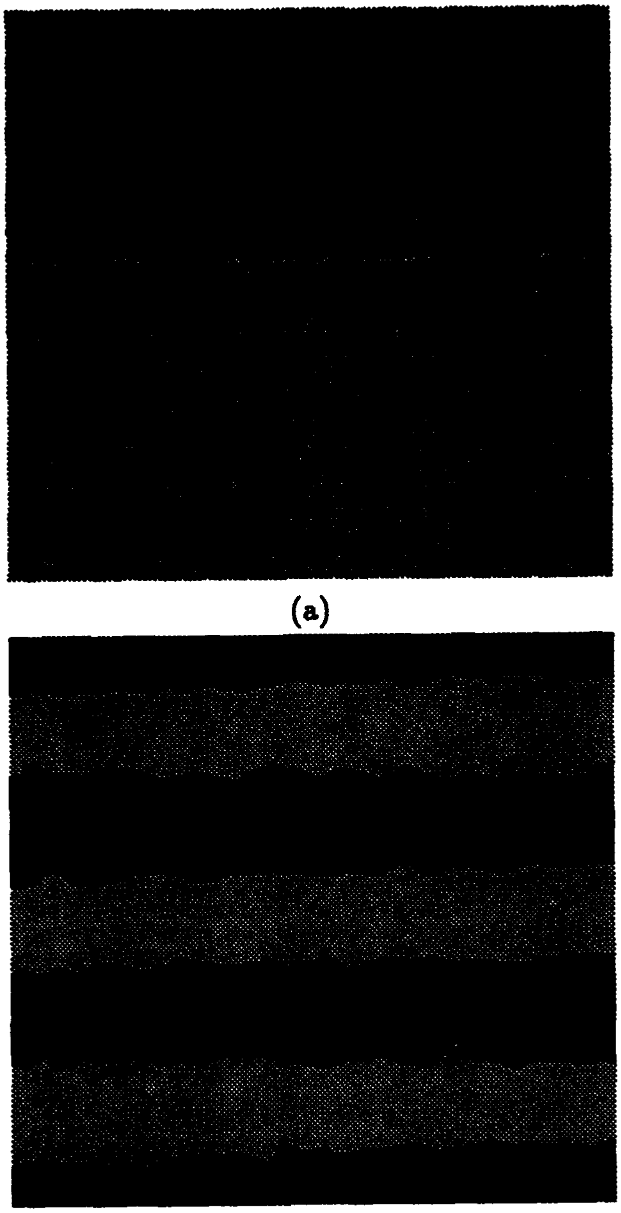
Texture Segmentation:

The multichannel amplitude responses are used as texture features. The original image data are taken as S_0. To begin with the first scale ($j = 1$), channel outputs $W_j^{(k)}(n,m)$, ($k = 1,2,3,4$), and $S_j(n,m)$ are computed down the scale until at a certain $j = J$ where $S_J(n,m)$ becomes insignificant. In our experimental study, Daubechies scaling function D_6 and the corresponding filter $h(n)$ were used. Also, only the first two scales and eight features were considered. These features are designated by $x_1 = W_1^1, x_2 = W_1^2, x_3 = W_1^3, x_4 = W_1^4, x_5 = W_2^1, x_6 = W_2^2, x_7 = W_2^3$ and $x_8 = W_2^4$. In an unsupervised mode, texture segmentation can be done by clustering data in the selected feature space and assigning cluster index to each pixel to form segmented regions.

Texture segmentation with multi-channel measurements based on modulated "wavelets" has been tested on three images. Figure 8(a) shows an image of Herringbone weave of 256 × 240 pixels. It contains seven strips of similar texture but with two different orientations. No significant responses appeared in channels 1, 2, 3, 4, 5 and 7. At each pixel, the maximum amplitude response appeared either at channel 6 (the texture with 45° orientation) or at channel 8 (with −45° orientation). Segmentation was obtained as shown in Figure 8(b). The border between segmented neighboring texture regions has small fluctuations of from 1 to 3 pixels. The orientation selective channels worked well for this type of texture discrimination.

Figure 9(a) shows an image (260x112) taken from a cut-away view of a cylindrical tube formed by laminated copper sheets sandwiched with layers of coils. The measurements from all eight channels were used in a clustering experiment with the K-means algorithm. Segmentation of regions of two different textures was obtained through assignment to two clusters. Cluster 1 refers to the metal sheet surface, cluster 2 refers to the region with coil cross-sections. The measurements from the high frequency channels (1 through 4) showed large variances, as given in Table 7; the texture border was not accurately obtained as shown in Figure 9(b).

The third experiment was performed on a cloud image (360x300) shown in Figure 10(a) which shows a part of a remotely sensed image of clouds over North Sea. The spread of the cold air is shown on the left and cloud streets of a cyclon appears on the right. Segmentation was done by clustering measurements of all eight channels into three clusters; the clustering result is summarized in Table 8 and shown in Figure 10(b). The responses from the high frequency channels obviously have more discriminating power. Cluster 2 (shown in grey) indicates the region of cyclon cloud streets, cluster 3 (shown in black) indicates the cold air region , and cluster 1 (shown in white) shows regions of between these two classes. The texture border of the cyclon cloud street was quite well localized as shown in Figure 10(c).



(a)

(b)

Figure 8: (a) The original textured image; (b) the result of texture segmentation.

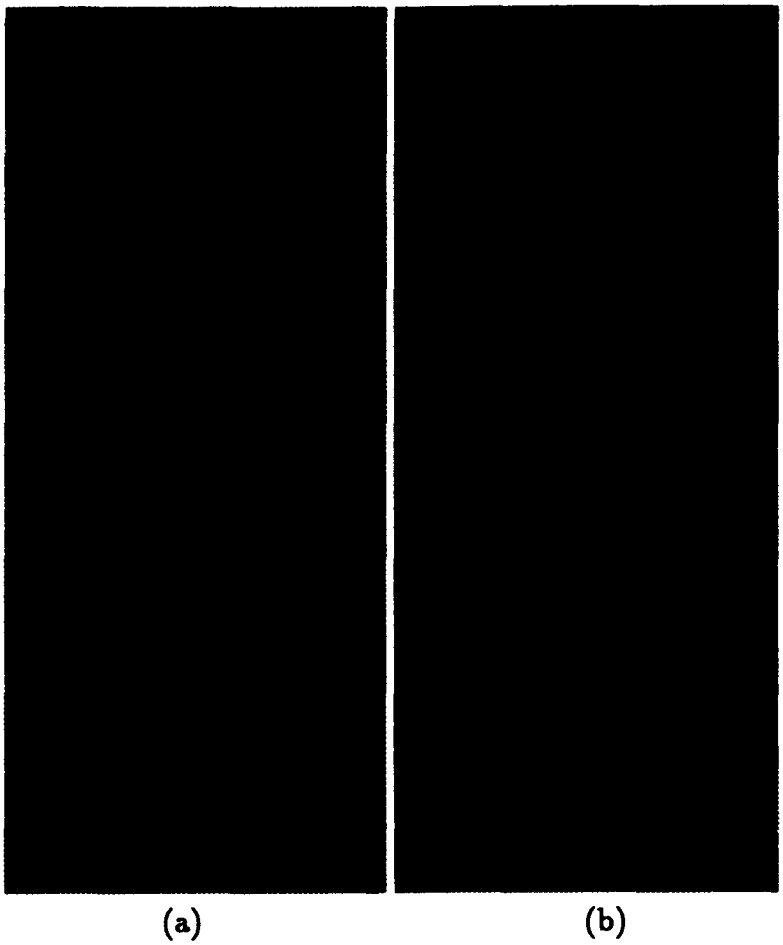


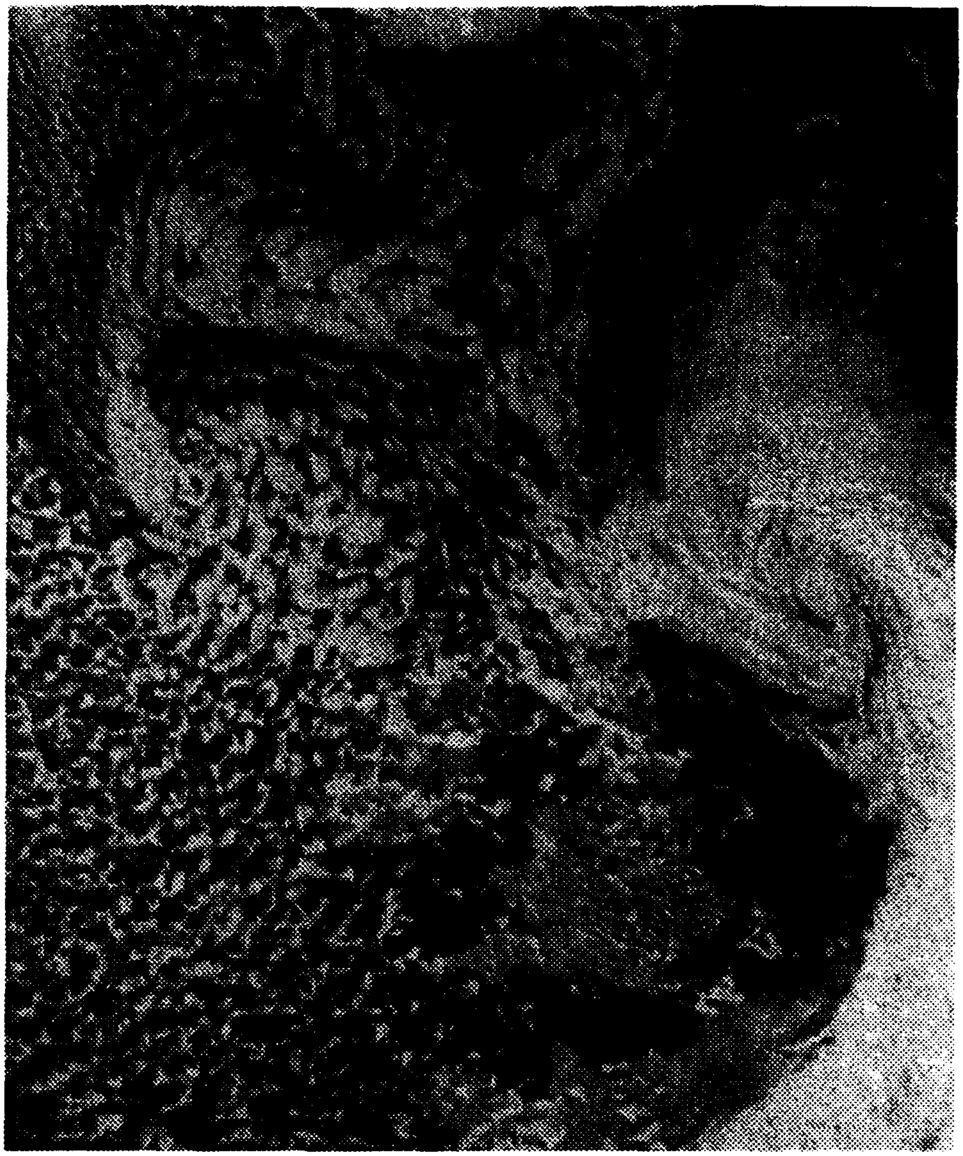
Figure 9: (a) The original image of a cut-away view of a cylindrical tube; (b) the result of texture segmentation based on k-means clustering with 200 iterations.

Table 7: Clustering result on the tube image in Figure 9.

features	cluster 1		cluster 2	
	mean	variance	mean	variance
x1	8.1	20.4	24.9	27.8
x2	9.1	10.3	21.7	28.1
x3	8.4	17.7	25.5	27.1
x4	11.3	22.5	28.8	33.4
x5	1.8	0.3	3.9	0.4
x6	1.9	0.2	3.7	0.6
x7	1.8	0.3	3.9	0.6
x8	2.3	0.4	4.9	1.0

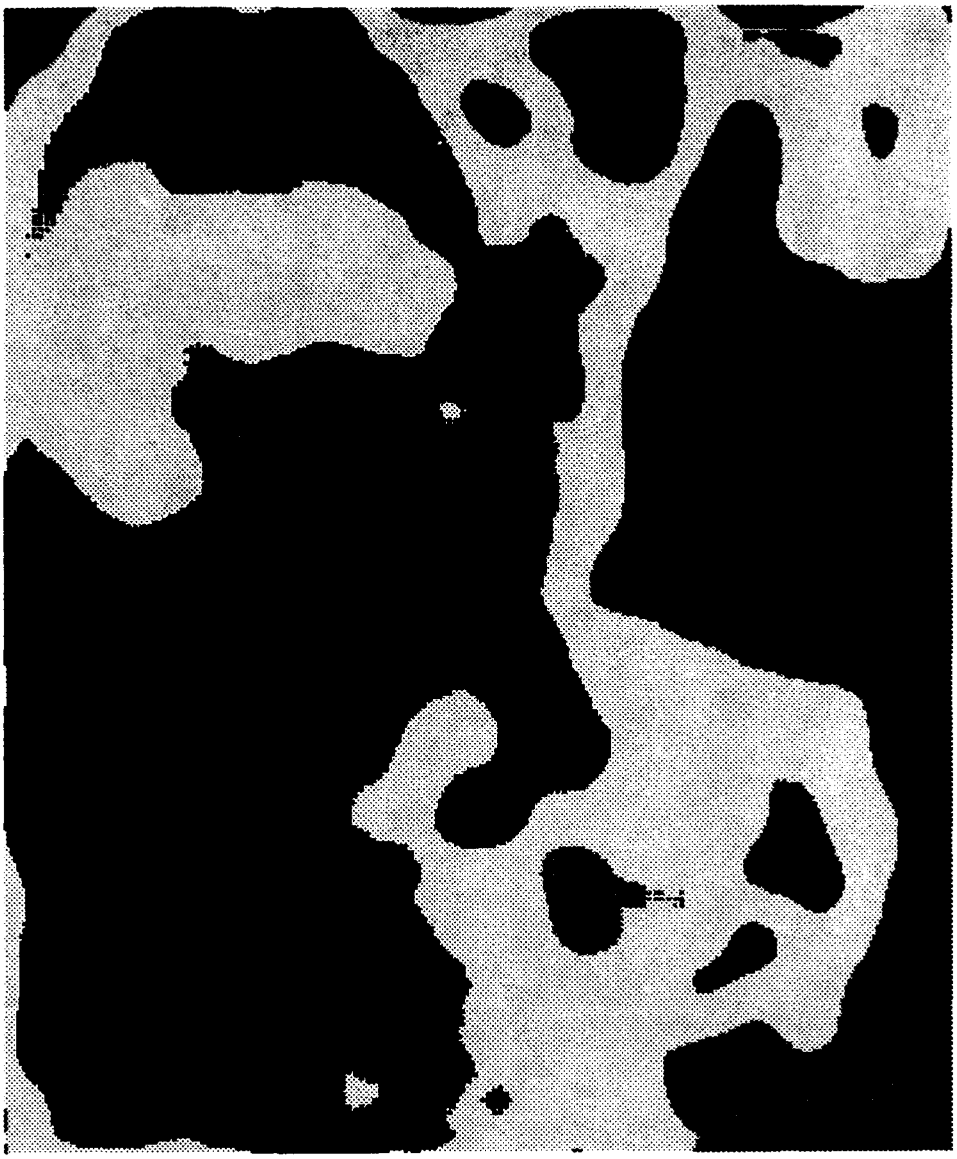
Table 8: Clustering result on the cloud image in Figure 10.

features	cluster 1		cluster 2		cluster 3	
	mean	variance	mean	variance	mean	variance
x1	18.1	18.9	7.8	14.1	28.2	24.3
x2	19.5	13.6	12.5	11.7	27.5	25.8
x3	17.6	22.1	8.7	11.8	27.1	21.0
x4	20.6	13.7	12.6	8.9	26.4	18.0
x5	3.1	0.4	1.8	0.3	4.3	0.4
x6	3.4	0.2	2.4	0.2	4.7	0.5
x7	3.1	0.8	1.9	0.3	4.2	0.4
x8	3.8	0.4	2.5	0.4	4.8	0.3



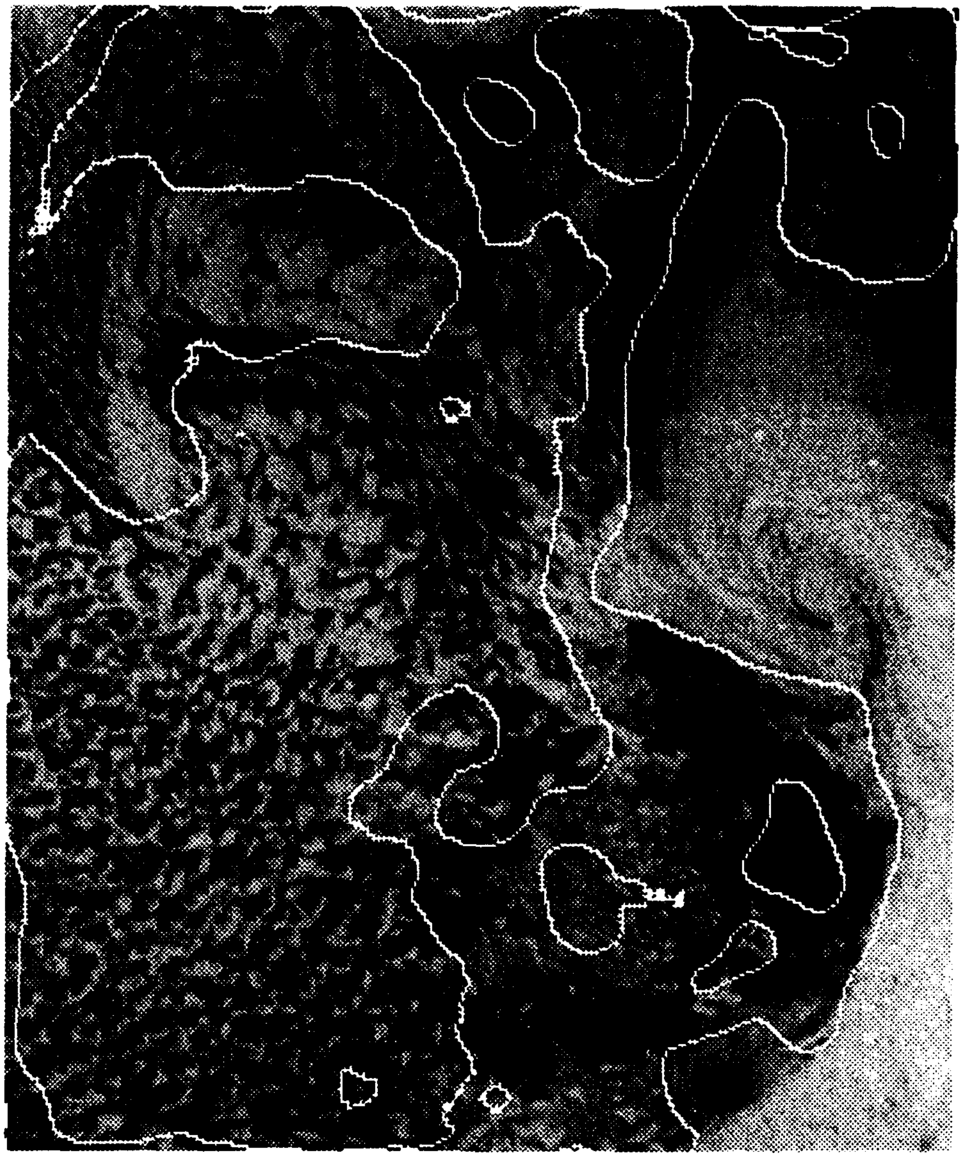
(a)

Figure 10: (a) The original cloud image showing a cyclon pattern; (b) the result of texture segmentation based on k-menas clustering with 50 iterations; (c) the original image with an overlay of the segmented texture boundary.



(b)

Figure 10: (Continued)



(c)

Figure 10: (Continued)

B.1.d Linearization of Segmented Objects at Multiple Resolutions

In most 2D or 3D image pattern recognition schemes, the object of interest is first segmented from the rest of the image. It is then compared to a database of objects and classified as the object in the database it most closely matches. Often each segmented object's silhouette is stored as a binary image. The relationships among the pixels in the silhouette determine its shape and ultimately its classification. Such a scheme may be used in the classification of military vehicles [CHEN91b, LILI89].

Typically some method is used to reduce the otherwise large number of pixel-to-pixel comparisons needed to successfully classify the silhouette. If a complex structured silhouette can be approximated by a small number of connected piecewise linear or planar segments, then the number of comparisons and hence the number of computations can be reduced by several orders of magnitude. The problem is how to construct such a linearization without losing the discriminating features of the silhouette.

In this study, the number of selected pixel locations in the silhouette is reduced by combining those pixels which share a common linear pattern (either vertical, horizontal, or diagonal) with their neighbors [MILI93]. This reduction is accomplished by examining the silhouette under multiple resolutions and classifying pixels as vertical, horizontal, or diagonal if they are part of such a pattern. This is done at each resolution. The silhouette can be reconstructed using these combined groups of pixels. The Haar wavelet is used to generate the multi-resolution decomposition.

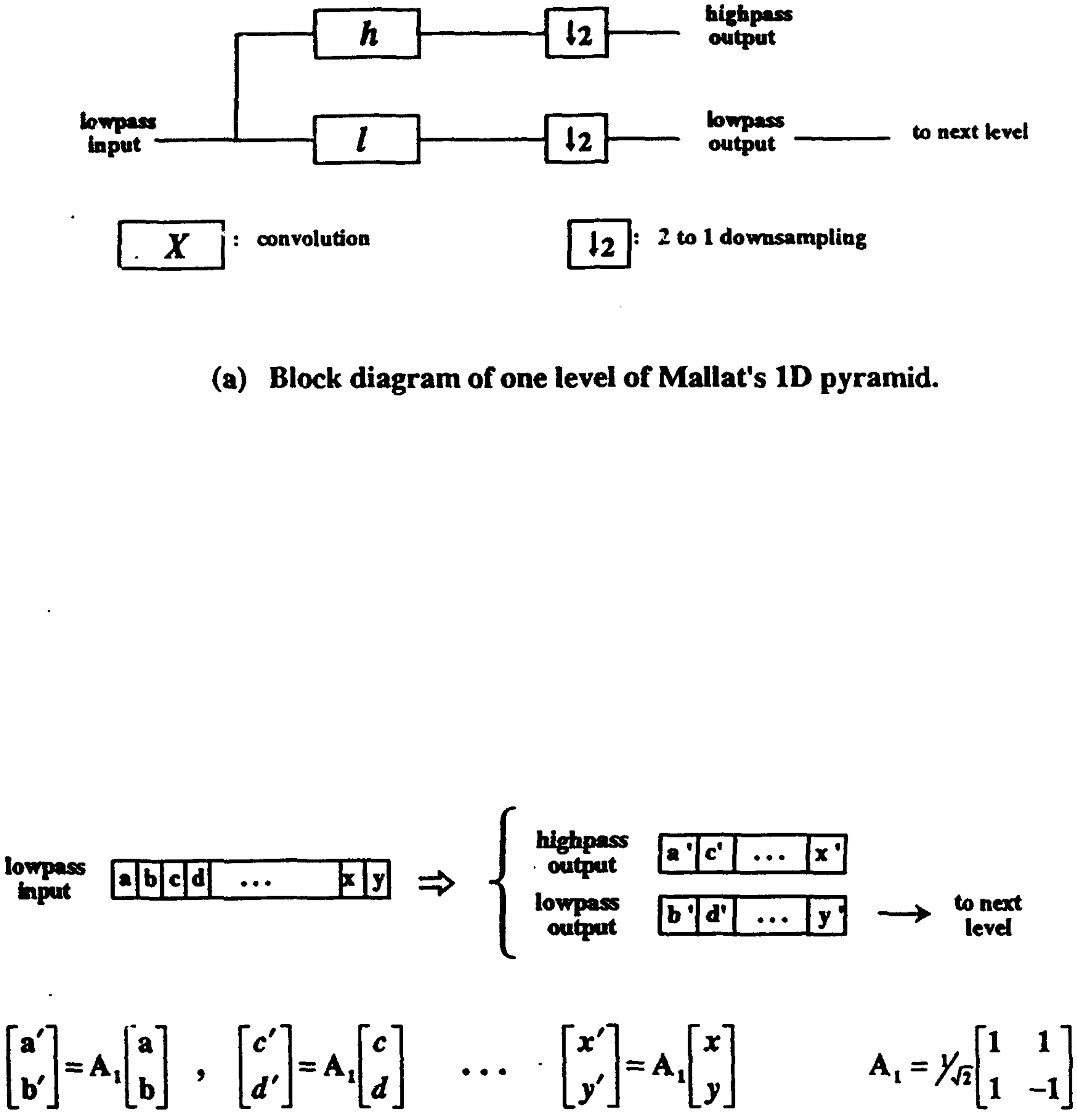
Haar Wavelet and Multi-Resolution Decomposition:

The Haar wavelet is well-known as being the orthogonal wavelet of the smallest support. Its limitation is that it can only approximate piecewise constant functions without error. However, in this case, the silhouettes can be considered as combinations of piecewise constant binary waveforms so the Haar wavelet is appropriate to use for decomposition.

The filters associated with the Haar wavelet and its corresponding scaling function are given respectively by $h = \frac{1}{\sqrt{2}}[1 \quad -1]$ and $l = \frac{1}{\sqrt{2}}[1 \quad 1]$. These represent the coefficients of two discrete FIR half-band filters, highpass and lowpass respectively. When an input signal is repeatedly convolved with these filters, the collection of output signals forms a collection of multi-resolution decompositions. Among these decompositions is the one described in Mallat's pyramid algorithm, where only the lowpass component at each level is further decomposed (see Figure 11(a)) [MALL89b, STRA89]. Since the filters are half-band, 2:1 down-sampling can be used at each stage to half the number of pixels in each output without losing signal information. With each successive level, the pixel-to-pixel resolution decreases by a factor of 2, thus creating the multiple resolutions.

Moving between levels on the pyramid may also be described by a single matrix multiplication per pair of pixels. This matrix operator is given by $A_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, the rows are just l and h. This matrix approach is shown in Figure 11(b). Note that $A_1^2 = I$ or $A_1^{-1} = A_1$. This means that the same operation is performed to go either up or down a level on the pyramid.

The algorithm can be extended to operate on higher dimensional input signals such as 2D and 3D binary images. This extension can be realized by considering the tensor products $l \otimes l, l \otimes h, h \otimes l$, and $h \otimes h$ for the 2D case and $l \otimes l \otimes l, l \otimes l \otimes h, l \otimes h \otimes l, l \otimes h \otimes h, h \otimes l \otimes l, h \otimes l \otimes h, h \otimes h \otimes l,$


h
↓2
highpass output
lowpass input
l
↓2
lowpass output
to next level
X : convolution
↓2 : 2 to 1 downsampling
lowpass input
a b c d ... x y
⇒
highpass output
a' c' ... x'
lowpass output
b' d' ... y'
to next level
$\begin{bmatrix} a' \\ b' \end{bmatrix} = A_1 \begin{bmatrix} a \\ b \end{bmatrix}$, $\begin{bmatrix} c' \\ d' \end{bmatrix} = A_1 \begin{bmatrix} c \\ d \end{bmatrix}$ ... $\begin{bmatrix} x' \\ y' \end{bmatrix} = A_1 \begin{bmatrix} x \\ y \end{bmatrix}$
$A_1 = 1/\sqrt{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

(a) Block diagram of one level of Mallat's 1D pyramid.

(b) Matrix representation of one level of Mallat's 1D pyramid.

Figure 11: Mallat's 1D pyramidal decomposition.

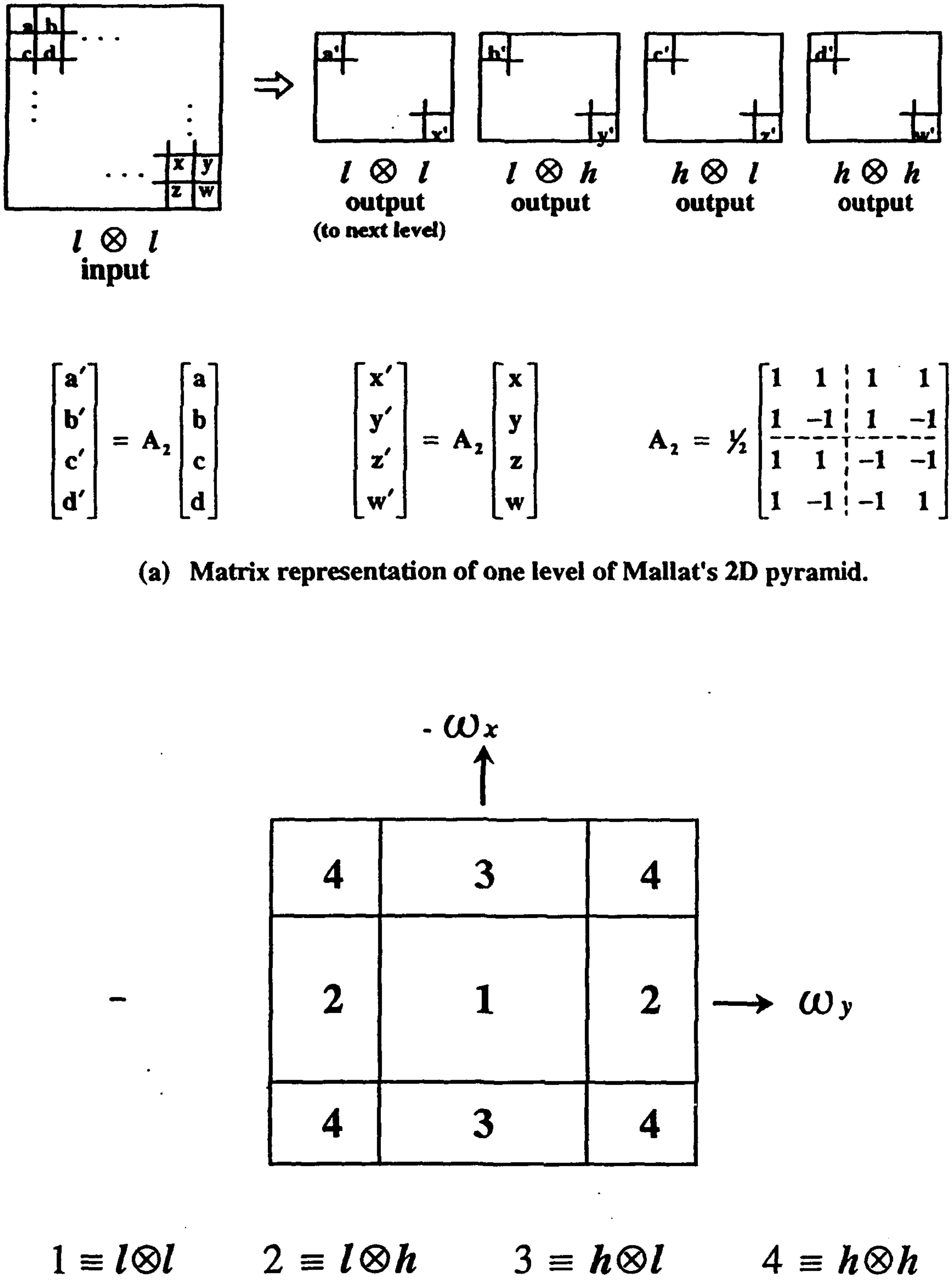
and $h \otimes h \otimes h$ for the 3D case. For example, $h \otimes l = \frac{1}{2} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$ and $h \otimes l \otimes h = (h \otimes l) \otimes h = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} = \frac{1}{2\sqrt{2}} \left\{ \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \right\}$. These matrices can also be thought of as the coefficients of 2D and 3D half-band FIR filters.

In the 2D case, the tensor products operate on 2 by 2 submatrices of the input image. The outputs of each stage of decomposition are 4 matrices which are half the size of the input matrix in each dimension. This follows since the down-sampling is now done in both the row and column dimensions. Again, the total storage is the same. As in the 1D case, moving between pyramid levels can be achieved by a matrix multiplication. Consider the following matrix

$$A_2 = \frac{1}{2} \left[\begin{array}{ccccc} 1 & 1 & \vdots & 1 & 1 \\ 1 & -1 & \vdots & 1 & -1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & \vdots & -1 & -1 \\ 1 & -1 & \vdots & -1 & 1 \end{array} \right].$$

This matrix is constructed by converting the tensor product matrices into row vectors via $\begin{bmatrix} a & b \\ c & d \end{bmatrix} \Rightarrow \begin{bmatrix} a & b & c & d \end{bmatrix}$, where each row in A_2 corresponds to a different tensor product. Two things to observe about this matrix are: 1) $A_2^2 = I$, 2) A_2 has the same structure as A_1, $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. One level of decomposition is shown in Figure 12(a). The frequency domain decomposition is shown in Figure 12(b). The outputs of the 1D filters are interpreted to be lowpass and highpass versions of the input. A similar interpretation can be given to the outputs of the 2D case. Since l and h are lowpass and highpass, $l \otimes h$ would be lowpass in the first dimension and highpass in the second dimension. Let the row index of the matrix be the first dimension. Then $l \otimes h$ will be sensitive to edges that are oriented columnwise (or vertically). Likewise, $h \otimes l$ will be sensitive to horizontal edges. The $l \otimes l$ output can be considered as the average intensity in each sub-image. Each pixel in the output is twice the average value of the pixels in the input image that are used in the average. The factor of 2 comes from the $\frac{1}{\sqrt{2}}$ scale factor in the definition of l and h.

The $h \otimes h$ output has two possible interpretations. If the edges are all aligned either vertically or horizontally, then the $h \otimes h$ output matrix will just show the ends or corners of the edges. However, this output component will also be sensitive to diagonal edges, i.e., edges that are neither vertical nor horizontal. Figure 13 shows the decomposition of two simple silhouettes of a square and a diamond; the first is oriented along rows and columns, and the second is rotated



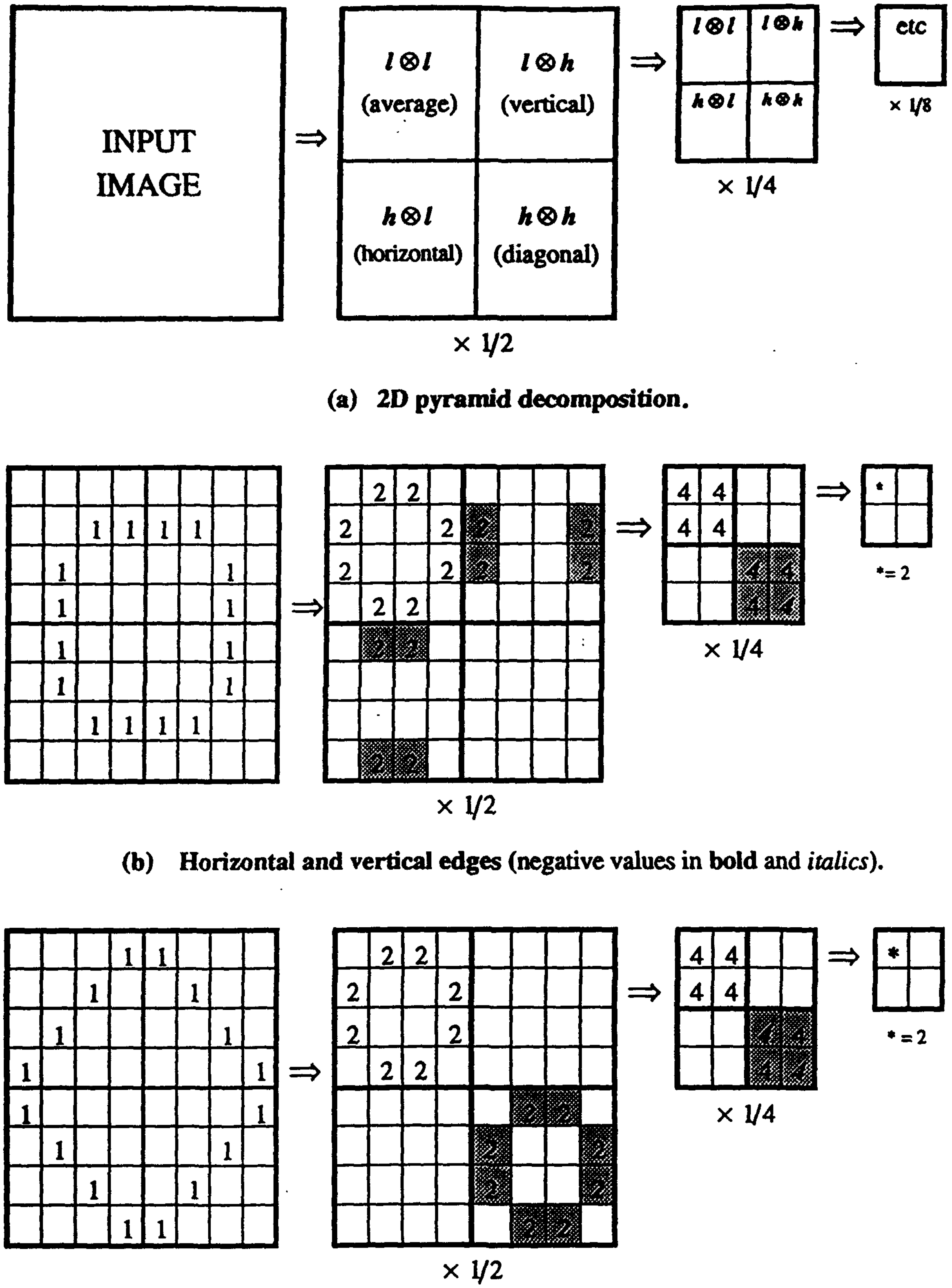
$$\begin{bmatrix} a' \\ b' \\ c' \\ d' \end{bmatrix} = A_2 \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = A_2 \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \qquad A_2 = \tfrac{1}{2}\left[\begin{array}{cc:cc} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \hdashline 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{array}\right]$$

(a) Matrix representation of one level of Mallat's 2D pyramid.

$1 \equiv l \otimes l \qquad 2 \equiv l \otimes h \qquad 3 \equiv h \otimes l \qquad 4 \equiv h \otimes h$

(b) Frequency domain decomposition of one level of 2D pyramid.

Figure 12: Mallat's 2D pyramidal decomposition.



(a) 2D pyramid decomposition.

(b) Horizontal and vertical edges (negative values in **bold** and *italics*).

(c) Diagonal edges (negative values in **bold** and *italics*).

Figure 13: Haar wavelet decomposition of simple silhouettes.

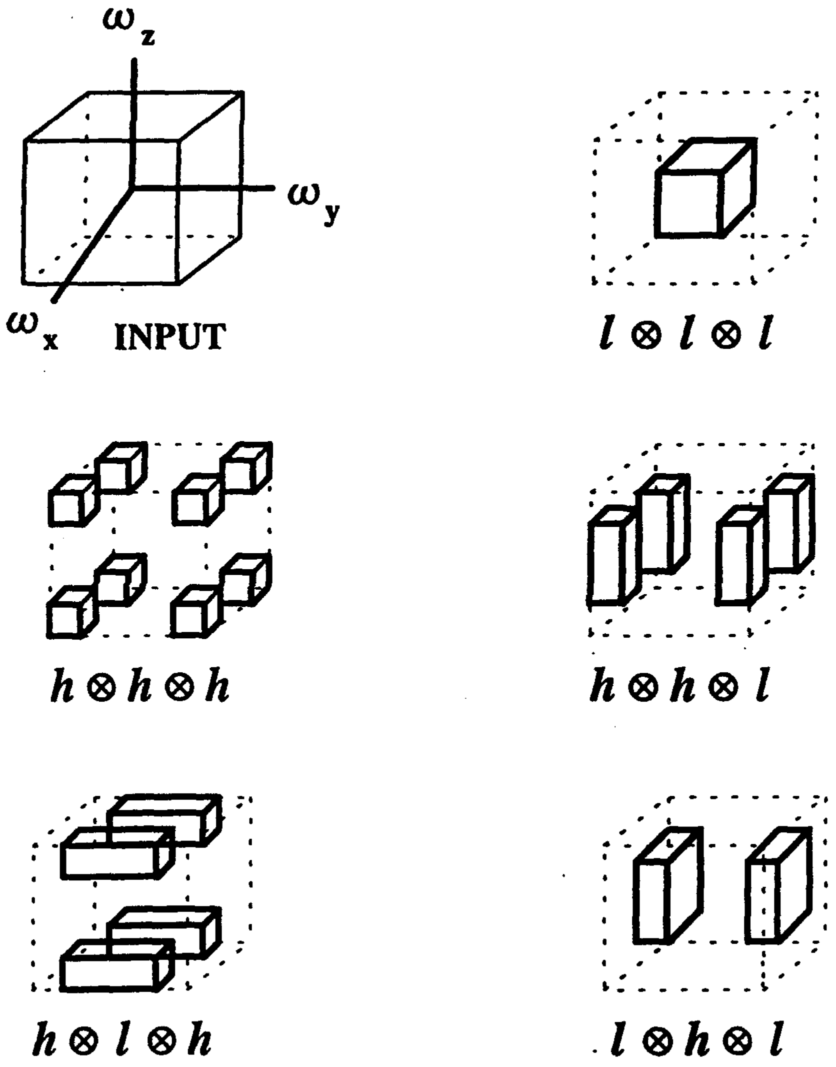
45 degrees.

In the 3D case, the tensor products operate on $2 \times 2 \times 2$ submatrices of the input matrix. There are now eight 3D output matrices, with each matrix dimension being one half of each input matrix dimension. So a $64 \times 16 \times 8$ input decomposes into eight 32×8 x 4 output matrices. Thus the total storage requirement is constant for all levels. Moving between levels on the 3D pyramid can also be described by multiplication by the following matrix:

$$A_3 = \frac{1}{2\sqrt{2}} \left[\begin{array}{cccccccccc} 1 & 1 & 1 & 1 & \vdots & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & \vdots & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & \vdots & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & \vdots & 1 & -1 & -1 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & 1 & \vdots & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & \vdots & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & \vdots & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & \vdots & -1 & 1 & 1 & -1 \end{array} \right].$$

Each row of this matrix is constructed by a 'flattening' of each tensor product $\left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \begin{bmatrix} e & f \\ g & h \end{bmatrix} \right\} \Rightarrow \begin{bmatrix} a & b & c & d & e & f & g & h \end{bmatrix}$. Again, $A_3^2 = I$, and it has the partitioned pattern $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. The 3D frequency domain decomposition is shown in Figure 14(a). Interpretation of each output matrix follows from the 2D case. For example, the $h \otimes l \otimes l$ output is a high-low-low filtered 3D image where the filter order is in correspondence with an x-y-z Cartesian system. Thus the $h \otimes l \otimes l$ output should be sensitive to sharp changes in the x dimension only, i.e., y-z planes. Accordingly, the $l \otimes h \otimes l$ output is sensitive to x-z planes and the $l \otimes l \otimes h$ output to x-y planes. The $l \otimes l \otimes l$ component indicates the average intensity at each previous level as in the 2D case.

If a 3D silhouette (meaning the surface of a 3D object) has edges which are parallel to the Cartesian axes, a rectangular parallelepiped for example, then the outputs which are generated from filters with only one lowpass component, namely, the $l \otimes h \otimes h, h \otimes l \otimes h$, and $h \otimes h \otimes l$ outputs, will correspond to edges in 3-dimensional space parallel to the lowpass filter axes. Refer to Figure 14(b) for the decomposition of a 3D cube. However, if the object has planes or edges which do not align with the Cartesian system, then the outputs with only one lowpass component will be sensitive to diagonal planes as shown in Figure 15.



INPUT

$l \otimes l \otimes l$

$h \otimes h \otimes h$

$h \otimes h \otimes l$

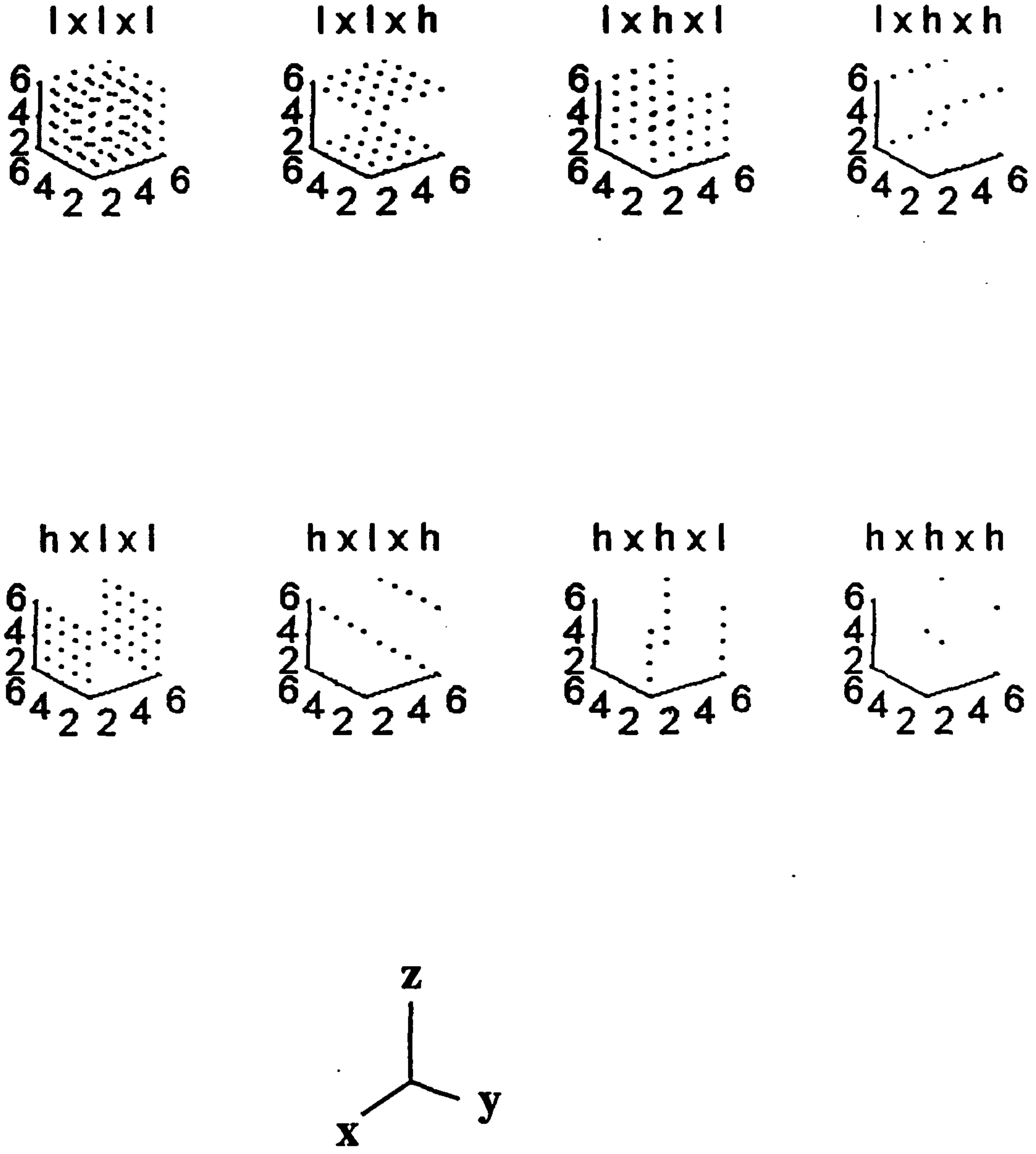
$h \otimes l \otimes h$

$l \otimes h \otimes l$

(a) Frequency domain decomposition of one level of 3D pyramid.

(only 5 of 8 outputs shown)

Figure 14: 3D pyramidal decomposition.



(b) Haar wavelet decomposition of a 3D cube (cube is oriented along the Cartesian axes).

Figure 14: (Continued).

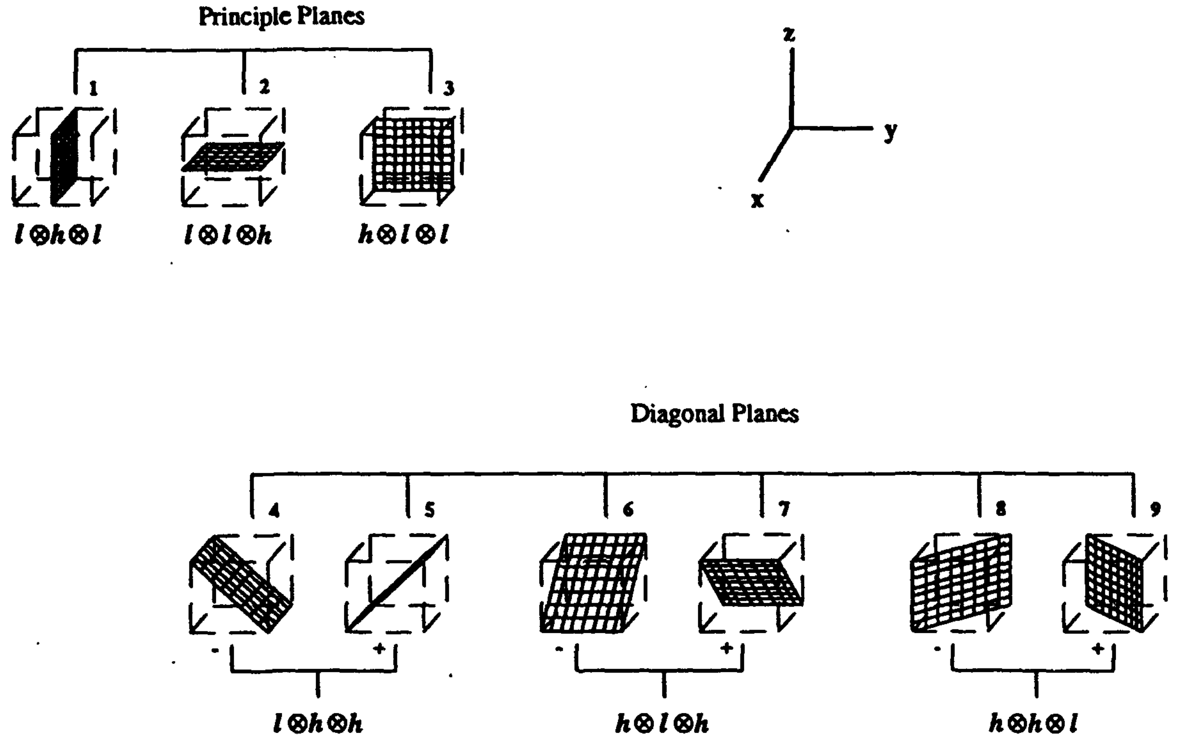


Figure 15: Possible 3D planar regions.

2D Silhouette Linearization:

Pixels in a 2D silhouette will be classified as belonging to one of four edge categories: vertical, horisontal, diagonal, or compound. Compound edges are made up of edges from a combination of the first three categories and consequently will be assigned more than one particular orientation. This labeling is done on each level of the multiresolution pyramid. Once the pixel labels have been combined for all resolutions, those with identical labels can be grouped together.

The segmentation algorithm is broken into four stages; 1) pixel thresholding, 2) pixel labeling, 3) pixel label group reduction, 4) silhouette segmentation. These stages are described below.

1. For each resolution level and each non-zero pixel value, p_y, at location y in the $l \otimes l$ image, find the ratio of amplitudes of each corresponding pixel p_y^1, p_y^2, p_y^3 in the $l \otimes h, h \otimes l, h \otimes h$ images to p_y. If $\frac{|p_y^k|}{|p_y|}$ > threshold, for any k=1,2,3, then save the value p_y^k in its respective sub-array, else set this pixel to zero. The value of the threshold is chosen to be $\frac{1}{2}$ for all resolutions.

2. At each resolution and for each non-zero pixel in the $l \otimes h, h \otimes l, h \otimes h$ sub-images, multiply those pixels in the original silhouette which are elements of the sub-image support of the non-zero pixel by one of four possible prime numbers 2,3,5,7. These primes correspond to the vertical, horizontal, negative diagonal, and positive diagonal labels, which in turn correspond to a non-zero $l \otimes h$ pixel value, a non-zero $h \otimes l$ pixel value, a negative $h \otimes h$ pixel value, and a positive $h \otimes h$ pixel value. After all resolutions are considered, decompose each silhouette pixel value into its product of primes $2^{x_1}3^{x_2}5^{x_3}7^{x_4}$. This is a unique decomposition from the prime factorization theorem. The set of prime exponents $\{x_i\}$, i=1,...,4, represents the number of times a pixel is assigned each orientation.

3. Several options are possible at this point to try to reduce the total number of segmented pixel groups. Typically, the prime exponent set elements are converted to a binary form by some thresholding operation. Then various combinations of morphological set erosion and dilation between these sets is performed.

4. Group the silhouette pixels by creating a new set of 2D coordinates. A new 2D coordinate is created at the midpoint of each pair of adjacent silhouette pixels that do not have identical $\{x_i\}$ values. Remove from the set those coordinates which lie less than one pixel from their nearest neighbors (these coordinates are part of the same segment boundary).

The above linearization algorithm is applied to two simple silhouettes, a square and a diamond, shown in Figure 13. The pixels that pass the labeling threshold are shaded. In Figure

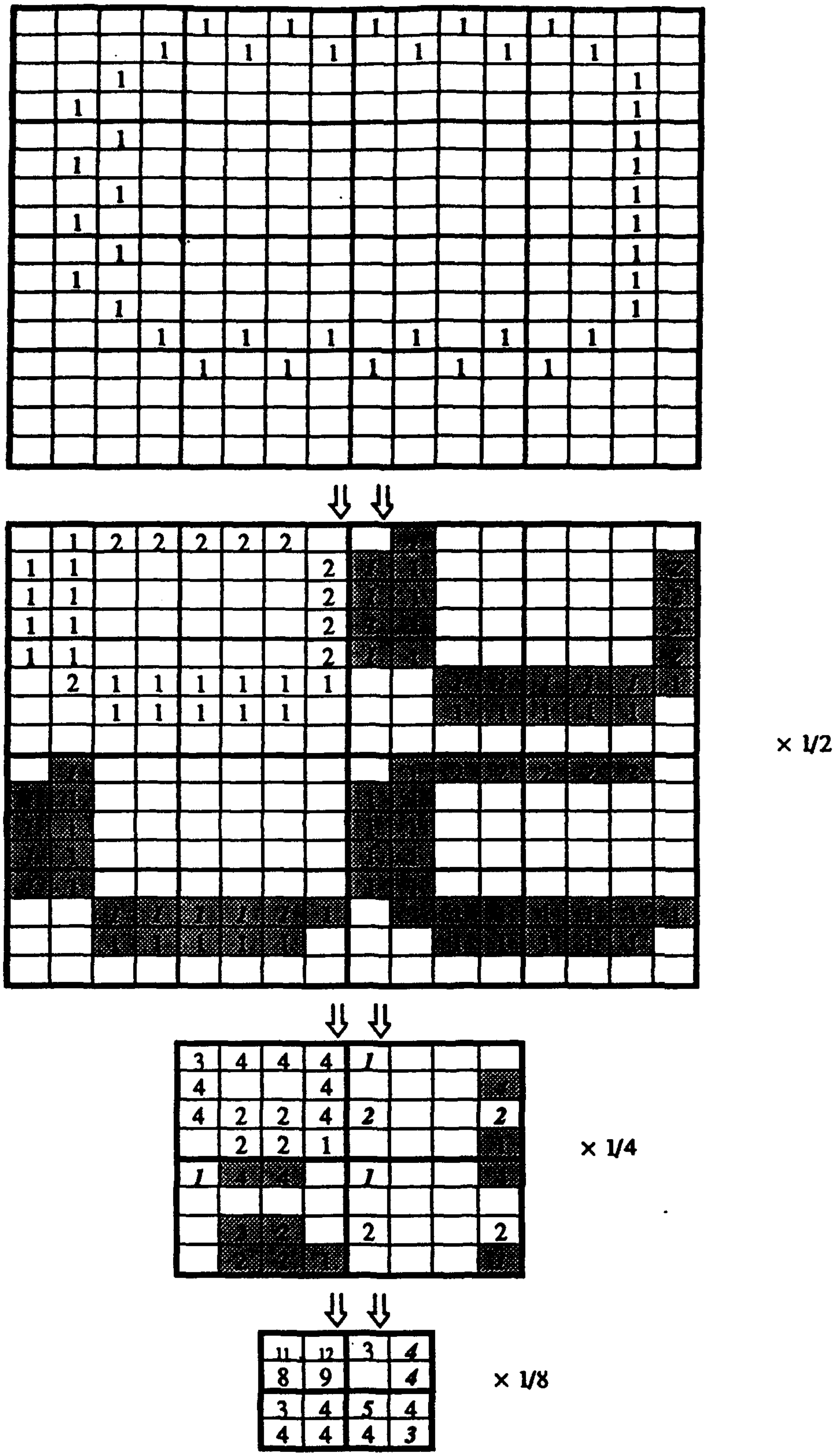
13(b), from left to right, the second picture (or bottom of the pyramid) shows vertically and horizontally labeled pixels, and the third picture, diagonally labeled pixels. To understand this, it is necessary to view the original image as groups of 2×2 and 4×4 submatrices. If grouped into 2×2 arrays, there are some groups in which the pixels are arranged in a purely vertical or horizontal fashion. In the 4×4 groupings, the pixels are arranged somewhat diagonally. In Figure 13(c), diagonal classifications are made on both levels of the pyramid since this silhouette has diagonally oriented 2×2 and 4×4 groupings. Note that both the square and the diamond are identical at the second pyramid level.

However, the classification process (step 1 in the algorithm) is highly dependent on the location of the input edges with respect to the inherent partitioning performed at each multiresolution level. Figure 16 shows the decomposition of two silhouettes. The first (Figure 16(a)) is similar to the square in Figure 13(b), and the second (Figure 16(b)) is similar to the diamond in Figure 13(c). In Figure 16(a), the right edge of the square is classified as vertical on the bottom two levels, and the top edge is labeled as diagonal on the bottom level, and as horizontal on the next level. These classifications are expected when we view the image by 2×2 and 4×4 groupings. However, the bottom and left edges do not have a sufficient number of pixels in the $l \otimes l$ bottom level, 2×2 windows. Only one pixel out of four is non-zero and it is not possible to associate orientation with only one pixel. Hence this pixel has compound orientation and is labeled in all bottom level subimages. The left edge is not labeled on the second pyramid level because there is equal amplitude in the $l \otimes l$ level 1, 2×2 window at the center of the left edge. At this level, in this window, there is no edge. Also the bottom edge is split. If the silhouette were shifted in the image one position to the right and down, with respect to the 2×2 windowing, then the left and bottom edges would have better segmentation, but the top edge would then be split. Figure 16(b) shows a similar situation happening with the diamond.

The linearization algorithm was applied to the silhouettes in Figures 13(b), (c), 16(a), and (b). Figure 17 displays the resulting dominant points (the original silhouette points are labeled by '.'s and the dominant point by 'O's).

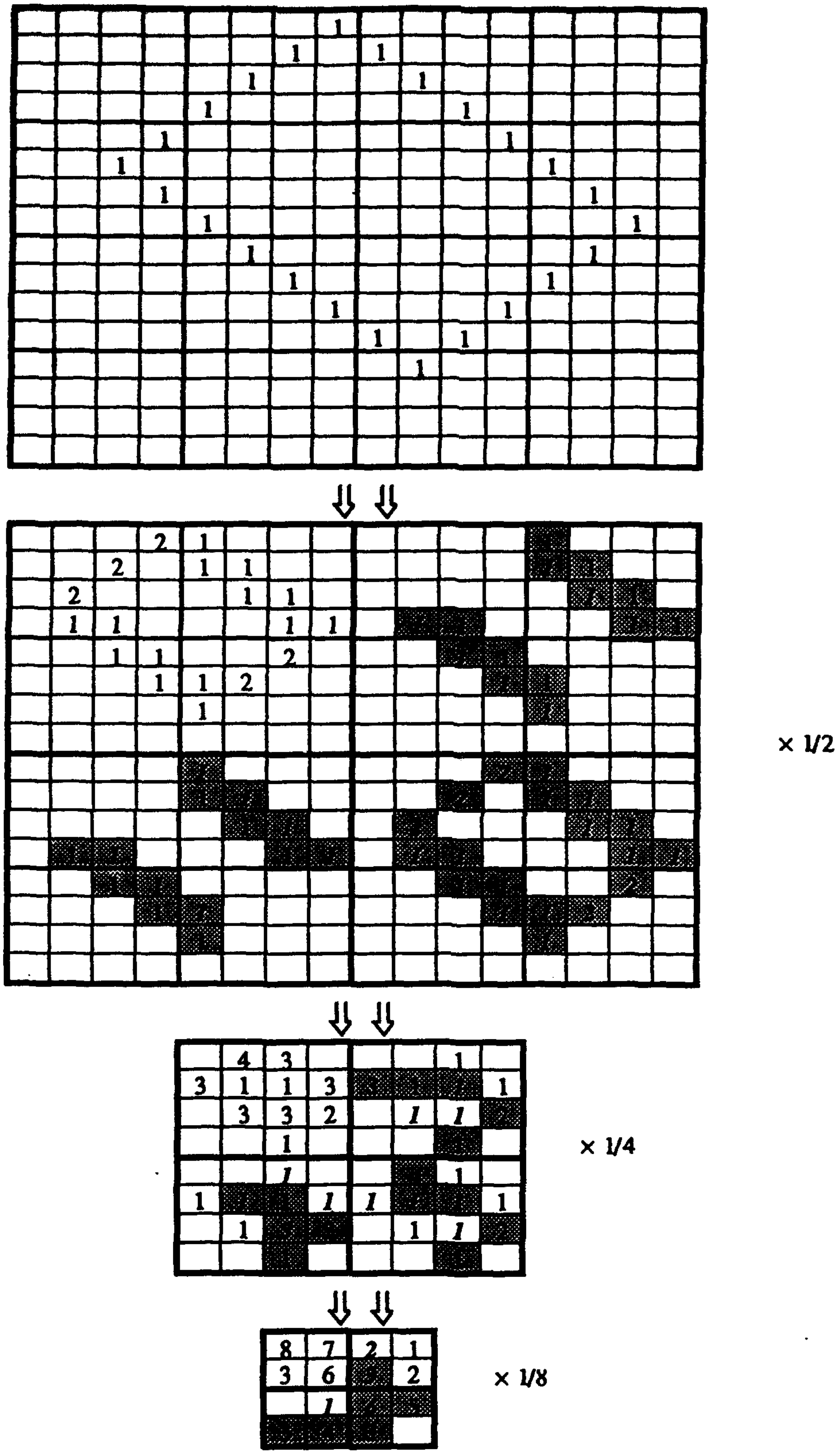
3D Surface Linearization:

The linearization of 3D objects is performed analogously to the classification of the 2D objects, except that now there are six different outputs to compare for each non-zero pixel value in the $l \otimes l \otimes l$ image (the $h \otimes h \otimes h$ sub-images are not used.). The six sub-images will be sensitive to nine different orientation planes (see Figure 15). Figure 18 shows the labeled pixels on the bottom level of the pyramid for the cube shown in Figure 14(b).



(a) **Decomposition of variant of square.**

Figure 16: Three-level decompositions of a square and a diamond.



(b) Decomposition of variant of diamond.

Figure 16: (Continued).

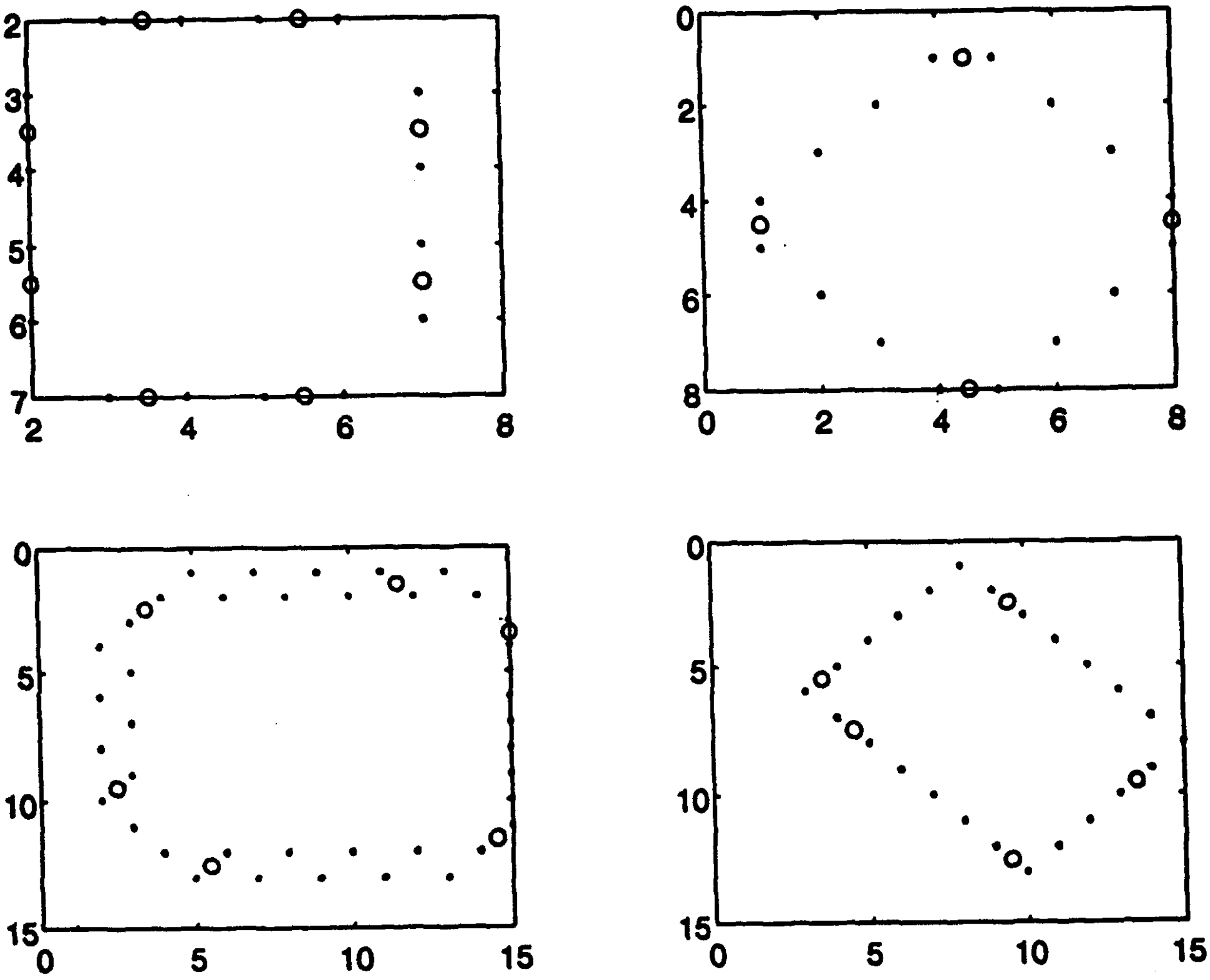
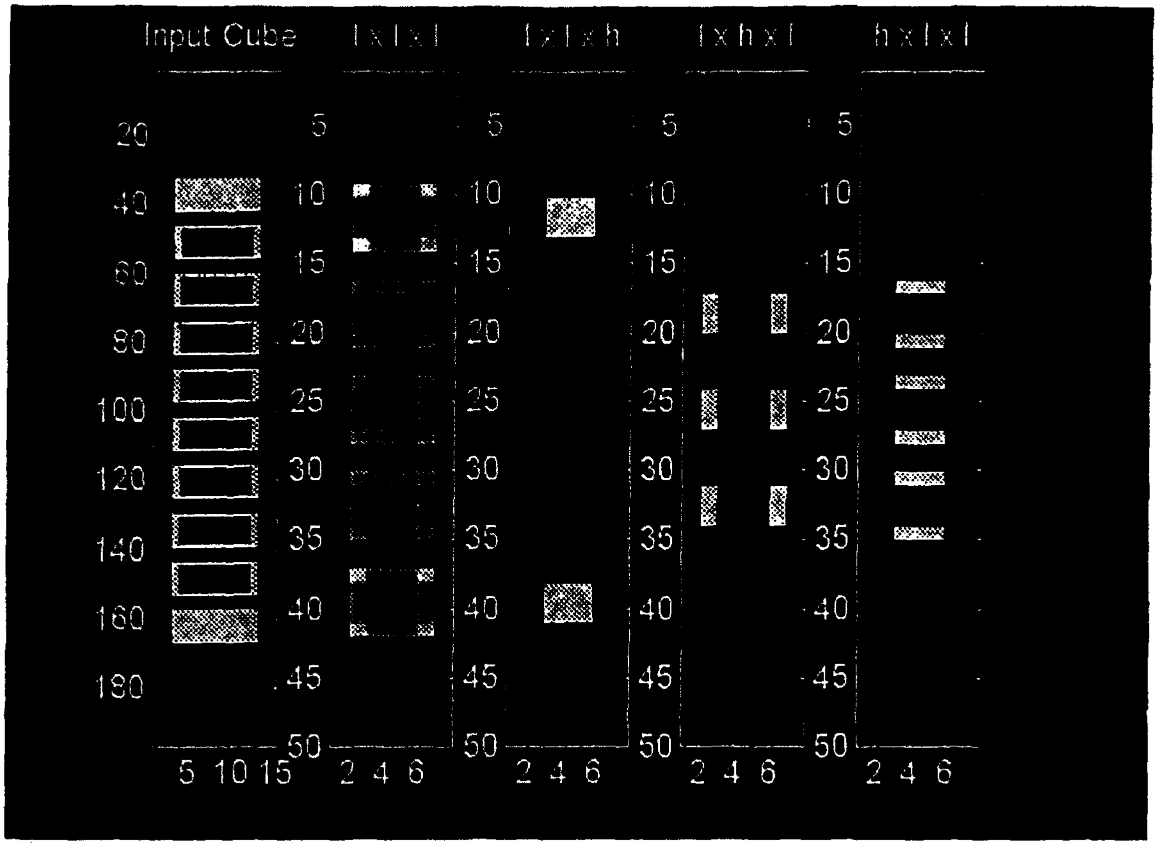


Figure 17: Dominant points of simple silhouettes found by using Haar wavelet.



First stage_decomposition of cube (after thresholding). The 3D object is presented in 2D fashion by slicing up the 'z' dimension. Only those outputs that contain labeled pixels are shown (3 of a possible 7). The last three images contain pixel magnitudes.

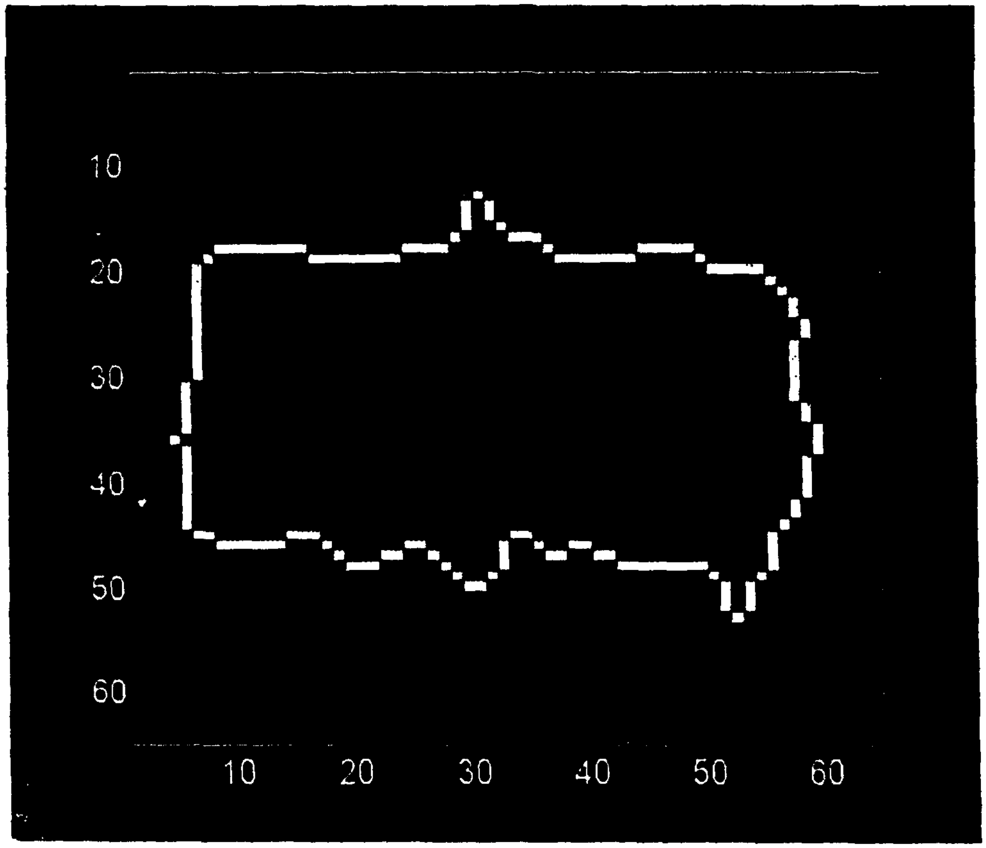
Figure 18: First stage decomposition of a cube.

Classification Example Using Laser Radar Data:

The above linearization scheme has been experimented on silhouettes of a military vehicle for possible applications to automatic target recognition. The data have been provided by the LADAR sensor. The sensor simultaneously collects laser radar ambiguous fine range, amplitude, and velocity information as well as infrared (IR) data. Figure 19(a) shows a silhouette of an armored personnel carrier (APC) extracted from the "fused" laser radar data. The dominant points, as determined by the linearization algorithm, are labeled by 'O's in Figure 19(b). The algorithm greatly reduces the number of pixels that need to be saved, while preserving the general shape of the APC. However, there are some dominant points that are missed, at the top center and bottom right. This is due to the partitioning problem mentioned previously.

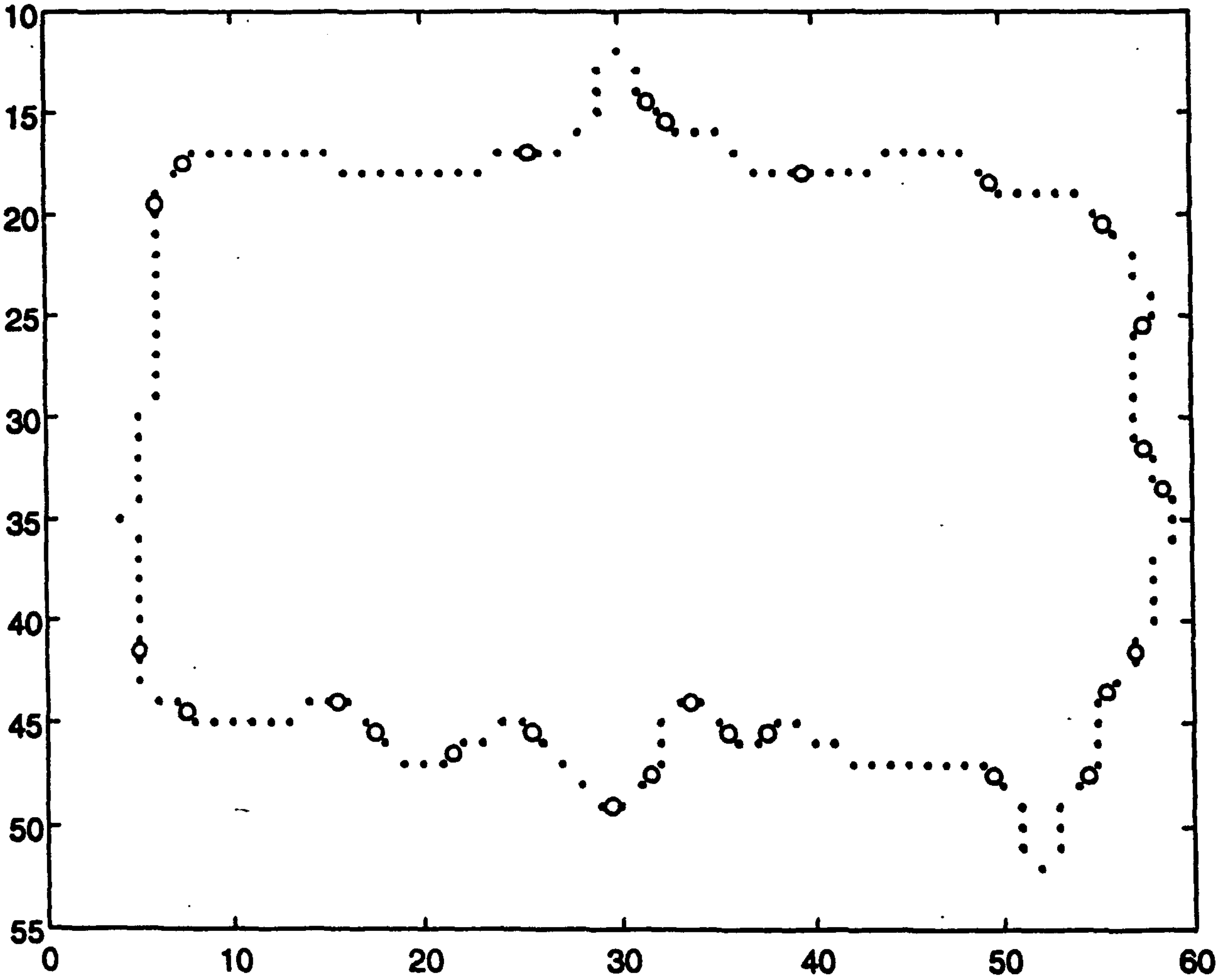
Future Investigation:

What has been described here is a rudimentary basis to support the idea of using Haar wavelets to assign different orientations to groups of pixels at multiple resolutions. These assignments can be used to group pixels of common orientation together to reduce the number of pixels at each resolution, hence reducing the number of pixels in the reconstructed silhouette. Comparisons have yet to be made with other methods of reducing 2D and 3D silhouettes [TEHC89, LEES93].



(a) 2D APC silhouette at an angle of 25 degrees.

Figure 19: A 2D silhouette of an armored personnel carrier(APC) and the determined dominant points.



(b) Dominant points of 2D APC silhouette (at 25 deg).

Figure 19: (Continued).

B.1.e Image Compression by Variable Rate Vector Quantization of Wavelet Transform Coefficients

The efficient multiresolution signal representation by wavelet transforms lends itself naturally to a very attractive new approach to image compression [DEVO92, WICK92b]. The advantage of the wavelet decomposition approach over the conventional subband coding is perhaps due to the property of vanishing moments of wavelets and regularity condition of wavelet filters. Orthogonal or biorthogonal wavelets are used in wavelet decompositions to suppress redundancy [ZETT90, LEWI92, ANTO92, HART93]. Wavelet transform coefficients are then quantized for encoding. Coding may also be based on edges detected by wavelet transform modulus maxima [MAZH91, MAZH92], and the error image resulted from reconstruction may be encoded with an orthogonal wavelet transform [FRMA92]. Wavelet packets provide adapted subband bases from which the best-basis can be chosen by minimizing an information cost function for the adapted subband coding [WICK92a, WICK92b, COWI92, LICH93]. Wavelet transform coefficients may be quantized with scalar quantization or a group of coefficients with an optimal bit allocation. Vector quantization has been used by Antonini, et al. for biorthogonal wavelet transforms [ANTO92] and by Liu, et al. for spline wavelet packets [LICH93]. All of these works have shown very high compression ratio with good reconstruction quality.

We have been interested in exploring applications to medical image compression where the quality of the decompressed images is of paramount importance. For archiving purpose, the information preserving compression is often required; the achievable compression ratio is currently limited to about 3:1 [LIGO91]. For communication purpose, however, some information loss may be accepted as long as the decompressed image is visually indistinguishable from its original by a panel of expert radiologists. As quantization of wavelet transform coefficients constitutes a lossy scheme, it is of special interest to examine how much compression can be obtained while maintaining high fidelity of the decompressed images. This section presents our study on image compression using the variable rate vector quantization of the Daubechies' orthogonal wavelet transform.

Variable Rate Vector Quantization:

Vector quantization(VQ) is an extension of the scalar quantization to the multi-dimensional case. A block of pixels, for example, 4×4 pixels, forming a vector of $k(=16)$ dimensions are quantized together to one of N reproduction vectors, so that any possible correlation existing among neighboring pixels may be better exploited than by quantizing each pixel individually. A vector quantizer Q performs a mapping from a vector x in the $k-$dimensional space R^k into a finite set C containing N points called codewords where $C = \{y_1, y_2, \ldots, y_N\}$ and $y_i \in R^k$ for all $i = 1, 2, \ldots, N$. R^k is partitioned into N regions or cells, $R_i, (i = 1, 2, \ldots, N)$, where

$R_i = \{x \in R^k | Q(x) = y_i\}$. The set C is termed the codebook. The rate of a vector quantizer is $r = (log_2 N)/k$ which gives a measure of the number of bits per vector component used to represent an input vector. A codebook is designed for its optimal representation by training through a set of images. It is optimal for a given bit rate if it minimizes the expected error or distortion $D = E\{d(x, y_i)\}$ between a source sequence and its representation. A widely used distortion measure is the mean square error. The code vector y_i associated with the partition region R_i may be determined as the centroid of the region R_i. The encoding is to select an appropriately matched code vector y_i to represent an input vector x; the index i of the code vector is stored or transmitted. The decoder performs a table-lookup procedure and generates the reproduction vector y_i which is the quantized approximation of the original input vector.

A popular approach to vector quantization is called the tree structured vector quantization where a sequence of binary search is performed for each input vector to determine the minimum distortion between the input vector and one of N codewords in the codebook (corresponding to N leaves in the binary tree). A binary tree search starts from the root of the tree. If the input vector is closer to the left child, it transmits a 0 and descends to the left child. If it is closer to the right child, it transmits a 1 and descends to the right child. Repeat the process with the selected child node and continue until a leaf node is reached. Then the next input vector is taken, and the process is repeated until the entire vector sequence is encoded. For a full balanced binary tree, all the leaves lie at the same depth, L, where $N = 2^L$, and an equal number of bits are assigned to each reproduction vector. This is called the fixed rate vector quantization, and the rate is L bits/vector.

The variable rate vector quantization is a generalization of the fixed rate vector quantization by assigning more bits to more active regions, such as edge regions, in an image and fewer bits to less active regions, such as regions with uniform background. Hence, with the same average bit rate, it is possible to reproduce better image quality by saving bits from the less active regions and using them to encode in more detail the highly active regions. In this case, the binary tree is unbalanced and has a varying depth for different leaves as illustrated in Figure 20. To design a variable rate vector quantizer, an unbalanced tree may be obtained by pruning a large fixed rate balanced tree, or it may be grown by splitting into two nodes the node which contributes the most distortion at each step. The former requires a very large initial tree such that the pruned subtree may result in low distortion at a given average rate. A large initial tree is expensive to generate, it also requires a large amount of training data to avoid the empty cell problem at the bottom of the tree. The latter can avoid the empty cell problem, and an unbalanced tree is grown as one trades off the average depth of the tree for the average distortion [RIGR91]. The tree is grown by splitting one node at a time. A good split is to achieve a large decrement in node impurity or distortion. Let $i(t)$ denote the impurity measure at node t, and s denote the binary test, i.e., splitting node t into two nodes: left node t_L and right node i_R. The decrement

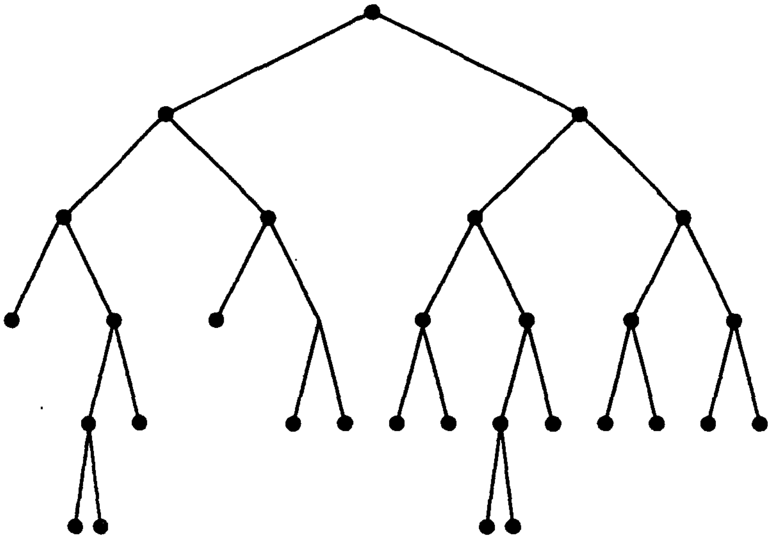


Figure 20: A tree stuctured vector quantizer with varying depth.

in node impurity after the split is given by

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R)$$

where p_L is the proportion of the training samples in node t that goes to the left and p_R is the proportion that goes to the right. p_L and p_R can be computed by $p_L = p(t_L)/p(t)$ and $p_R = p(t_R)/p(t)$ where $p(t), p(t_L)$ and $p(t_R)$ are the probabilities of training samples being in nodes t, t_L and t_R respectively, and $p(t) = p(t_L) + p(t_R)$. $\Delta i(s,t)$ represents, in effect, the ratio of the change in distortion to the change in rate(depth). When growing a tree, we split the with the largest $\Delta i(s,t)$ to get the maximum decrease in distortion for an increase in rate(dep Thus, the variable rate tree growing for codebook design can be summarized by the following algorithm:

1. Generate an initial node by calculating the centroid of the training sequence.

2. Split the initial node by using the Linde-Buzo-Gray algorithm [LIND80] to generate a left node(t_L) and a right node(t_R).

3. Design left(t'_L) and right(t'_R) nodes from a previous node t, i.e., consider the left and right children from its parent.

4. Calculate $\Delta i(s,t)$ for all children nodes.

5. Split the particular t with the largest $\Delta i(s,t)$.

6. Repeat steps 3-5 until the desired average rate is achieved.

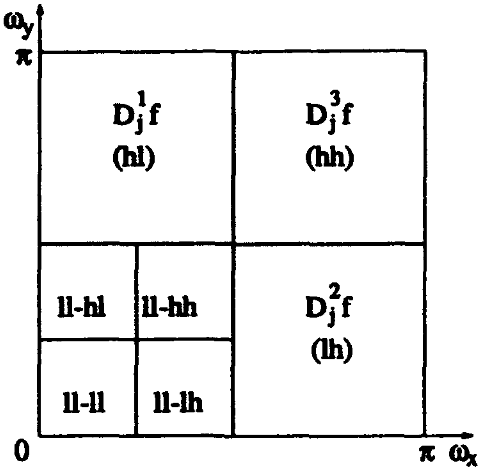


Figure 21: Wavelet decomposition in frequency domain after two levels of decomposition.

Variable Rate VQ of Orthogonal Wavelet Transforms:

Let us consider the variable rate vector quantization of wavelet transforms of digital images. Daubechies' compactly supported orthonormal wavelet transform is considered: orthogonality for reducing the redundancy, and compact support for yielding a fast algorithm. In terms of the tensor product of two 1-dimensional wavelet transforms applied first horizotally and then vertically, a given image is decomposed, at the next resolution, into one low-frequency subimage, ll, and three decorrelated high-frequency subimages, lh, hl and hh, corresponding respectively to the horizontal high frequency component, vertical high frequency component and the component of high frequencies in both directions. The ll subimage is further decomposed into three subimages, $ll - lh, ll - hl$ and $ll - hh$, and one coarse subimage $ll - ll$ at the next lower resolution, etc. as shown in Figure 21. Thus, for the wavelet pyramidal decomposition into J resolution levels, there will be a total of $3J$ orientationally detail subimages and one coarse subimage. One may generate $(3J+1)$ codebooks for encoding corresponding subimages. Since the error introduced in vector quantization at each resolution level will add up in the decompression and reconstruction stage, only a few levels of decomposition will be used in order to limit the error accumulation and to achive the high quality of reconstruction. In our experimental study described below, only two levels of decomposition were used. By pooling together vectors in subimages of the same orientation but at different resolutions, three orientational codebooks are generated through training for use in encoding high-frequency subimages of various resolution levels. They yield better decompressed images when compared with what can be obtained by the multi-resolution, multi-orientation codebooks; this is so because it is equivalent to introducing more training vectors resulting in a better codebook for each orientation. The coarse subimage

$ll - ll$ is scalar quantized in this study.

Experimental Results:

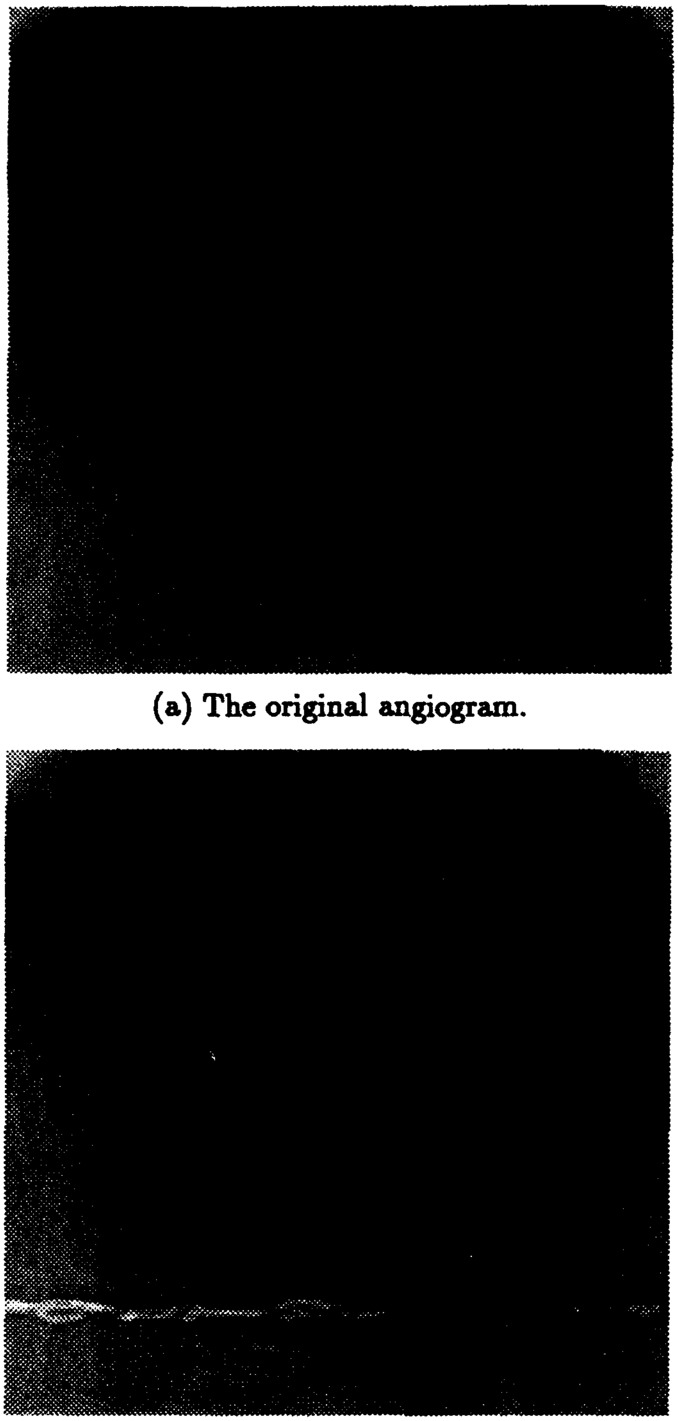
We have performed experiments on encoding and decoding two types of medical images, angiograms and MRI head images to examine the reconstruction quality and compression ratio achieved with the variable rate vector quantization of orthogonal wavelet transforms [MINC93]. Each image is of 256×256 pixels with 8-bit gray levels. Daubechies' orthogonal wavelet transform D_4 was used in image decompositions into two resolution levels, and 4×4 blocks were used in the vector quantization. For each type of images, three codebooks, one for each orientation (horizontal, vertical and diagonal), were designed for the desired bit rate. For angiograms, six images were used in training; and for MRI head images, five images were used in training. In each case, one test image was used. The overall quality of a decompressed image was measured by the signal-to-noise ratio(SNR) given by

$$SNR = 10 \log_{10} \frac{(\frac{1}{n})^2 \sum_i^n \sum_j^n (X_{i,j})^2}{(\frac{1}{n})^2 \sum_i^n \sum_j^n (X_{i,j} - \hat{X}_{i,j})^2}$$

The experimental results are summarized in Table 9. Figure 22(a) shows an original angiogram used in the test; at the encoding bit rate of 0.90 (compression ratio of 8.9:1), the reconstructed angiogram with SNR of 39.69 is shown in Figure 22(b). Figure 23(a) shows an original MRI head image; at the encoding bit rate of 0.91 (compression ratio of 8.8:1), the reconstructed MRI image with SNR of 25.71 is shown in Figure 23(b). Visual examination of both reconstructed images suggests a reconstruction of very high fidelity. The MRI head images have complex structures of the brain and, hence, are less compressible. Our preliminary experimental work shows similar or a slightly improved result in comparison to what has been reported by Riskin, et al [RISK90]. The reconstruction quality may be further improved by using a more regular wavelet and using more training images. The acceptability for medical applications must be ultimately determined by expert radiologists.

Table 9: Test results on Compression Ratio and Reconstruction SNR of an angiogram and a MRI head image.

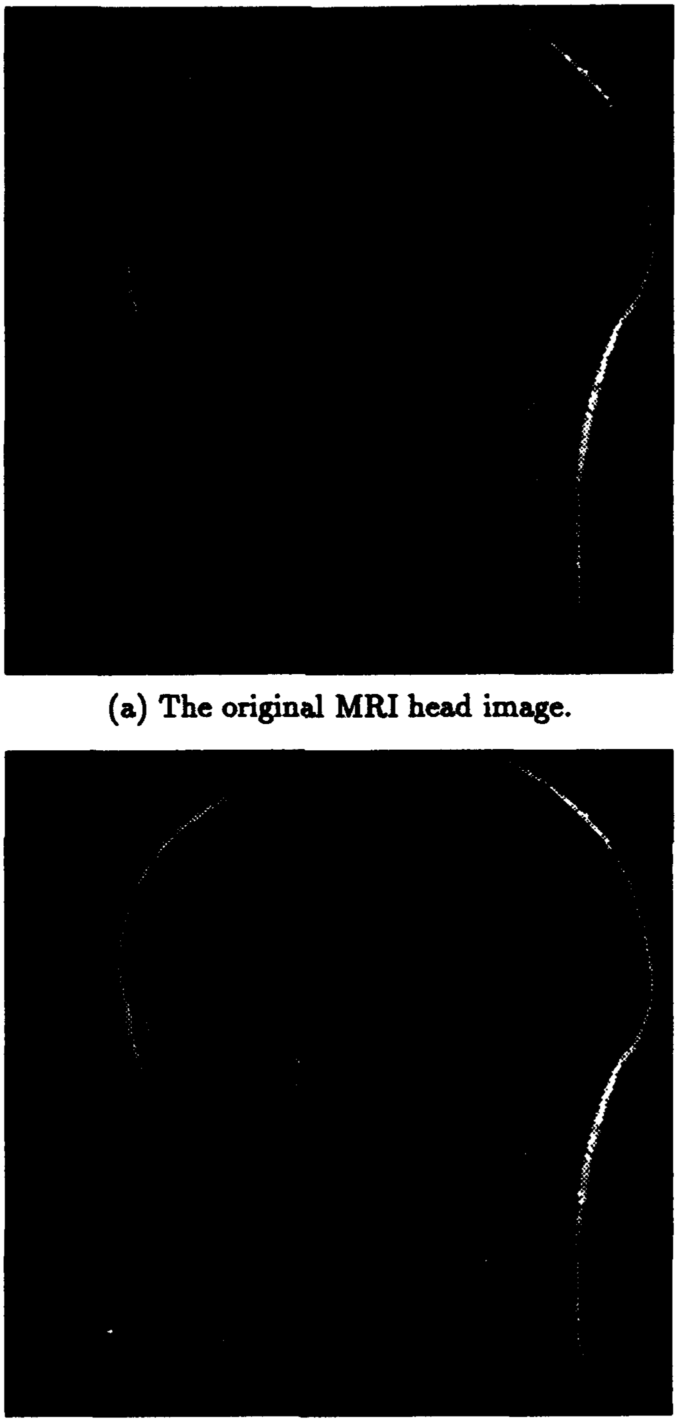
Angiogram			MRI Head Image		
Rate	CR	SNR	Rate	CR	SNR
1.31	6.1:1	42.93	1.39	5.8:1	28.84
0.9	8.9:1	39.69	0.91	8.8:1	25.71



(a) The original angiogram.

(b) The reconstructed angiogram.

Figure 22: A test angiogram and the reconstructed angiogram obtained at the encoding rate of 0.90bpp and SNR=39.69.



(a) The original MRI head image.

(b) The reconstructed MRI head.

Figure 23: A test MRI head image and the reconstructed image obtained at the encoding rate of 0.91bpp and SNR=25.71.

B.1.f Adaptive Image Halftoning based on Wavelet Transform

Image halftoning is a process to faithfully represent a multi-level picture on a bi-level physical device. In recent years, high resolution monochrome computer terminals and hard-copy devices such as laser printers have been developed, which physically reproduce an image in a binary fashion, but give a gray level impression. There is an increasing demand to improve the halftoning technique in publishing, telecommunication and computer industries [PAPP91]. Among various halftoning approaches, the most frequently used one is called ordered dithering which may be further classified into two classes. The first class uses dispersed-dots in a halftoned image, where the number of dots set-on is proportional to the gray value for an input image of constant gray value, and the dots are distributed spatially in the output such that the spectral energy of the output is as high as possible. The second class of ordered dithering uses clustered-dots, where the distribution of dots is designed in such a way that, at one half of the maximum gray value, the output forms a checkerboard pattern of alternating black and white regions [STOF81, ULIC87]. Digital electronic screening uses a screen function, which simulates an obscure physical plate with holes of pre-defined shape and spatial frequency, to the image to be rendered. The input image is first preprocessed (up-sampled) to the same dimension as that of the output. The gray scale image is then converted to an array of binary dots by comparing the normalized gray values at each location with the screen function which serves as a threshold array. For a fixed spatial frequency of the screen, which is determined by the physical display or printing device, the halftoning process is completely controlled by the screen function. The digital halftoning using this approach is implemented by considering an equilateral quadrangle of area $q^2/2$ inscribed inside a $q \times q$ square. The screen function is embedded in the quadrangle which serves as its support and whose elements are labeled by integers from 1 to $q^2/2$. Such a quadrangle is appropriately tiled to cover the whole image. Let a square mask of $q \times q$ sequentially scans through the entire up-sampled image. The output at any location covered by the mask is determined by the following thresholding process: if the normalized gray value at that point is larger than or equal to the labeled value of the corresponding point in the mask, value 1 is assigned to the output; otherwise, value 0 is assigned.

The screen function is designed on the basis of constant gray values, so as to display the average intensity at a local area as faithfully as possible. Presently, in many laser printers and monochrome display devices, a single screen function is used regardless any possible [illegible] variation of local image intensity. It is satisfactory to represent the low spatial frequency information in continuous tone images. It is less effective, however, for regions where large intensity variation is present, such as edge regions. Edge enhancement has been suggested by placing a stripe of black dots along the edge of the darker region together with a neighboring stripe of white dots around the lighter region [ANAS82]. Although edges are enhanced by using this approach, artifacts may be created in some regions of the image. We have developed a screen function which is adaptive to the image intensity variation [ZHAN93, SUZH93]. Wavelet transform is used for extracting information from the input image to control the screen function to be used at different locations.

We have utilized the wavelet of Mallat and Zhong [MALL89a, MAZH92] where $\phi(x)$ is a symmetric cubic spline with compact support $[-2,2]$ and $\psi(x)$ is also a cubic spline being antisymmetric with respect to $x = 0.25$. The corresponding filter coefficients $\{h(k)\}$, $\{g(k)\}$ and $\{l(k)\}$ are listed in Table 10. This wavelet $\psi(x)$ is the first derivative of a certain smoothing function $\theta(x)$. The discrete wavelet transform of a function $f(x)$ at the scale 2^j represents the sampled derivative of $f(x)$ smoothed by a set of filters where the last smoothing function is $\theta_{jn}(x) = 2^{-j/2}\theta(2^{-j}x - n)$. Let the discrete wavelet transform coefficients be computed without subsampling. Let image rows of the $(j-1)^{th}$ smoothed image be first filtered by the high-pass filter G, the columns of the resulting image be then filtered by L; and let this operation be designated as filtering by the separable filter LG to give a horizontally oriented high frequency component image W_1^j. Similarly, filtering by GL results in a vertically oriented high frequency component image W_2^j. At each pixel, W_1^j and W_2^j are combined giving a magnitude

$$W^j = [(W_1^j)^2 + (W_2^j)^2]^{\frac{1}{2}}$$

and an angle

$$\alpha^j = \arg(W^j) = \tan^{-1}\left(\frac{W_2^j}{W_1^j}\right)$$

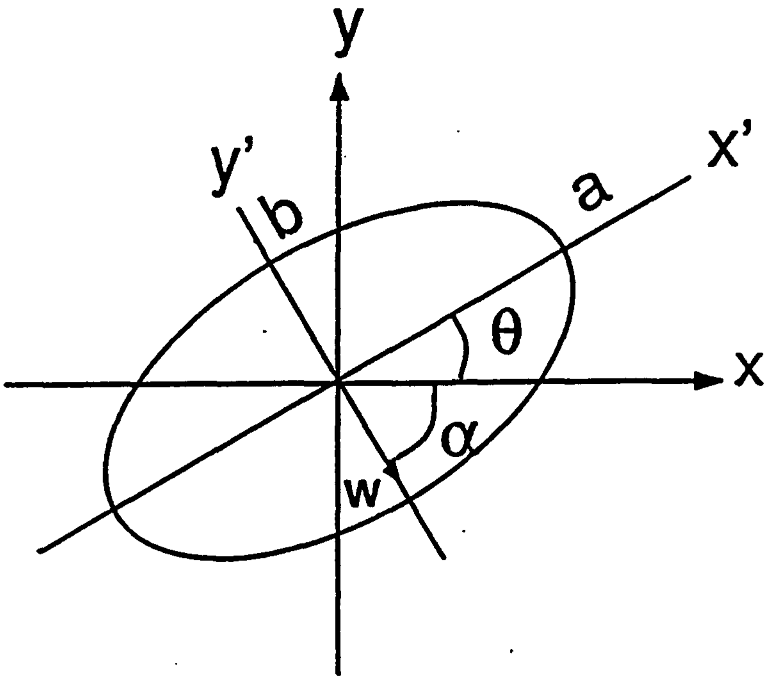
which represent the magnitude and direction of the local intensity variation in the image at the j^{th} scale. W^j and α^j provide an estimate of the gradient of the smoothed image at the j^{th} scale. They are used to control the screen function which will be discussed below. At edge pixels where W^j is maximum, the screen function will be made to adapt to the edge information. We chose $j = 2$ to ensure sufficient smoothing to reduce the noise effect in the gradient estimate. In the following discussions, we drop the subscription j so that W and α refer to W^2 and α^2 respectively.

Screen Function Design:

We follow the clustered-dot approach in the design of a screen function. The more concentrated the dots are, the better visual effect of the grayness will be. The gray information at a pixel, including gray value and its gradient, is represented by the number, density and distribution pattern of the set-on dots. Let us consider that the distribution pattern is an ellipse with minor axis b and major axis a at an orientation θ, as shown in Figure 24. Let the compression index of the ellipse be designated by $c = b/a$. The screen function is defined by the elliptical

Table 10: Filter Coefficients for Cubic Spline Wavelet

k	$h(k)$	$g(k)$	$l(k)$
-5	0	0	0
-4	0	0.00021	0.00003
-3	0	-0.00008	0.00727
-2	0.0625	-0.01643	0.03118
-1	0.25	-0.10873	0.06623
0	0.375	-0.59261	0.79113
1	0.25	0.59261	0.06623
2	0.062	0.10873	0.03118
3	0	0.01643	0.00727
4	0	0.00008	0.00003
5	0	-0.00021	0



$$c = b/a$$

Figure 24: An elliptic model for screen function whose orientation, compression and size parameters are adaptable to the wavelet transform magnitude, angle and pixel gray level respectively.

model within an equilateral quadrangular support. It is controlled by the estimated gradient which reflects the local intensity variation. The orientation of the ellipse is controlled by

$$\theta = \alpha + \frac{\pi}{2},$$

and the compression index c is controlled by the gradient magnitude such that it is more compressed for larger gradient. In a region with uniform gray value or very small gradient, c is made to approach to 1 so that the elliptic pattern becomes a circular pattern. On the other hand, at an edge pixel, the maximum W occurs, c approaches to 0 and the highly elongated ellipse tends to become a thin line segment depicting a sharp intensity variation across the edge. To relate the estimated gradient magnitude W to the compression parameter c, we must normalize W by a chosen maximum value W_{max} in the image, so as to set

$$c = 1 - w_n, \qquad 0 < c \leq 1$$

where $w_n = W/W_{max}$ is the normalized wavelet magnitude, $0 \leq w_n < 1$. The size of the ellipse (that is, the number of points in the elliptical region) is in direct proportion to the pixel gray value. In the limiting cases, either all points in the support region are set on giving the brightest impression, or none of the points is set on giving the darkest impression. In general, the screen function has larger elliptic patterns for higher gray levels. For pixels with higher gradient, it is adapted to become more compressed and oriented in a direction orthogonal to the gradient direction.

An ellipse centered at the origin is described by the following equation in the (x', y')-coordinate axes shown in Figure 24,

$$(\frac{x'}{a})^2 + (\frac{y'}{b})^2 = 1$$

where the major axis of the ellipse lies on the x'-axis and the minor axis lies on the y'-axis. Hence,

$$b^2 = \left(\frac{b}{a}\right)^2 x'^2 + y'^2 = c^2 x'^2 + y'^2$$

For a fixed value of c, a larger ellipse has larger values of b and a. All pixels on an ellipse of fixed orientation and fixed compression index may be considered as having the same quadratic "distance", denoted by $d = b^2$, to the center of the ellipse. A point on a larger ellipse (with larger b value) is more distant from the center than any point on a smaller ellipse (of the same orientation and same compression index). The rotational transformation between the (x, y)-coordinates and (x', y')-coordinates shown in Figure 24 is given by

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Then

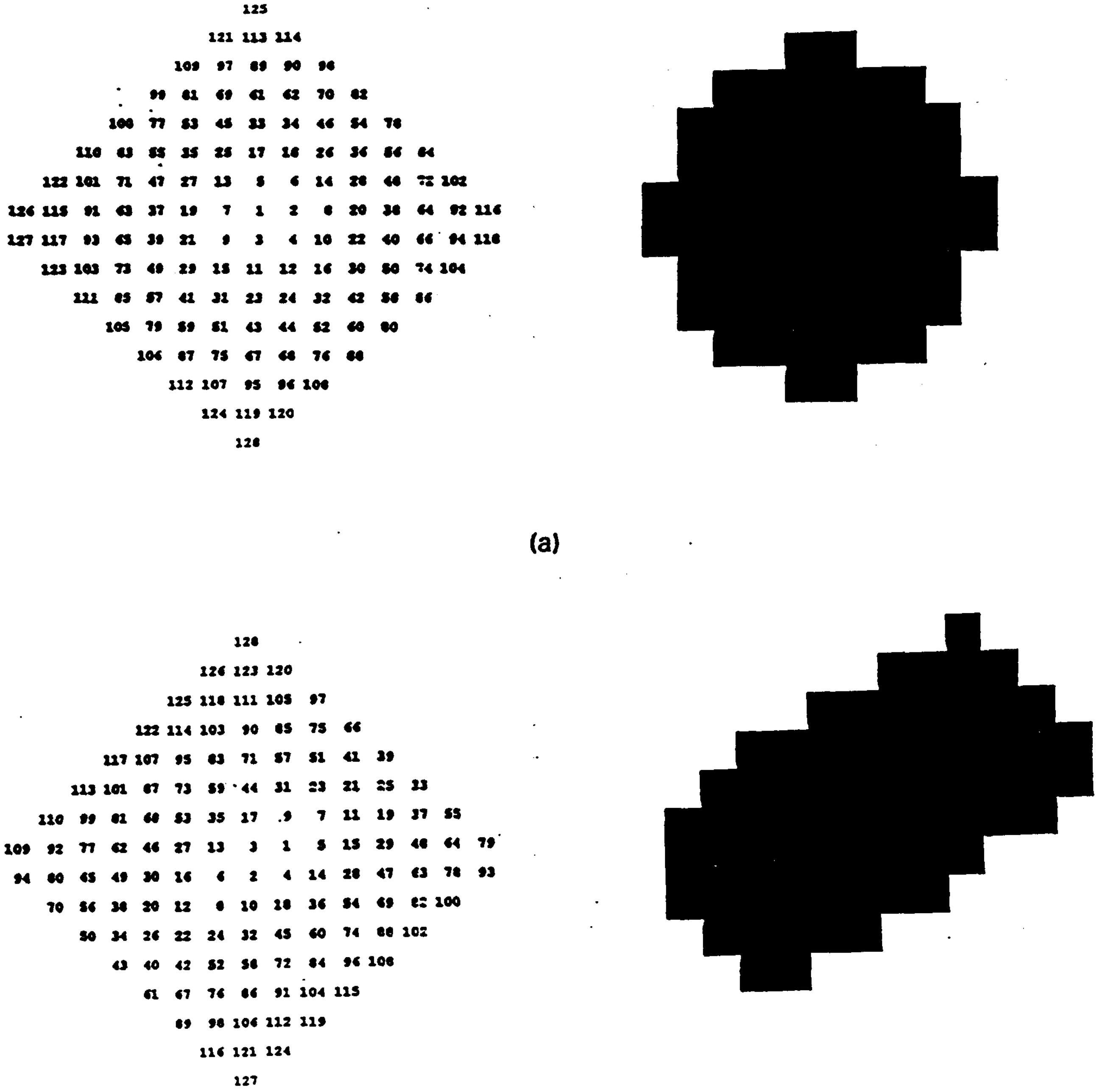
$$d = b^2 = (c^2 \cos^2\theta + \sin^2\theta)x^2 + (c^2 \sin^2\theta + \cos^2\theta)y^2 - 2(1 - c^2)(\sin\theta\cos\theta)xy$$

The elliptic screen function for a fixed orientation and a fixed compression is defined over an equilateral quadrangular support inside a $q \times q$ square matrix. Let q be chosen to be a power of 2, for example, $q = 8, 16, 32, etc.$, so the center of the ellipse (cluster center) is set at $(\frac{q}{2}+0.5, \frac{q}{2}+0.5)$. Within the support, each point is labeled in the increasing order accordiing to its "distance" (b^2) to the cluster center. If two points have the same "distance", they are resolved by assigning the smaller label to the point which is scanned first.

The label runs from 1 to $q^2/2$. The same labeled pattern in this quadrangle is tiled to fill up the remaining corner regions of the $q \times q$ matrix. This labeled matrix serves as a threshold array to be compared with the pixel gray value I which should be normalized to be within a range of $q^2/2$ units. If the input image has 256 possible gray values, the normalized gray value should be $(I/256)$ $(q^2/2)$. A point in the output image is set on if the normalized gray value is greater than or equal to the labeled-value of the corresponding point in the threshold array. This results in an elliptic cluster of a specific orientation and compression, as illustrated in Figure 25 (a) and (b). Figure 25 shows two different screen functions ($q = 16$) and their resulting cluster patterns. The values of θ and c in the screen function shown in Figure 25(a) are 45° and 1.0 respectivelly, while those values in Figure 25(b) are 30° and 0.4 respectively. . The labeled arrays are shown in the left of both figures. The normalized gray value of the pixel under consideration is 68 in each case, the resulting cluster patterns to represent these two pixels with different gradient information are shown in the right part of Figure 25(a) and (b) respectivelly. For a higher gray value, more distant points in the screen function will be set on, leading to a larger elliptic cluster. This screen function, being adaptive to the magnitude and orientation of the local gray level variation as estimated by the use of wavelet transform, can provide a halftone image with enhanced visualization.

Experimental Result:

The proposed adaptive halftoning method was experimented on the Lenna image, the result is shown in Figure 26(b). Figure 26(a) shows the original image consolidated to 256 × 256 pixels with 8-bit gray levels. In order to reduce the computational cost in our experimental study, the image pixels were classified into two classes according to the value of W. Those pixels in the nearly uniform region with $W \leq 1.0$ were classfied into class I; those pixels in regions of significant local intensity variation were classified into class II. A pre-computed screen function (with $c = 1$) was applied to pixels of class I, and the adaptive screen function was applied to pixels of class II. In this experiment, we set $q = 8$; the resultting halftone image shown in Figure 26(b) gave an



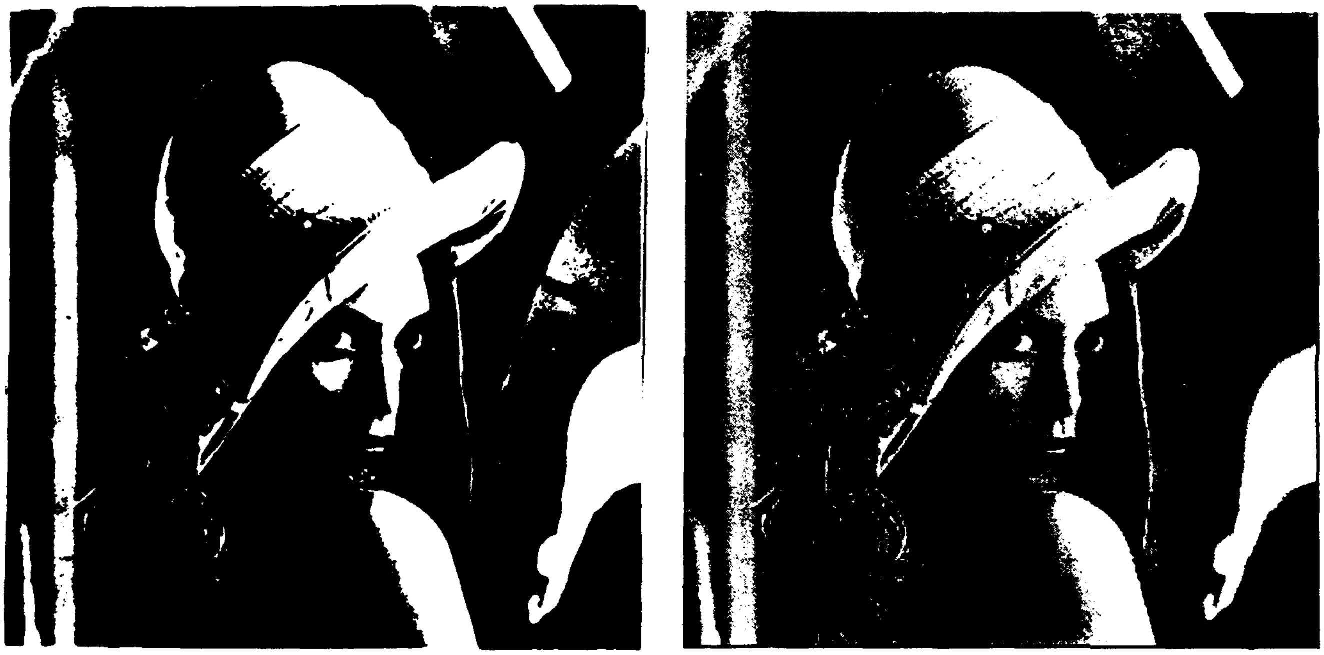
```
                     125
                 121 113 114
              109  97  89  90  98
            99  81  69  61  62  70  82
        100  77  53  45  33  34  46  54  78
     110  83  55  35  25  17  18  26  36  56  84
   122 101  71  47  27  13   5   6  14  28  48  72 102
126 115  91  63  37  19   7   1   2   8  20  38  64  92 116
127 117  93  65  39  21   9   3   4  10  22  40  66  94 118
   123 103  73  49  29  15  11  12  16  30  50  74 104
     111  85  57  41  31  23  24  32  42  58  86
        105  79  59  51  43  44  52  60  80
           106  87  75  67  68  76  88
              112 107  95  96 108
                 124 119 120
                     128
```

(a)

```
                     128
                 126 123 120
              125 118 111 105  97
           122 114 103  90  85  75  66
        117 107  95  83  71  57  51  41  39
     113 101  87  73  59  44  31  23  21  25  33
   110  99  81  68  53  35  17  .9   7  11  19  37  55
109  92  77  62  46  27  13   3   1   5  15  29  48  64  79
 94  80  65  49  30  16   6   2   4  14  28  47  63  78  93
    70  56  38  20  12   8  10  18  36  54  69  82 100
       50  34  26  22  24  32  45  60  74  88 102
          43  40  42  52  58  72  84  96 108
             61  67  76  86  91 104 115
                89  98 106 112 119
                   116 121 124
                       127
```

(b)

Figure 25: Screen functions (on the left) and the corresponding cluster patterns (on the right): (a) $c = 1$, $\theta = 45^o$, pixel gray value = 68; (b) $c = 0.4$, $\theta = 30^o$, pixel gray value=68.



(a)

(b)

Figure 26: An experimental result of the adaptive halftoning on the Lenna image: (a) the original image(256 × 256 × 8-bit) and (b) the resulting halftone image ($\tilde{q}$=8).

impression of only 32 gray levels, instead of 256 levels in the original image. The clarity of the display was nevertheless achieved. A comparison was made with the halftone image generated by the conventional single screen method ($c = 1$), a small degree of improvement in clarity was observed in the hair region. Further improvement is under consideration on on optimal threshold assignment for reducing contouring artifacts and aliasing.

B.1.g Artificial Neural Networks Based on Wavelet Transforms for System Identification

Our work on systolic architecture for computing signal decomposition and signal reconstruction [CHUA92] led us to explore the pipelined processing of discrete wavelet transforms in artificial neural networks for on-line system identification. Although not an image processing problem, this work is reported here to reflect another application of wavelet transforms.

Let us consider a class of separable nonlinear dynamic systems of Wiener-Volterra type which is modeled by a linear dynamic component followed by a static nonlinearity. The input-output relationship is governed by a functional mapping. Ths system identification problem is concerned with identifying the characterestics of an unknown dynamic system, or functional mapping, from measurements of its input and output. We have constructed three structured feedforward artificial neural networks, based on wavelet transforms, for identifying 1) stable linear dynamic systems, time invariant or time varying, 2) static nonlinear function approximation, and 3) separable nonlinear dynamic systems respectively.

For a discrete-time linear dynamic system characterized by its time-varying impulse response $h(k, m)$, which denotes the impulse response at time k when the impulse input is applied at time m, the system output $y(k)$ is given by

$$y(k) = \sum_{m=0}^{k} h(k,m)x(m)$$

where $\{x(m)\}$ denotes the input sequence. Let $h_k(k-m) = h(k,m)$, $\tilde{\mathbf{h}}_{\mathbf{k}} = [h_k(k), h_k(k-1), \ldots, h_k(0)]^T$ and $\mathbf{x} = [x(o), x(1), \ldots, x(k)]^T$. Furthermore, let us consider wavelet transforms of the input sequence $\mathbf{x}$ and the reversed impulse response sequence $\tilde{\mathbf{h}}_{\mathbf{k}}$ over a time interval $[0, N-1]$ by using the Daubechies' orthonormal wavelet D_4, for example, with low-pass filter coefficients $\{c_1, c_2, c_3, c_4\}$ and high-pass filter coefficients $\{g_1, g_2, g_3, g_4\}$. The J-scale orthonormal wavelet transform can be represented by a matrix operator $\mathbf{A_J}$,

$$\mathbf{A_J} = \prod_{j=J}^{1} \mathbf{A}^{\mathbf{j}}_{\mathbf{N/2^{j-1}}}.$$

where

$$\mathbf{A}^{\mathbf{j}}_{\mathbf{N/2^{j-1}}} = \begin{bmatrix} \mathbf{A}^{\mathbf{1}}_{\mathbf{N/2^{j-1}}} & 0 \\ 0 & \mathbf{I}_{\mathbf{N-N/2^{j-1}}} \end{bmatrix}$$

$\mathbf{I_q}$ is a $q \times q$ identity matrix and

$$\mathbf{A_N^1} = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 & c_2 & c_3 & c_4 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & c_1 & c_2 & c_3 & c_4 \\ c_3 & c_4 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & c_1 & c_2 \\ g_1 & g_2 & g_3 & g_4 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & 0 & g_1 & g_2 & g_3 & g_4 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & g_1 & g_2 & g_3 & g_4 \\ g_3 & g_4 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & g_1 & g_2 \end{bmatrix}$$

the latter expression is obtained on the hypothesis that the signal sequence is $N-$periodic, so that $\mathbf{A_J}$ is a unitary matrix. Then

$$y(k) = \tilde{\mathbf{h}}_\mathbf{k}^\mathbf{T} \cdot \mathbf{x} = (\mathbf{A_J}\tilde{\mathbf{h}}_\mathbf{k})^T \mathbf{A_J} \cdot \mathbf{x} = \mathbf{w_J^T} \cdot \mathbf{z}$$

where $\mathbf{z}$ is a $N \times 1$ vector containing wavelet transform coefficients of J scales of the input sequence, and $\mathbf{w_J}$ is a $N \times 1$ vector containing the wavelet transform coefficients of J scales of the reversed impulse response sequence. Based on the above-mentioned analysis, a three-layered artificial neural network is constructed where N hidden neurons have nonlinearities characterized by Daubechies orthogonal wavelet at successive scales and the scaling function at the coarest level; these artificial neurons take input data $\{x(n)\}$ with fixed connection weights are used to perform the Daubechies' orthonormal wavelet transform of J scales on the windowed input sequence $\{x(n)\}$. Only one set of connection weights $\{w_i\}$ leading to the output neurons are to be tranied. The system inpulse response kernel is then obtained by taking the inverse wavelet transform of the trained weights, which can be computed by

$$\tilde{\mathbf{h}}_\mathbf{k} = \mathbf{w_J^T} \cdot \mathbf{g}(k)$$

where

$$\mathbf{g} = [\phi_{J,0}, \phi_{J,1}, \dots, \phi_{J,2^{-J}N-1}, \psi_{J,0}, \dots, \psi_{J,2^{-J}N-1}, \dots, \psi_{1,0}, \dots, \psi_{1,N/2-1}]^T$$

In practice, both input sequence and impulse response sequence are not N-periodic, the wrap-around entries in the first few columns in matrix $\mathbf{A_N^1}$ should be removed; hence, $\mathbf{A_J}$ will no longer be unitary and the transform will introduce some errors in the first few terms of $\tilde{\mathbf{h}}_\mathbf{k}$, i.e., the trailing terms in the impulse response sequence. For asymptotically stable systems, these terms are usually very small in comparison to other terms in the sequence.

Because the hidden neurons in this structure provide orthogonal channels to represent the input signal characteristics, only those weights corresponding to changing characteristics of the input-output data pairs need to be adjusted as successive blocks of N data pairs are brought into training. This leads to great improvement in training time and accuracy by one order of magnitude, when compared with the use of the conventional multilayered neural networks for system identification.

Another three-layered feedforward neural network has been constructed for identification of a nonlinear function where the scaled cubic spline functions are used as radial basis functions in the special neurons in the hidden layer. The connection weights from the input to the hidden layer are all fixed to be equal to one; only the connection weights from the hidden layer to the output neurons are to be trained from the measured input-output data pairs $\{x, y\}$ which are first sorted in an ascending order of the input variable x. The local maxima of the wavelet transform modulus of the inpout-output data and their locations, $x_i's$, are used to determine the required number of special neurons and neuron bias -$x_i's$ at the given scale level. As more blocks of training data become available, structures at finer scale levels will be determined and trained until a desired approximation accuracy is obtained. By cascading this network to the network constructed for linear system identification, s special structure of four-layered artificial neural network is obtained for identification of a class of nonlinear dynamic systems. Simulation studies have been performed to compare with the conventional multilayered networks for nonlinear system identification. The performance improvements, both in training speed and in accuracy, make this approach potentially useful for on-line dynamic system identification. The details of this work are described in a paper which has been submitted for publication [HOLI92].

B.2 Parallel Image Processing

In this project we have considered several parallel architectures as targets for our parallel algorithms: 2D mesh; 3D mesh; special purpose systolic arrays and special forms of compound graph networks. For the mesh architectures we have considered both traditional and reconfigurable mesh architectures. The 2D and 3D mesh architectures received the most attention. For these architectures a variety of algorithms and algorithm classes have been considered including wavelet transform based algorithms. In section B.2.b we have evaluated time complexity of certain embeddings of wavelet transformed 2D images into 2D mesh architectures for a variety of algorithm classes. In section B.2.c we have given special attention to the general problem of embedding 2D meshes into 3D meshes. In section B.2.d we have addressed a variety of issues relating to 3D image processing and the use of 3D meshes in parallel image processing including: the embedding of 3D multiscale transforms into 3D mesh; a theoretical analysis of the expected speedup when using 3D mesh; 3D shrinking to a residue algorithms including subfields methodology; 3D connected component labeling algorithms; and the segementation problem in magnetic resonance imaging. In section B.2.e we examine fundamental issues for parallel reduction and reduction-augmentation operators in 2D and 3D image spaces including: the identification of constraints on operator support size and shape and the identification of efficient connectivity preservation tests. In section B.2.f we illustrate efficient wavelet decompositions and reconstructions for 1D signals in special systolic arrays. Finally, in section B.2.g we explore the potential for using graph compound networks to enhance the communication capability of mesh architectures.

B.2.a Parallel Architectures

Multiresolution image processing is typically a natural fit to pyramid computing architectures [AHUJ84, BURT83, TANI83]. Wavelet forms of image representation for $n \times n$ images can place a multiresolution representation into an $n \times n$ mesh architecture with one coefficient per processing element (PE) [MALL89b]. Examples of such mesh architectures have been developed [DUFF86, FOUN90], and are likely to be more readily available than pyramid architectures in the future. Further, reconfigurable 2D meshes have been shown to emulate pyramid architectures well [LIST91, CANT88, ALBA91, MARE91]. Thus, $n \times n$ mesh architectures with or without reconfigurability appear to be desirable targets for 2D wavelet based algorithm development. The 3D mesh architecture is becoming commercially available [JACK] and offers a more compact packaging of large arrays, as compared to 2D mesh, which leads to several advantages [NOAK90]. The 3D mesh utilizes all three available spatial dimensions for interconnection and is in some sense the ideal massively parallel architecture which satisfies typical VLSI constraints. This mesh is particularly apt for processing 3D images and may also be useful for 2D image processing. The

2D and 3D mesh architectures were the primary targets for parallel image processing research in the project; although a few other special purpose architectures were considered.

The 2D mesh model (MESH2) used in this analysis includes 4-neighbors connections (as illustrated in Figure 27a). The reconfigurable 2D mesh model (RMESH2) is illustrated in Figure 27b where each position of the mesh is occupied by a PE (circle) and a switch (square). Each PE is 4-connected to other PE's with a traditional mesh interconnection and its associated switch handles interaction with separate row and column busses. (This model is similar in power to the polymorphic torus described in [LIST91]) Typical switch settings allow busses to bypass PE's achieving direct connections between distant PE's, multiple paths along any bus, etc. By allowing for up to 4 independent connections between PE's and associated switches, the PE's can utilize the row and column busses to form 4 direct connections with PE's along the same row and column. With this model using the same embedding of the pyramid into the mesh as in [CANT88], each pyramid level above the base is able to function independently with direct connections to 4-neighbors within each pyramid level provided by segments of the row and column busses. Applications explored so far utilize primarily the capability of dividing a given row or column bus into sets of distinct busses.

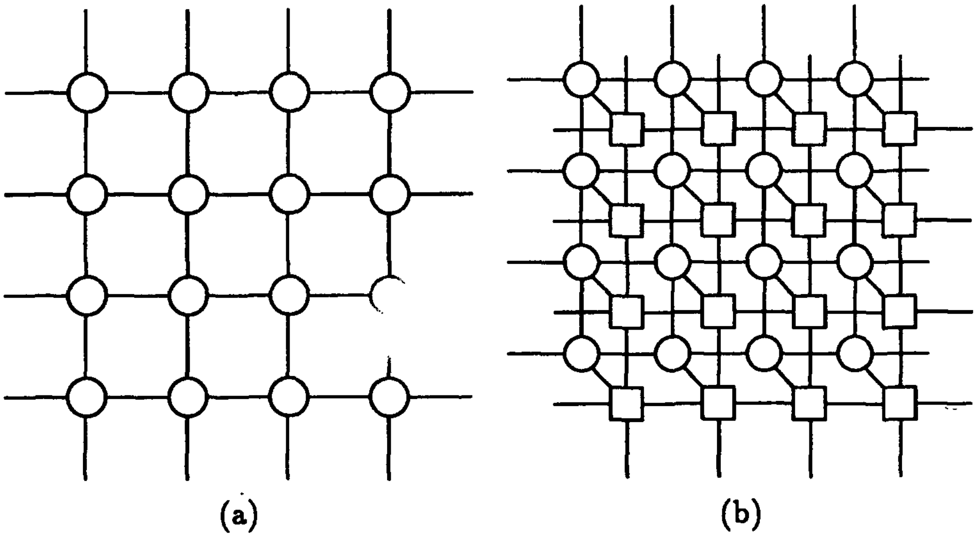


Figure 27: 2D mesh computing models: (a) MESH2 a 4 × 4 piece of a 2D mesh (b) RMESH2 a 4 × 4 piece of a reconfigurable mesh model. Circles represent PE's and squares represent switches.

The 3D mesh models, MESH3 and RMESH3, are the obvious extensions from the 2D case, with 6-neighbor interconnections. In reconfigurable 3D mesh (RMESH3) applications we will report the specific amount of reconfigurability required, e.g. say only within planes orthogonal to a given axis.

Certain special purpose architectures are considered. A systolic model is discussed in section B.2.f which is composed of a set of 1D arrays of special purpose PE's [CHUA92]. Certain forms of compound graph interconnection networks are proposed to enhance mesh communication capability [HAMD91a, HAMD91b] and are discussed in section B.2.g.

B.2.b Embeddings of Wavelet Transforms in 2D Mesh

In common applications wavelet transform decomposition initially effectively convolves four 2D filters with the original image, A in Figure 28, producing four resulting images at 1/2 of the resolution of A, i.e. A_1, D_1^1, D_1^2, D_1^3. The new image A_1 is contrived to be a 'low-pass filtered' lower resolution version of the original image, A. The *detail* images D_1^j represent difference images which can be used to reconstruct A; but typically also represent useful image analysis features. Successive iterations at levels $i = 1, 2, \ldots$ of decomposition apply the four filters to the lower resolution version of A, A_i. Separable filters are typically used in the fundamental 2D wavelet definition and the effective 2D filter convolutions are realized with separate 1D filters along rows and columns. Results presented here will assume such separable filters. A more complete presentation is found in [HALL93c].

In order to efficiently utilize such wavelet transforms in parallel mesh architectures, we need to identify where to place the wavelet transform coefficients into such meshes. This is the embedding problem which we address in this section. Such embeddings will impact on the time complexity achievable by various parallel algorithms typically by affecting the communication overhead required to access elements in the support of operators used by the algorithms. In the following we consider certain such embeddings into 2D meshes (with and without reconfigurability) and evaluate the communication overhead for a variety of algorithm classes. We will demonstrate the superiority of one of these embeddings for most evaluations.

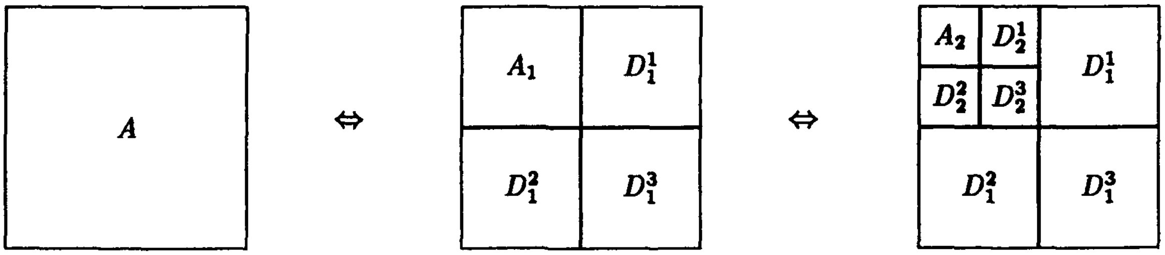


Figure 28: Wavelet transform decomposition and reconstruction where the original image A has been taken to a level 2 representation. See the text.

2D Mesh Embeddings:

Two particular embeddings are considered as illustrated in Figure 29. The original image is thought of as level 0 with size $n \times n$ $(n = 2^s)$ and successive levels, $i = 1, 2, \ldots$, refer to the successively lower resolution image representations characteristic of the wavelet transform [MALL89b]. In the embedding illustrated in Figure 29a (following [MALL89b]) the detail images D_i^1, D_i^2 and D_i^3 at level i of the multiscale transform and the lower resolution version of the image, A_i, are concentrated into subblocks within the mesh. This embedding is denoted Block-Concentrated (BC) and is illustrated to level 3 in Figure 29a. This BC embedding leaves separate detail images and A_i in contiguous blocks but at say level 3 in Figure 29a the four components corresponding to a given local region in the image are separated substantially. Thus, local operators which would use all components would encounter a communication overhead getting access to these widely separated components. This problem grows worse at lower levels. To counter this a second embedding has been developed, denoted Distributed (DIST), and is illustrated in Figure 29b where the individual components of A_i, and the detail images at level i (denoted a_i, d_i^1, d_i^2 and d_i^3) are distributed uniformly over the mesh. The DIST embedding is developed recursively, e.g. at level 0 2×2 regions of the original image are replaced with,

$$\begin{array}{cc} a_1 & d_1^1 \\ d_1^2 & d_1^3 \end{array}.$$

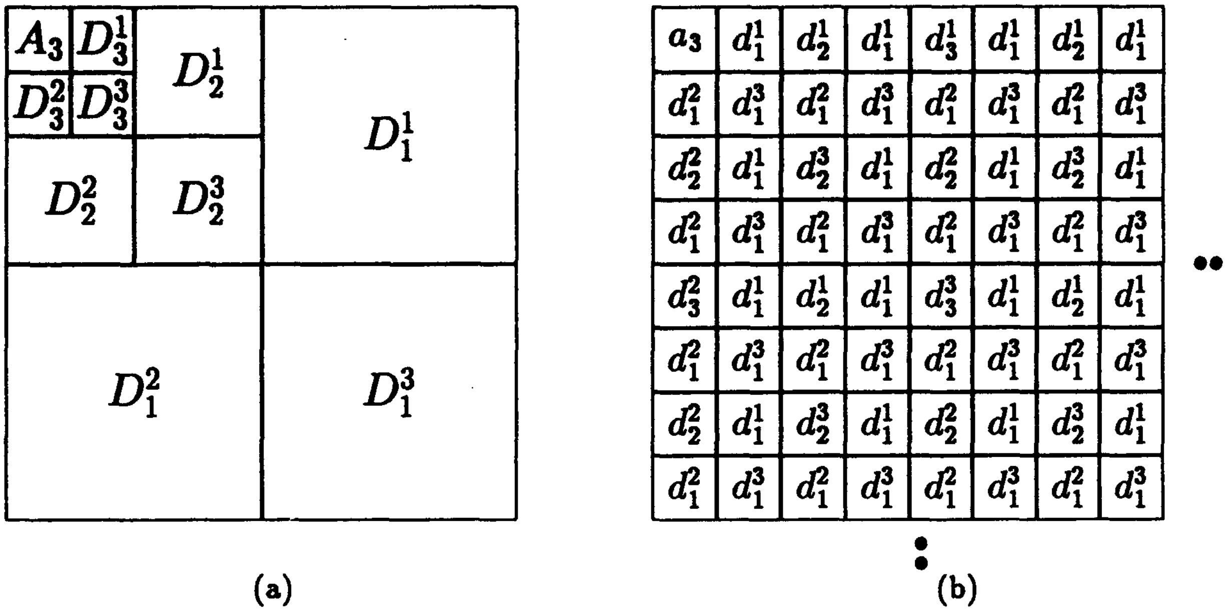
The level 1 individual components are thus distributed in this fashion:

$$\begin{array}{ccccc} a_1 & d_1^1 & a_1 & d_1^1 & \ldots \\ d_1^2 & d_1^3 & d_1^2 & d_1^3 & \ldots \\ a_1 & d_1^1 & a_1 & d_1^1 & \ldots \\ d_1^2 & d_1^3 & d_1^2 & d_1^3 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \end{array}$$

Level 2 is developed by replacing appropriate 2×2 regions of a_1's with

$$\begin{array}{cc} a_2 & d_2^1 \\ d_2^2 & d_2^3 \end{array}$$

and this process is continued recursively to the desired level, e.g. level 3 in Figure 29b. Specific detail image components at level $i (i > 0)$ corresponding to the same base image location are mapped to PE's in the mesh which are 2^i distance apart for the DIST case and 2^{s-i+1} distance apart for the BC case. (Distance is based on 4-adjacency which matches the connectivity assumed for the 2D mesh models) Thus for processing requiring access to all four image components, BC has higher communication cost at lower levels and lower communication cost at higher levels while the opposite holds for DIST.



a_3	d_1^1	d_2^1	d_1^1	d_3^1	d_1^1	d_2^1	d_1^1
d_1^2	d_1^3	d_1^2	d_1^3	d_1^2	d_1^3	d_1^2	d_1^3
d_2^2	d_1^1	d_2^3	d_1^1	d_2^2	d_1^1	d_2^3	d_1^1
d_1^2	d_1^3	d_1^2	d_1^3	d_1^2	d_1^3	d_1^2	d_1^3
d_3^2	d_1^1	d_2^1	d_1^1	d_3^3	d_1^1	d_2^1	d_1^1
d_1^2	d_1^3	d_1^2	d_1^3	d_1^2	d_1^3	d_1^2	d_1^3
d_2^2	d_1^1	d_2^3	d_1^1	d_2^2	d_1^1	d_2^3	d_1^1
d_1^2	d_1^3	d_1^2	d_1^3	d_1^2	d_1^3	d_1^2	d_1^3

Figure 29: Wavelet transform embeddings in a 2D mesh architecture: (a) Block-Concentrated embedding (BC); (b) Distributed (DIST) embedding.

Time Performance Evaluations:

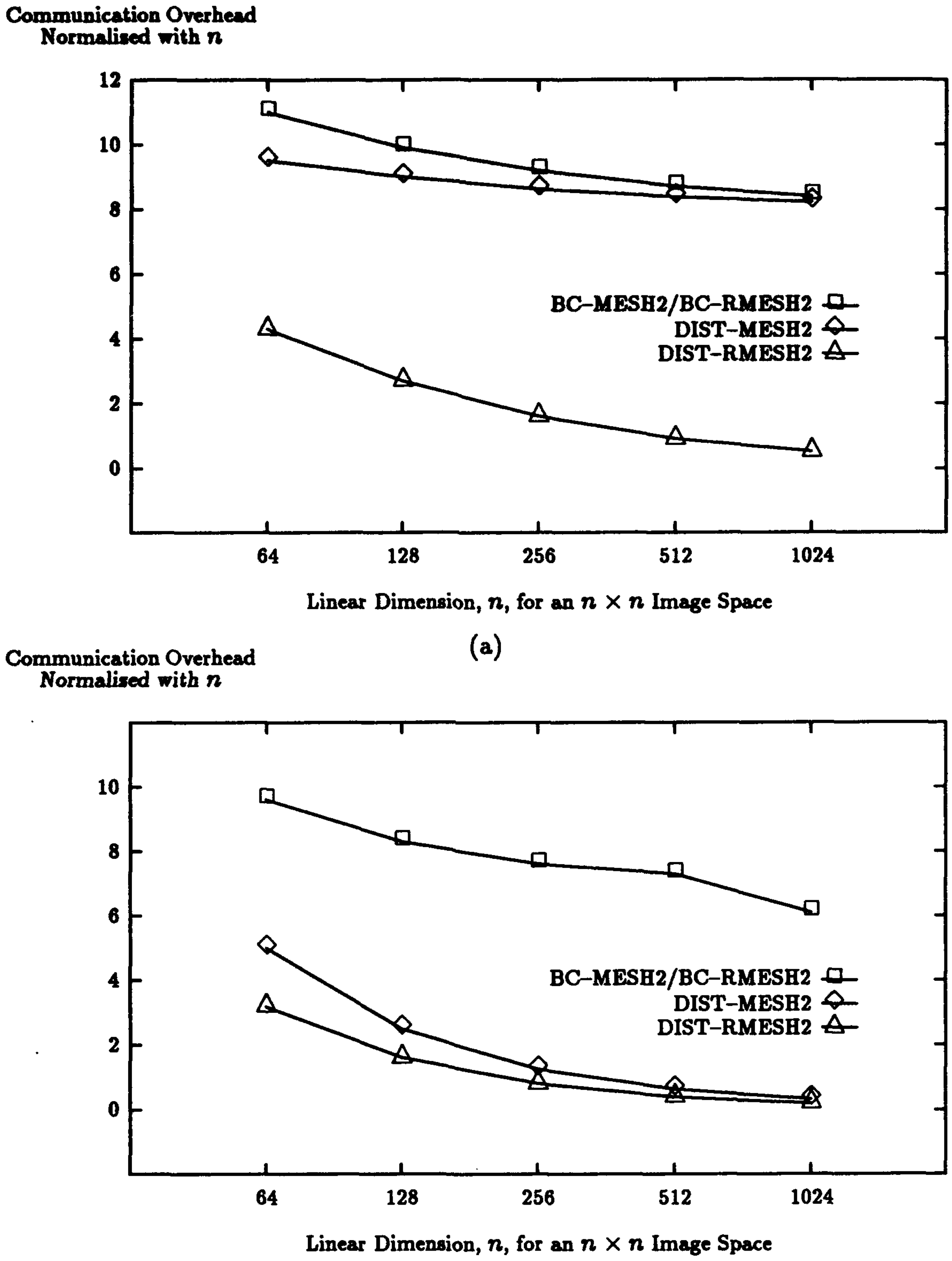
Time performance evaluations are performed for a variety of algorithm classes using the following assumptions: (1) Only communication steps are considered which includes filter coefficient broadcasts; (2) In one communication step a MESH2 PE can read one datum from one of its row or column PE neighbors; (3) In one communication step an RMESH2 PE can read one datum from one of its row or column PE neighbors or from one of the PE's which it connects to on its row or column of switches; (4) In RMESH2 each reconfiguration of the switch array costs one communication step; and (5) Filters in wavelet decomposition and reconstruction algorithms are assumed separable. Numbers of communication steps are typically normalized against the linear dimension n of the given $n \times n$ image.

Wavelet Decompositions and Reconstructions:

First, time performance results are reported for algorithms which decompose a given image into its wavelet transform and reconstruct the original image from its wavelet transform. The decomposition process requires a set of linear convolutions along rows and columns of the mesh with special data movements to place the results in the desired locations. The reconstruction process requires the insertion of rows and columns of zeros in the mesh, row and column linear convolutions, and data movements to restore image components to their original locations. The convolution algorithm used in this analysis is similar to that in [RANK90] and filter coefficients are broadcast from the master controller. The communication overhead is evaluated as the sum of the communication steps (e.g. movements of data from PE to neighboring PE) plus the number of coefficients broadcast to PE's. A linear convolution with a filter of size M has communication overhead of typically $2M - 1$ using these assumptions.

The communication overhead for decomposition and reconstruction can be expressed in closed form, but the relative tradeoffs are easier to see with graphical presentation of communication overhead for specific cases. (See [HALL93c] for more detail) Some typical decomposition (reconstruction) results are illustrated in Figure 30 (Figure 31) for a linear filter size of 8. The results are normalized by expressing number of communication steps as a multiple of the linear dimension of the image, n. In Figure 30a and Figure 31a the wavelet transforms are computed to a level where the lowest resolution image is the size of the filter, i.e. 8×8 for these results. In Figure 30b and Figure 31b the transforms are performed only to 3 levels of resolution for each image space size. In the former case more processing at higher levels of the multiresolution structure is required than for the latter case. In multiresolution processing where only a few levels of resolution are used the Figure 30b and Figure 31b data are more representative, but where larger numbers of levels of resolution are required the Figure 30a and Figure 31a data are more representative.

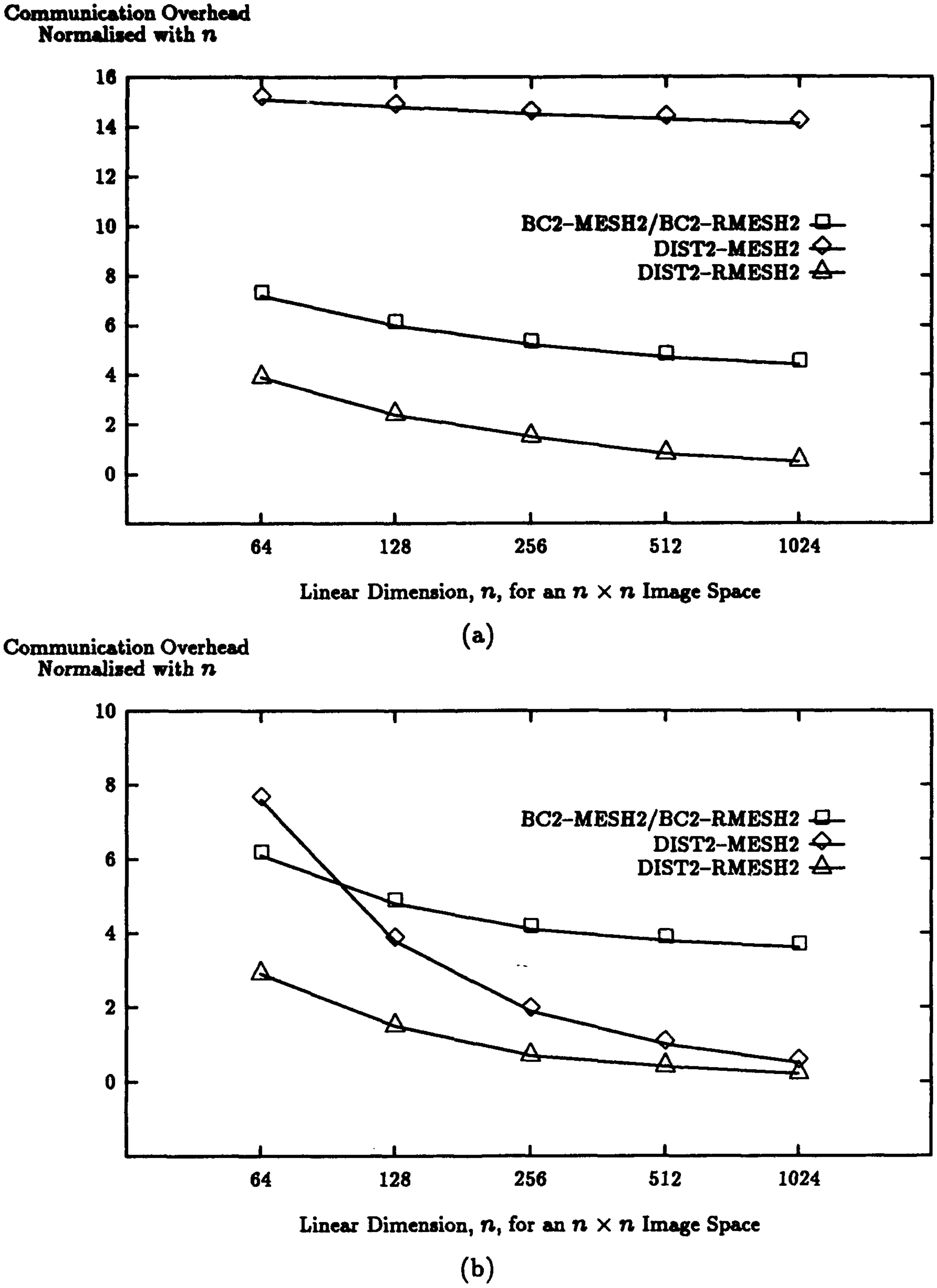
In Figure 30a and 31a examples, where higher level processing is required, for MESH2 mesh models DIST typically requires greater communication overhead than BC. But, for the RMESH2 model DIST is able to fully utilize reconfigurability achieving time performance essentially independent of image space size; whereas, BC performance cannot be improved by reconfigurability. In the Figure 30b and 31b examples the processing is restricted to a fixed number of levels and DIST performance typically is superior to that for BC for either mesh model and for larger images DIST performance is close for either mesh model. Thus, where one can avoid building a multiresolution representation with a large number of levels the DIST embedding appears to offer superior promise as a time efficient wavelet transform embedding for regular or reconfigurable mesh architectures; whereas, DIST is uniformly superior for reconfigurable models.



(a)

(b)

Figure 30: Communication overhead required by wavelet decomposition: (a) taken to 8 × 8 lowest resolution, (b) taken for only 3 levels of resolution. (Filter size is 8)



(a)

(b)

Figure 31: Communication overhead required by wavelet reconstruction: (a) taken from 8×8 lowest resolution and (b) working on images with only 3 levels of resolution. (Filter size is 8)

General local operators:

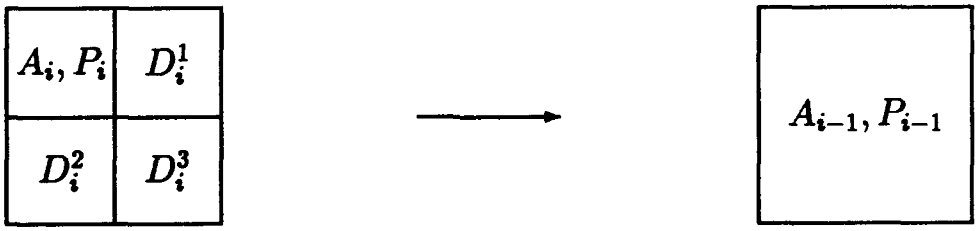
We next consider a class of operators with small local support, e.g. $m \times m$ support. Such operators might be used to compute arbitrary 2D inner products, local texture measures, etc. We assume that these local functions are applied at various resolution levels in the wavelet transform image representation as shown in Figure 32. We assume that the function is taken over all D_i^j and A_i at the given resolution level i. Figure 33 illustrates communication overhead for BC and DIST embeddings at different image resolutions. For MESH2 BC is ideal at the highest levels (i.e. small r) while DIST is ideal at the lower levels. DIST is able to fully utilize reconfigurability and communication overhead is not a function of level in the transform.

A Canonical planning model:

A canonical "planning" task model is defined next. Here we imagine a top-down process on a wavelet transform of an image which begins at some reasonably high level, k, in the transform. The "plan" is envisioned as some datum per pixel estimate of the image space, e.g. a likelihood of belonging to a given texture region; and is imagined to be computed as a local function, as defined earlier.

1. First, at the top level an initial plan is developed and stored at PE's at this level, one datum/PE;

2. In a typical iteration, i, the planning function is realized as a local function of A_i and the detail images and the previous plan, P_i; and reconstruction is performed to the next highest resolution image:



This task is a mixture of local function evaluation, repositioning of intermediate results and reconstruction.

Figure 34 reports communication overhead results for a planning task using a 5×5 local function support and beginning at level k where A_k is 8×8. For MESH2 DIST is superior to BC over all image instances considered. For RMESH2 DIST is able to fully utilize reconfigurability and communication overhead is not a function of level in the transform.

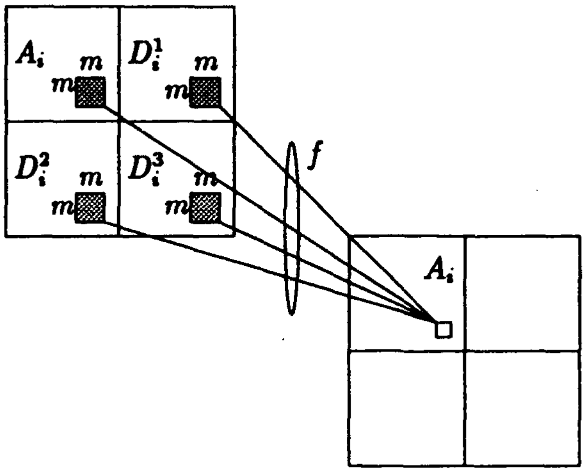


Figure 32: Local function model. Local function f has $m \times m$ supports in A and the detail images at some level i in the transform. It places its results in the PE's holding A.

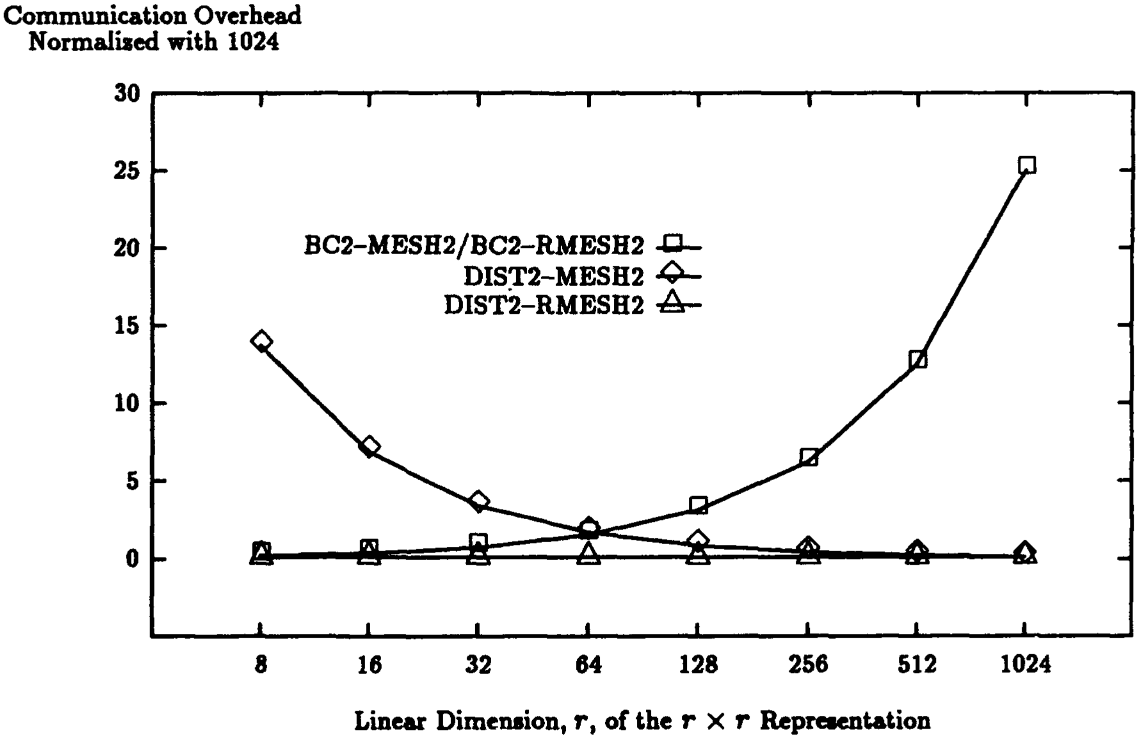


Figure 33: Communication overhead for 5 × 5 local function operator applied at different resolutions of a 1024 × 1024 image.

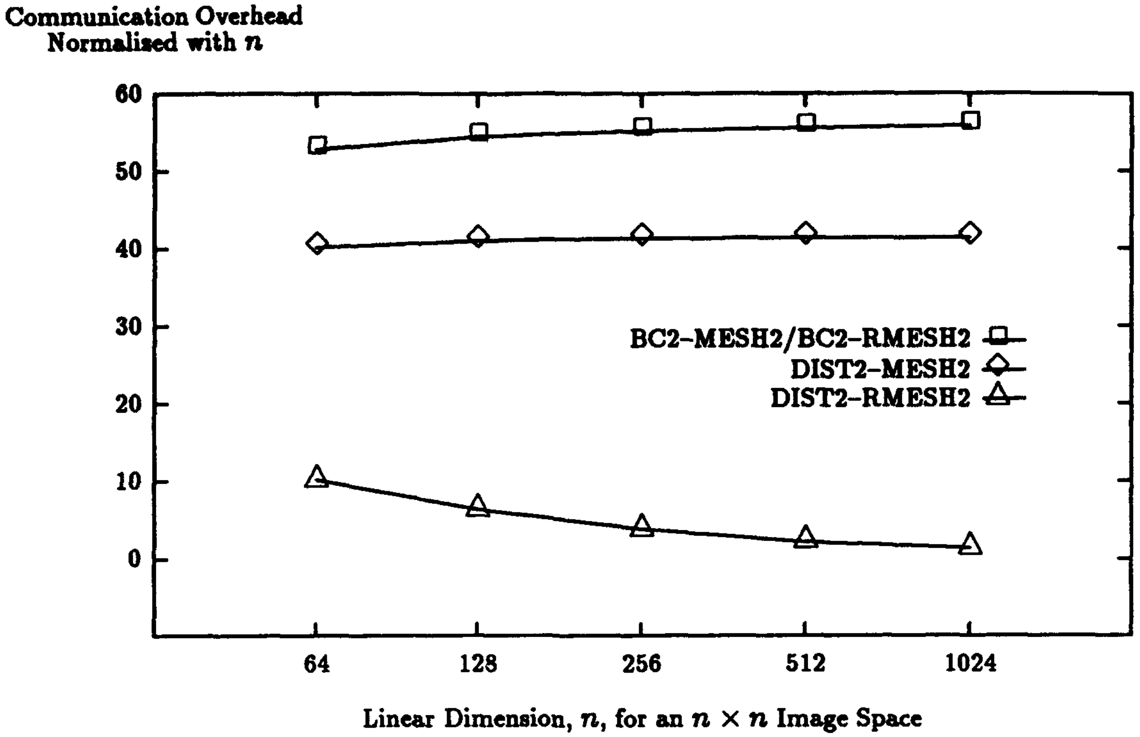


Figure 34: Communication overhead for top-down planning using a 5×5 local function support beginning at the 8×8 image representation.

Mapping larger 2D images into smaller 2D meshes:

In practical applications the 2D mesh size will be fixed and for larger images we will typically realize several pixels/PE. For this case we illustrate some results which compare communication overhead for the two embeddings for the fundamental linear convolution operator. We assume in this analysis that the original image is divided as symmetrically as possible among the PE's of the mesh maintaining image topography. Thus, a 1024 × 1024 image is placed in a 128 × 128 PE array such that each PE realizes an 8 × 8 subimage of the original image. We will term this the *Sub-block* multiple pixels mapping. One measure of mapping efficiency is the relative balance of load we place at each PE – this is typically measured as the maximum number of pixels realized by any PE in the mesh. BC load remains unchanged for all but the highest levels of image representation while DIST load falls quickly for higher levels. (more detail is given in [HALL93c]) The DIST load is optimally small in general for all levels. This optimal load balance leads to substantial advantages for DIST. The convolution cost is a function of worse case PE load and communication costs to obtain access to required neighboring pixels. BC maintains good locality of required pixels, but is hampered by poor load balance and is substantially inferior to DIST. Further, BC performance cannot be improved with reconfiguration. DIST has essentially minimal PE load, but pixel locality is lost for sufficiently high level. This degrades DIST performance somewhat for MESH2 but for the RMESH2 model this lost locality in the mesh can be ameliorated by reconfiguration. These results for linear convolution imply that the DIST embedding is typically superior for wavelet decomposition and reconstruction when larger images are processed by smaller meshes and this superiority increases as larger subimages are realized at PE's. (These results readily extend to the 3D case considered in section B.2.d.)

We propose an alternative mapping to Sub-block which comes as a corollary to the discussion on "Embedding 2D meshes into 3D meshes" details of which can be found in section B.2.c. The alternative mapping (denoted TDFM) maps to the mesh from the larger image rectangular blocks of pixels of the size of the smaller mesh. The mapping begins by taking the mesh-sized block at the upper left corner of the image and maps it to the mesh. Then it maps all the other blocks on top of the mesh in a snake–like manner. (More details are given in section B.2.c) This TDFM mapping can be used in conjunction with the BC and DIST embeddings and improves BC load balancing for MESH and RMESH models of computation and for DIST for MESH.

We will illustrate a simple multiple pixels/PE case where an 8 × 8 2D mesh will be embedded on a 4 × 4 mesh as shown in Figure 35. Figures 36 and 37 show how the wavelet coefficients will be distributed at level–3 for the BC and DIST cases respectively. Figure 36(a) shows that for BC typically PE's each realize pixels within one level of the transform and as a consequence when processing is being performed at levels greater than 0 many PE's are idle. Figure 37(a) shows that DIST distributes pixels over levels in a well balanced manner. For the alternative TDFM mapping Figure 36(b) shows that now BC PE's have a much better balance of wavelet coefficients over levels resulting in typically many less idle PE's. Further, as illustrated in Figure 37(b) DIST PE's when used with TDFM maintain their good balance.

Discussion:

The DIST embedding appears to be generally superior to the BC embedding for non-reconfigurable meshes. For reconfigurable meshes BC is unable to utilize reconfigurability to improve performance, while DIST is generally able to utilize reconfigurability well, frequently achieving optimal performance. DIST appears to be an ideal solution to the 2D transform–2D mesh wavelet embedding problem.

11	12	13	14	15	16	17	18
21	22	23	24	25	26	27	28
31	32	33	34	35	36	37	38
41	42	43	44	45	46	47	48
51	52	53	54	55	56	57	58
61	62	63	64	65	66	67	68
71	72	73	74	75	76	77	78
81	82	83	84	85	86	87	88

(a)

11,12,21,22	13,14,23,24	15,16,25,26	17,18,27,28
31,32,41,42	33,34,43,44	35,36,45,46	37,38,47,48
51,52,61,62	53,54,63,64	55,56,65,66	57,58,67,68
71,72,81,82	73,74,83,84	75,76,85,86	77,78,87,88

(b)

11,18,88,81	12,17,87,82	13,16,86,83	14,15,85,84
21,28,78,71	22,27,77,72	23,26,76,73	24,25,75,74
31,38,68,61	32,37,67,62	33,36,66,63	34,35,65,64
41,48,58,51	42,47,57,52	43,46,56,53	44,45,55,54

(c)

Figure 35: Multiple pixels/PE mapping of an 8 × 8 2D image to a 4 × 4 2D mesh (a) Original Image; (b) Sub-block method; (c) TDFM method.

$a_3^3, d_3^1, d_3^2, d_3^3$	$d_2^1, d_2^1, d_2^1, d_2^1$	$d_1^1, d_1^1, d_1^1, d_1^1$	$d_1^1, d_1^1, d_1^1, d_1^1$
$d_2^2, d_2^2, d_2^2, d_2^2$	$d_2^3, d_2^3, d_2^3, d_2^3$	$d_1^1, d_1^1, d_1^1, d_1^1$	$d_1^1, d_1^1, d_1^1, d_1^1$
$d_1^2, d_1^2, d_1^2, d_1^2$	$d_1^2, d_1^2, d_1^2, d_1^2$	$d_1^3, d_1^3, d_1^3, d_1^3$	$d_1^3, d_1^3, d_1^3, d_1^3$
$d_1^2, d_1^2, d_1^2, d_1^2$	$d_1^2, d_1^2, d_1^2, d_1^2$	$d_1^3, d_1^3, d_1^3, d_1^3$	$d_1^3, d_1^3, d_1^3, d_1^3$

(a)

$a_3^3, d_1^3, d_1^1, d_1^2$	$d_3^1, d_1^3, d_1^1, d_1^2$	$d_2^1, d_1^3, d_1^1, d_1^2$	$d_2^1, d_1^3, d_1^1, d_1^2$
$d_3^2, d_1^3, d_1^1, d_1^2$	$d_3^3, d_1^3, d_1^1, d_1^2$	$d_2^1, d_1^3, d_1^1, d_1^2$	$d_2^1, d_1^3, d_1^1, d_1^2$
$d_2^2, d_1^3, d_1^1, d_1^2$	$d_2^2, d_1^3, d_1^1, d_1^2$	$d_2^3, d_1^3, d_1^1, d_1^2$	$d_2^3, d_1^3, d_1^1, d_1^2$
$d_2^2, d_1^3, d_1^1, d_1^2$	$d_2^2, d_1^3, d_1^1, d_1^2$	$d_2^3, d_1^3, d_1^1, d_1^2$	$d_2^3, d_1^3, d_1^1, d_1^2$

(b)

Figure 36: Distribution of the wavelet coefficients at level–3 on the 2D mesh (a) BC with the Sub-block Method; (b) BC with the TDFM method.

$a_3^3, d_1^1, d_1^2, d_1^3$	$d_2^1, d_1^1, d_1^2, d_1^3$	$d_3^1, d_1^1, d_1^2, d_1^3$	$d_2^1, d_1^1, d_1^2, d_1^3$
$d_2^2, d_1^1, d_1^2, d_1^3$	$d_2^3, d_1^1, d_1^2, d_1^3$	$d_2^2, d_1^1, d_1^2, d_1^3$	$d_2^3, d_1^1, d_1^3, d_1^3$
$d_3^2, d_1^1, d_1^2, d_1^3$	$d_2^1, d_1^1, d_1^2, d_1^3$	$d_3^3, d_1^1, d_1^2, d_1^3$	$d_2^1, d_1^1, d_1^3, d_1^3$
$d_2^2, d_1^1, d_1^2, d_1^3$	$d_2^3, d_1^1, d_1^2, d_1^3$	$d_2^2, d_1^1, d_1^2, d_1^3$	$d_2^3, d_1^1, d_1^2, d_1^3$

(a)

$a_3^3, d_1^1, d_1^3, d_1^2$	$d_1^1, d_2^1, d_1^2, d_1^3$	$d_2^1, d_1^1, d_1^3, d_1^2$	$d_1^1, d_3^1, d_1^2, d_1^3$
$d_1^2, d_1^3, d_1^1, d_2^2$	$d_1^3, d_1^2, d_2^3, d_1^1$	$d_1^2, d_1^3, d_1^1, d_2^3$	$d_1^3, d_1^2, d_2^2, d_1^1$
$d_2^2, d_1^1, d_1^3, d_1^2$	$d_1^1, d_2^3, d_1^2, d_1^3$	$d_2^3, d_1^1, d_1^3, d_1^2$	$d_1^1, d_2^2, d_1^2, d_1^3$
$d_1^2, d_1^3, d_1^1, d_3^2$	$d_1^3, d_1^2, d_2^1, d_1^1$	$d_1^2, d_1^3, d_1^1, d_2^1$	$d_1^3, d_1^2, d_3^3, d_1^1$

(b)

Figure 37: Distribution of the wavelet coefficients at level–3 on the 2D mesh (a) DIST with the Sub-block method; (b) DIST with the TDFM method.

B.2.c Embedding 2D Meshes into 3D meshes

We now want to explore the potential for 3D mesh architectures in computer vision applications. First we consider the mapping of 2D meshes into 3D meshes. Embeddings of one architecture onto another allow an algorithm designed for an architecture/graph G (guest) to be simulated on another architecture/graph H (host). Such transformations make it possible to run an algorithm already designed for (perhaps a non–existent) architecture on a currently available architecture. We have studied such embeddings where a 2D mesh algorithm is embedded on a 3D mesh architecture. We have mentioned that most parallel algorithm work has been done for 2D mesh architectures. An embedding therefore will make it possible to automatically realize all the algorithms developed so far for 2D mesh architectures on the 3D mesh. The success of such an embedding can be measured by means of expansion, dilation, and congestion. Expansion is the size of the host graph relative to the size of the guest graph. Dilation is the maximum "stretch" of an edge of the guest graph on the host architecture. Congestion is the maximum number of stretched paths sharing an edge in the host graph. Dilation and congestion reveal information about the communication delay of the embedding. Expansion tells us how many processors are active in the host embedding, i.e. the processor utilization. We have studied dilation–1 2D–3D embeddings which tend to maximize the processor utilization while minimizing the dilation and congestion. Therefore the communication cost of a 2D algorithm will be the same on the 3D architecture.

The Two–Dimensional Flip (TDF) Embedding:

We propose a new 2D–3D mesh embedding termed the Two–Dimensional Flip (TDF) embedding. We will assume that G and H are two mesh connected arrays of processors, such that, $G = g_1 \times g_2$, $H = h_1 \times h_2 \times h_3$, and $g_1 \cdot g_2 \leq h_1 \cdot h_2 \cdot h_3$. be considered.) We will also assume that $g_1 = g_2 = N^{1/2}$, $h_1 = h_2 = h_3 = N^{1/3}$ for simplicity. In other words the $N^{1/2} \times N^{1/2}$ 2D mesh will be split into exactly $N^{1/3}$, $N^{1/3} \times N^{1/3}$ sub-planes and will be stacked on top of each other in the 3D mesh of size $N^{1/3} \times N^{1/3} \times N^{1/3}$. However, it is desirable not to add any extra overhead by the embedding. Therefore, we will provide good row connectivity and propose the following mapping.

Split the 2D mesh into partitions that are the size of the 3D mesh's planes. Then order those partitions in order to stack them on the 3D mesh's planes. Let $k = h_1^{1/2} = N^{1/6}$ and $P_{(i,j)}$ be the partition number of size $N^{1/3} \times N^{1/3}$ in the 2D mesh:

$P_{(1,1)}$	$P_{(1,2)}$	$\cdots$	$P_{(1,k-1)}$	$P_{(1,k)}$
$P_{(2,1)}$	$P_{(2,2)}$	$\cdots$	$P_{(2,k-1)}$	$P_{(2,k)}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$P_{(k-1,1)}$	$P_{(k-1,2)}$	$\cdots$	$P_{(k-1,k-1)}$	$P_{(k-1,k)}$
$P_{(k,1)}$	$P_{(k,2)}$	$\cdots$	$P_{(k,k-1)}$	$P_{(k,k)}$

Now, embed the partitions, $P_{(i,j)}$, into the the 3D mesh in the following order,

$$\begin{array}{ccccc}
P_{(1,1)}, & P^C_{(1,2)}, & \cdots & P_{(1,k-1)}, & P^C_{(1,k)}, \\
P^*_{(2,k)}, & P^R_{(2,k-1)}, & \cdots & P^*_{(2,2)}, & P^R_{(2,1)}, \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
P_{(k-1,1)}, & P^C_{(k-1,2)}, & \cdots & P_{(k-1,k-1)}, & P^C_{(k-1,k)}, \\
P^*_{(k,k)}, & P^R_{(k,2)}, & \cdots & P^*_{(k,2)}, & P^R_{(k,1)},
\end{array}$$

so that each partition in this sequence forms a plane in the 3D mesh, where the operators, F^R, F^C and F^*, are such that, for a given arbitrary sized matrix, F, of size $n \times m$,

$$\begin{aligned}
F^R &\equiv \left\{ \text{interchange } i\text{-th row with } (n-i+1)\text{-th row for } i = 1, \lfloor \tfrac{n}{2} \rfloor \right\}, \\
F^C &\equiv \left\{ \text{interchange } j\text{-th column with } (m-j+1)\text{-th column for } j = 1, \lfloor \tfrac{m}{2} \rfloor \right\}, \\
F^* &\equiv F^{R^C} = F^{C^R}.
\end{aligned}$$

Notice that row partitions are located at consecutive planes which means that connectivity is preserved due to the snake-like placement of the partitions. Unfortunately this also means that column partitions will be separated depending on which partition they reside. No extra overhead is paid for 2D row convolutions but each shift operation in the column convolution takes $2k-1$ communication steps instead of 1. Thus, if only row-connectivity is required this is a dilation–1 embedding, otherwise it is dilation–$(2N^{1/6}-1)$.

Reconfigurable TDF (RTDF) Embeddings:

The substantial advantage of TDF embedding can be seen when the buses along each plane of the 3D mesh architecture perpendicular to the planes in which the sequences are embedded are reconfigurable. This is best illustrated with an example. We examine the embedding of an 8×8 2D mesh onto a $4 \times 4 \times 4$ 3D mesh ($N = 64$) as illustrated in Figure 38. The thick line that connects 44 to 54 is the necessary reconfiguration to make this embedding a dilation-1 embedding and it has to be repeated at consecutive planes to connect 43 to 53, 42 to 52, and 41 to 51.

11	12	13	14	15	16	17	18
21	22	23	24	25	26	27	28
31	32	33	34	35	36	37	38
41	42	43	44	45	46	47	48
51	52	53	54	55	56	57	58
61	62	63	64	65	66	67	68
71	72	73	74	75	76	77	78
81	82	83	84	85	86	87	88

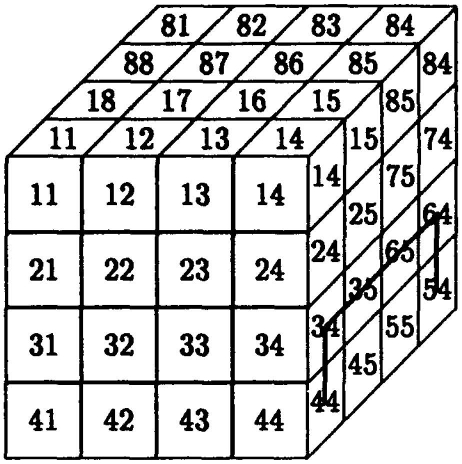


Figure 38: TDF Embedding of an 8 × 8 2D mesh onto a 4 × 4 × 4 3D mesh

In general, this reconfiguration for a single plane is shown in Figure 39. The solid and dotted lines show different paths of reconfiguration. Notice that the paths are non-conflicting, i.e., one can reconfigure the system to honor these connections once in the beginning and no further overhead is spent for the rest of the algorithm. It has the disadvantage that the data at the outer layers have to travel a distance which is a multiple of the size of the base mesh ($4(N^{1/3} - 1)$). The embedding also works for nonsquare base meshes. Further, the connections could be custom built into the architecture of the 3D mesh and activated upon a special controller request to enable any 2D algorithm to be executed on the 3D mesh architecture with no extra overhead. Since a single communication of the 2D mesh takes exactly one communication step on the 3D dilation-1 embedding, any performance results for the 2D case are also applicable here. The TDF embedding is not particularly efficient for the non-reconfigurable 3D mesh case suggesting a strong motivation for this level of reconfigurability.

The TDF embedding can be viewed as an alternative mapping for larger 2D images onto smaller 2D meshes. Imagine that rather than stacking each partition on the third dimension of a 3D mesh we collapse them to a base mesh where the PE's at the base emulate the corresponding data on the partitions above them. For example, for the case of 8 × 8 into 4 × 4, the PE that emulates the element 14 will also emulate 15, 85, and 84. (Figure 38) This has been addressed in section B.2.b.

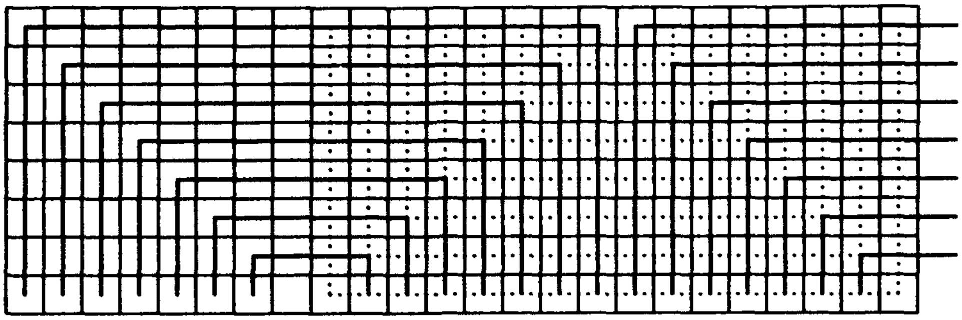


Figure 39: Paths required from reconfiguration capability along a single plane for the RTDF embedding

Convolution costs:

Consider the linear convolution costs where the filter is of size M. The communication overheads are as follows:

	Convolution along Rows	Convolution along Columns
2D	$M + (M - 1)$	$M + (M - 1)$
TDF	$M + (M - 1)$	$M + (2N^{1/6} - 1)(M - 1)$
RTDF	$M + (M - 1)$	$M + (M - 1)$

We see that extra overhead is required only for the operations involving communication along the columns of the wavelet transform in the non–reconfigurable case. Row operations are the same in the 3D mesh as a result of the embedding. If the extra overhead caused by the $2N^{1/6} - 1$ factor in column communication is small relative to the total cost then this embedding for a 3D non-reconfigurable architecture is feasible.

Transpose costs:

Parallel image transpose is used extensively in image processing especially in algorithms that require sorting. Transpose on a 2D mesh costs $2(N^{1/2} - 1)$ communication steps. On the other hand, communication costs are reduced significantly if the 2D mesh is embedded into the 3D mesh. Moreover, the transpose algorithm does not require the use of the reconfigurable buses.

Let the sequence,

$$\begin{array}{ccccc}
P_{(1,1)}, & P^C_{(1,2)}, & \cdots & P_{(1,k-1)}, & P^C_{(1,k)}, \\
P^*_{(2,k)}, & P^R_{(2,k-1)}, & \cdots & P^*_{(2,2)}, & P^R_{(2,1)}, \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
P_{(k-1,1)}, & P^C_{(k-1,2)}, & \cdots & P_{(k-1,k-1)}, & P^C_{(k-1,k)}, \\
P^*_{(k,k)}, & P^R_{(k-1,k)}, & \cdots & P^*_{(k,2)}, & P^R_{(k,1)}
\end{array}$$

denote the TDF embedding where each element of the sequence is a partition in the 3D mesh. Suppose that we are trying to transpose the following partitioned matrix given by,

$$\begin{bmatrix}
P_{(1,1)} & P_{(1,2)} & \cdots & P_{(1,k-1)} & P_{(1,k)} \\
P_{(2,1)} & P_{(2,2)} & \cdots & P_{(2,k-1)} & P_{(2,k)} \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
P_{(k-1,1)} & P_{(k-1,2)} & \cdots & P_{(k-1,k-1)} & P_{(k-1,k)} \\
P_{(k,1)} & P_{(k,2)} & \cdots & P_{(k,k-1)} & P_{(k,k)}
\end{bmatrix}^T = \begin{bmatrix}
P^T_{(1,1)} & P^T_{(2,1)} & \cdots & P^T_{(k-1,1)} & P^T_{(k,1)} \\
P^T_{(1,2)} & P^T_{(2,2)} & \cdots & P^T_{(k-1,2)} & P^T_{(k,2)} \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
P^T_{(1,k-1)} & P^T_{(2,k-1)} & \cdots & P^T_{(k-1,k-1)} & P^T_{(k,k-1)} \\
P^T_{(1,k)} & P^T_{(2,k)} & \cdots & P^T_{(k-1,k)} & P^T_{(k,k)}
\end{bmatrix}$$

Now if the transposed image is TDF–embedded, the TDF sequence becomes,

$$\begin{array}{ccccc}
P^T_{(1,1)}, & (P^T_{(2,1)})^C, & \cdots & P^T_{(k-1,1)}, & (P^T_{(k,1)})^C, \\
(P^T_{(k,2)})^*, & (P^T_{(k-1,2)})^R, & \cdots & (P^T_{(2,2)})^*, & (P^T_{(1,2)})^R, \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
P^T_{(1,k-1)}, & (P^T_{(2,k-1)})^C, & \cdots & P^T_{(k-1,k-1)}, & (P_{(k,k-1)})^T)^C, \\
(P^T_{(k,k)})^*, & (P^T_{(k-1,k)})^R, & \cdots & (P^T_{(2,k)})^*, & (P^T_{(1,k)})^R
\end{array}$$

and since $(F^T)^R = (F^C)^T$, $(F^T)^C = (F^R)^T$, and $(F^*)^T = (F^T)^*$, the TDF–sequence becomes,

$$\begin{array}{ccccc}
P^T_{(1,1)}, & (P^R_{(2,1)})^T, & \cdots & P^T_{(k-1,1)}, & (P^R_{(k,1)})^T, \\
(P^*_{(k,2)})^T, & (P^C_{(k-1,2)})^T, & \cdots & (P^*_{(2,2)})^T, & (P^C_{(1,2)})^T, \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
P^T_{(1,k-1)}, & (P^R_{(2,k-1)})^T, & \cdots & P^T_{(k-1,k-1)}, & (P_{(k,k-1)})^R)^T, \\
(P^*_{(k,k)})^T, & (P^C_{(k-1,k)})^T, & \cdots & (P^*_{(2,k)})^T, & (P^C_{(1,k)})^T
\end{array}$$

If we then reorder the sequence in the following way,

$$\begin{array}{ccccc}
P^T_{(1,1)}, & (P^C_{(1,2)})^T, & \cdots & P^T_{(1,k-1)}, & (P^C_{(1,k)})^T, \\
(P^*_{(2,k)})^T, & (P^R_{(2,k-1)})^T, & \cdots & (P^*_{(2,2)})^T, & (P^R_{(2,1)})^T, \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
P^T_{(k-1,1)}, & (P^C_{(k-1,2)})^T, & \cdots & P^T_{(k-1,k-1)}, & (P_{(k-1,k)})^C)^T, \\
(P^*_{(k,k)})^T, & (P^R_{(k-1,k)})^T, & \cdots & (P^*_{(k,2)})^T, & (P^R_{(k,1)})^T
\end{array}$$

Notice that this sequence is the same sequence where all the partitions are transposed in the original embedding. Thus, we have reduced the transpose of the 2D image into the transpose of the planes of the 3D mesh which can all be done at the same time in parallel. The only extra cost is the reordering cost of the planes which is small compared to the rotation costs.

Algorithm **Transpose** *for TDF embedded 2D images:*

- Transpose each plane in parallel, $P_{(i,j)}$, (communication cost $2(N^{1/3}-1)$)
- Reorder planes. (communication cost $N^{1/6}(N^{1/6}-1)$)

Figure 40 compares the communication overhead of a single parallel transpose executed on a 2D mesh with respect to a TDF–embedded 2D mesh.

Discussion:

We have shown efficient 2D-3D embeddings which achieve dilation-1 for certain forms of reconfigurable 3D meshes. This allows algorithm development work on 2D mesh architectures to be readily realized on 3D meshes. Thus, a 3D mesh architecture can provide a host to both 2D and 3D applications with no extra communication cost in a heterogeneous processing environment. We show that for strongly global classes of algorithms like the transpose a 3D mesh architecture will run 2D applications faster than a 2D mesh.

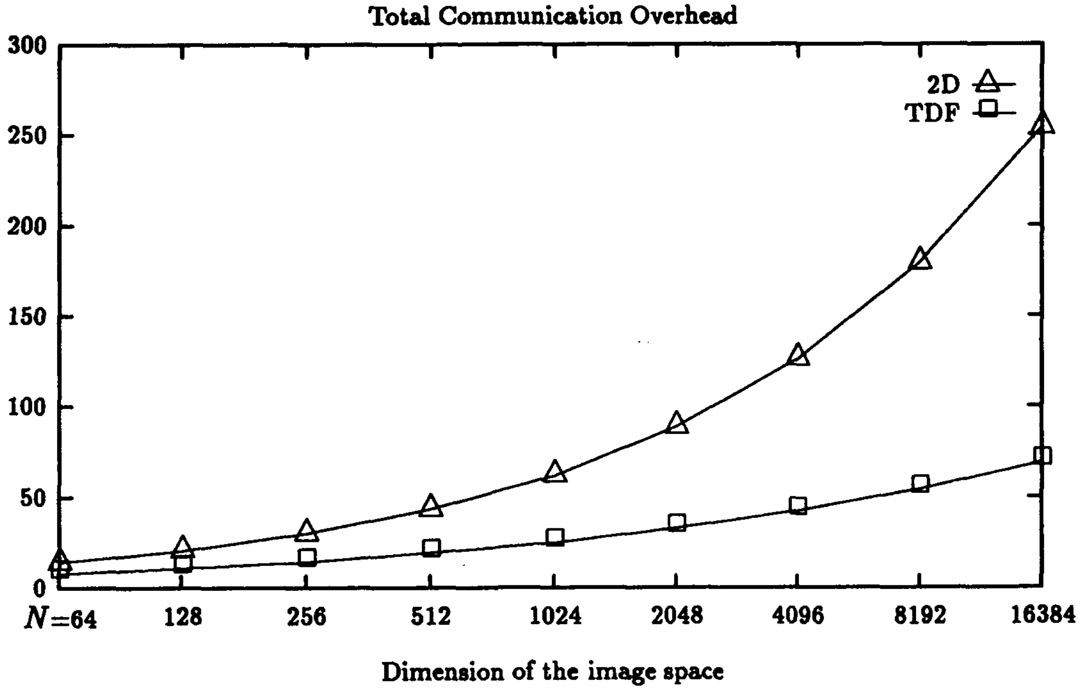
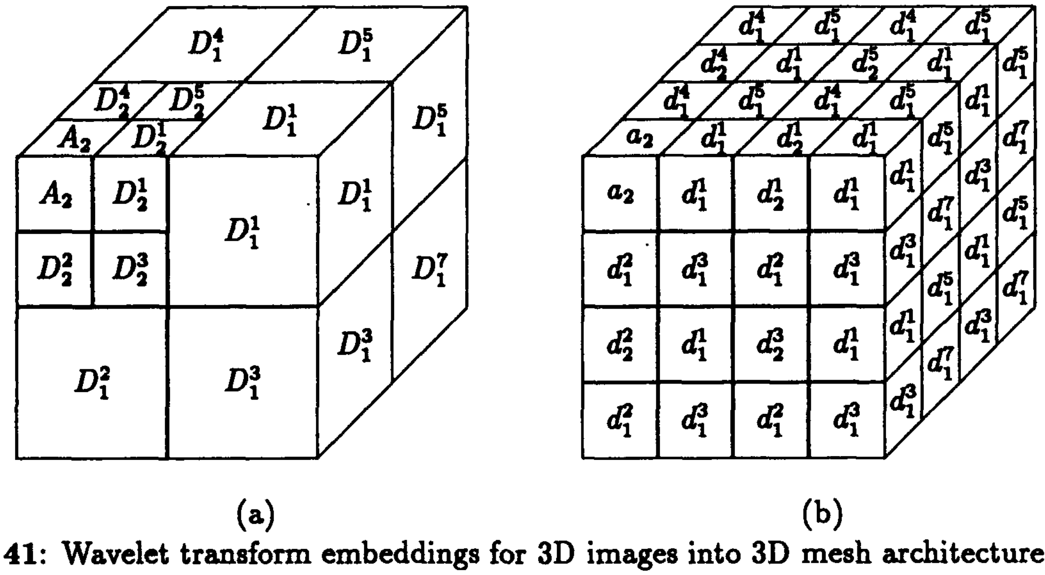


Figure 40: Communication Overhead Costs Transpose Costs for a Single Parallel Image Transpose

B.2.d 3D Mesh and 3D Image Issues and Applications

Embeddings of 3D Images and Transforms in 3D Mesh Architectures:

Certain of our 2D mesh results have been extended to the case where the target architecture is a 3D mesh (with and without reconfigurability). The 3D mesh models, MESH3 and RMESH3, are the obvious extensions from the 2D case, with 6-neighbor interconnections. We are looking for successful applications where we make minimal demands on reconfigurability. We extend the BC and DIST 2D embeddings in the most obvious manner and denote these embeddings BC3 and DIST3, as illustrated in Figure 41. (Here we assume a decomposition into A_i and 7 detail images, D_i^j.) The advantages identified for DIST in 2D cases tend to follow for 3D meshes. Whereas in the 2D image case it might be reasonable to expect that a 2D mesh is available with size close to image space size, in the 3D image case this is very unlikely to be the case. For example, a large commercial 3D mesh currently has size 32 × 32 × 16, while a large 3D magnetic resonance image could be 256 × 256 × 128. In this case if we wish to place the entire image in the mesh in a regular manner we would be realizing an 8 × 8 × 8 subimage at each PE. Thus, the analysis presented earlier on embedding larger 2D images into 2D meshes is quite appropriate to this case and illustrates the substantial advantage of DIST3. For example, for a linear convolution along one axis, the DIST3 relative performance advantage (as compared to BC3) improves over that for DIST2 [HALL93c].



D_1^4
D_1^5
D_2^4
D_2^5
D_1^1
D_1^5
A_2
D_2^1
A_2
D_2^1
D_1^1
D_1^1
D_2^2
D_2^3
D_1^7
D_1^3
D_1^2
D_1^3
d_1^4 d_1^5 d_1^4 d_1^5
d_2^4 d_1^1 d_2^5 d_1^1
d_1^4 d_1^5 d_1^4 d_1^5
a_2 d_1^1 d_2^1 d_1^1
a_2 d_1^1 d_2^1 d_1^1
d_1^2 d_1^3 d_1^2 d_1^3
d_2^2 d_1^1 d_2^3 d_1^1
d_1^2 d_1^3 d_1^2 d_1^3
d_1^1 d_1^3 d_1^1 d_1^3
d_1^5 d_1^7 d_1^5 d_1^7
d_1^1 d_1^3 d_1^1 d_1^3
d_1^5 d_1^7 d_1^5 d_1^7

(a) (b)

Figure 41: Wavelet transform embeddings for 3D images into 3D mesh architectures: (a) BC3 – the 3D extension of BC; (b) DIST3 - the 3D extension of DIST.

Expected Speedup for Three–Dimensional Mesh Architectures:

Next we will discuss the communication overhead of the 3D mesh architecture by means of expected speedup analysis[SCHE91]. Amdahl's law, which gives a measure of expected speedup, S, for an algorithm that has both serial and parallel components, can be written as,

$$S = \frac{s+p}{s+p/N}$$

where s is the time spent in serial parts of the problem and p is the time spent by a sequential computer in those parts of the problem that can be divided among N processors of a parallel processing machine. If $s+p=1$ (expressed as fractions of total time) then,

$$S = \frac{N}{1+s(N-1)} \tag{1}$$

which indicates that if s, the time spent in serial parts of the algorithm, is small the speedup S approaches N. In other words if you spent 10% of your total time in serial operations the maximum speedup you can achieve by parallelizing your algorithm is 10 even for very large number of processors. ($S \approx 1/s$ for N large)

Another version of this equation is given by Gustafson where p is now the total time spent by a parallel processing system in solving the problem,

$$S = \frac{s+pN}{s+p}$$

which becomes,

$$S = N - s(N-1)$$

for $s+p=1$. This is consistent with equation (1) for small s. We are primarily interested in the communication overhead imposed by the interconnection scheme of a parallel architecture. This depends on the architecture as well as the implementation of the algorithm. In order to include the effect of communication we will introduce the definition the communication overhead, c, to the equations as the time spent communicating between processors relative to the total time spent performing parallel computations. Thus c will be a function of the algorithm A and the machine M, i.e. $c = c(A, M)$.

Assume that the algorithm A needs to make x data transfers and p parallel computational steps. Let p_{ij} be the probability that a particular communication step is a data transfer from processor i to the processor j and t_{ij} be the time it takes to transfer data from processor i to j.

Therefore a measure of expected time for a single communication step is given as,

$$E(t) = \sum_{i=1}^{N}\sum_{j=1}^{N} t_{ij}p_{ij}.$$

It is very likely that both t_{ij} and p_{ij} are functions of the distance between the processors i and j. Therefore it is reasonable to assume that the time it takes to send a message from one processor to the other is proportional to the number of hops between them, i.e. $t_{ij} = t(d_{ij}) = Kd_{ij}$ where K is a scaling factor. Also in a strongly global class algorithm[STOU86] all processors are equally likely to to communicate, i.e. $p_{ij} = p(d_{ij}) = 1/N^2$. This reduces $E(t)$ to,

$$E(t) = \frac{K}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij} = Kd$$

where d is now the average distance between processors. Then an average fractional communication overhead for the algorithm A and machine M can be written as,

$$c(A,M) = \frac{xE(t)}{p + xE(t)} = \frac{xKd}{p + xKd} \qquad (2)$$

where the former is the ratio of communications time to total parallel processing and communications time as mentioned before. If we normalize this equation with respect to the number of processing steps then equation (2) becomes,

$$c(A,M) = \frac{\rho d}{1 + \rho d}$$

where $\rho = xK/p$.

The following can be derived for a 2D mesh, 3D mesh and a hypercube,

$$c(A, 2Dmesh) \approx \frac{2\rho N^{1/2}}{3 + 2\rho N^{1/2}}$$
$$c(A, 3Dmesh) \approx \frac{\rho N^{1/3}}{1 + \rho N^{1/3}}$$
$$c(A, hypercube) \approx \frac{\rho \log_2 N}{2 + \rho \log_2 N}$$

This shows that as expected both 2D and 3D mesh architectures suffer from communication overhead more than the hypercube but also that the 3D mesh is better than the 2D mesh. For large values of N both overheads converge to 1. Figure 42 shows the communication overhead for these architectures for reasonable values of N (64...16K) as a function of ρ. Notice that for large ρ (i.e. where number of data transfers is much larger than the number of parallel computational steps) the communication overheads approach 1 even for reasonable values of N. But for smaller values of ρ there is a greater variety of behavior among the architectures. Hypercube shows the effectiveness of it's interconnection scheme by requiring less communication overhead for strongly-global classes of algorithms. However, 3D mesh performs significantly better than the 2D mesh for all cases.

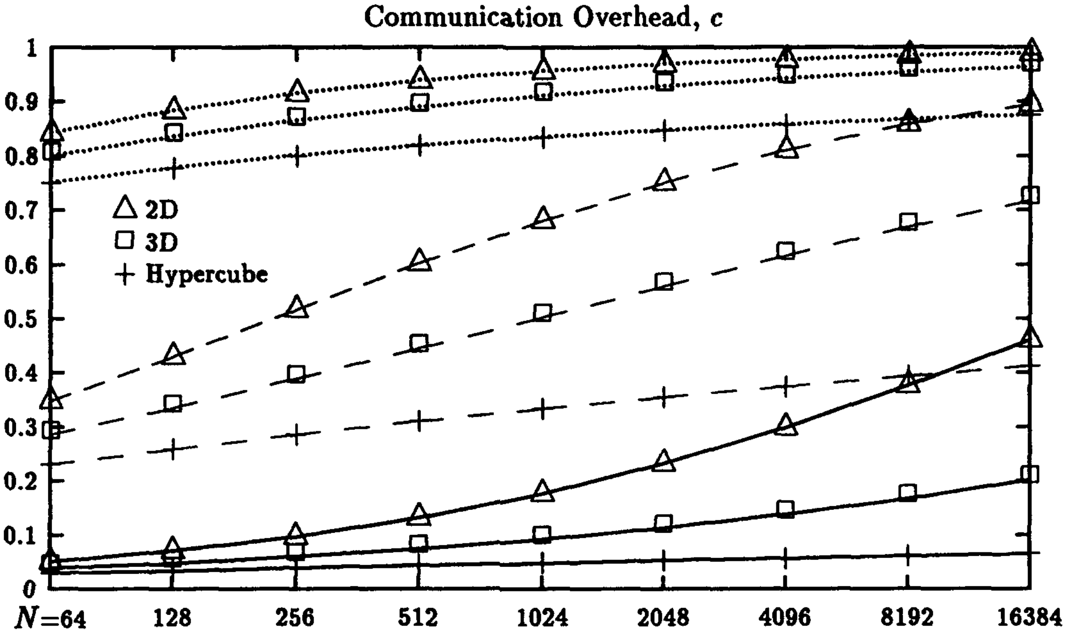


Figure 42: Communication Overheads of 2D mesh, 3D mesh and Hypercube ($\rho = 0.01$ for solid lines, $\rho = 0.1$ for dashed lines, $\rho = 1$ for dotted lines)

Now that we have a measure of the communication overhead c we will try to introduce that into the speedup S so that we can see the effect of communication overhead on the overall mapping of an algorithm A on the machine M. If the total time spent in parallel part of the algorithm is denoted by p_T then we can assume that c is the fraction of p_T spent performing communications, thus,

$$p = (1 - c)p_T.$$

Using the same formulation as above speedup S becomes,

$$S = (1 - c)N - s(N - 1 - c)$$

for Gustafson's formulations.

What is interesting in this formula is that both N, the number of PE's, and s, the serial part of the algorithm, are now affected by the communication overhead c in the way they contribute to the speedup. Furthermore, if s is small an upper bound on the maximum expected speedup can be determined by,

$$S \leq (1-c)N.$$

A similar analysis can also be done for Amdahl's formulation which results in a similar type of expression for the upper bound on the maximum expected speedup. The former indicates that speedup may significantly reduce as a function of communication overhead. Using expressions found above for the average fractional communication overheads we obtain the following,

$$\begin{aligned} S_{2D} &= \frac{3N}{3+2\rho N^{1/2}} \approx \frac{1.5N^{1/2}}{\rho} \\ S_{3D} &= \frac{N}{1+\rho N^{1/3}} \approx \frac{N^{2/3}}{\rho} \\ S_{hypercube} &= \frac{N}{2+\rho \log_2 N} \approx \frac{N}{\rho \log_2 N} \end{aligned}$$

Figures 43, 44 and 45 compare the expected speedup as a function of ρ for an algorithm A running on a machine M (2D mesh, 3D mesh, hypercube) with N PE's. As expected hypercube gives better speedup results for larger values of ρ. Recall that larger ρ means the number data transfers in the algorithm is much larger than the number of parallel computational steps. For the case where $\rho = 1$ 3D mesh performs slightly better than the hypercube up to 2048 PE's but hypercube gets better for larger values of PE's. Although not shown here as ρ increases further hypercube outperforms mesh architectures for all cases. However, as ρ decreases, which means that as communication becomes less of a burden in the algorithm, hypercube looses its attraction over the mesh architectures. As the expressions already suggested, the 3D mesh is substantially better than the 2D mesh architecture for all cases. In particular when $\rho < 0.1$ the 3D mesh begins to perform better than the hypercube for most cases considered. This suggests that the 3D mesh architecture is feasible and will perform better for that range of algorithms. Also notice that the absolute value of the maximum speedup decreases as ρ increases which means that you cannot expect more speedup by increasing the number of PE's when the algorithm requires a substantial amount of data transfers.

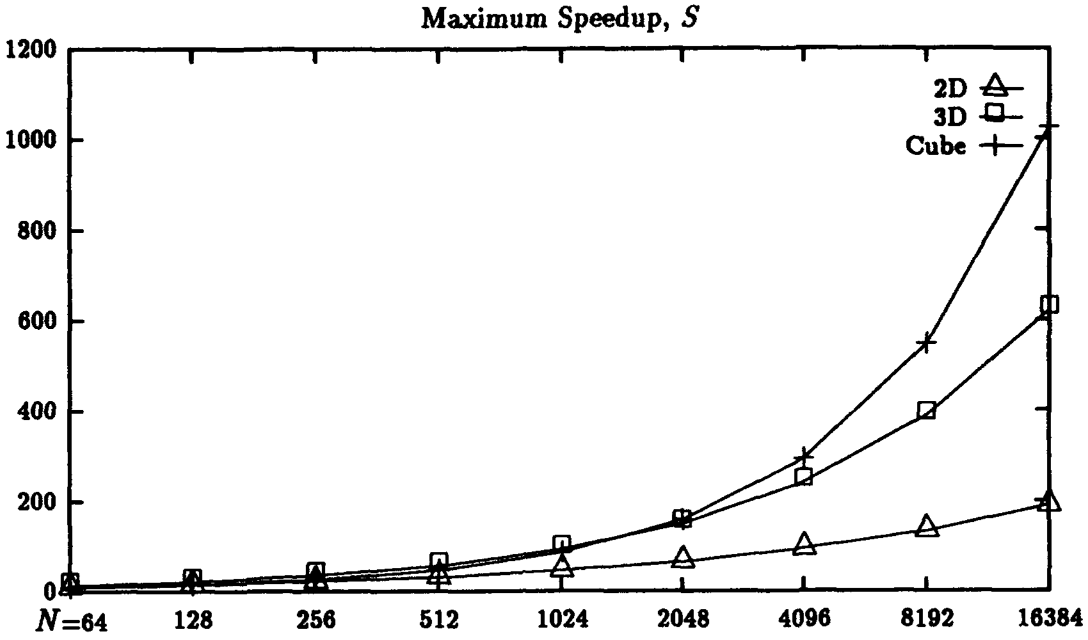


Figure 43: Maximum Expected Speedup of 2D mesh, 3Dmesh and Hypercube ($\rho = 1$)

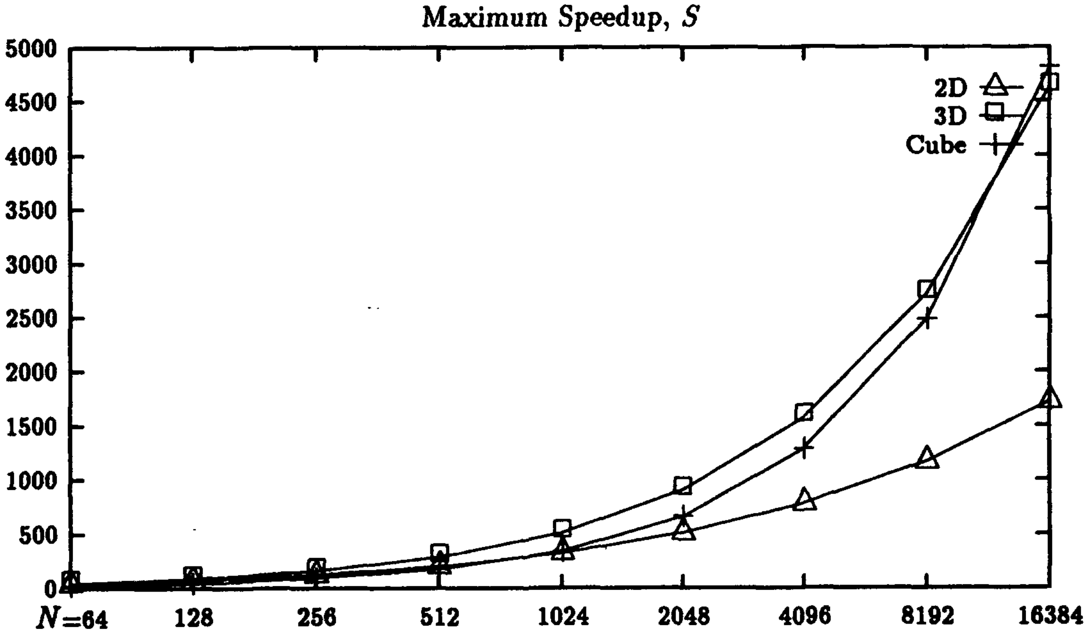


Figure 44: Maximum Expected Speedup of 2D mesh, 3D mesh and Hypercube ($\rho = 0.1$)

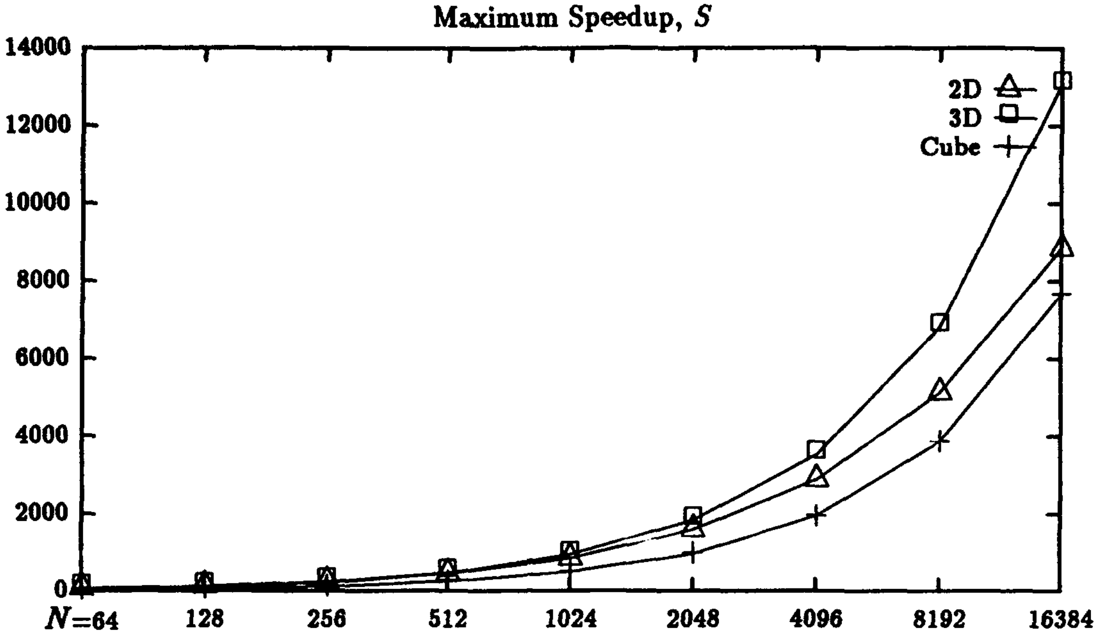


Figure 45: Maximum Expected Speedup of 2D mesh, 3D mesh and Hypercube ($\rho = 0.01$)

We have given a measure of communication overhead to analyse and compare 2D and 3D mesh architectures. Although our main purpose was to give a basis for determining the feasibility of the 3D mesh architecture, hypercube data was also included in the analysis as an example of a communication efficient parallel architecture. For the classes of algorithms we have considered the 3D mesh interconnection scheme gave substantially better results than its 2D counterpart and even in some cases it promised to be a competitor to the hypercube. The analysis suggested that there is a strong dependency of the algorithm to the architecture on which it will be executed, which suggests that the algorithm needs to be machine–tuned for better performance.

3D Shrinking to a Residue:

The problem of shrinking the individual components of a binary image to single point residues has been used to do component counting and labeling in 2D images. A solution to this problem was presented over 2 decades ago for 3D images, [ARCE72] but this solution did not preserve the connectivity of the foreground. (Certain distinct objects would be merged into one object by the algorithm) We have developed new approaches to parallel 3D shrinking to a residue using subfields methodology [HALK92] obtaining good parallel computation time and the first connectivity preserving 3D parallel shrinking algorithms reported. We have also identified fundamental limits on the realizability of parallel 3D shrinking algorithms using local operators [HALK92]. The use of ***subfields*** (i.e. partitions of the image space) where only part of the image space is altered in one parallel iteration has been found to ease verification of connectivity preservation in 2D and 3D thinning and 2D shrinking and in 2D instances offers particularly fast parallel performances. Thus, subfields methodology may offer a powerful tool in 3D parallel algorithm development where the design of correct and fast parallel algorithms is particularly challenging. Shrinking to a residue is typically defined for binary images composed of foreground (sets of ones) and background (sets of zeros) regions. In shrinking we wish to reduce each foreground connected component to a single point - termed a residue. This process entails preserving the connectivity properties of the foreground but not the background, since cavities and holes are removed in the process. An example is shown below for a 3 × 3 × 3 cube using the classical shrinker of Arcelli and Levialdi. Succesive parallel iterations are given in row major order with the original image in the upper left corner - labeled iteration 0.

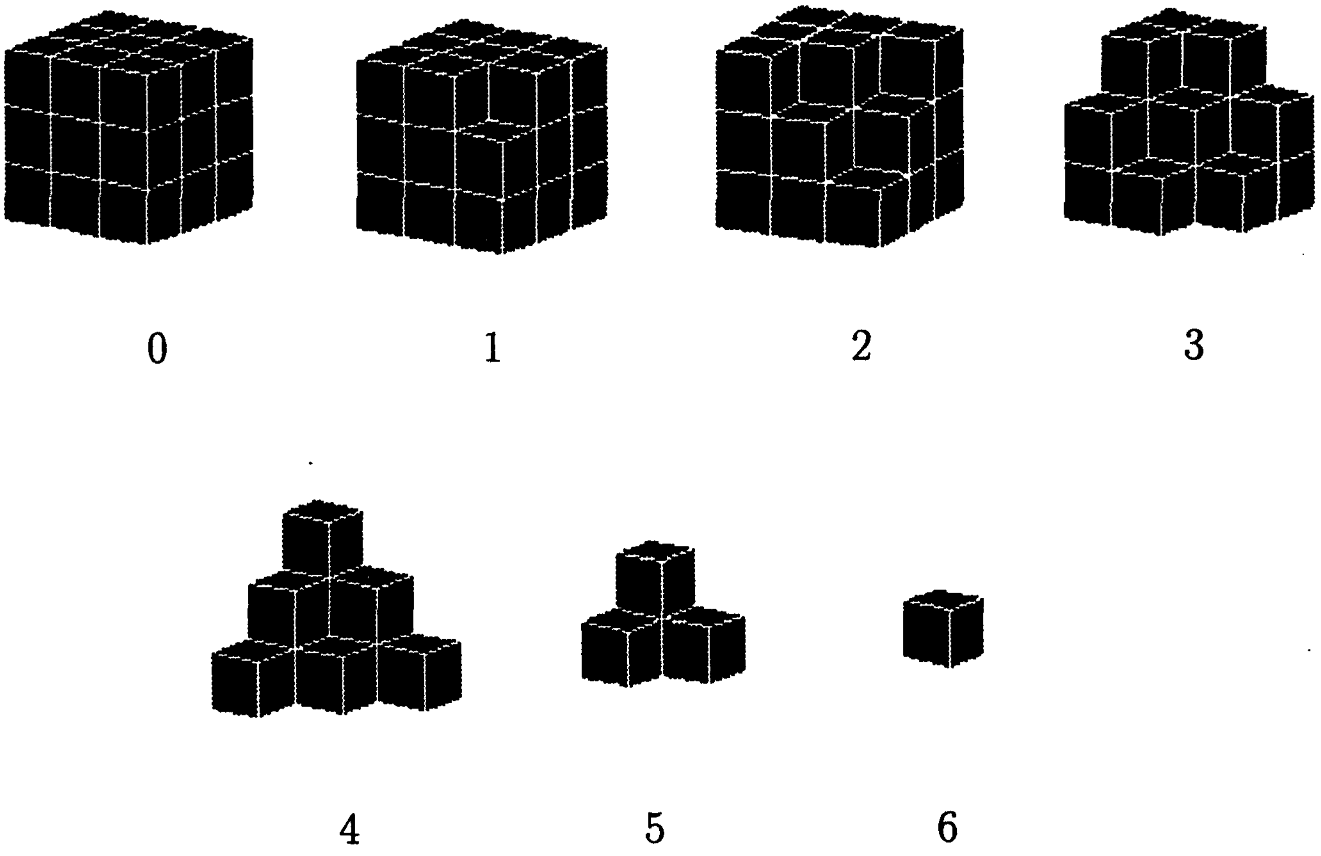


Figure 46: Example of 3D Shrinking to a Residue

For 2D images parallel shrinking to a residue algorithms using operators with local support are known which preserve connectivity and are provably convergent. In contrast we conjecture that connectivity preserving 3D shrinking to a residue based on local operators cannot succeed for arbitrary images. For example, we believe that there is no parallel connectivity preserving local shrinking operator which can shrink 2 unconnected closed curves which interleave (e.g. as the links of a chain) to separate residues. This follows because a local operator cannot in general determine whether or not two distinct curve segments in a local region are part of interleaved closed curves and cannot know whether or not to "unhook" the interleaved chains. As a result we cannot in practice design a parallel shrinking algorithm which for arbitrary 3D images simultaneously preserves connectivity and guarantees that all components of S are shrunk to single point residues. Thus, convergence to a correct solution for two-subfields shrinking algorithms is still an open question. Even if we cannot prove convergence in general for small support operators it remains to be seen if convergence can be proved for constrained image classes.

Several two-subfields shrinking algorithms have been implemented in parallel emulations and have been tested on sets of artificial images. Our tests indicate that a member of this class achieves an improvement in parallel computation time typically requiring ≈72–98% of the iterations required by the original algorithm of Arcelli and Levialdi [ARCE72] while preserving all connectivity properties. Convergence tends to fail for about 5% of our test cases and the experimental investigation for more robust algorithms is on-going.

We conclude that two-subfields approaches appear to offer some promise in the design of parallel connectivity preserving 3D image processing algorithms. Open issues include: the determination of image classes for which convergent shrinking algorithms can be designed and the development of efficient computations for $3 \times 3 \times 3$ local operators where table lookup is not feasible.

3D Connected Component Labeling:

We have studied the problem of developing parallel algorithms for the labeling of components in 3D images where the target architectures are the 3D mesh with and without reconfigurability [CHAN93]. We summarize these results in this section. The labeling of connected components is a fundamental step in image processing. In typical recognition tasks, images are given preliminary segmentations which produce essentially binary images with foreground elements representing particular regions of interest. Frequently we are interested in connected regions and in particular the maximal connected regions. Component labeling algorithms identify each distinct such region and assign a unique label to each element of that region.

There exist many methods in 2D, both sequential and parallel, to label components [ALNU91, HARA92]. On an $n \times n$ 2D Mesh Connected Computer (MCC) with one pixel per processing element (PE), the algorithms with the best asymptotic time complexity are the divide and conquer technique [NASS80], and shrink-expand techniques [CYPH90,ALNU91] which are based on the classical Levialdi shrinking algorithm [LEVI72]. Both of these algorithms have an $O(n)$ worse case time complexity. But the divide and conquer technique has a rather large constant of proportionality for time complexity and uses global broadcasting operations, while the shrink-expand techniques have smaller constants and use small local operators. For these reasons, shrink-expand techniques are considered to be the best practical solution on 2D mesh architectures which are not reconfigurable.

It would be nice to be able to extend such notions to 3D images on 3D mesh architectures. Unfortumately, the shrinking to a residue problem has not been solved for general 3D images and as we indicated in the previous section may not be solvable with local operators. Thus, to exploit these 2D shrink-expand approaches in 3D meshes, we will need to do apply them within 2D layers of the image.

When a reconfigurable 2D mesh architecture is available, there are very good component labeling algorithms available [LIST91] which can produce $O(\log n)$ performance using radix-based methods. Such performance is achievable when broadcasts on a bus can be claimed to be $O(1)$. But the worse case bus length is $O(n^2)$ (for space filling curves like spirals) and the real bus broadcast time could substantially increase the time complexity. For example, if $O(\textit{bus length})$ broadcasting cost is assessed, the component labeling complexity becomes $O(n^2 \log n)$. We show how to extend these radix-based methods to 3D images and meshes with appropriate reconfigurability.

We have considered five distinct 3D component labeling algorithms. Algorithm 1 applies shrink-expand labeling to 2D planes with merging operations between planes. Algorithm 2 uses just merging steps with a form of dynamic broadcasting in a 3D form of divide and conquer. Algorithm 3 uses another variant of the methods in Algorithm 2. Algorithm 4 uses a mixture of shrink-expand approaches in planes with propagation labeling. Finally, Algorithm 5 is the extension of 2D radix-based labeling to 3D.

We summarize worse case asymptotic computation time below:

Methods	3D Mesh without reconfigurability	Reconfigurable 3D Mesh with $O(1)$ broadcast time	Reconfigurable 3D Mesh with $O(\textit{bus length})$ broadcast time
Algorithm 1	$O(n^2 logn)$	$O(n^2 logn)$	$O(n^3 logn)$
Algorithm 2	$O(n^2)$	$O(n^2)$	$O(n^3)$
Algorithm 3	$O(n^2 logn)$	$O(n^2 logn)$	$O(n^3 logn)$
Algorithm 4	$O(n^3)$	$O(n^3)$	$O(n^3)$
Algorithm 5	not possible	$O(\log n)$	$O(n^3 logn)$

As expected radix-based approaches are preferable when the required reconfiguration is available and the actual broadcast times on the (potentially quite long) configured busses are not too high. When reconfiguration is not available, algorithm 2 shows the best asymptotic performance. Both algorithm 2 and algorithm 4 could have superior worse case time complexity if broadcast times are as bad as $O(bus\ length)$. We have not shown that reconfiguration can be used by algorithms 1-4 to im᾽rove worse case time, but reconfiguration can improve average time performance for these algorithms. An average time analysis is under development.

Segmentation of Magnetic Resonance Images:

Magnetic resonance imaging (MRI) is one of the most significant advances in medical imaging. [RAYA90, UDUP90] A multi-channel representation of head MRI images consists typically of three categories: proton density(PD), T1-weighted, and T2-weighted images. The head MRI images are usually partitioned into the following regions: white matter, gray matter, CSF, miscellaneous substances (such as tumor, bone, blood, etc.) and background. Our research interest is to estimate different structures and to extract the structure of soft-tissue in the head MRI images so that detailed classification of the substance regions can be generated.

Image segmentation on head MRI images can be divided into two major steps: one is brain region extraction; the other is tissue classification. First of all, the brain region is extracted from its surrounding bone(or skull) by using either an automatic segmentation algorithm or a morphological operation. [RUSS92] Then, the tissue classification can be performed by using three preprocessing techniques such as four-level quantization, refined regularization, and edge-preserved smoothing followed by a region-based segmentation which is usually a pre-defined threshold scheme. We will focus on the tissue classification problem in the following presentation.

It is more difficult to perform the tissue classification than the brain region extraction because the boundaries among the white matter, the gray matter and the CSF are not clear and well-defined. That is, the contrast among soft tissues is commonly weaker than that among hard tissues. In order to overcome this problem, some techniques for tissue classification are involved:

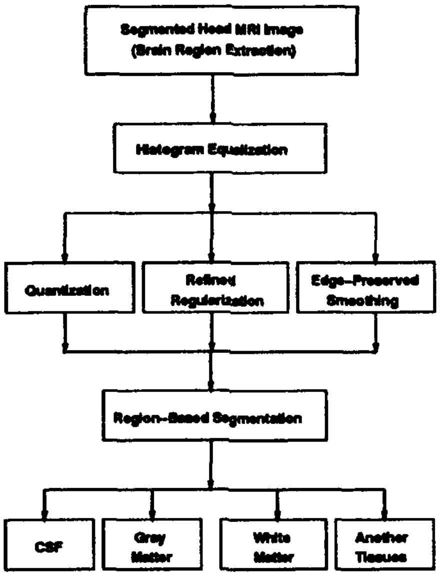
- **Histogram Equalization:** Histogram equalization[GONZ87] is typically a contrast enhancement process and is used to identify particular structures contained in the head MRI images.

- **Four-level Quantization:** In this approach, the thresholds T_0, T_1, T_2, and T_3 are chosen to give a quantized version of the head MRI image. The histogram of this quantized image will contain four dominant peaks. Then, four distinct regions can be extracted by simply picking up these four peaks to belong to various tissues in the head MRI images.

- **Refined Regularization:** The technique of refined regularization was presented in [POGG82], [GOKM90]. The technique of refined regularization is to minimize a so called membrane function where two terms are included as a measure of closeness between solution and data and a measure of smoothness. A regularization parameter λ is used to provide a tradeoff between these two measures.

- **Edge-preserved Smoothing:** The technique of edge-preserved smoothing was originally presented in [CHEN91a, CHEN92]. Edge-preserved smoothing(or adaptive smoothing) is designed to avoid the disadvantage of linear filtering which smooth edges as well as noise. It is achieved by repeatly convolving the signal with a very small average mask weighted by a measure of the signal continuity at each point.

Procedures of the tissue classification can be described as below:



The head MRI images after the brain region extraction are called the segmented head MRI images. In general, the boundaries among soft tissues in the segmented head MRI images are still very fuzzy. Therefore, histogram equalization can be applied to the segmented head MRI images to enhance the contrast among various tissues. That is, we not only get rid of the bone part which originally surrounds the brain region but also stretch the signal levels among various tissues in the brain region. After the histogram equalization, three various preprocessing techniques, say four-level quantization, refined regularization and edge-preserved smoothing, are utilized. These three techniques provide somewhat different descriptions about the boundaries among the soft tissues. In fact, the four-level quantization doesn't actually smooth the signal. But the refined regularization and the edge-preserved do smooth the signal in order to suppress the noise. A final region-based segmentation can be applied to acquire the results of the tissue classification. Generally speaking, the region-based segmentation using four-level quantization is done by picking up the four dominant peaks which represent background, white matter, gray matter and CSF respectively. The region-based segmentation using refined regularization or edge-preserved smoothing is done by applying the four-evel quantization on the resulting images after the preprocessing. In the experiment, the three main tissues such as white matter, gray matter and CSF are classified. Experimental results are given in Figure 47.

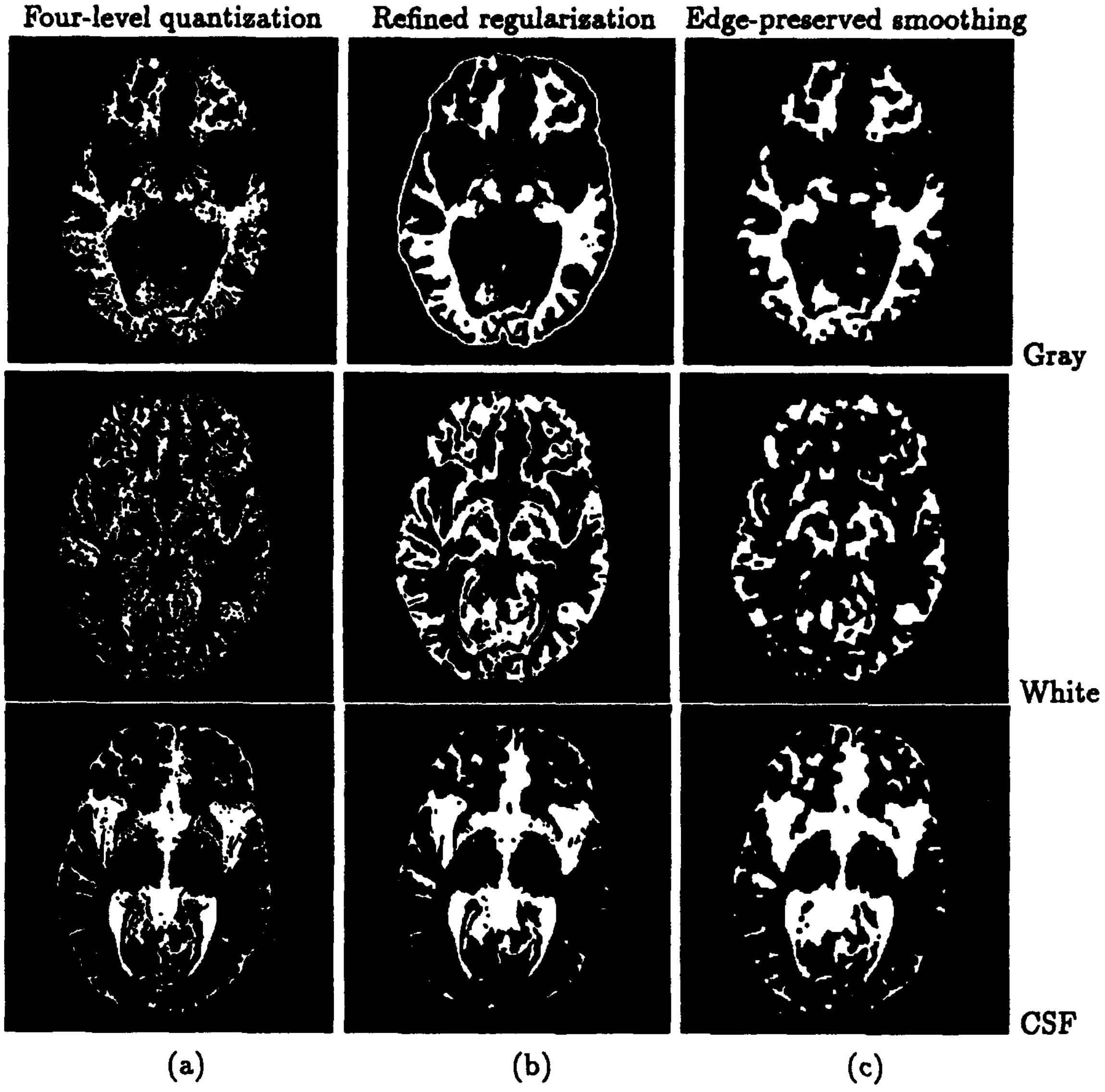


Figure 47: Results of tissue classification using (a) Four-level quantization (b) Refined regularization (c) Edge-preserved smoothing.

It can be observed that the results using three various approaches are somewhat different in visualization. Since the results using four-level quantization didn't deal with the noise, the resulting regions which correspond to various tissues are not well-defined and clear. However, the processes of the refined regularization and the edge-preserved smoothing tend to remove the noise from the head MRI images. Therefore, they are better in visualization based on the same region-based segmentation. However, it is not fair to say which approach provides the best segmentation results because the head MRI images contain too many complex patterns or shapes. A solution for this is to build certain brain models which may be ideal then perturb them so that performance evalu^t:on can be performed to compare the tissue classification algorithms.

We now consider the potential for improving performance evaluations using perturbation models to create test images. Performance evaluations, or say quantitative evaluations, are importan^t for measuring how well an algorithm performs. In general, it is very difficult to evaluate the performance or goodness of the tissue classification algorithms because ideal segmentation results are not typically known. Therefore, we need to build suitable models which form test images so as to provide a general study of the performance of various algorithms. We term such models, perturbation models, which can be also related to noisy models. Perturbation, or contrast adjustment, affects the sharpness and the quality of images. Usually, the perturbation destroys the ability to define an edge in images. Therefore, an edge can no more easily seen because the contrast is decreasing. Here, we are interested in certain perturbation models which change objects or structures in images by using certain perturbation operators which happen *locally* rather than *globally*.

Let us discuss how to build the perturbation models for gray-level images. Suppose we are given a gray-level image as

$$f(x,y) \ \ where \ \ 0 \le x \le N_x - 1, \ 0 \le y \le N_y - 1 \tag{3}$$

Let us perturb the image to be an image $f'(x,y)$ using a perturbation operator as follow:

$$f'(x,y) = O\{f(x,y)\} \tag{4}$$

Then,

$$f'(x,y) = f(x,y) + n(x,y) \tag{5}$$

where $n(x,y)$ is the adding noise. In our experiments, Gaussian noise will be used. Let us partition the whole image Ω into a structure of interest Ψ and a non-structure of interest Ψ', then

$$f'(x,y) = \begin{cases} f(x,y) + n(x,y) & if \ (x,y) \in \Psi \\ f(x,y) & if \ (x,y) \in \Psi' \end{cases} \tag{6}$$

Several prerequisite definitions are as follows:

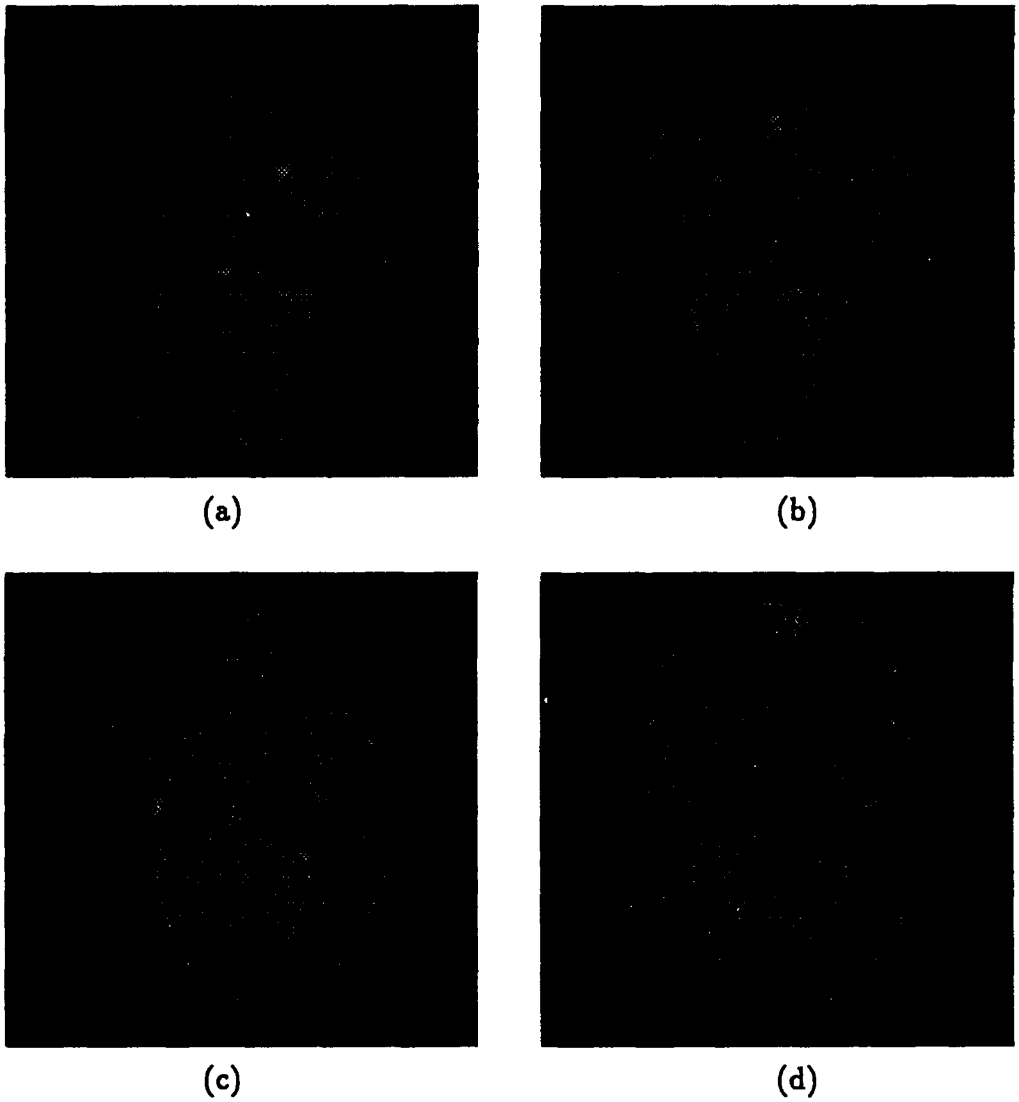
- **Edge Pixel:** In order to define edge pixels in gray-level images. An edge-based segmentation algorithm(or edge detection) is necessary. A simple approximation such as Sobel operator[GONZ87] can be utilized.

- **Near-Edge Pixel:** A pixel p is a near-edge pixel if and only if p(can be either 0 or 1) is 8-adjacent to an edge or more edges or p is an edge pixel.

All the near-edge pixels in the image are chosen to be the structure of interest Ψ. Therefore, a number of Gaussian random numbers which equals to the number of near-edge pixels in the image can be generated with various variances σ. These Gaussian random numbers are added to the original gray-level image to provide the perturbation models.

In the experiments, an ideal brain model which is given in Figure 48(a) was generated. The corresponding intensities for the white matter, the gray matter and the CSF are 85, 125, and 165 respectively. Also, three ideal segmentation images each of which contains only one substance can be generated. Then, the perturbation models are built by adding local Gaussian noise with various σ. Typical perturbation models using $\sigma = 10$, $\sigma = 20$, and $\sigma = 40$ can be shown in Figure 48(b), (c) and (d). It can be observed that the structure of near-edges in the perturbation model is changed due to the effect of the adding Gaussian random noise. The larger the variance σ, the more noisy the perturbation models will become.

After the perturbation models are built, one can apply the tissue classification algorithms so as to provide performance evaluations. Again, the four-level quantization, the refined regularization, and the edge-preserved smoothing are applied to the perturbation models with various σ to yield segmentation results. Then, error measurements which are defined by counting number of different pixels between the ideal segmentation image and the segmentation result by using exclusive OR logical operations. Therefore, three error measurements for the white matter, the gray matter and the CSF can be created. Also, total error measurements are defined by summing up all the three error measurements. Experimental results are given in Figure 49. It can be observed that the error measurements monotonically increase with respect to the variance σ. The errors caused by the four-level quantization increased much faster than the errors caused by either the refined regularization or the edge-preserved smoothing. Therefore, the quantization method is not robust when dealing with the noise even though it gives less errors when the variance σ is small. That is, the refined regularization or the edge-preserved smoothing yield a better tradeoff when dealing with the noise. Usually, the contrast among soft tissues in real human brain images is very fuzzy. That is, the differences between the edge and the noise in real human brain is not well-separated and can be very hard to classify. A better tradeoff is to choose the refined regularization for the segmentation purpose. This work is on-going with the long-term goal of identifying useful 3D texture operators for MRI segmentation.



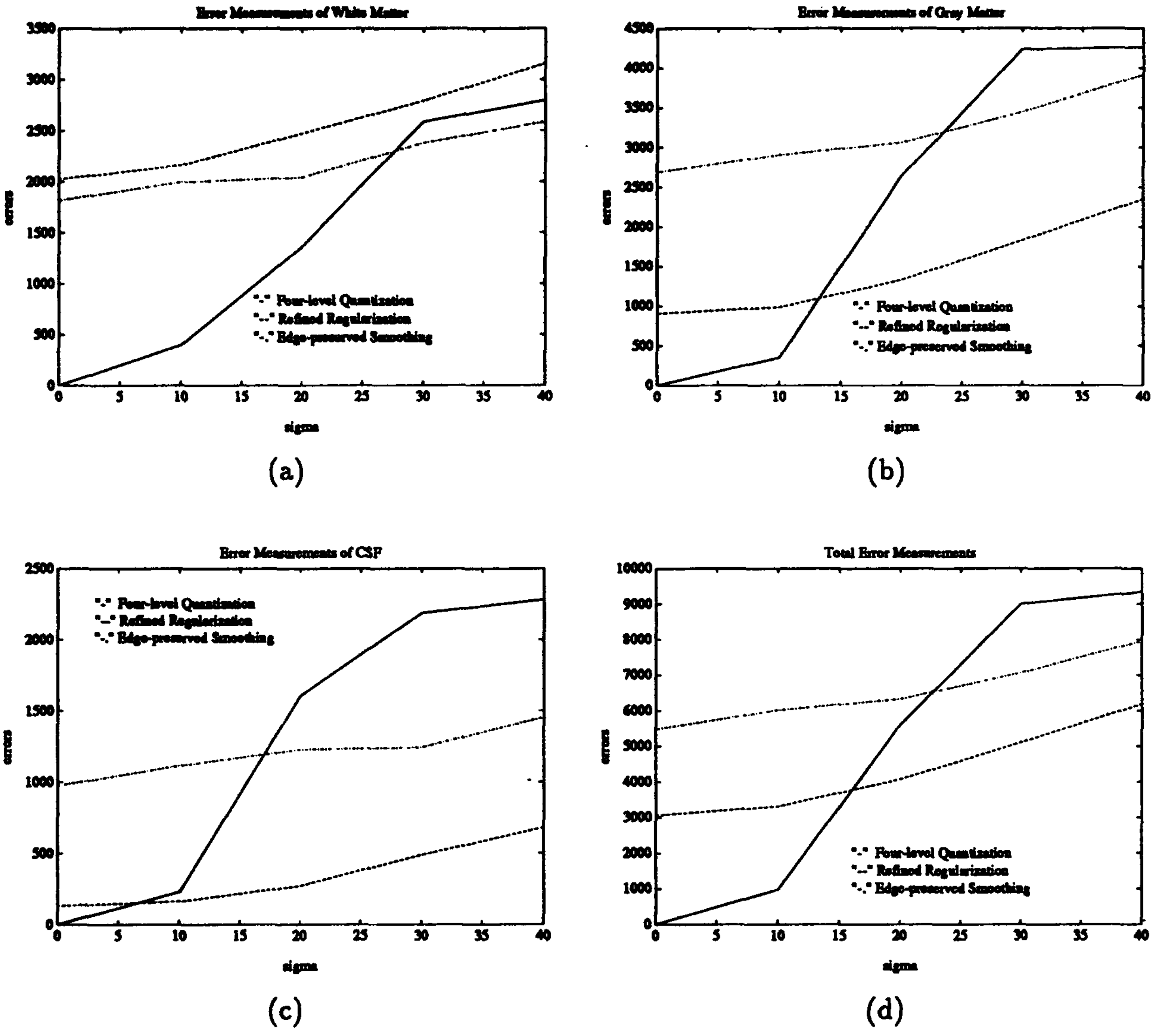
(a)

(b)

(c)

(d)

Figure 48: Results of perturbation model for gray-level image (a) Ideal Model (b) Perturbation model using $\sigma = 10$, (c) Perturbation model using $\sigma = 20$ (d) Perturbation model using $\sigma = 40$.



(a)

(b)

(c)

(d)

Figure 49: Error measurements of various algorithms for (a) White matter (b) Gray matter (c) CSF (d) Total Error Measurements.

B.2.e Topology Preservation for Paralleı Algorithms

It is a fundamental goal in the application of parallel computing to make algorithms as fully parallel as possible in order to utilize the potential of a large number of processors. To achieve this one must develop a deep understanding of the communication demands and fundamental nature (e.g. required support sizes, potential for fully parallel operation with limited supports, sufficient conditions for correct operation etc.) of the particular algorithms under study. Often in the processing of images we will want to transform binary images such that the topology (sometimes understood as connectivity) of an image is not altered or is altered only in some constrained way. Example algorithms are thinning [GUOH92], shrinking to a residue [GOKH90], and algorithms like connected component labeling which make use of shrinking [CYPH90]; but, the problem of topology preservation is fundamental in more general contexts including mathematical morphology approaches to image processing. [SERR82] It is well understood how to preserve topology for sequential applications of 2D or 3D operators [KONG89]. Topology preservation for parallel algorithms is less well understood in the community and published algorithms frequently fail to preserve topology [GONG90b], [HALL92a], or are inadequately proven. This can lead to serious failure in subsequent processing, revealing a need for greater attention to these issues. The identification of good tests for connectivity preservation can aid in developing useful design spaces which characterize classes of acceptable algorithms from which designers can choose desirable instances. Support limit constraints can help designers focus on the smallest acceptable operator supports. A few of our results on support limits for topology preserving operators, characterization of design spaces, and topology preservation tests are given in the following.

Fundamental Limits in 2D Thinning:

In [HALL93b] we examine fundamental limits on fully parallel 2D thinning operators. We have shown that eleven–pixel supports are the smallest possible for an 8-4 image space and, in particular, 3×3 supports are insufficient. Further, the possible locations for the eleven pixels are tightly constrained. It is shown that the optimally small support of any fully parallel thinning operator, **O**, is composed of 11 pixels which include the pixel, p, under consideration by **O**; the eight neighbors of p ($< w >$ in the illustration below); and two additional pixels satisfying one of the following conditions (given with reference to the illustration below):

a. Exactly one pixel is drawn from the set $\{x_i \mid i = 1, \cdots, 6\}$ and exactly one pixel is drawn from the set $\{y_i \mid i = 1, \cdots, 6\}$; or

b. The two pixels are one of the following pairs: $\{z_1, x_1\}$, $\{z_1, y_6\}$, $\{z_2, x_3\}$, $\{z_2, y_1\}$, $\{z_3, y_3\}$, $\{z_3, x_4\}$, $\{z_4, x_6\}$, or $\{z_4, y_4\}$.

The denoted region around p is illustrated below:

$$\begin{array}{ccccc}
z_1 & x_1 & x_2 & x_3 & z_2 \\
y_6 & w & w & w & y_1 \\
y_5 & w & p & w & y_2 \\
y_4 & w & w & w & y_3 \\
z_4 & x_6 & x_5 & x_4 & z_3
\end{array}$$

The results in [HALL93b] can also be used to demonstrate that operators with the infinite support $p \cup < s >$ illustrated in Figure 50 (and its 90° rotation) cannot provide adequate fully parallel thinning. Of course, any subset of this inadequate support is also inadequate; thus, a very large class of inadequate supports has been identified.

$$\begin{array}{ccccccccccccccc}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & s & s & s & s & & & & s & s & s & s & \cdot & \cdot \\
\cdot & \cdot & s & s & s & s & & & & s & s & s & s & \cdot & \cdot \\
\cdot & \cdot & s & s & s & s & & & & s & s & s & s & \cdot & \cdot \\
 & & & & & & s & s & s & & & & & & \\
\cdot & \cdot & s & s & s & s & s & p & s & s & s & s & s & \cdot & \cdot \\
 & & & & & & s & s & s & & & & & & \\
\cdot & \cdot & s & s & s & s & & & & s & s & s & s & \cdot & \cdot \\
\cdot & \cdot & s & s & s & s & & & & s & s & s & s & \cdot & \cdot \\
\cdot & \cdot & s & s & s & s & & & & s & s & s & s & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}$$

Figure 50: An infinite set of pixels, $p \cup < s >$, which cannot be a support for any connectivity preserving fully parallel thinning operator.

Design Spaces for 2D Connectivity Preserving Reduction Operators:

We have also characterized a robust class of connectivity preserving fully parallel thinning algorithms [HALL93b] based on connectivity preservation tests developed in [HALL92b] and [RONS88]. A typical fully parallel thinning operator would satisfy the following deletion conditions (termed FPT):

a. p is 8-simple

b. p is not an endpoint

There will be certain cases where p must be preserved from deletion in order to preserve connectivity. Consider 4-adjacent 1's p and q both of which satisfy the FPT deletion conditions, but which are not an 8-deletable set and should not both be deleted. We determine that the only such cases are those shown in Figure 51, where pairs of adjacent pixels, $\{a, b\}$, are chosen to be 8-simple (i.e. $a = 1$ and $b = 0$ is not allowed) and the pixels labeled x can be either 0's or 1's. Certain pairs of these conditions are mirror images or 180° rotations of each other. Preservation conditions must be added to the FPT deletion conditions to guarantee that at least one of p and q is not deleted in each case. Finally, we must avoid the complete deletion of a 2 × 2 squ component of 1's. These FPT preservation conditions reveal the design space for all connectiv preserving fully parallel thinning operators which satisfy the FPT deletion conditions.

$$\begin{array}{c}\begin{array}{cccc} a & 0 & 1 & x \\ b & p & q & b \\ x & 1 & 0 & a \end{array}\\[1em] (1)\end{array}\qquad\begin{array}{c}\begin{array}{cccc} a & 0 & 1 & x \\ b & p & q & 0 \\ x & 1 & 1 & x \end{array}\\[1em] (2)\end{array}\qquad\begin{array}{c}\begin{array}{cccc} 0 & 0 & 1 & x \\ 0 & p & q & 0 \\ 0 & 0 & 1 & x \end{array}\\[1em] (3)\end{array}\qquad\begin{array}{c}\begin{array}{cccc} x & 1 & 0 & a \\ b & p & q & b \\ a & 0 & 1 & x \end{array}\\[1em] (4)\end{array}\qquad\begin{array}{c}\begin{array}{cccc} x & 1 & 0 & a \\ 0 & p & q & b \\ x & 1 & 1 & x \end{array}\\[1em] (5)\end{array}$$

$$\begin{array}{c}\begin{array}{cccc} x & 1 & 0 & 0 \\ 0 & p & q & 0 \\ x & 1 & 0 & 0 \end{array}\\[1em] (6)\end{array}\qquad\begin{array}{c}\begin{array}{cccc} x & 1 & 1 & x \\ b & p & q & 0 \\ a & 0 & 1 & x \end{array}\\[1em] (7)\end{array}\qquad\begin{array}{c}\begin{array}{cccc} x & 1 & 1 & x \\ 0 & p & q & b \\ x & 1 & 0 & a \end{array}\\[1em] (8)\end{array}\qquad\begin{array}{c}\begin{array}{cccc} x & 1 & 1 & x \\ 0 & p & q & 0 \\ x & 1 & 1 & x \end{array}\\[1em] (9)\end{array}$$

Figure 51: Cases where the 1's p and q are both FPT–deletable but $\{p, q\}$ is not 8-deletable. The 90° rotations of each illustrated case also satisfy these conditions. Pixels labeled x are 0 or 1; and either $a = 0$ or $b = 1$ in each adjacent pair a, b

These results apply naturally to any connectivity preserving reduction operator by removing the endpoint condition from the FPT conditions. It is easy to show examples of algorithms satisfying these conditions in which the preservation conditions are relatively simple. For example:

Algorithm FPTN

The following parallel reduction operator is applied repeatedly. A pixel $p = 1$ is deleted if it satisfies the FPT deletion conditions, and its neighborhood does not satisfy any of the following conditions (or the 90° rotations of a–e):

$$\begin{array}{ccc} & 1 & c \\ 0 & p & 1 \\ & 1 & d \end{array} \qquad \begin{array}{ccc} d & 1 & \\ 1 & p & 0 \\ c & 1 & \end{array} \qquad \begin{array}{cc} 0 & 1 \\ p & 1 \\ 1 & 0 \end{array} \qquad \begin{array}{cc} 1 & 0 \\ p & 1 \\ 0 & 1 \end{array} \qquad \begin{array}{cccc} & 1 & 1 & \\ 0 & p & 1 & 0 \\ & 1 & 1 & \end{array} \qquad \begin{array}{cccc} 0 & 0 & 0 & \\ 0 & p & 1 & 0 \\ 0 & 1 & 1 & \\ & 0 & & \end{array}$$

(a) (b) (c) (d) (e) (f)

where $\{c, d\}$ contains at least one 0. The algorithm terminates when no deletions occur at an iteration.

Condition (a) preserves p from deletion in cases (5), (6), and (8) (Figure 51); condition (b) preserves q in cases (2), (3), and (7); conditions (c), (d) and (e) preserve q in cases (1), (4) and (9), respectively; and condition (f) prevents deletion of the 2×2 square. This operator has an optimally small eleven–point support.

Fundamental Limits in 2D Reduction-only Shrinking:

We can also apply reduction operators to the shrinking to a residue problem; but, we can show that no local reduction-only shrinking operator, O, can successfully shrink all images to residues. In particular it can be shown that there exists no fully parallel connectivity preserving reduction operator which can reduce a component in S with holes to a residue. Given this we will restrict our attention to simply connected components in images. We can devise fully parallel shrinking algorithms using operators, O, with local supports; but, the supports must be sufficiently large. Rosenfeld has shown that 3×3 supports are not adequate and has shown certain adequate supports for 4-8 spaces [ROSE70]. For example, he has shown that the support $< s > \cup\, p$ below

$$\begin{array}{cccc} & s & & \\ s & s & s & \\ s & p & s & s \\ s & s & s & \end{array}$$

(or its 90° rotations) is required for fully parallel reduction-only shrinking operators in 4-8 spaces and he demonstrates the existence of such an operator.

Similar arguments are used now to address the required support size for the 8-4 case. If we assume the shrinking operator, O, must delete exactly one 1 in each of the 2 pixel components:

$$\begin{array}{c} a \\ b \end{array} \qquad \begin{array}{cc} a & b \\ & \end{array} \qquad \begin{array}{cc} a & \\ & b \end{array} \qquad \begin{array}{cc} & a \\ b & \end{array}$$

where $a = b = 1$. For example let 1's denoted a be deleted by O, then 1's denoted b must not be deleted by O. If the support for O applied at p does not include $< e >$ below:

$$\begin{array}{cccc}
e & e & e & \\
e & 1 & 0 & 0 \\
e & 0 & p & 0 \\
 & 0 & 0 & 0
\end{array}$$

then O cannot delete p. But then we can create an irreducible image as an arc composed of all non-8-simple points except for 2 endpoints each satisfying the above condition for p, e.g.

$$\begin{array}{ccccccccccc}
 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & & \\
1 & & & & & & & & 1 & & \\
1 & & & & & & & & & 1 & \\
 & p & & & & & & & & & p
\end{array}\quad .$$

No interior points can be deleted by O (as this would violate FC1) and the image is irreducible. Similar arguments can be applied to each 2 pixel component and one can demonstrate that the following 19 point support, $p \cup < s >$,

$$\begin{array}{ccccc}
s & s & s & s & s \\
s & s & s & s & s \\
s & s & p & s & s \\
s & s & s & s &
\end{array}$$

is necessary for any operator O which preserves b's in the two-pixel components above. There are 16 distinct ways one can delete or preserve the a's and b's in the two-pixel components each of which leads to a slightly different constraint on the support for O. The minimum support size is 19 points as illustrated above. Further, there exist operators using this support which will correctly shrink all simply connected components in 8-4 spaces, e.g.:

Algorithm ROSHR

The following reduction operator is applied repeatedly. A pixel p=1 is deleted if

a. $C(p) = 1$ and

b. Either of the following conditions is satisfied:

1. $p_2 = 0$ or

2. $N_8^*(p)$ contains exactly one 1;

c. Except, do not delete p for any of the following 4 conditions:

$$\begin{array}{cccc}
0 & 0 & 0 & \\
0 & 1 & 0 & 0 \\
0 & 0 & p & 0 \\
 & 0 & 0 & 0
\end{array}\quad
\begin{array}{cccc}
 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & p & 0 & 0 \\
0 & 0 & 0 &
\end{array}\quad
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & p & 0 \\
0 & 0 & 0
\end{array}\quad
\begin{array}{cccc}
 & & & \\
0 & 0 & 0 & 0 \\
0 & 1 & p & 0 \\
0 & 0 & 0 & 0
\end{array}\; .$$

The algorithm stops when the image space contains only residues.

This is closely related to the classical parallel thinning algorithm in [ROSE75], but deletes only north border 8-simple points and endpoints.

Designers of parallel algorithms typically wish to use operators with small support. These results on support limitations illustrate to algorithm designers exactly how small their operator supports can be and illustrate the relative "shape" of such supports.

Connectivity Preservation Tests for Parallel 2D Reduction–augmentation Operators:

Reduction-augmentation operators transform binary images by taking certain foreground pixels to the background (reduction) and certain background pixels to the foreground (augmentation). Such operators have found application in parallel shrinking to a residue algorithms [LEVI72,GOKH90] (which are in turn fundamental operators in certain connected component labeling approaches [CYPH90,ALNU91] and as separate reduction and augmentation operations in image smoothing (say using mathematical morphology operators [HARA92], concavity filling [SKLA76], etc. The notion of a connected component as a fundamental region (or object) in an image is frequently used in computer vision research. Thus, we are often interested in operators which preserve connectivity (topology). Connectivity preservation tests for reduction-only operators (and by *symmetry* augmentation-only operators) are well understood and well developed for 2D images [RONS88,HALL92b]. But, it can be desirable to perform reduction and augmentation operators simultaneously. For example, this would allow all PE's in a 2D mesh with an image embedded one pixel/PE to be active in a given application of a reduction-augmentation operator. Thus, there is some need to develop approaches for verifying such connectivity preserving parallel operators. Simultaneous reduction and augmentation raises some new issues beyond reduction-only approaches. For example, it is not as easy to be clear about what it means to preserve connectivity as compared to reduction-only cases.

In this part of our work (reported more fully in [HALL94]) we have presented a new approach for characterizing connectivity preservation properties for parallel reduction-augmentation operators. From this special classes of reduction-augmentation operators have been defined and tests are demonstrated for verifying these connectivity preservation properties for arbitrary operators within these classes. The tests are illustrated with a proof of Levialdi's shrinking to a residue algorithm [LEVI72]. A few results are summarized below.

It is not immediately obvious how to define connectivity preservation for a reduction–augmentation operator. Consider, for example, what happens when the Levialdi shrinker [LEVI72] is applied to the image

$$\begin{array}{ccc} & & c \\ a & q & \\ & b & \end{array}$$

where $a = b = c = 1$ and $q = 0$. All of the 1's are reduced by LEV and q is augmented to a 1. Thus, the components $\{a, b\}$ and $\{c\}$ of 1's have been completely reduced, and a new component $\{q\}$ has been created. How can we say that connectivity has been "preserved"? Of course, we can allow the reduction of $\{c\}$ since it is a residue; but we need to establish a basis on which we can say that $\{q\}$ is the successor component to $\{a, b\}$. In general we need to understand how we relate augmented 0's to the components of 1's which existed when the augmentation occurred.

A simple way to do this is to imagine that components of 1's are originally labeled with distinct component labels. (This is a theoretical notion useful in evaluating algorithm correctness.) Then we assign component labels to augmented 0's based on some rules. Similarly, if we are also concerned about preservation of background connectivity, we would imagine an original labeling of components of 0's and utilize rules to assign component labels to reduced 1's. Classes of rules are developed here which encompass typical reduction–augmentation operators. Typically we assign labels to altered pixels based on the labels of special sets of 1's or 0's in the neighborhood of the altered pixel. We define the following special class of operators with restricted augmentation condition:

Definition B..1 *Operators which satisfy the following conditions are termed* augmentation–elementary*:*

1. *For any given 0, q, we define a set of sets of 1's, each defined over $N_k(q)$, at least one of which is* necessary *to allow the augmentation of q. These sets of 1's are termed the* augmentation sets*. The set of augmentation sets is denoted $R_A(q)$ and each set in $R_A(q)$ is denoted $R_a^i(q)$. (We will leave out the superscript if there is no ambiguity.)*
2. *We require that each $R_a^i(q)$ is m-connected and m-adjacent to q and each pair of $R_a^i(q)$'s is m-connected.*

Thus, for augmentation-elementary operators the set of 1's in $R_A(q)$ is m-connected in $N_k^*(q)$.

When we are concerned solely about foreground connectivity preservation (as in shrinking) we only constrain the operators to be augmentation–elementary. Similar conditions restricting reduction allow preservation of background connectivity also. Now we restate foreground connectivity preservation conditions based on the new labeling metaphor:

FC1L. O must guarantee that 1's bearing the same label are m-connected in S;

FC2L. O must guarantee that no foreground label becomes extinct;

FC3L. O must guarantee that 1's carrying distinct labels are not m-connected in S;

FC4L. O must guarantee that no new label is generated.

Now we state the main connectivity tests. Variable m refers to the foreground connectivity: $m = 8$ for 8-4 image spaces; $m = 4$ for 4-8 image spaces; and $m = 6$ for hexagonal image spaces. Variable $k = 8$ for rectangu!.r 8-4 and 4-8 spaces and $k = 6$ for hexagonal spaces. $N^*_m(p)$ refers to the m-neighbors of pixel p. The special class of augmentation-elementary operators defined above guarantees FC4L, so we need not define a test for that condition.

FC1L Test: Whenever a 1, p, can be reduced by O and non-m-adjacent x, y belong to $N^*_m(p)$, where x and y are 1's or 0's augmentable by O (such that p belongs to or is m-adjacent to an augmentation set of such 0's), then there is an m-path from x to y in $N^*_k(p)$ containing 1's not reduced by O or 0's augmented by O. Such augmentable 0's must carry the same label as p.

FC2L Test 1: Whenever a 1, p, can be reduced by O, there are pixels of $N^*_m(p)$ which are 1's not reduced by O or 0's augmented by O carrying the label of p.

For operators which satisfy FC1L there is an alternative, possibly simpler, test for FC2L:

FC2L Test 2: Determine for each m-component, G, composed solely of mutually m-adjacent 1's that either one or more 1's are not reduced or that there are augmentable 0's, q, m-adjacent to G such that $R_a(q)$ meets G or contains 1's m-adjacent to G.

FC3L Test: FC3L is satisfied if both of the following conditions are satisfied:

1. Whenever a 0, q, is augmentable by O then any 1 in $N^*_m(q)$ which is not reduced by O carries the same label as q.
2. Whenever a 0, q, is augmentable by O then any other augmentable 0 in $N^*_m(q)$ will carry the same label as q.

Special classes of augmentation operators for which FC3L is even easier to confirm are given in [HALL94].

The tests are all local in nature and the required case checking is of small to moderate complexity. *These tests offer algorithm designers* a convenient way to verify connectivity preservation for prospective new reduction-augmentation operators. The restrictions due to the special reduction–augmentation operator classes, for which the results apply, are typical of those which appear in existing reduction-augmentation algorithms, so the tests are believed to apply to robust classes of operators. The tests would be particularly useful for the design of connectivity preserving algorithms for fine grain realizations in 2D mesh architectures.

3D Connectivity Preservation with Parallel Subfields Operators :

There is still much to do in characterizing parallel topology preserving algorithms for 3D images. The existence of 3D meshes particularly increases interest in a complete understanding of topology preservation for 3D images undergoing transformation by parallel operators since such architectures offer us the opportunity to process local operators fully in parallel. Sufficient conditions for connectivity preservation have been derived for classes (particularly subfields classes) of parallel ***reduction*** algorithms, where parallel operators only transform object pixels to background pixels, and classes of ***reduction-augmentation*** algorithms, where parallel operators can transform both object and background pixels. [HALK92], [HALL92c] These derived conditions enable the characterization of "design spaces" [HALL93b] for such classes helping algorithm designers focus on appropriate sets of conditions. We will summarize a few of the new connectivity preservation tests in the following.

Subfields approaches have found some application in 3D image spaces as well as in 2D image spaces. [HAFF84], [HALK92] We will find here that connectivity preservation is substantially simplified with the use of subfields. We will consider two of the possible subfields definitions for rectangular 3D image spaces, a two and a four subfields partition. Associate an index over the range $[0, n-1]$ for each of the 3 orthogonal axes of the image space where the image space is of size $n \times n \times n$. Our two-subfields partition (2SF) assigns voxels to subfields 1 or 2 in alternate planes, along any given major axis, in the following manner:

1 2 1 2	2 1 2 1 ...
2 1 2 1	1 2 1 2 ...
1 2 1 2	2 1 2 1 ...
odd planes	even planes

Note that an important characteristic of 2SF is that no pair of 6-adjacent voxels belongs to the same subfield. Thus, since only one subfield of voxels can be changed within one parallel iteration, we can guarantee for 2SF that 6-adjacent voxels are not changing simultaneously. Our four-subfields definition 4SF is defined similarly assigning voxels to subfields 1, 2, 3 or 4 in alternate planes, along any given major axis, in the following manner:

1 3 1 3	4 2 4 2 ...
2 4 2 4	3 1 3 1 ...
1 3 1 3	4 2 4 2 ...
odd planes	even planes

For 4SF only the eight 26-neighbors of p can change simultaneously with p.

Details of the notation can be found in [HALL92c]. We are considering binary image spaces where voxel values are assigned from the binary set $\{0, 1\}$. The set S of 1's is referred to as the *foreground*, its complement, the set S' of 0's, the *background*. $N_i(p)$ refers to the set containing p and its i-adjacent neighbors.

We can characterize connectivity preservation (topology preservation) in a variety of ways. We have focussed on the following characterization where a 3D reduction operator, O, is said to be ***connectivity preserving*** if all of the following properties are satisfied in each iteration:

C1: O must not transform a 26-component of S into two or more distinct 26-components of S;

C2: O must not completely delete a 26-component of S;

C3: O must not merge distinct 6-components of S' into a single 6-component of S'; and

C4: O must not create a new 6-component in S'.

C5: O must not remove holes from S.

C6: O must not create holes in S.

We will use the following terms to help characterize 3D simple points, i.e. those points which can be deleted while preserving connectivity.

$NC1_{26}(p) \equiv$ the number of 26-components of 1's in $N^*_{26}(p)$.
$NC0_{18}(p) \equiv$ the number of 6-components of 0's in $N^*_{18}(p)$ which are 6-adjacent to p.

We say that a 3D reduction operator, O, belongs to *Class-B* if it only deletes 1's, p, which satisfy the following:

1. $NC1_{26}(p) = 1$ and
2. $NC0_{18}(p) = 1$.

These Class-B conditions constitute the 3D simple point characterization given in [MALA92].

We find that the following conditions are sufficient to preserve connectivity properties C1 – C4 for Class-B 2SF operators.

TC1-26: Given 2 1's, p and q, which are 18-adjacent but not 6-adjacent, p and q are not both deleted in one iteration when any of the following conditions hold for these $2 \times 2 \times 2$ cubes (viewed along each of the 3 major axes):

$$\begin{array}{cc} a & b \\ c & d \end{array} \quad \begin{array}{cc} p & 0 \\ 0 & q \end{array} \quad \begin{array}{cc} e & f \\ g & h \end{array} \quad \text{or} \quad \begin{array}{cc} a & b \\ c & d \end{array} \quad \begin{array}{cc} 0 & p \\ q & 0 \end{array} \quad \begin{array}{cc} e & f \\ g & h \end{array}$$

where $\{a, b, c, d\}$ and $\{e, f, g, h\}$ each contain at least one 1. Treating p and q symmetrically there are 6 distinct cases for these conditions.

TC2-26: No 2, 3 or 4 voxel component composed solely of mutually 18-adjacent, but not 6-adjacent, voxels can be completely reduced in 1 iteration.

Operators would typically be designed to satisfy these two tests by preserving from deletion certain 1's, which otherwise would be reduced by a Class-B operator.

We find that the following condition is sufficient to preserve connectivity properties C1-C4 for Class-B 4SF operators:

T4SF-26: No 2-voxel component composed of 26-adjacent, but not 18-adjacent, voxels can be completely reduced in 1 iteration. There are 4 such components.

Such operators need only preserve one 1 in each of the four 2-voxel test components. We offer arguments in [HALL92c] claiming that C5 and C6 are also satisfied. Given this we see that almost all Class-B 4SF operators are connectivity preserving and that little fuss is entailed in verifying such operators. Subfields models of computation appear quite promising especially as we expect that most 3D mesh applications will realize several voxels per PE which allows subfields models to be realized without additional computational cost. This is achieved by guaranteeing that all PE's hold voxels from each subfield so that there are no idle PE's in the mesh.

B.2.f Wavelet Transforms in Systolic Arrays

Systolic arrays can offer particularly good time performance for algorithms where the input signals can be continuously applied while the array remains full and active. We have developed particularly effective designs for the computation of wavelet decomposition and reconstruction for 1D signals.[CHUA92] These designs will handle all orthonormal wavelet bases. Given a signal window size of $W = 2m$, the systolic architecture is composed of m layers of 1D arrays. The maximum row length is p for row 0, where p is approximately 1/2 of the filters' support size. The VLSI area complexity of the rows decreases by 1/2 for each successive row; thus, overall area complexity is proportional to $2p$. This is very close to optimal. For large input signal length, N, the computation or reconstruction is completed in $\approx N$ clock periods. Multiple signals can also be pipelined through the systolic arrays with no degradation in computation time. These results are very promising for 1D signals.

B.2.g Compound Graph Networks for Enhancing Mesh Architectures

Although the pyramid emulations discussed earlier are a very promising cost-effective approach, pyramids have fundamental limitations when processing demands are strongly global [STOU86] as for sorting or many Hough transform applications. In these cases the pyramid connectivity (and that for the associated reconfigurable meshes emulating the pyramid) is inadequate slowing performance to that comparable to a mesh alone. Thus, in applications where better time performance is needed on strongly global tasks, networks with higher connectivity are desirable. The well-known binary hypercube is a powerful highly connected network useful for stongly global computations, but is relatively expensive to build in large instances. Certain forms of graph compounds have been identified which can produce performance approaching the hypercube but with substantially reduced hardware cost. (or network degree) [HAMD91a, HAMD91b] These graph compounds can be used to augment communication capability in a mesh to help achieve higher performance.

The RGC network [HAMD91b] is a constant degree network which is envisioned as an augmentation of a traditional 2-D mesh architecture. The RGC is formed by compounding a given number of atom graphs with a complete interconnection and this compounding is done for any desired number of levels. An example is illustrated in Figure 52 where the atom graph is a binary hypercube. (referred to as an RGC-CUBE) Table 11 illustrates the comparative asymptotic performance on various basic image processing algorithms (operating on an nxn image of size $N = n^2$ pixels) for two instances of an RGC-CUBE as compared to the 2-D mesh, pyramid, Mesh of Trees (MOT) [NATH83] and Mesh with global mesh (MGM) [CARL88], [PRAS89]. These results are not bad and RGC-CUBE appears to offer some potential in image processing, but most of the algorithms studied do not fully utilize the full interconnection capability of the RGC-CUBE.

The RCC class of interconnection networks is another form of complete graph compound, but with substantially greater bandwidth and node degree which grows with network size. [HAMD91a, HAMD91b] The RCC-CUBE instance in this class is able to efficiently emulate binary hypercube and achieves time performance very close to hypercube on a variety of strongly global algorithms, i.e. sorting and PRAM emulation, while maintaining a substantially lower degree and hardware cost. The RCC-FULL instance [HAMD91b] is able to outperform binary hypercube by performing sorting and PRAM emulation in $O(logN)$ time while still maintaining a reduced node degree as compared to hypercube. These high connectivity networks appear to be promising cost-effective alternatives to binary hypercube when strongly global problems must be solved.

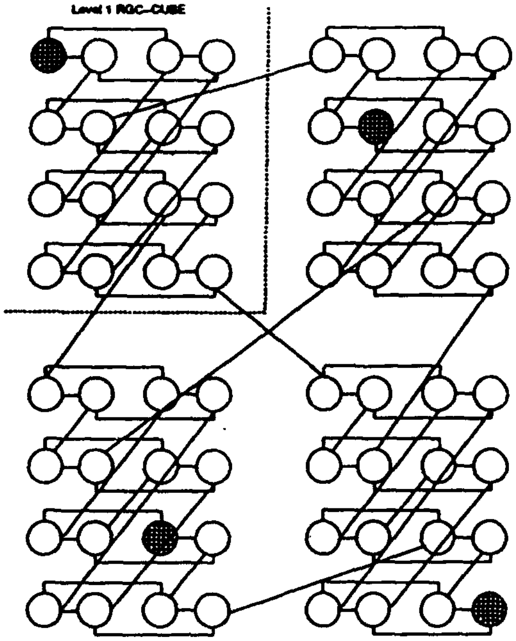


Figure 52: A 64-node example of a 2-RGC-CUBE. Shaded nodes (PE's) are available for expansion to higher level networks.

Network	Mesh	Pyramid	MOT	MGM	2-RGC-CUBE	3-RGC-CUBE
Global Maxima	$O(N^{1/2})$	$O(\log(N))$	$O(\log(N))$	$O(N^{1/4})$	$O(\log(N))$	$O(\log(N))$
Connected Component Labeling	$O(N^{1/2})$	$O(N^{1/4})$	$O(\log^3(N))$	$O(N^{1/4}\log^2(N))$	$O(N^{1/6}\log(N))$	$O(N^{1/4}\log(N))$
Convex Hull	$O(N^{1/2})$	$O(\log^2(N))$	$O(\log(N))$	$O(N^{1/4})$	$O(N^{1/6}\log(N))$	$O(N^{1/4}\log(N))$
Diameter	$O(N^{1/2})$	$O(N^{1/4}\log(N))$	$O(\log(N))$	$O(N^{1/4})$	$O(N^{1/6}\log(N))$	$O(N^{1/4}\log(N))$
Smallest Enclosing Rectangle	$O(N^{1/2})$	$O(N^{1/4}\log(N))$	$O(\log(N))$	$O(N^{1/4})$	$O(N^{1/6}\log(N))$	$O(N^{1/4}\log(N))$
Smallest Enclosing Circle	$O(N^{1/2})$	$O(N^{1/4}\log(N))$	$O(\log(N))$	$O(N^{1/4})$	$O(N^{1/6}\log(N))$	$O(N^{1/4}\log(N))$

Table 11: Asymptotic Time Complexities for Several Architectures for a Variety of Fundamental Image Processing Algorithms

References

[AHUJ84] N. Ahuja and S. Swamy, Multiprocessor pyramid architectures for bottom-up image analysis, *IEEE Trans. Pattern Anal. and Mach. Intell.*, PAMI-6, 1984, 463-475.

[ALBA91] M.G. Albanesi, V. Cantoni, U. Cei, M. Ferretti and M. Mosconi, Embedding pyramids into mesh arrays, in [LIST91].

[ALNU91] H.M. Alnuweiri and V.K. Prasanna, Fast image labeling using local operators on mesh-connected computers, *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-13, 1991, 202-207.

[ANAS82] D. Anastassiou and K. S. Pennington, Digital Halftoning of Images, *IBM J. Res. Develop.*, Vol. 26, No. 6, 1982, 687-697.

[ANTO92] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, Image Coding Using Wavelet Transform, *IEEE Trans. on Image Processing.* vol. 1, no. 2, April 1992, 205-220.

[ARCE72] C. Arcelli and S. Levialdi, Parallel shrinking in three dimensions, *Comput. Graphics Image Process.*, 1, 1972, 21-30.

[BOVI90] A. C. Bovik, M. Clark and W. S. Geisler, Multichannel Texture Analysis Using Localized Spatial Filters, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 12, No. 1, 1990, 55-73.

[BOXH78] G.E.P.Box, W.G.Hunter, and J.S.Hunter, Statistics for Experimenters, John Wiley & Sons, New York 1978.

[BURT83] P.J. Burt, The pyramid as a structure for efficient computation, in A. Rosenfeld, ed., *Multi-Resolution Systems for Image Processing*, 1983.

[CANN86] J. Canny, A Computational Approach to Edge Detection, *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 8, No. 6, 1986, 679-698.

[CANT88] V. Cantoni and S. Levialdi, Multiprocessor computing for images, *Proc IEEE*, 76, 1988, 959-969.

[CARL88] D.A. Carlson, Modified mesh-connected parallel computers, *IEEE Trans. Computers*, C-37, 1988, 1315-1321.

[CATT92] F. Catte, P-L Lions, J-M Morel and T. Coll, Image Selective Smoothing and Edge Detection by Nonlinear Diffusion, *SIAM J. Numer. Anal.*, Vol. 29, no. 1, 1992, 182-193.

[CHAN92] T. Chang and C. C. J. Kuo, A Wavelet Transform Approach to Texture Analysis, *Proc. IEEE ICASSP*, San Francisco, CA. 1992, 661-664.

[CHAN93] A. Chandra and R.W. Hall, 3D Component Labeling in 3D Mesh Architectures, submitted to *CVPR-94*, TR-LCVPR-93-11, Dept. of Electrical Engineering, University of Pittsburgh.

[CHEN91a] P. Saint-Marc, J.-S. Chen, and G. Medioni, Adaptive Smoothing: A General Tool for Early Vision, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 13, NO. 6, 1991, 514-529.

[CHEN91b] S. W. Chen and A. K. Jain, Object Extraction from Laser Radar Imagery, *Pattern Recognition*, Vol. 1, No. 6, 1991, 587-600.

[CHEN92] J.-S. Chen, Generalized Adaptive Smoothing for Multiscale Edge Detection, *SPIE Vol. 1708 Applications of Artifical Intelligence X: Machine Vision and Robotics*, 1992.

[CHUA92] H. Y. H. Chuang, H. J. Kim and C. C. Li, Systolic Architecture for Discrete Wavelet Transforms with Orthonormal Bases, *Proc. SPIE Conf. Vol. 1708, Appl. Artificial Intell. X: Machine Vision and Robotics*, Orlando, FL, April 1992, 157-164.

[CHUI92] C. K. Chui, *An Introduction to Wavelets*, Academic Press, San Diego, CA, 1992.

[COIF92] R. R. Coifman, Y. Meyer and V. Wickerhauser, Size Properties of Wavelet Packets, in *Wavelets and Their Applications*, eds., M. B. Ruskai, et al, Jones and Bartlett Publ., Boston, MA, 1992, 453-470.

[COLE92] D. Colella and C. Heil, The Characterization of Continuous, Four-Coefficient Scaling Functions and Wavelets, *IEEE Trans. Information Theory*, Vol. 38, No. 2, 1992, 876-881.

[COWI92] R. R. Coifman and M. V. Wickerhauser, Entropy-Based Algorithms for Best Basis Selections, *IEEE Trans. on Information Theory*, Vol. 38, 1992, 713-718.

[CYPH90] R. Cypher, J.L. Sanz, L. Snyder, Algorithms for image component labeling on SIMD mesh-connected computers, *IEEE Trans. Computers*, 39, 1980, 276-281.

[DAUG85] J. G. Daugman, Uncertainty Relation for Resolution in Space, Spatial Frequency, and Orientation Optimized by Two-dimensional Visual Cortical Filters, *J. Opt. Soc. Am.*, Vol.2, No.7, 1985, 1160-1169.

[DAUB88] I. Daubechies, Orthonormal bases of compactly supported wavelets, *Comm. on Pure and Appl. Math.*, vol. 41, 1988, 909-996.

[DAUB90] I. Daubechies, The wavelet transform, time-frequency localization and signal analysis, *IEEE Trans. on Information Theory*, vol. 36, no. 5, 1990, 961-1005.

[DAUB91a] I. Daubechies and J. C. Lagarias, Two-Scale Difference Equations: I. Existence and Global Regularity of Solutions, *SIAM J. Math. Anal.*, Vol. 22, No. 2, 1991, 554-572.

[DAUB91b] I. Daubechies, S. Jaffard, and J.-L. Journé, A simple Wilson orthonormal basis with exponential decay, *SIAM Jour. Math. Anal.*, vol. 22, no. 5, 1991, 1388-1410.

[DAUB92a] I. Daubechies and J. C. Lagarias, Two-Scale Difference Equations: II. Local Regularity, Infinite Products of Matrices and Fractals, *SIAM J. Math. Anal.*, Vol. 22, No. 4, 1992, 1031-1079.

[DAUB92b] I. Daubechies, *Ten Lectures on Wavelets*, vol. CBMS 61 of *CBMS-NSF Series in Applied Mathematics*, SIAM, Philadelphia, PA, 1992.

[DEVO92] R. A. DeVore, B. Jawerth and B. J. Lucier, Image Compression Through Wavelet Transform Coding, *IEEE Trans. on Information Theory*, vol. 38, no. 2, March 1992, 719-746.

[DUFF86] M.J. B. Duff and T.J. Fountain, Eds., *Cellular Logic Image Processing*, New York: Academic Press, 1986.

[FARR91] F. Farrokhnia and A. K. Jain, A Multi-Channel Filtering Approach to Texture Segmentation, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Maui, HI. 1991, 364-370.

[FOBA92] F. L. Fontaine and S. Basu, Multiresolution Approach to Solving Diffusion Equation for Edge Detection, *Proc. IEEE Sym. on Circuits and Systems*, San Diego, CA, 1992, 975-978.

[FOUN90] T.J. Fountain, The CLIP7 Programme, in *Multiprocessor Computer Architectures*, T.J. Fountain and M.J. Shute, eds., North-Holland, Amsterdam, 1990.

[FRMA92] J. Froment and S. Mallat, Second Generation Compact Image Coding with Wavelets, *in Wavelets: A Tutorial in Theory and Applications*, ed. C. K. Chui, Academic Press, 1992, 655-678.

[GOKM90] M. Gokmen, Edge Detection Using Regularization Theory, *PhD Dissertation*, Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA, 1990.

[GOKH90] M. Gökmen and R. W. Hall, Parallel shrinking algorithms using 2-subfields approaches, *Computer Vision Graphics Image Process.*, 52, 1990, 191-209.

[GOLI92] M. Gokmen and C. C. Li, Multiscale Edge Detection Using First Order R-Filter, *Proc. 11th International Conference on Pattern Recognition*, Vol. III, Hague, The Netherlands, 1992, 307-310.

[GOLI93] M. Gokmen and C. C. Li, Edge Detection and Surface Reconstruction Using Refined Regularization, *IEEE Trans. on Pattern Anal. and Mach. Intell.*, Vol. 15, no. 5, 1993, 492-499.

[GONG90a] W. Gong and G. Bertrand, A simple parallel 3D thinning algorithm, *Proc. 10th Intl. Conf. Patt. Recog.*, Vol 1, Atlantic City, 1990, 188-190.

[GONG90b] W. Gong and G. Bertrand, A note on "Thinning of $3-D$ images using the safe point thinning algorithm (SPTA), *Patt. Recog. Lett.*, 11, 1990, 499-500.

[GONZ87] R.C. Gonzalez and P. Wintz, *Digital Image Processing*, Addison-Wesley Publishing Co., 1987.

[GUOH92] Z. Guo and R.W. Hall, "Fast Fully Parallel Thinning Algorithms", *CVGIP: Image Understanding*, 55, 1992, 317-328.

[HAFF84] K.J. Hafford and K. Preston, Three-dimensional skeletonization of elongated solids, *Comput. Vision Image Process.*, 27, 1984, 78-91.

[HALL92a] R.W. Hall, Comments on "A parallel-symmetric thinning algorithm" by Bourbakis, *Patt. Recog.*, 25, 1992, 439-441.

[HALL92b] R.W. Hall, Tests for connectivity preservation for parallel reduction operators, *Topology and Its Applications*, 46, 1992, 199-217.

[HALL92c] R.W. Hall, Connectivity preserving parallel operators in 2D and 3D images, *Vision Geometry*, SPIE, Boston, Massachusetts, November 1992, 172-183.

[HALL93a] R.W. Hall, Ş. Küçük and M. Hamdi, Wavelet transform embeddings in mesh architectures, *Proceedings of the 1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, June 1993, 596-597.

[HALL93b] R.W. Hall, Optimally small operator supports for fully parallel thinning algorithms, *IEEE Trans. Patt. Anal. Mach. Intell*, 15, 1993, 828-833.

[HALL93c] R.W. Hall, Ş. Küçük and M. Hamdi, Wavelet Transform Embeddings in Mesh Architectures, submitted to *Pattern Recognition*, TR-LCVPR-93-10, Dept. of Electrical Engineering, University of Pittsburgh.

[HALL94] R.W. Hall, Connectivity Preservation Tests for Parallel Reduction-Augmentation Algorithm, submitted to *ICPR-94*, TR-LCVPR-94-01, Dept. of Electrical Engineering, University of Pittsburgh.

[HALK92] R.W. Hall and Ş. Küçük, Parallel 3D Shrinking Algorithms Using Subfields Notions, *Proceedings 11-th International Conference on Pattern Recognition*, The Hague, The Netherlands, Aug. 30 - Sept. 3, 1992, 395-398.

[HAMD91a] M. Hamdi, Communication-efficient interconnection networks for parallel computations, Ph.D. Dissertation, Electrical Engineering Dept., University of Pittsburgh, July 1991.

[HAMD91b] M. Hamdi and R.W. Hall, Compound graph networks for parallel image processing, *Computer Architecture for Machine Perception*, B. Zavidovique and P-L. Wendel, eds., Paris, France, December 1991, 355-366.

[HARA92] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision, Vol.1* Addison Wesley, 1992, chap. 2.

[HART93] F. Hartung, Wavelet and Subband Coding of Images: A Comperative Study, *Proc. SPIE, Vol. 2034, Mathematical Imaging: Wavelet Applications in Signal and Image Processing*, San Diego, CA, July 1993.

[HOLI92] J. W. Hong and C. C. Li, Structured Networks Based on Wavelet Transforms for System Identification, submitted to *IEEE Trans. on Neural Networks*; *Tech. Report TR-SP-92-03, Department of Electrical Engineering, University of Pittsburgh*, Pittsburgh.

[HSLI93] H. C. Hsin and C. C. Li, An Experiment on Texture Segmentation using Modulated Wavelets, *Proc. IEEE Intern. Conf. on Systems, Man and Cybernetics*, Vol. 4, Le Touquet, France, 1993, 529-534; *Tech. Report TR-CV-93-04, Department of Electrical Engineering, University of Pittsburgh*, Pittsburgh.

[JACK] J.H. Jackson, The data transport computer: A 3-dimensional massively parallel SIMD computer, WAVETRACER, Inc., Acton, MA.

[JULE83] B. Julesz and J. R. Bergen, Textons, The Fundamental Elements in Preattentive Vision and Perception of Textures, *Bell System Technical Journal*, Vol. 62, No. 6, 1983, 1619-1645.

[KIKL92] H. Kim, W. Kim and C. C. Li, A Performance Study of Two Wavelet-Based Edge Detectors, *Proc. 11th International Conference on Pattern Recognition*, Vol. III, Hague, The Netherlands, 1992, 302-306.

[KILI93] H. Kim and C. C. Li, A Non-orthogonal Wavelet Edge Detector with Four Filter-Coefficients, *Proc. SPIE, Vol. 2034, Math. Imaging: Wavelet Applications in Signal and Image Processing*, San Diego, CA, 1993, 115-126. *Tech. Report TR-CV-93-02, Department of Electrical Engineering, University of Pittsburgh*, Pittsburgh.

[KIML93] W. Kim and C. C. Li, Uncertainty Constants of Daubechies' and B-spline Wavelets, *Tech. Report TR-CV-93-05, Department of Electrical Engineering, University of Pittsburgh*, Pittsburgh, June 1993.

[KONG89] T.Y. Kong and A. Rosenfeld, Digital topology: Introduction and survey, *Comput. Vision Graphics Image Process.*, 48, 1989, 357-393.

[LAIN92] A. F. Laine and J. Fan, Adaptive Approach for Segmentation by Multichannel Wavelet Frames, *Proc. SPIE Vol. 2034, Math. Imaging: Wavelet Applications in Signal and Image Processing*, San Diego, CA, 1993, 228-299.

[LAWA92] K. S. Lau and J. Wang, Characterization of L^p-Solutions for The Two-Scale Dilation Equations, Preprint, Department of Mathematics and Statistics, University of Pittsburgh, 1992.

[LAWT91] W. Lawton, Necessary and Sufficient Conditions for Constructing Orthonormal Wavelet Bases, *J. Math. Phys.*, Vol. 32, 1991, 57-61.

[LEES93] J.S.Lee, Y.N.Sun, C.H.Chen, and C.T.Tsai, Wavelet Based Corner Detection, *Pattern Recognition*, Vol.26, No.6, June 1993, 853-865.

[LEVI72] S. Levialdi, On shrinking binary picture patterns, *Comm. ACM* 15, 1972, 7-10.

[LEWI92] A. S. Lewis and G. Knowles, Image Compression Using the 2-D Wavelet Transform, *IEEE Trans. on Image Processing*, vol. 1, 1992, 244-250.

[LICH93] Q. Liu, A. K. Chau and C. K. Chui, High-Rate Image Compression using Spline-Wavelet Packets and Vector Quantization, *Proc. SPIE, Vol. 2034, Mathematical Imaging: Wavelet Applications in Signal and Image Processing*, San Diego, CA, 1993, 194-204.

[LIGO91] C. C. Li, M. Gokmen, A. D. Hirschman and Y. Wang, Information Preserving Image Compression for Archiving NMR Images, *Computerized Medical Imaging and Graphics*, vol. 15, no. 4, 1991, 277-283.

[LILI89] R.Y.Li, Y.Y.Li, and J.D.Leonard, Model-based Target Reconition Using Lasar Radar Imagery, *Proc. SPIE, Vol. 1099, Advances in Image Compression and Automatic Target Recognition*, 1989, 17-26.

[LIND80] Y. Linde, A. Buzo, and R. M. Gray, An Algorithm for Vector Quantizer Design, *IEEE Transactions on Communications*, vol. 28, Jan. 1980, 84-95.

[LIST91] H. Li and Q.F. Stout, eds. *Reconfigurable Massively Parallel Computers*, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[MAHI80] D. Marr and E. Hildreth, Theory of Edge Detection, *Proc. Royal Soc. London*, B, Vol. 207, 1980, 187-217.

[MALA92] G. Malandain and G. Bertrand, Fast characterization of 3D simple point, *Proc. 11th IAPR International Conf. Pat. Recog.*, Vol. III, The Hague, The Netherlands, 1992, 232-235.

[MALL89a] S. Mallat and S. Zhong, Complete Signal Representations with Multiscale Edges, Computer Science Tech. Report No. 483, Courant Institute of Mathematical Sciences, New York University, November, 1989.

[MALL89b] S. G. Mallat, Multifrequency Channal Decompositions of Images and Wavelet Models, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 37, Dec. 1989, 2091-2110.

[MALL89c] S. Mallat, A Theory for Multiresolution Signal Decomposition: The Wavelet Representation, *IEEE Tran. Pattern Anal. Mach. Intell.*, Vol. 11, No. 7, July 1989, 674-693.

[MALL89d] S. Mallat, Multiresolution Approximations and Wavelet Orthonormal Bases of $L^2(R)$, *Trans. Am. Math. Soc.*, Vol. 315, No. 1, 1989, 69-88.

[MAZH92] S. Mallat and S. Zhong, Characterization of Signals from Multiscale Edges, *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 14, No. 7, 1992, 710-732.

[MAHW92] S. Mallat and W. L. Hwang, Singularity Detection and Processing with Wavelets, *IEEE Trans. Information Theory*, Vol. 38, No. 2, 1992, 617–643.

[MAZH91] S. Mallat and S. Zhong, Compact Image Coding from Edges with Wavelets, *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1991, 2745-2748.

[MARE91] M. Maresca and H. Li, Polymorphic VLSI arrays with distributed control, in [LIST91].

[MEYE93] Y. Meyer, *Wavelets: Algorithms and Applications*, translated by Robert D. Ryan, SIAM, Philadelphia, PA, 1993.

[MILI93] J. Miller and C. C. Li, Segmentation of Object Silhouettes Using The Haar Wavelet at Multiple Resolutions, Tech. Report TR-CV-93-06, Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA, November, 1993.

[MINC93] C. H. Min, Image Compression Using Variable-Rate Vector Quantization of Wavelet Transform Coefficients, MS Thesis, Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA, 1993.

[NACH75] J. Nachmais and A. Weber, Discrimination of simple and complex gratings, *Vision Res.* Vol. 15, 1975, 217-223.

[NASS80] D. Nassimi and S. Sahni, Finding connected components and connected ones on a mesh-connected parallel computer, *SIAM Journal of Computing*, 9, 1980, 744-757.

[NATH83] D. Nath, S.N. Maheshwari and P.C.P. Bhatt, Efficient VLSI networks for parallel processing on orthogonal trees, *IEEE Trans. Computers*, C-32, 1983, 569-581.

[NOAK90] M. Noakes and W.J. Dally, System design of the j-machine, *Advanced Research in VLSI - Proceedings of the Sixth MIT Conference*, 1990, 179-213.

[PAPP91] T. N. Pappas and D.L. Neuhoff, Model-Based Halftoning, *Proc. SPIE, Vol. 1453, Human Vision, Visual Processing, and Digital Display II*, 1991, 244-255.

[PEMA90] P. Perona and J. Malik, Scale-Space and Edge Detection Using Anisotropic Diffusion, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, Vol. 12, no. 7, 1990, 629-639.

[POGG82] T. Poggio, H. Voorhees, and A. Yuille, A Regularized Solution to Edge Detection, *Tech. Report, M.I.T. Artifical Intell. Lab.*, AI Memo 833, 1982.

[PRAS89] V.K. Prasanna-Kumar and D. Reisis, Image computations on meshes with multiple broadcast, *IEEE Trans. Pattern Anal and Mach. Intell.*, PAMI-11, 1989, 1194-1202.

[PRES84] K. Preston and M.J.B. Duff, *Modern Cellular Automata*, Plenum, New York, 1984.

[RANK90] S. Ranka and S. Sahni, Convolution on mesh connected multicomputers, *IEEE Trans. Pattern Anal. and Mach. Intell.*, PAMI-12, 1990, 315-318.

[RAYA90] Sai Prasad Raya, Low-Level Segmentation of 3D-Magnetic Resonance Brain Images-A Rule-Based System, *IEEE Trans. on Medical Imaging*, Vol. 9, NO. 3, 1990, 327-337.

[RIGR91] E. A. Riskin and R. M. Gray, A Greedy Tree Growing Algorithm for Design of Variable Rate Vector Quantizers, *IEEE Trans. on Signal Processing*, vol. 39, no. 11, 1991, 2500-2507.

[RISK90] E. A. Riskin, E. M. Daly and R. M. Gray, Variable Rate Vector Quantization for Medical Image Compression, *IEEE Trans. on Medical Imaging*, vol. 9, no. 3, Sep. 1990, 290-298.

[RONS88] C. Ronse, Minimal test patterns for connectivity preservation in parallel thinning algorithms for binary digital images, *Discrete Applied Mathe.*, 21, 1988, 67-79.

[ROSE70] A. Rosenfeld, Connectivity in digital pictures. *J. ACM*, 17, 1970, 146-160.

[ROSE75] A. Rosenfeld, A characterization of parallel thinning algorithms, *Information and Control*, 29, 1975, 286-291.

[RUSS92] C. Russ, *The Image Processing Handbook*, CRC Press, Boca Raton, 1992.

[SCHE91] I.D. Scherson and P.F. Corbett, Communications Overhead and the Expected Speedup of Multidimensional Mesh–Connected Parallel Processors, *Journal of Parallel and Distributed Computing*, JPDC–11, 1991, 86–96.

[SERR82] J. Serra, *Image Analysis and Mathematical Morphology*, Vol. 1, Academic Press, London, 1982.

[SHAH91] J. Shah, Segmentation by Nonlinear Diffusion, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Maui, HI, 1991, 202-207.

[SKLA76] J. Sklansky, L.P. Cordella and S. Levialdi, Parallel detection of concavities in cellular blobs, *IEEE Tran. Computers*, C-25, 187-196, 1976.

[STOF81] J.C. Stoffel and J.F. Moreland, A Survey of Electronic Techniques for Pictorial Reproduction, *IEEE Trans. on Comm.*, Vol. COM-29, No. 12, 1981, 1898-1925.

[STOU86] Q.F. Stout, Algorithms for meshes and pyramids, in *Intermediate-Level Image Processing*, M.J.B. Duff, ed., Academic Press, London, 1986.

[STOU88] Q.F. Stout, Mapping vision algorithms to parallel architectures, *Proc IEEE* 76, 1988, 982-995.

[STRA89] G. Strang, Wavelets and Dilation Equations: A Brief Introduction, *SIAM Review*, Vol. 31, No. 4, 1989, 614-627.

[SULI93a] M. Sun, C. C. Li and R. J. Sclabassi, Edge Localization in Images by Symmetrical Wavelet Transforms, *Proc. SPIE, Vol. 2034, Math. Imaging: Wavelet Applications in Signal and Image Processing*, San Diego, CA, 1993, 92-103.

[SULI93b] M. Sun, C. C. Li and R. J. Sclabassi, Symmetrical Wavelet Transform for Edge Localization, submitted to *Optical Engineering*; *Tech. Report TR-CV-93-03, Dept. of Electrical Engineering, University of Pittsburgh*, Pittsburgh.

[SUZH93] H. Szu, Y. Zhang, M. Sun and C. C. Li, Neural Network Adaptive Digital Image Screen Halftoning(DISH) based on Wavelet Transform Preprocessing, *Proc. INNS World Congress on Neural Networks* Japan, 1993.

[TANI83] S.L. Tanimoto, A pyramidal approach to parallel processing, *Proc. 10th Ann. Intl. Symp. Comp. Arch.*, 1983, 372-378.

[TEHC89] C.H.Teh and R.T.Chin, On the Detection of Dominant Points on Digital Curves, *IEEE Tran. Pattern Anal. Mach. Intell.*, Vol.11, No.8, August 1989, 859-872.

[UDUP90] J. K. Udupa and G. T. Herman, *3D Imaging in Medicine*, CRC Press, Boca Raton, Florida, 1990.

[ULIC87] R. Ulichney, Digital Halftoning, The MIT Press, Cambridge, Massachusetts, 1987.

[UNSE89] M. Unser and M. Eden, Multiresolution Feature Extraction and Selection for Texture Segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 11, No 7, 1989, 717-728.

[WHPI93] R. T. Whitaker and S. M. Pizer, A Multi-scale Approach to Nonuniform Diffusion, *CVGIP: Image Understanding*, Vol. 57, no.1, 1993, 99-110.

[WICK92a] M. V. Wickerhauser, Lectures on Wavelet Packet Algorithms, *Tech. Report, Dept. of Mathematics, Washington University*, St. Louis, MO, November, 1991.

[WICK92b] M. Wickerhauser, High-Resolution Still Picture Compression, *Tech. Report, Dept. of Mathematics, Washington University*, St. Louis, MO, April, 1992.

[ZETT90] W. R. Zettler, J. Huffman and D. C. P. Linden, Application of Compactly Supported Wavelets to Image Compression, *Proc. SPIE, Vol. 1244, Image Processing Algorithms and Techniques*, 1990, 150-160.

[ZHAN93] Y. Zhang, M. Sun and C. C. Li, Adaptive Image Halftoning based on Wavelet Transform, *Proc. SPIE, Vol. 1961, Visual Information Processing II*, Orlando, Fl, 1993, 366-376.

C. Research Publications

1. H. Y. H. Chuang, H. J. Kim and C. C. Li, Systolic Architecture for Discrete Wavelet Transforms with Orthogonal Bases, *Proceedings of SPIE Conference, Vol. 1708, Applications of Artificial Intelligence X: Machine Vision and Robotics*, Orlando, FL, April 1992, pp. 157-164.

2. M. Gökmen and C. C. Li, Multiscale Edge Detection Using First Order R-Filter, *Proceedings of the International Conference on Pattern Recognition*, The Hague, The Netherlands, August 1992, pp. 307-310.

3. H. J. Kim, W. Kim and C. C. Li, A Performance Study of Two Wavelet-Based Edge Detectors, *Proceedings of the International Conference on Pattern Recognition*, The Hague, The Netherlands, August 1992, pp. 302-306.

4. M. Gökmen and C. C. Li, Edge Detection and Surface Reconstruction Using Refined Regularization, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 5, pp. 492-499, 1993.

5. Y. Zhang, M. Sun and C. C. Li, Adaptive Image Halftoning Based on Wavelet Transform, *Proceedings of SPIE Conference, Vol. 1961, Visual Information Processing II*, Orlando, FL, April 1993, pp. 366-376.

6. H. J. Kim and C. C. Li, A Non-orthogonal Wavelet Edge Detector with Four Filter-Coefficients, *Proceedings of SPIE Conference, Vol. 2034, Mathematical Imaging: Wavelet Applications in Signal and Image Processing*, San Diego, CA, July 1993, pp. 115-126.

7. M. Sun, C. C. Li and R. J. Scalabassi, Edge Localization in Images by Symmetrical Wavelet Transforms, *Proceedings of SPIE Conference, Vol. 2034, Mathematical Imaging: Wavelet Applications in Signal and Image Processing*, San Diego, CA, July 1993, pp. 92-103.

8. H. C. Hsin and C. C. Li, An Experiment on Texture Segmentation Using Modulated Wavelets, *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, Vol. 4, Le Touquet, France, October 1993, pp.529-534.

9. J. W. Hong and C. C. Li, Structured Networks Based on Wavelet Transforms for System Identification, Submitted to IEEE Trans. on Neural Networks.

10. M. Sun, C. C. Li and R. J. Sclabassi, Symmetrical Wavelet Transform for Edge Localization, *Submitted to Optical Engineering.*

11. M. Hamdi and R.W. Hall, "Compound Graph Networks for Parallel Image Processing," *Computer Architecture for Machine Perception*, B. Zavidovique and P-L. Wendel, eds., Paris, France, December 1991, pp. 355-366.

12. M. Hamdi, *Communication-Efficient Interconnection Networks for Parallel Computations*, PH.D. Dissertation, University of Pittsburgh, Department of Electrical Engineering, July 1991.

13. R.W. Hall and Ş. Küçük, Parallel 3D Shrinking Algorithms Using Subfields Notions, *Proceedings 11-th International Conference on Pattern Recognition*, The Hague, The Netherlands, Aug. 30 - Sept. 3, 1992, pp. 395-398.

14. R.W. Hall, Ş. Küçük and M. Hamdi, Wavelet Transform Embedding in Mesh Architectures, *1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, June 1993, pp. 596-597.

15. R.W. Hall, Connectivity preserving parallel operators in 2D and 3D images, *SPIE Vision Geometry Conference*, Boston, Massachusetts, November 1992, pp. 172-183.

16. R.W. Hall, Optimally small operator supports for fully parallel thinning algorithms, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-15, pp. 828-833, 1993.

17. R.W. Hall, Ş. Küçük and M. Hamdi, Wavelet Transform Embeddings in Mesh Architectures, submitted to *Pattern Recognition*, (TR-LCVPR-93-10).

18. A. Chandra and R.W. Hall, 3D Component Labeling in 3D Mesh Architectures, submitted to *CVPR-94*, (TR-LCVPR-93-11).

19. R.W. Hall, Connectivity Preservation Tests for Parallel Reduction-Augmentation Algorithm, submitted to *ICPR-94*, (TR-LCVPR-94-01).

D. Participating Professional Personnel

Professor Ching-Chung Li, Principal Investigator
Professor Richard W. Hall, Co-principal Investigator
Dr. M. Gokmen, Post-doctoral Research Fellow, 1990-91
M. Hamdi, Graduate Research Assistant, 1990-91
H. J. Kim, Graduate Research Assistant, 1990-93
S. Kucuk, Graduate Research Assistant, 1991-93
W. K. Kim, Graduate Research Assistant, 1992-93
Y. H. Chang, Graduate Research Assistant, 1992-93
A. Chandra, Graduate Research Assistant, 1992-93
T. P. Wang, Graduate Research Assistant, 1992
H. C. Hsin, Graduate Research Assistant, 1993
J. T. Miller, Graduate Research Assistant, 1993
M. A. Al-Mohimeed, Graduate Student, 1991-92
J. W. Hong, Graduate Student, 1992
Y. P. Zhang, Graduate Student, 1992-93
C. H. Min, Graduate Student, 1993

E. Interactions

Our research group has been in close interaction with Professor K. S. Lau and Professor C. J. Lennard of our Mathematics Department who have conducted regular seminars on wavelet theory during the past three years. Professor H. Y. H. Chuang of our Computer Science Department participated in our research on the part of systolic architecture of discrete orthonormal wavelet transforms. Professor M. Sun of our Department of Neurological Surgery has been in active collaboration with us on applications of wavelet transforms to both image processing and biomedical problems. One M. S. thesis "Dipole Localization in Human Brain via the Wavelet Transform of Multi-channel EEG" (by F. C. Tsui) was jointly supervised by Dr. Sun and Professor Li. There has been significant interaction with T. Y. Kong (Queens College, CUNY) and A. Rosenfeld (University of Maryland) on the subject of topology preservation and support limitations for parallel operators. With regard to industrial applications, we have recently discussed with engineers of ALCOA Technical Center on potential applications of wavelet transforms to computer-aided design projects.