INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

IMAGE ANALYSIS IN FOURIER SPACE

By
Steven J. Desjardins. B.Sc.. M.Sc.
May 2002

A Thesis
submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy in Mathematics[1]

[1]The Ph.D. Program is a joint program with Carleton University. administered by the Ottawa-Carleton Institute of Mathematics and Statistics

Abstract

General results on Fourier Transforms, tight frame wavelets and pseudodifferential operators are presented to provide a theoretical framework for the applications. Known and new tight frame wavelets that are characteristic and tapered characteristic functions in Fourier Space are constructed in Cartesian and polar Fourier Space with frame bound 1. These wavelets are used to localize singularities in images.

A review of the use of the diffusion equation in de-noising images is presented. A new method, which applies a multi-directional diffusion in Fourier Space is given. Properties of this new product filter method are described and the product filter's de-noising ability is evaluated. The product filter algorithm is also compared to other techniques, including MATLAB's built-in filters and two recent wavelet techniques.

Finally, a summary of the results of a study of the de-noising abilities of third-order partial differential equations is presented.

Acknowledgements

I would like to thank Dr. Rémi Vaillancourt for all of his help and encouragement over the last five years. I would also like to thank Dr. Wulf Rossmann. Dr. Erhard Neher. Dr. Philip Scott and the Staff of the Department of Mathematics and Statistics. And. most of all. I thank Seleena. because without her. none of this would have been possible.

Dedication

This thesis is dedicated to my wonderful wife, Seleena,

and to the memory of our dear friend, Othello.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is primarily concerned with the localization of singularities and the reduction of noise in natural and geometric images via their representations in Fourier Space.

Tight frame wavelets with frame bound 1 are constructed by paving Fourier Space with box functions (characteristic functions of rectangular regions) and new tapered box functions (smooth and overlapping). A new radial paving is also considered via polar coordinates in Fourier Space. These wavelets are used to localize singularities in the Barbara and zigzag images. The radial wavelets are used to localize a circular singularity. These results are presented in Chapter 2.

A new method for using the diffusion equation to de-noise images is presented in Chapters 3 to 5. A multi-directional diffusion in Fourier Space is applied minimally to noisy images. This new technique is evaluated and compared to other methods, including MATLAB's built-in filters and two wavelet techniques.

The results of a study of the de-noising abilities of third-order partial differential equations (PDE's) are summarized in Chapter 6.

A theoretical background is provided by presenting some general results on frame wavelets, Fourier Transforms and pseudodifferential operators. The purpose of this

background material is to provide a mathematical context for the image processing methods developed in the thesis.

1.1 Frame Wavelets

The concept of frames for Hilbert Spaces generalizes the notion of an orthonormal basis and a Riesz basis in the sense that a frame $Z = \{z_i \mid i \in I\}$, where I is an index set, provides a stable representation for signals f by means of an expansion $f = \sum_i c_i(f) z_i$, but $\{z_i\}$ is not necessarily an orthonormal or independent set. Frames provide a useful model to obtain signal decompositions in cases where redundancy, robustness, oversampling, and irregular sampling play a role (see [9], [10]).

Nonsmooth orthonormal multiwavelets have been constructed [8] for the purpose of performing microlocal analysis of tempered distributions in $\mathbb{R}^n$. In $\mathbb{R}^2$, the multiwavelets, which consist of characteristic functions of squares, have perfect localization in Fourier Space but poor localization in x-Space. It is shown in [29] that no smooth wavelets exist in the Hardy Space, $H^2(\mathbb{R})$, the space of L^2 functions whose Fourier Transforms have support on the positive real axis. To obtain good localization in x-Space and Fourier Space in $\mathbb{R}^2$, the block wavelets are smoothed by convolution or tapering. Several constructions are presented for different pavings of the Fourier Space.

For numerical applications, the construction of smooth frame wavelets for $H^2(\mathbb{R})$ given in [29] with good localization in both x and Fourier Spaces is generalized to $L^2(\mathbb{R}^2)$ by properly tapering the orthonormal multiwavelets given in [8] so that the twelve wavelet frame functions

$$\{w^1(x), w^2(x), \ldots, w^{12}(x)\}$$

and their scaled versions in Fourier Space satisfy the identity

$$G(\xi) := \sum_{\substack{\ell \in \mathbb{L} \\ j \in \mathbb{Z}}} |\widehat{\psi^\ell}(2^j \xi)|^2 \equiv 1. \qquad \xi \neq 0. \tag{1.1}$$

where

$$\mathbb{L} = \{1, 2, \ldots, 12\}.$$

It is noted that G is invariant under dyadic scaling, that is, $G(2^k \xi) = G(\xi)$ for all k and all ξ.

This work could be considered in the context of almost orthogonal decompositions by means of ψ Transforms presented, for instance, in [21] and [22].

1.2 Fourier Transforms

The continuous Fourier Transform $\widehat{f}(\xi)$ of a function $f(x)$ defined over $\mathbb{R}^2$ and the Inverse Fourier Transform of $\widehat{f}(\xi)$ will be

$$\widehat{f}(\xi) = \mathcal{F}\{f(x)\} = \int e^{-i\xi \cdot x} f(x)\, dx. \qquad f(x) = \mathcal{F}^{-1}\{\widehat{f}(\xi)\} = \frac{1}{(2\pi)^2} \int e^{ix \cdot \xi} \widehat{f}(\xi)\, d\xi. \tag{1.2}$$

For numerical applications on domains with pixels at integer points, say, over squares with sides of length 1, the standard formulae with harmonic analysts,

$$\widehat{f}(\xi) = \int e^{-2\pi i \xi \cdot x} f(x)\, dx. \qquad f(x) = \int e^{2\pi i x \cdot \xi} \widehat{f}(\xi)\, d\xi \tag{1.3}$$

will be used.

A fundamental result that is used in this thesis is concerned with the Fourier Transform of parallel straight lines and parallel straight segments. The continuous Fourier Transform of a line impulse distribution in $\mathbb{R}^2$ is a line impulse distribution at a right angle with respect to the original line impulse distribution. It is enough to show this result for a line impulse along the horizontal x_1-axis.

Proposition 1 *Let*

$$f(x_1, x_2) = 1_{x_1} \otimes \delta(x_2)$$

be a line impulse distribution along the x_1-axis. Then the Fourier Transform

$$\widehat{f}(\xi_1, \xi_2) = 2\pi\delta(\xi_1) \otimes 1_{\xi_2}$$

is a line impulse distribution along the ξ_2-axis. The Fourier Transforms of parallel line impulse distributions differ by the phase of their elements.

Proof. For $t \in \mathbb{R}$ we have $\widehat{1_t} = 2\pi\delta(t)$ and $\widehat{\delta}(t) = 1_t$. Hence, by the definition of the tensor product of distributions, the Fourier Transform acts separately on each component: thus

$$\mathcal{F}\{f(x_1, x_2)\} = \widehat{1_{x_1}} \otimes \widehat{\delta}(x_2) = 2\pi\delta(\xi_1) \otimes 1_{\xi_2}.$$

Let $f(x_1, x_2)$ be a line impulse distribution along the line $x_2 = r$, that is,

$$f(x_1, x_2) = 1_{x_1} \otimes \delta(x_2 - r).$$

Since $\widehat{\delta}(t - r) = e^{ir\omega} 1_t$,

$$\mathcal{F}\{f(x_1, x_2)\} = \widehat{1_{x_1}} \otimes \widehat{\delta}(x_2 - r) = 2\pi\delta(\xi_1) \otimes e^{ir\xi_2} 1_{\xi_2}. \square$$

The Discrete Fourier Transform (DFT) $X(k_1, k_2)$ of a sequence $x(n_1, n_2)$ and the Inverse Discrete Fourier Transform of $X(k_1, k_2)$ are

$$X(k_1, k_2) = \sum_{n_2=1}^{N} \sum_{n_1=1}^{N} x(n_1, n_2)\, e^{-2\pi i(k_1-1)(n_1-1)/N}\, e^{-2\pi i(k_2-1)(n_2-1)/N} \tag{1.4}$$

and

$$x(n_1, n_2) = \frac{1}{N^2} \sum_{k_2=1}^{N} \sum_{k_1=1}^{N} X(k_1, k_2)\, e^{-2\pi i(k_1-1)(n_1-1)/N}\, e^{-2\pi i(k_2-1)(n_2-1)/N}. \tag{1.5}$$

A result similar to Proposition 1 holds for the Discrete Fourier Transform of a line impulse. Hereafter, the MATLAB convention of a colon, that is, $1 : N$ means $1, 2, \ldots, N$, will be used for convenience.

Proposition 2 *Let*

$$x(n_1, n_2) = \begin{cases} 1. & n_1 = 1. n_2 = 1 : N \\ 0. & otherwise. \end{cases}$$

be a line impulse along the first row of an $N \times N$ matrix. Then the Discrete Fourier Transform

$$X(k_1, k_2) = \begin{cases} 1. & k_1 = 1 : N. \quad k_2 = 1. \\ 0. & otherwise. \end{cases}$$

is a line impulse along the first column of the matrix. The Fourier Transforms of parallel line impulses differ by the phase of their elements.

Proof. A simple summation gives

$$\begin{aligned} X(k_1, k_2) &= \sum_{n_2=1}^{N} \sum_{n_1=1}^{N} x(n_1, n_2)\, e^{-2\pi\imath(k_1-1)(n_1-1)/N}\, e^{-2\pi\imath(k_2-1)(n_2-1)/N} \\ &= \sum_{n_2=1}^{N} e^{-2\pi\imath(k_2-1)(n_2-1)/N} \\ &= \begin{cases} N. & k_1 = 1 : N. \quad k_2 = 1. \\ 0. & \text{otherwise.} \end{cases} \end{aligned}$$

Let $x(n_1, n_2)$ be a line impulse along row r. Then

$$\begin{aligned} X(k_1, k_2) &= \sum_{n_2=1}^{N} \sum_{n_1=1}^{N} x(n_1, n_2)\, e^{-2\pi\imath(k_1-1)(n_1-1)/N}\, e^{-2\pi\imath(k_2-1)(n_2-1)/N} \\ &= e^{-2\pi\imath(k_1-1)(r-1)/N} \sum_{n_2=1}^{N} e^{-2\pi\imath(k_2-1)(n_2-1)/N} \\ &= \begin{cases} N\, e^{-2\pi\imath(k_1-1)(r-1)/N}. & k_1 = 1 : N. \quad k_2 = 1. \\ 0. & \text{otherwise.} \end{cases} \end{aligned}$$

which is a modulated line impulse in the vertical direction. □

The Discrete Fourier Transform of a constant segment parallel to the horizontal axis produces an approximation to the cardinal sine in the horizontal direction which is modulated in the vertical direction. as a simple computation shows.

Let

$$x(n_1, n_2) = \begin{cases} 1, & n_1 = r_1, \quad n_2 = r_2 : r_3, \\ 0, & \text{otherwise.} \end{cases}$$

be a segment impulse along the r_1-th row of an $N \times N$ matrix. Then the Discrete Fourier Transform

$$\begin{aligned} X(k_1, k_2) &= \sum_{n_2=1}^{N} \sum_{n_1=1}^{N} x(n_1, n_2)\, e^{-2\pi i(k_1-1)(n_1-1)/N}\, e^{-2\pi i(k_2-1)(n_2-1)/N} \\ &= e^{-2\pi i(k_1-1)(r_1-1)/N} \left[\sum_{n_2=r_2}^{r_3} e^{-2\pi i(k_2-1)(n_2-1)/N} \right]. \end{aligned}$$

where the term in square brackets is an approximation to the cardinal sine. By means of the MATLAB command `fftshift`, the vertical peak line of $X(k_1, k_2)$ is in the centre of the matrix. The larger N is and the longer the segment is, the closer the surface is to a modulated line impulse.

1.3 Pseudodifferential Operators

Given a function $f(x)$ and a symbol $g(x, \xi)$, the classical pseudodifferential operator G is defined by the formula

$$Gf(x) = \frac{1}{(2\pi)^2} \int e^{ix\cdot\xi} g(x, \xi) \widehat{f}(\xi)\, d\xi. \tag{1.6}$$

or

$$Gf(x) = \frac{1}{(2\pi)^2} \iint e^{i(x-y)\cdot\xi} g(x, \xi) f(y)\, dy\, d\xi.$$

If we define the kernel

$$k(x, y) = \frac{1}{(2\pi)^2} \int e^{iy\cdot\xi} g(x, \xi)\, d\xi.$$

then G has the integral operator representation

$$Gf(x) = \int k(x, x-y) f(y)\, dy.$$

Friedrichs [24], p. 15, has introduced the so-called cokernel

$$\sigma(\chi,\xi) = \int e^{-ix\cdot\chi} g(x,\xi)\,dx$$

to define

$$Gf(x) = \frac{1}{(2\pi)^4} \iint e^{ix\cdot\chi} \sigma(\chi-\xi,\xi)\widehat{f}(\xi)\,d\xi\,d\chi.$$

Finally, harmonic analysts [26], p. 304–305, consider the operator G as a time-frequency operator with weight or spreading function

$$\sigma(\chi,y) = \int e^{-ix\cdot\chi} k(x,y)\,dx;$$

thus

$$\begin{aligned} Gf(x) &= \frac{1}{(2\pi)^2} \iint \sigma(\chi, x-y)\, e^{ix\cdot\chi} f(y)\,dy\,d\chi \\ &= \frac{1}{(2\pi)^2} \iint \sigma(\chi,u)\, e^{ix\cdot\chi} f(x-u)\,du\,d\chi \\ &= \frac{1}{(2\pi)^2} \iint \sigma(\chi,u)(M_\chi T_u f)(x)\,du\,d\chi. \end{aligned}$$

where the translation operator T and the modulation operator M are defined by the formulae

$$T_y f(x) = f(x-y). \qquad M_\omega f(x) = e^{ix\cdot\omega} f(x).$$

In the context of this thesis, the representation (1.6) will be the most convenient. For instance, if one wants to find the singular support of a function $f(x_1,x_2)$ in the time domain, one may construct a symbol $g(x,\xi)$ such that $Gf(x)$ has significant values over the singular support of f and is otherwise negligible. (The singular support of f is the closure of the set of all points where f is not a C^∞ function.) If the symbol $g(x,\xi)$ is sufficiently smooth and has appropriate decay at infinity, the singular support of Gf is contained in the singular support of f.

Taking the Fourier Transform of the heat equation for $u(x_1, x_2, t)$.

$$u_t = u_{x_1 x_1} + u_{x_2 x_2},$$

one has

$$\widehat{u}_t = -|\xi|^2 \widehat{u},$$

which admits the solution, in the frequency domain,

$$\widehat{u}(\xi, t) = \widehat{u}(\xi, 0)\, e^{-|\xi|^2 t}$$

or, in the time domain,

$$u(x, t) = \frac{1}{(2\pi)^2} \int e^{ix\cdot\xi}\, e^{-|\xi|^2 t} \widehat{u}(\xi, 0)\, d\xi. \tag{1.7}$$

This is a pseudodifferential operator representation of the solution to the heat equation, with symbol $\exp(-|\xi|t^2)$, independent of x. Since the heat equation is a hypoelliptic equation, to give a meaning to this pseudodifferential operator over $L^2(\mathbb{R}^2)$ when the equation has variable coefficients, one needs to appeal to the Calderón-Vaillancourt theorem [12]. However, in this thesis, the discretized versions of these pseudodifferential operators (1.6) and (1.7) have meaning over a finite matrix even if its symbol depends also on x. The discretization of more general pseudodifferential operators may be done by means of pseudodifference operators [41, 42].

The product filter, introduced in Chapter 3, applies the diffusion equation in the form (1.7) in many directions simultaneously to a matrix representing a noisy image.

1.4 Noise and the Diffusion Equation

In the modern world, the storage and transmission of data have become of paramount importance. In particular, the collecting, storage, transfer and transmission of digital images have become vital in the spread of information and commerce because of the

pervasiveness of the Internet. But the collecting and transfer or transmission of these digital images often introduce noise that distorts the image.

A greyscale image can be represented by the set: $\{(x_1, x_2, I(x_1, x_2))\}$ where (x_1, x_2) represents the pixel coordinates ($1 \leq x_1 \leq m, 1 \leq x_2 \leq n$, for integers m and n) and $I(x_1, x_2)$ is the corresponding intensity value. Typically, $0 \leq I(x_1, x_2) \leq 255$ for integer $I(x_1, x_2)$ or $0 \leq I(x_1, x_2) \leq 1$ for real $I(x_1, x_2)$. Such an image can also be represented by the matrix $[I(i, j)]$, where $I(i, j)$ represents the intensity at pixel (i, j). For comparison, a colour picture could be represented by $\{(x_1, x_2, I_R(x_1, x_2), I_G(x_1, x_2), I_B(x_1, x_2))\}$, where $I_R(x_1, x_2)$, $I_G(x_1, x_2)$ and $I_B(x_1, x_2)$ represent the red, green and blue intensities, respectively.

The effect of noise is to distort some, or possibly all, of the intensity values in the following way: noise, represented by $N(x_1, x_2)$, is added to the intensity values to produce distorted intensities, $I(x_1, x_2) + N(x_1, x_2)$. The noise itself can be of different types. It can be white noise, which follows a Gaussian distribution, in which all pixels are affected to some extent. It can be salt and pepper noise, where some pixels have their intensity values replaced by the minimum value (typically 0) or the maximum possible value, resulting in black and white dots in the image (and hence the name). There can also be noises of more random natures, affecting some pixels to different degrees. All calculations in this thesis were done with MATLAB. In the MATLAB environment, images are treated like matrices. The types of noise used are defined and explained in Chapters 4 and 5.

When presented with a noisy image, the information is often intact – *i.e.* a viewer can see what the image is supposed to represent, though fine details may potentially be lost. In extreme cases, the noise may be severe enough to *destroy* the image and render it unintelligible. In either case, it would be preferable to have the undistorted (pre-noise) original image.

If the original image and/or a detailed knowledge of the noise process that has distorted the image (*i.e.* its intensity and/or distribution) is available, it may be possible to remove the noise and restore the image. However, in real-world applications, the original clean image is unknown and the noise process may not be well-understood. The goal would still be the same - to remove the noise and restore the image. Because the true natures of the image and noise are unknown (only their sum, $I(x_1, x_2) + N(x_1, x_2)$, is known), any attempt to remove the noise must proceed with caution, lest any more damage be done to the image.

One of the first problems encountered is how to identify the noise. Conceptually, the noise would correspond to transient features in the image. But this is unhelpful since the edges (contours of regions in the image) and fine details would also be of a transient nature and yet highly important. So, any procedure that aims to eliminate noise by eliminating transient features may further damage the image.

Many different techniques have been used to de-noise images [1, 2, 3, 4, 5, 6, 8, 11, 13, 14, 15, 19, 23, 30, 31, 32, 33, 34, 35, 36, 39, 40, 43]. One common method is to use a filter matrix. Essentially, the image, as a matrix, is multiplied by another matrix, the filter. A simple example is an averaging filter, where each pixel's intensity is replaced by a (possibly weighted) average of its neighbours' intensities.

The heat, or diffusion, equation in two dimensions,

$$u_t = u_{x_1 x_1} + u_{x_2 x_2},$$

has also been used in the reduction of noise, the rationale being that noise represents pertubations to the image. Application of the diffusion equation will smooth over the pertubations. The problem with this approach is that edge data and fine details may be smoothed over as well, further distorting the image.

Several attempts have been made to correct this limitation [2, 3, 4, 5, 6, 14, 35, 40]

and to try to enhance edges by running diffusion backwards in time in the vicinity of an edge. Perona and Malik [35] were the first to try such an approach. Their idea was to replace the diffusion equation with an anisotropic diffusion equation.

$$u_t = \nabla \cdot (g(|\nabla u|)\, \nabla u).$$

where $g(\cdot)$ is a non-negative, monotonically decreasing function with $g(0) = 1$. Diffusion is controlled by the function $g(\cdot)$. Along an edge, the gradient is large in magnitude and normal to it (since the edge is a contour). Diffusion is encouraged within regions (where ∇u is small), but not across the boundaries of regions (edges). So, $g(\cdot)$ is larger within regions and smaller at edges. The goal is to smooth in directions parallel to the edge, but not perpendicular to it to preserve the edge and to try to run the diffusion backwards perpendicularly to the edge to enhance it.

Another group of researchers, including Alvarez, Lions and Morel [2, 3, 4, 5, 6, 14], have furthered this work and corrected limitations in Perona and Malik's scheme. This group discovered that Perona and Malik's scheme will actually enhance and not remove some types of noise and it is unstable, as the solutions to slightly different initial conditions may diverge. Also, Perona and Malik's scheme will require pre-filtering in the case of noisy images. They suggested some further extensions [14]:

$$u_t = \nabla \cdot (g(|\nabla G_\sigma * u|)\, \nabla u).$$

where

$$G_\sigma(x) = C\sigma^{-1/2} \exp(-|x|^2/4\sigma)$$

is a Gaussian with variance σ and $*$ is convolution, where

$$f * g = \int_{-\infty}^{\infty} f(x)\, g(t - x)\, dx.$$

This model is like Perona and Malik's, with the function $g(\cdot)$ to control edge enhancement, but now there is a different argument that is a superior estimator. The need for

pre-filtering noise is eliminated. A more improved scheme is:

$$u_t = g(|G * \nabla u|)|\nabla u|\nabla \cdot \frac{\nabla u}{|\nabla u|}.$$

where G is a smoothing kernel (like a Gaussian) [5]. This last scheme corrects the drawbacks of the Perona and Malik scheme and the term

$$|\nabla u|\nabla \cdot \frac{\nabla u}{|\nabla u|}$$

ensures that diffusion proceeds in directions orthogonal to ∇u, not in the direction of ∇u. The term $g(|G * \nabla u|)$ controls the edge enhancement as in their previous scheme above. This group went on to formalize a set of axioms for image processing [2, 3, 4].

Torkamani-Azar and Tait [40] have suggested

$$u_t = \nabla \cdot (g(\nabla[h * u]) \nabla u).$$

where

$$h(x_1, x_2) = \frac{\beta}{2} \exp(-\beta(|x_1| + |x_2|)).$$

and β is a constant. Their method was also developed to correct the limitations of Perona and Malik and to be simpler to implement when discretized. Better smoothing is achieved than in Perona and Malik [40]. Torkamani-Azar and Tait found that there was a trade-off between sharpening edges and removing noise when choosing the value of the constant β. Smaller β led to better noise removal, whereas larger β preserved edges better. It was thus desirable to run several iterations with small β for the first run and larger β for the rest to remove noise on the first pass and then enhance the edges after.

The results from these schemes are good. Noise is significantly reduced and edges are preserved or enhanced. To implement these schemes, the partial differential equations (PDE's) have to be discretized into difference equations. The intricacies in the

above schemes stem from the desire to distinguish edges from noise and preserve or enhance the edges while diffusing the noise away. The methods above attempt to control the direction of diffusion using the gradient of the image and then diffuse a little in some areas, diffuse more in others and run the diffusion backwards in time in other regions of the image.

It was thought that it might be possible to remove noise with PDE's in a much simpler way while preserving edges and details. The alternative approach is this: diffuse in all directions by a small amount, thereby reducing the distortion caused by the noise and yet not damaging the image details too much at the same time. And so, a multi-directional diffusion was attempted in Fourier Space.

Torkamani-Azar and Tait give a convenient expression for the signal-to-noise ratio (SNR) that will be used in the thesis [40]:

$$\mathrm{SNR} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} u(i,j)^2}{\sum_{i=1}^{m}\sum_{j=1}^{n}[u(i,j)-U(i,j)]^2} = \frac{\|u\|_F^2}{\|u-U\|_F^2}, \tag{1.8}$$

where $[u(i,j)]$ and $[U(i,j)]$ represent the original and noisy images, respectively, as $m \times n$ matrices and $\|\cdot\|_F$ is the Frobenius matrix norm. Ideally, if noise were perfectly removed from a noisy image, the result would be $u = U$ and SNR is infinite. In general, a higher SNR value signifies a better result, though visual observation is the true measurement, since two matrices may have the same norm and yet appear completely different when viewed as images.

Related to the signal-to-noise ratio is another quality measure, called the peak signal-to-noise ratio (PSNR), which is measured in decibels (dB). The formula for PSNR is [33, 38]:

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right). \tag{1.9}$$

where MSE, mean square error, is

$$\mathrm{MSE} = \frac{1}{m\,n}\|u-U\|_F^2.$$

It is believed that a PSNR of 30 dB for a processed image represents an excellent recovery of the original. However, numerical measures cannot replace visual inspection. PSNR is calculated and used in a comparison of the new product filter algorithm method with a wavelet technique (see Chapter 5).

Chapter 2

Tight Frame Wavelets

2.1 Necesary and Sufficient Conditions for Tight Frame Wavelets

Given $f \in L^2(\mathbb{R}^2)$, let $f_{jk}(x)$ denote the scaled and shifted function

$$f_{jk}(x) = 2^j f(2^j x - k), \qquad j \in \mathbb{Z}, \quad k \in \mathbb{Z}^2. \tag{2.1}$$

Its Fourier Transform is

$$\widehat{f}_{jk}(\xi) = 2^{-j} e^{-ik\cdot\xi} \widehat{f}(2^{-j}\xi) = e^{-ik\cdot\xi} \widehat{f}_j(\xi). \tag{2.2}$$

where the last expression defines $\widehat{f}_j$. Let $\mathbb{L}$ be a finite index set. A system

$$\{\psi^\ell_{jk}\}_{\ell\in\mathbb{L}, j\in\mathbb{Z}, k\in\mathbb{Z}^2} \subset L^2(\mathbb{R}^2)$$

is called a *tight wavelet frame* with frame bound A and $\Psi = \{\psi^\ell\}_{\ell\in\mathbb{L}}$ is called a *tight wavelet frame function* if

$$f(x) = \frac{1}{A} \sum_{\substack{\ell\in\mathbb{L} \\ j\in\mathbb{Z} \\ k\in\mathbb{Z}^2}} \langle f, \psi^\ell_{jk}\rangle \, \psi^\ell_{jk}(x), \qquad \forall f \in L^2(\mathbb{R}^2). \tag{2.3}$$

where

$$\langle f, g\rangle = \int_{\mathbb{R}^2} f(x)\overline{g(x)}\, dx.$$

Recall that a system $\{\psi^\ell_{jk}\}_{\ell\in\mathbb{L},j\in\mathbb{Z},k\in\mathbb{Z}^2} \subset L^2(\mathbb{R}^2)$ is called an *orthonormal wavelet basis* and $\Psi = \{\psi^\ell\}_{\ell\in\mathbb{L}}$ is called an *orthonormal wavelet function* if the system $\{\psi^\ell_{jk}\}_{\ell\in\mathbb{L},j\in\mathbb{Z},k\in\mathbb{Z}^2}$ is an orthonormal basis for $L^2(\mathbb{R}^2)$, which is equivalent to saying that the system $\{\psi^\ell_{jk}\}_{\ell\in\mathbb{L},j\in\mathbb{Z},k\in\mathbb{Z}^2}$ is a tight wavelet frame with frame bound 1 and $\|\psi^\ell\|_{L^2(\mathbb{R}^2)} = 1$ for $\ell \in \mathbb{L}$.

The following theorem, which is essentially Theorem 1 stated and proved in [20], gives necessary and sufficient conditions to have a tight wavelet frame in $\mathbb{R}^2$.

Theorem 1 *Suppose* $\{\psi^1, \psi^2, \ldots, \psi^L\} \subset L^2(\mathbb{R}^2)$, *then*

$$\|f\|^2_{L^2(\mathbb{R}^2)} = \sum_{\substack{\ell\in\mathbb{L}\\ j\in\mathbb{Z}\\ k\in\mathbb{Z}^2}} \left|\langle f, \psi^\ell_{j,k}\rangle\right|^2 \tag{2.4}$$

for all $f \in L^2(\mathbb{R}^2)$ *if and only if the functions* $\{\psi^1, \psi^2, \ldots, \psi^L\}$ *satisfy the following two equalities:*

$$\sum_{\substack{\ell\in\mathbb{L}\\ j\in\mathbb{Z}}} \left|\widehat{\psi^\ell}(2^j\xi)\right|^2 = 1. \qquad a.e.\, \xi \in \mathbb{R}^2. \tag{2.5}$$

$$t_q(\xi) = 0. \qquad a.e.\, \xi \in \mathbb{R}^2.\ \forall q \in \mathbb{Z}^2\backslash(2\mathbb{Z})^2. \tag{2.6}$$

where

$$t_q(\xi) := \sum_{\substack{\ell\in\mathbb{L}\\ j\in\mathbb{Z}_+}} \widehat{\psi^\ell}(2^j\xi)\overline{\widehat{\psi^\ell}(2^j(\xi+2\pi q))}. \qquad \mathbb{Z}_+ := \mathbb{N}\cup\{0\}.$$

and $q \in \mathbb{Z}^2\backslash(2\mathbb{Z})^2$ *means that at least one component* q_j *is odd.*

Any function $f \in L^2(\mathbb{R}^2)$ admits the *tight wavelet frame expansion*

$$f(x) = \sum_{\substack{\ell\in\mathbb{L}\\ j\in\mathbb{Z}\\ k\in\mathbb{Z}^2}} \langle f, \psi^\ell_{jk}\rangle\, \psi^\ell_{jk}(x). \tag{2.7}$$

By using the localization of the wavelet frame functions in Fourier Space, one can study the directions of growth of $\widehat{f}(\xi)$ by looking at the size of the frame coefficients

$$\langle f, \psi^{\ell}_{jk} \rangle = \text{const} \langle \widehat{f}, \widehat{\psi}^{\ell}_{jk} \rangle. \tag{2.8}$$

This equality follows from Plancherel's formula. By using the localization of the frame functions in x-Space, one can localize the singular support of $f(x)$ by varying ℓ, j and k in (2.8).

An alternate formulation of the problem is by means of a pseudodifferential operator

$$Pf(x) = \frac{1}{(2\pi)^2} \int e^{ix\cdot\xi} p(x,\xi) \widehat{f}(\xi)\, d\xi. \tag{2.9}$$

The problem is to find a symbol p such that Pf is strongly localized on the singular support of f and negligible where f is smooth. Pseudodifferential operators with smooth symbols do not extend the singular support of f: that is, the singular support of Pf is contained in the singular support of f. In Section 2.8 below, the symbol will involve only the matrix Q^2_ξ which will be a discretized version of $\widehat{\psi^2_\xi}$ defined by (2.2) and the values of the shift parameter k in the summation in (2.7) will be determined indirectly by the singular support of f through the size of $|\langle \widehat{f}, \widehat{\psi^2_{\xi,k}} \rangle|$.

2.2 Smooth Tight Frame Wavelets

In this section, very general orthogonal box wavelets are smoothed by convolution with the tensor product of even C^∞ functions with integral one to produce tight frame wavelets.

The following notation, which is taken from [7] for n-dimensional wavelets, will be used in the construction of 2-dimensional tight frame wavelets.

- $H = \{\pm 1\}^2$ is a parametrization of the 4 quadrants in $\mathbb{R}^2$. For example, in $\mathbb{R}^2$, $(+1,+1)$, $(-1,+1)$, $(-1,-1)$, and $(+1,-1)$ correspond to the first, second, third,

and fourth quadrants, respectively.

- For $\eta = (\eta_1, \eta_2) \in H$, denote by Q_η the unit square $\prod_{k=1}^{2} [0, \eta_k]$, where $[0, -1]$ stands for the interval $[-1, 0]$.

- $E = \{0, 1\}^2 \setminus \{(0, 0)\}$ is the set of 3 vertices of the 2-dimensional unit square less the origin.

- For $\varepsilon = (\varepsilon_1, \varepsilon_2) \in E$ and $\eta = (\eta_1, \eta_2) \in H$, denote the elementwise product by

$$\varepsilon * \eta := (\varepsilon_1 \eta_1, \varepsilon_2 \eta_2).$$

- For $\varepsilon = (\varepsilon_1, \varepsilon_2) \in E$, $\eta = (\eta_1, \eta_2) \in H$, and $j \in \mathbb{Z}_+$, define the square

$$2^j (Q_\eta + \varepsilon * \eta) := \left\{ \left(2^j (x_1 + \varepsilon_1 \eta_1), 2^j (x_2 + \varepsilon_2 \eta_2)\right) \mid (x_1, x_2) \in Q_\eta \right\}.$$

Then let $\mathcal{Q}_{j,\varepsilon,\eta}$ be the collection of unit squares that cover $2^j (Q_\eta + \varepsilon * \eta)$ with overlaps of measure zero, that is,

$$\mathcal{Q}_{j,\varepsilon,\eta} := \left\{ \prod_{k=1}^{2} [\eta_k (\ell_k - 1), \eta_k \ell_k] + 2^j (\varepsilon * \eta) \mid 1 \le \ell_1, \ell_2 \le 2^j, \ \ell_1, \ell_2 \in \mathbb{N} \right\},$$

where $[-(\ell_k - 1), -\ell_k]$ stands for the interval $[-\ell_k, -(\ell_k - 1)]$.

- Given an indexing set K and a collection $\{Q_k\}_{k \in K}$ of subsets of $\mathbb{R}^2$, define

$$\mathcal{Q} := \{Q_k\}_{k \in K} \qquad \text{and} \qquad \cup(\mathcal{Q}) := \bigcup_{k \in K} Q_k.$$

- Define

$$\pi \mathcal{Q}_{j,\varepsilon,\eta} := \{\pi Q \mid Q \in \mathcal{Q}_{j,\varepsilon,\eta}\}.$$

- Let $\mathbb{Z}_+^{E \times H}$ denote the set of all functions from $E \times H$ to $\mathbb{Z}_+$.

- For a nonnegative integer $N \in \mathbb{Z}_+$, let $\mathbb{Z}_N := \{0, 1, \ldots, N\}$ and denote the set of all functions from $E \times H$ to $\mathbb{Z}_N$ by $\mathbb{Z}_N^{E \times H}$.

To state Theorem 2, let $\vartheta(t)$ be a $C_0^\infty(\mathbb{R})$ function of one variable satisfying

$$\vartheta(t) \geq 0. \quad \vartheta(t) = \vartheta(-t). \quad \int_{-\infty}^{\infty} \vartheta(t)\,dt = 1. \quad \vartheta(t) = \begin{cases} 1. & |t| \leq \frac{1}{3}. \\ 0. & |t| \geq \frac{2}{3}. \end{cases}$$

For $\alpha > 0$ and $\xi = (\xi_1, \xi_2) \in \mathbb{R}^2$, denote

$$\vartheta_\alpha(\xi) = \frac{1}{\alpha^n} \prod_{j=1}^{2} \vartheta\Big(\frac{\xi_j}{\alpha}\Big).$$

Theorem 2 *Let $j \in \mathbb{Z}_-$, $\varepsilon \in E$ and $\eta \in H$. Fix a positive number α satisfying $0 < \alpha < 1/2$. For $Q \in \mathcal{Q}_{j,\varepsilon,\eta}$, define $\lambda_Q(\xi)$ by*

$$\lambda_Q(\xi) := \vartheta_{\pi\alpha} * \chi_{\pi Q}(\xi) = \int_{\mathbb{R}^2} \vartheta_{\pi\alpha}(\xi - \zeta)\chi_{\pi Q}(\zeta)\,d\zeta.$$

where $\chi_{\pi Q}$ is the characteristic function of the square πQ. For $\rho \in \mathbb{Z}_-^{E\times H}$, let

$$\widetilde{\mathcal{Q}}_\rho := \bigcup_{(\varepsilon,\eta)\in E\times H} \pi\mathcal{Q}_{\rho(\varepsilon,\eta),\varepsilon,\eta},$$

and

$$\tau_\rho(\xi) := \sum_{\substack{Q\in\widetilde{\mathcal{Q}}_\rho \\ j\in\mathbb{Z}}} |\lambda_Q(2^j\xi)|^2.$$

and for $Q \in \widetilde{\mathcal{Q}}_\rho$, define $\omega_Q(x)$ by

$$\widehat{\omega_Q}(\xi) := \tau_\rho(\xi)^{-1/2}\lambda_Q(\xi).$$

Then $\Psi := \{\omega_Q\}_{Q\in\widetilde{\mathcal{Q}}_\rho}$ is a tight wavelet frame function.

Proof. To show that

$$\Psi := \{\psi^1, \psi^2, \ldots, \psi^L\}$$

is a tight wavelet frame function with frame bound 1, it suffices to show that

$$\|f\|_{L^2(\mathbb{R}^2)}^2 = \sum_{\substack{Q\in\widetilde{\mathcal{Q}}_\rho \\ j\in\mathbb{Z} \\ k\in\mathbb{Z}^2}} \big|\langle f, (\omega_Q)_{j,k}\rangle\big|^2.$$

for all $f \in L^2(\mathbb{R}^2)$. By Theorem 1. this is equivalent to showing that equalities (2.5) and (2.6) hold.

Since

$$\bigcup_{\substack{Q\in\widetilde{\mathcal{Q}}_\rho \\ j\in\mathbb{Z}}} 2^j\pi Q = \mathbb{R}^2\backslash\{0\}.$$

then $\tau_\rho(\xi) > 0$ for $\xi \neq 0$. This shows that the $\widehat{\omega_Q}(\xi)$ are well-defined. because $0 \notin$ supp $\widehat{\omega_Q}$. Therefore. the dyadic scale independence of τ_ρ. namely.

$$\tau_\rho(2^j\xi) = \tau_\rho(\xi). \qquad j \in \mathbb{Z}.$$

implies the first equality:

$$\sum_{\substack{Q\in\widetilde{\mathcal{Q}}_\rho \\ j\in\mathbb{Z}}} |\widehat{\omega_Q}(2^j\xi)|^2 = \tau_\rho(\xi)^{-1} \sum_{\substack{Q\in\widetilde{\mathcal{Q}}_\rho \\ j\in\mathbb{Z}}} |\lambda_Q(2^j\xi)|^2 = 1. \qquad \xi \in \mathbb{R}^2\backslash\{0\}.$$

For $Q \in \widetilde{\mathcal{Q}}_\rho$ and $0 < \alpha < 1/2$. supp $\widehat{\omega_Q}$ is contained in a $2\pi\alpha/3$-neighbourhood of πQ. and this implies that

$$\widehat{\omega_Q}(2^j\xi)\widehat{\omega_Q}(2^j(\xi - 2\pi q)) \equiv 0. \qquad j \in \mathbb{Z}_-.\ q \in \mathbb{Z}^2\backslash(2\mathbb{Z})^2.$$

since the supports do not overlap. □

The points $(\varepsilon, \eta) \in E \times H$ can be thought of as rough directions of analyticity. By choosing ρ so that $\rho(\varepsilon, \eta)$ is large. that is. by taking the set $2^{\rho(\varepsilon,\eta)}\,(Q_\eta + \varepsilon, * \eta)$ to be a large square. the Fourier Transform of each function ω_Q for $Q \in \mathcal{Q}_{\rho(\varepsilon,\eta),\varepsilon,\eta}$ has support contained in a square $2\pi Q \in 2\pi\mathcal{Q}_{\rho(\varepsilon,\eta),\varepsilon,\eta}$ which subtends a very small angle as viewed from the origin (see Fig. 2.1).

Remark 1 *Given an annular ring or shell of functions surrounding the origin and whose dyadic dilations cover* $\mathbb{R}^2 \setminus \{0\}$. *there is a ring whose functions have sufficiently small support so that a shift by* 2π *in any direction will result in non-overlap with the*

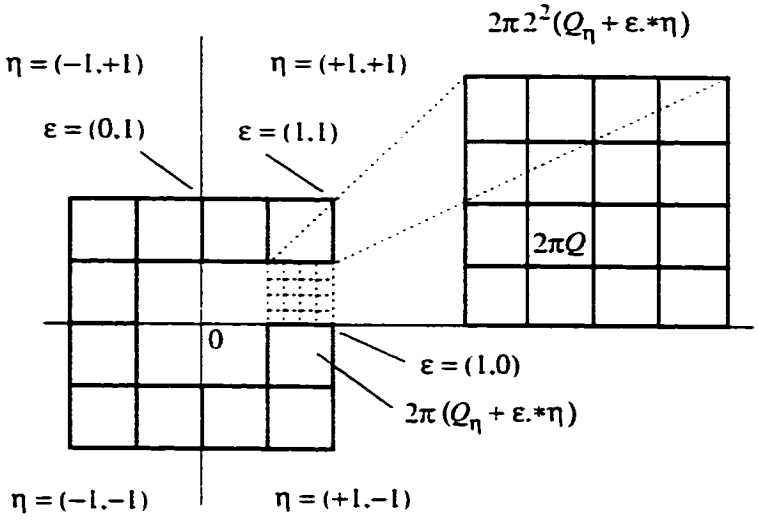

Figure 2.1: An example of Q_ρ.

unshifted function and hence equality (2.6) *is satisfied. Thus we obtain a tight wavelet frame by Theorem 2 and if equality* (2.5) *holds, then the frame bound is equal to 1.*

2.3 Multiresolution Analysis for Wavelets with Box Fourier Transforms

This section is concerned with the two-dimensional generalization of Examples I and K on pages 386 and 390, respectively, of [29].

Define the classical Hardy Spaces $H^2(\mathbb{R}_\pm)$ by

$$H^2(\mathbb{R}_\pm) = \left\{ f \in L^2(\mathbb{R}) \mid \widehat{f}(\xi) = 0 \quad \text{a.a. } \xi \leq (\geq) 0 \right\}.$$

In these examples, a wavelet function ψ_- and a scaling function ϕ_- for orthonormal wavelets of $H^2(\mathbb{R}_-)$ are defined by

$$\widehat{\psi}_- = \chi_{[2\pi,4\pi]}, \qquad \widehat{\phi}_+ = \chi_{[0,2\pi]}.$$

From the two-scale relation

$$\widehat{\phi}_+(2\xi) = m_0(\xi)\widehat{\phi}_+(\xi)$$

it is found that the corresponding lowpass filter is

$$m_0(\xi) = \chi_{[0,\pi]}(\xi) = \widehat{\phi}_+(2\xi)$$

on $[0, 2\pi)$, and extended 2π-periodically. From the two-scale relation

$$\widehat{\psi}_+(2\xi) = e^{i\xi}\overline{m_0(\xi+\pi)}\widehat{\phi}_+(\xi) = m_1(\xi)\widehat{\phi}_+(\xi)$$

it is found that the corresponding highpass filter is

$$m_1(\xi) = e^{i\xi}\overline{m_0(\xi+\pi)} = e^{i\xi}\widehat{\psi}_+(2\xi)$$

on $[0, 2\pi)$, and extended 2π-periodically.

By the same argument, there are a wavelet function ψ_- and a scaling function ϕ_- for orthonormal wavelets of $H^2(\mathbb{R}_-)$. Since

$$L^2(\mathbb{R}) = H^2(\mathbb{R}_+) \oplus H^2(\mathbb{R}_-),$$

$\{\psi_+, \psi_-\}$ and $\{\phi_+, \phi_-\}$ can be regarded as multiwavelet functions and multiscaling functions, respectively, of $L^2(\mathbb{R})$.

For the two-dimensional case, we can take the tensor product of multiresolution analyses for one-dimensional multiwavelets. Then the four multiscaling functions ϕ^1, ϕ^2, ϕ^3, ϕ^4 are defined by

$$\widehat{\phi}^1(\xi_1, \xi_2) := \widehat{\phi}_+(\xi_1)\widehat{\phi}_+(\xi_2), \qquad \widehat{\phi}^2(\xi_1, \xi_2) := \widehat{\phi}_+(\xi_1)\widehat{\phi}_-(\xi_2),$$
$$\widehat{\phi}^3(\xi_1, \xi_2) := \widehat{\phi}_-(\xi_1)\widehat{\phi}_+(\xi_2), \qquad \widehat{\phi}^4(\xi_1, \xi_2) := \widehat{\phi}_-(\xi_1)\widehat{\phi}_-(\xi_2).$$

To each multiscaling function, there correspond three multiwavelet functions. These

12 multiwavelet functions w^ℓ, $\ell = 1, 2, \ldots, 12$, are defined by

$$\begin{aligned}
\widehat{w}^1(\xi_1,\xi_2) &:= \widehat{w}_-(\xi_1)\widehat{\phi}_-(\xi_2), & \widehat{w}^2(\xi_1,\xi_2) &:= \widehat{w}_+(\xi_1)\widehat{w}_+(\xi_2),\\
\widehat{w}^3(\xi_1,\xi_2) &:= \widehat{\phi}_+(\xi_1)\widehat{w}_+(\xi_2), & \widehat{w}^4(\xi_1,\xi_2) &:= \widehat{\phi}_-(\xi_1)\widehat{w}_-(\xi_2),\\
\widehat{w}^5(\xi_1,\xi_2) &:= \widehat{w}_-(\xi_1)\widehat{w}_+(\xi_2), & \widehat{w}^6(\xi_1,\xi_2) &:= \widehat{w}_-(\xi_1)\widehat{\phi}_+(\xi_2),\\
\widehat{w}^7(\xi_1,\xi_2) &:= \widehat{w}_+(\xi_1)\widehat{\phi}_-(\xi_2), & \widehat{w}^8(\xi_1,\xi_2) &:= \widehat{w}_-(\xi_1)\widehat{w}_-(\xi_2),\\
\widehat{w}^9(\xi_1,\xi_2) &:= \widehat{\phi}_-(\xi_1)\widehat{w}_-(\xi_2), & \widehat{w}^{10}(\xi_1,\xi_2) &:= \widehat{\phi}_+(\xi_1)\widehat{w}_-(\xi_2),\\
\widehat{w}^{11}(\xi_1,\xi_2) &:= \widehat{w}_+(\xi_1)\widehat{w}_-(\xi_2), & \widehat{w}^{12}(\xi_1,\xi_2) &:= \widehat{w}_-(\xi_1)\widehat{\phi}_-(\xi_2).
\end{aligned}$$

Four square annuli, each made of 12 wavelets with box Fourier Transform $\widehat{w}_j^\ell$, are shown in Fig. 2.2, with artificially added spacing around boxes for immediate visual perception of the boxes. The three families in quadrant α ($\alpha = 1, 2, 3, 4$), can be generated by the corresponding scaling function ϕ^α. The scaling function satisfies the identity

$$|\widehat{\phi}^\alpha(\xi)|^2 = \sum_{j=1}^{\infty}\sum_{\beta=1}^{3} |\widehat{w}^{3(\alpha-1)+\beta}(2^j\xi)|^2.$$

In this case, since the $\widehat{w}^\ell$ are characteristic functions of disjoint sets, the squares and absolute values may be removed to get

$$\widehat{\phi}^\alpha(\xi) = \sum_{j=1}^{\infty}\sum_{\beta=1}^{3} \widehat{w}^{3(\alpha-1)+\beta}(2^j\xi).$$

It follows that each ϕ^α ($\alpha = 1, 2, 3, 4$), being the characteristic function of one of the four central squares in Fig. 2.2, satisfies the two-scale equation

$$\widehat{\phi}^\alpha(2\xi) = m_0^\alpha(\xi)\widehat{\phi}^\alpha(\xi) \tag{2.10}$$

with lowpass filter

$$m_0^\alpha(\xi) = \widehat{\phi}^\alpha(2\xi) = \sum_{j=2}^{\infty}\sum_{\beta=1}^{3} \widehat{w}^{3(\alpha-1)+\beta}(2^j\xi) = \chi_{\mathrm{supp}\,\widehat{\phi}^\alpha(2\cdot)}(\xi) \tag{2.11}$$

on $[0, 2\pi) \times [0, 2\pi)$, and extended $(2\pi \times 2\pi)$-periodically. Then each wavelet $\widehat{\omega}^\ell$, $\ell = 3(\alpha - 1) + \beta$, where $\alpha = 1, 2, 3, 4$, and $\beta = 1, 2, 3$, satisfies the two-scale equation

$$\widehat{\omega}^\ell(2\xi) = m_1^\ell(\xi)\widehat{\phi}^\alpha(\xi) \tag{2.12}$$

with highpass filter

$$m_1^\ell(\xi) = \widehat{\omega}^\ell(2\xi) = \chi_{\text{supp}\,\widehat{\omega}^\ell(2\cdot)}(\xi) \tag{2.13}$$

on $[0, 2\pi) \times [0, 2\pi)$, and extended $(2\pi \times 2\pi)$-periodically.

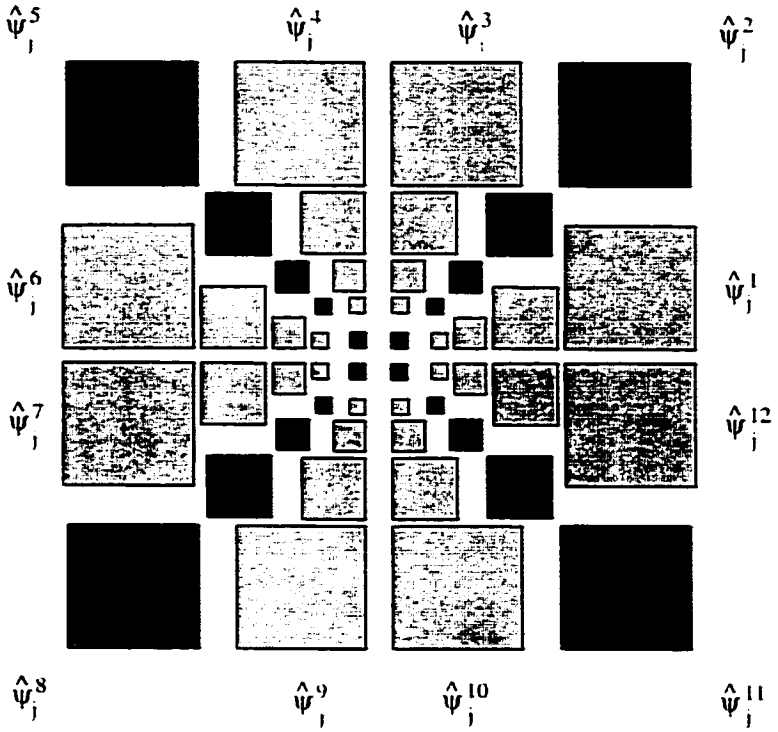

Figure 2.2: Twelve orthonormal wavelets $\{\widehat{\omega}_J^\ell\}$ in Fourier Space.

Because of the form of the two-scale equations (2.10) and (2.12), and the lowpass and highpass filters (2.11) and (2.13), which are the periodized characteristic functions of the supports of $\widehat{\phi}^\alpha(2\xi)$ and $\widehat{\omega}^\ell(2\xi)$, respectively, a multiwavelet multiresolution analysis could be generated by one scaling function consisting of the characteristic function of the central square made of the four central squares in Fig. 2.2.

Similarly, one can use 12 scaling functions of the form

$$\widehat{\phi}^{\ell}(\xi) = \sum_{j=1}^{\infty} \widehat{\psi}^{\ell}(2^j\xi).$$

with the two-scale relation

$$\widehat{\phi}^{\ell}(2\xi) = m_0^{\ell}(\xi)\widehat{\phi}^{\ell}(\xi)$$

and lowpass filter

$$m_0^{\ell}(\xi) = \widehat{\phi}^{\ell}(2\xi) = \sum_{j=2}^{\infty} \widehat{\psi}^{\ell}(2^j\xi) = \chi_{\mathrm{supp}\,\widehat{\phi}^{\ell}(2\cdot)}(\xi)$$

on $[0, 2\pi) \times [0, 2\pi)$, and extended $(2\pi \times 2\pi)$-periodically. The wavelet $\widehat{\psi}^{\ell}$ satisfies the two-scale equation

$$\widehat{\psi}^{\ell}(2\xi) = m_1^{\ell}(\xi)\widehat{\phi}^{\ell}(\xi)$$

with highpass filter

$$m_1^{\ell}(\xi) = e^{i(\xi_1-\xi_2)}\widehat{\psi}^{\ell}(2\xi) = e^{i(\xi_1-\xi_2)}\chi_{\mathrm{supp}\,\widehat{\phi}^{\ell}(2\cdot)}(\xi)$$

on $[0, 2\pi) \times [0, 2\pi)$, and extended $(2\pi \times 2\pi)$-periodically.

In the terminology of image processing, these are infinite impulse response filters.

2.4 Painless Smooth Tight Frame Wavelets

The argument in this section is in the spirit of the one-dimensional tight frames given in [28], which are themselves taken from [16].

Let Q_0 be the square $[-\pi, \pi] \times [-\pi, \pi]$ centered at the origin and consider a "square annulus" made of the 12 squares, $Q_1, \ldots, Q_{12}$, with sides of length π surrounding Q_0, similar to what is shown in Fig. 2.4.

Let ε be given such that $0 < \varepsilon < \pi/2$, and let $\widetilde{Q}_\ell$ be an enlarged version of the square Q_ℓ by at most a band of width ε along each side. For example, for $Q_1 = [\pi, 2\pi] \times [0, \pi]$,

$$\widetilde{Q}_1 = [\pi - \varepsilon, 2\pi + \varepsilon] \times [-\varepsilon, \pi + \varepsilon].$$

Let $g_\ell \in L^2(\mathbb{R}^2)$ be such that:

- $\widehat{g}_\ell$ is continuous.
- $\widehat{g}_\ell$ is supported in $\widetilde{Q}_\ell$.
- $\widehat{g}_\ell$ is nonzero in the interior of $\widetilde{Q}_\ell$, and
- $\widehat{g}_\ell$ is identically 1 on Q_ℓ.

Define

$$G(\xi) = \sum_{\ell=1}^{12} \sum_{j=-\infty}^{\infty} |\widehat{g}_\ell(2^{-j}\xi)|^2. \qquad \xi \in \mathbb{R}^2. \tag{2.14}$$

For any given point ξ, there are only finitely many nonzero terms in the series in equation (2.14). Since each $\widehat{g}_\ell$ is bounded, then G is bounded above. For any given point $\xi \neq 0$, there is at least one term in equation (2.14) such that $2^{-j}\xi$ lies in some Q_ℓ. Hence $G(\xi) \geq 1$ except at the origin, where it is zero.

Define the functions

$$\widehat{\omega}_\ell(\xi) = \frac{\widehat{g}_\ell(\xi)}{\sqrt{G(\xi)}}. \qquad \ell = 1, \ldots, 12. \tag{2.15}$$

It is clear that $\widehat{\omega}_\ell(\xi)$ is in $L^2(\mathbb{R}^2)$. For each point $\xi \neq 0$ we have

$$\sum_{\ell=1}^{12} \sum_{j=-\infty}^{\infty} |\widehat{\omega}_\ell(2^{-j}\xi)|^2 = 1 \tag{2.16}$$

since G is invariant under dyadic scaling. (Note that this remains true even when $G(\xi)$ is unbounded.)

For $k \in \mathbb{Z}^2$, the functions

$$e_k(\xi) = \frac{1}{2\pi} e^{-i\xi \cdot k}. \qquad \xi \in \mathbb{R}^2. \tag{2.17}$$

form an orthonormal basis for $L^2([0, 2\pi]^2)$. The fact that $\widetilde{Q}_\ell$ is contained in a square with sides of length 2π will be used below.

As in (2.1), define the functions

$$\psi^\ell_{j,k}(x) = 2^j \psi^\ell(2^j x - k), \qquad j \in \mathbb{Z}, \quad k \in \mathbb{Z}^2, \quad \ell = 1, \ldots, 12. \tag{2.18}$$

Applying the Fourier Transform

$$\widehat{g}(\xi) = \int g(x)\, e^{-i\xi \cdot x}\, dx. \tag{2.19}$$

to (2.18), we obtain

$$\widehat{\psi}^\ell_{j,k}(\xi) = 2^{-j} e_k(2^{-j}\xi)\, 2\pi \widehat{\psi}^\ell(2^{-j}\xi). \tag{2.20}$$

Although it follows from Theorem 2 that the $\{\psi^\ell_{jk}\}$ form a tight frame with frame bound 1, an independent proof is given. For $f \in L^2(\mathbb{R}^2)$, by Plancherel's theorem

$$\begin{aligned}
\sum_{k\in\mathbb{Z}^2} |\langle f, \psi^\ell_{j,k}\rangle|^2 &= \frac{1}{(2\pi)^4} \sum_{k\in\mathbb{Z}^2} |\langle \widehat{f}, \widehat{\psi}^\ell_{j,k}\rangle|^2 \\
&= \frac{1}{(2\pi)^2} \sum_{k\in\mathbb{Z}^2} \left| \int \widehat{f}(\xi)\, 2^{-j} \overline{\widehat{\psi}^\ell(2^{-j}\xi)}\, e_{-k}(2^{-j}\xi)\, d\xi \right|^2 \\
&= \frac{1}{(2\pi)^2} \int |\widehat{f}(\xi)|^2\, |\widehat{\psi}^\ell(2^{-j}\xi)|^2\, d\xi.
\end{aligned}$$

where the last equality follows from the fact that $\widehat{\psi}^\ell(2^{-j}\xi)$ is supported in $2^j \tilde{Q}_i$ and $\{e_k(2^{-j}\xi)\}$ is an orthonormal basis for $L^2(2^j\tilde{Q}_i)$.

Consequently,

$$\begin{aligned}
\sum_{\ell=1}^{12} \sum_{j=-\infty}^{\infty} \sum_{k\in\mathbb{Z}^2} |\langle f, \psi^\ell_{j,k}\rangle|^2 &= \frac{1}{(2\pi)^2} \sum_{\ell=1}^{12} \sum_{j=-\infty}^{\infty} \int |\widehat{f}(\xi)|^2\, |\widehat{\psi}^\ell(2^{-j}\xi)|^2\, d\xi \\
&= \frac{1}{(2\pi)^2} \int |\widehat{f}(\xi)|^2 \left(\sum_{\ell=1}^{12} \sum_{j=-\infty}^{\infty} |\widehat{\psi}^\ell(2^{-j}\xi)|^2 \right) d\xi \\
&= \frac{1}{(2\pi)^2} \int |\widehat{f}(\xi)|^2\, d\xi \\
&= \|f\|^2_{L^2(\mathbb{R}^2)}.
\end{aligned}$$

Thus, the functions

$$\{\psi^\ell_{j,k}\}_{j\in\mathbb{Z}, k\in\mathbb{Z}^2, \ell=1,\ldots,12}$$

form a tight wavelet frame for $L^2(\mathbb{R}^2)$ with frame bound 1.

2.5 A Ring of 12 Tapered Frame Wavelets

In this section, tight frame wavelets with scaling function in the sense of Mallat [33], p. 83, satisfying (1.1) identically are constructed. This fact accelerates computation in the construction of the frame. In view of the numerical implementation in Section 2.8, it will be more convenient to consider squares with sides of length $1/2$ instead of length π and use definition (1.3) of the Fourier Transform.

Taper functions of one variable must be defined. Partition the time axis with points $\{a_j\}$ $(a_j < a_{j-1})$ into intervals, such that the jth interval is $[a_j, a_{j-1}]$ and has length $L_j = a_{j-1} - a_j$. Around each endpoint of an interval, say a_j, allow a transition region $[a_j - \varepsilon_j, a_j + \varepsilon_j]$ of width $2\varepsilon_j$; in this region, the window function $b_j(t)$ over interval j rises smoothly from 0 to 1, and the window function $b_{j-1}(t)$ over interval $j-1$ decreases smoothly from 1 to 0. The window or bell function, $b_j(t)$, is nonzero for t in the region $(a_j - \varepsilon_j, a_{j-1} + \varepsilon_{j-1})$ and it is 1 for t in $[a_j + \varepsilon_j, a_{j-1} - \varepsilon_{j-1}]$. The window functions over two adjacent intervals overlap in the transition region.

A window function, $b_j(t)$, has the following properties:

(i) $0 \le b_j(t) \le 1$ for all t and

$$b_j(t) = \begin{cases} 1 \text{ if } a_j + \varepsilon_j \le t \le a_{j-1} - \varepsilon_{j-1} \\ 0 \text{ if } t \le a_j - \varepsilon_j \text{ or } t \ge a_{j-1} + \varepsilon_{j-1} \end{cases}$$

where $\varepsilon_j \ge 0$ and $\varepsilon_j + \varepsilon_{j-1} \le L_j$;

(ii) $b_j^2(a_j + t) + b_j^2(a_j - t) = 1$ if $|t| \le \varepsilon_j$;

(iii) $b_j(a_j + t) = b_{j-1}(a_j - t)$ if $|t| \le \varepsilon_j$.

Condition (i) says that the window functions are simply smoothed versions of the rectangular window and b_j can be specified if it is known in the transition regions

$[a_j - \varepsilon_j, a_j + \varepsilon_j]$ and $[a_{j+1} - \varepsilon_{j+1}, a_{j+1} + \varepsilon_{j+1}]$. So only the window function in the transition regions needs to be considered and it is referred to as a taper function.

Without any loss of generality, consider a taper function $b(t)$ over the transition region $[0, 1]$, that is, $2\varepsilon_j = 1$ and $a_j = 0.5$, since a taper function in an arbitrary region $[a_j - \varepsilon_j, a_j + \varepsilon_j]$ can be obtained by a simple change of variables $b((t - (a_j - \varepsilon_j))/2\varepsilon_j)$. Condition (ii) is

$$b^2(t) + b^2(1 - t) = 1.$$

Condition (iii) involves adjacent windows. It says that the two adjacent windows in the overlapping region are symmetric about the endpoint a. In other words, the tail end of the window $j - 1$ is the reflection of the beginning of the window j.

Figure 2.3 (left) shows three tapered characteristic functions: (a) of the interval $[0, 1]$ with transition width $1/2$ at both ends, (b) of the interval $[1, 2]$ with transition widths $1/2$ at the left end and 1 at the right end, and (c) of the interval $[2, 4]$ with transition widths 1 at the left end and 2 at the right end. It is seen in Fig. 2.3 (right) that the square root of the sum of the square of these three functions is one over the overlapping tapered parts.

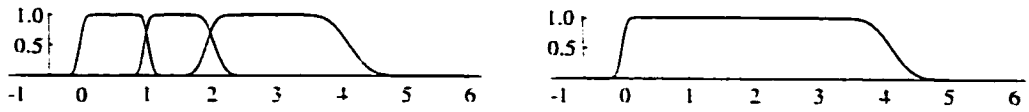

Figure 2.3: Left: three overlapping tapered characteristic functions; right: square root of sum of squares of the three functions.

Taper functions of two variables are defined by the tensor product of two taper functions of one variable.

$$b(s, t) = b_1(s)b_2(t).$$

The tapered characteristic functions of the 12 squares of side $1/2$ of ring $\mathcal{R}^{[0]}$, shown in Fig. 2.4, have transition widths 4ε on the outside edges and 2ε on all the other edges.

Similarly, the three squares of side 1 of the second ring $\mathcal{R}^{[1]}$, in the first quadrant, produced by dilation by 2, have transition widths 8ε on the three outside edges and 4ε on the remaining edges.

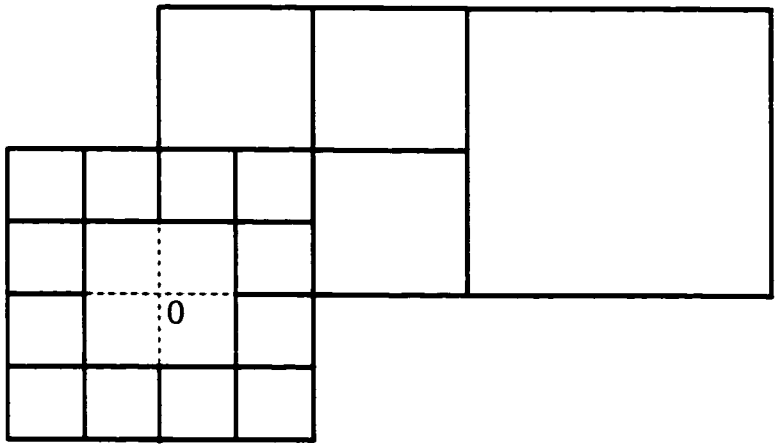

Figure 2.4: Twelve squares with sides of length 1/2 of ring $\mathcal{R}^{[0]}$, 3 squares with sides of length 1 of ring $\mathcal{R}^{[1]}$ and 1 square with sides of length 2 of ring $\mathcal{R}^{[2]}$ in the first quadrant.

In general, the ring $\mathcal{R}^{[j]}$, parametrized by $j \in \mathbb{Z}$, is the support of tapered characteristic functions

$$\widehat{\omega}_j^{\ell}(\xi) := \widehat{\omega}^{\ell}(2^j\xi), \qquad \ell = 1, 2, \ldots, L,$$

where the Fourier Transform is defined by (1.3). For fixed j, the only rings that intersect with $\mathcal{R}^{[j]}$ are $\mathcal{R}^{[j-1]}$ and $\mathcal{R}^{[j+1]}$. Given one ring $\mathcal{R}^{[j]}$, the other rings are simply obtained by dilation.

To prove identity (1.1), only points where two, three and four tapered wavelet frame functions overlap need to be checked. Identity (1.1) at an arbitary point ξ in the intersection of the four tapered parts of $\widehat{\omega}_1^2$ and $\widehat{\omega}_2^{\ell}$ ($\ell = 1, 2, 3$) will be checked. Since tapering is of the same width for all four transition regions, it is obtained by the same tapering function b. Denote by lh, rh, bv and tv the left and right horizontal distances and bottom and top vertical distances, respectively, from the beginning of

the tapering. Then, proceeding counter-clockwise from the top right corner of frame $\widehat{\omega}_1^2$.

$$\begin{aligned} b^2(lh)b^2(bv) + b^2(bv)b^2(rh) + b^2(rh)b^2(tv) + b^2(tv)b^2(lh) \\ &= b^2(bv)[b^2(lh) + b^2(rh)] + b^2(tv)[b^2(rh) + b^2(lh)] \\ &= b^2(bv) + b^2(tv) \\ &= 1. \end{aligned}$$

The treatment of smooth frames for $H^2(\mathbb{R})$ given in Section 8.4 of [29] is generalized to two dimensions to show that the $\omega_{j,k}^\ell$ form a tight wavelet frame with frame bound 1 for $L^2(\mathbb{R}^2)$.

If $0 < \varepsilon \leq 1/2$, the system $\{\omega_{j,k}^\ell\}$, $j \in \mathbb{Z}$ and $k \in \mathbb{Z}^2$, is a wavelet frame for $L^2(\mathbb{R}^2)$. For $f \in L^2(\mathbb{R}^2)$, Plancherel's theorem is used to obtain

$$\begin{aligned} \sum_{k\in\mathbb{Z}^2} |\langle f, \omega_{j,k}^\ell\rangle|^2 &= \sum_{k\in\mathbb{Z}^2} \left| \int \widehat{f}(\xi) 2^{-j} \overline{\widehat{\omega}^\ell(2^{-j}\xi)}\, e^{-2\pi i 2^{-j} k\cdot\xi}\, d\xi \right|^2 \\ &= \int |\widehat{f}(\xi)|^2\, |\widehat{\omega}^\ell(2^{-j}\xi)|^2\, d\xi. \end{aligned}$$

where the last equality follows from the fact that $\widehat{\omega}^\ell(2^{-j}\xi)$ is supported in $2^j\tilde{Q}_i$ and $\{e^{-2\pi i 2^{-j}k\cdot\xi}\}$ is an orthonormal basis for $L^2(2^j\tilde{Q}_i)$. Consequently,

$$\begin{aligned} \sum_{\ell=1}^{12} \sum_{j=-\infty}^{\infty} \sum_{k\in\mathbb{Z}^2} |\langle f, \omega_{j,k}^\ell\rangle|^2 &= \sum_{\ell=1}^{12} \sum_{j=-\infty}^{\infty} \int |\widehat{f}(\xi)|^2\, |\widehat{\omega}^\ell(2^{-j}\xi)|^2\, d\xi \\ &= \int |\widehat{f}(\xi)|^2 \left(\sum_{\ell=1}^{12} \sum_{j=-\infty}^{\infty} |\widehat{\omega}^\ell(2^{-j}\xi)|^2 \right) d\xi \\ &= \int |\widehat{f}(\xi)|^2\, d\xi \\ &= \|f\|^2_{L^2(\mathbb{R}^2)}. \end{aligned}$$

The tapered characteristic function $\widehat{\phi}(\xi)$ of the unit square inside ring $\mathcal{R}^{[0]}$, with transition width 2ε is a scaling function. It satisfies the two-scale relation

$$\widehat{\phi}(2\xi) = m_0(\xi)\widehat{\phi}(\xi)$$

with lowpass filter

$$m_0(\xi) = \widehat{\phi}(2\xi)$$

on $[-1/2, 1/2) \times [-1/2, 1/2)$, and extended (1×1)-periodically. The function $\widehat{\phi}(\xi)$ satisfies the identity

$$|\widehat{\phi}(\xi)|^2 = \sum_{j=1}^{\infty} \sum_{\ell=1}^{12} |\widehat{\psi}^{\ell}(2^j\xi)|^2.$$

The wavelet $\widehat{\psi}^{\ell}$ satisfies the two-scale equation

$$\widehat{\psi}^{\ell}(2\xi) = m_1^{\ell}(\xi)\widehat{\phi}^{\ell}(\xi).$$

The highpass filter is a partially tapered characteristic function of the form

$$m_1^{\ell}(\xi) = b_1(\xi_1)b_2(\xi_2)$$

on $[-1/2, \varepsilon + 1/2) \times [-1/2, 1/2)$, and extended (1×1)-periodically, where $b_1(\xi)$ is the characteristic function $\chi_{[-1/2, -1/2+\varepsilon]}(\xi_1)$, tapered at the left end with taper region of length ε, and $b_2(\xi_2)$ is the characteristic function $\chi_{[-1/4, 1/4]}(\xi_2)$, tapered at both ends with taper region of length ε.

Finer angular localization in Fourier Space is achieved by rings of 48 wavelet frame functions by dividing each of the previous 12 squares into four squares as shown in Fig. 2.5. These functions are tapered and have transition widths 4ε on the outside edges and 2ε on all the other edges. It is easy to see that identity (1.1) is satisfied.

2.6 Pseudo-multiresolution of Polar Frame Wavelets

Polar rings offer an arbitrary number of angular resolutions. A first ring with a few angular functions and part of the second ring are shown in Fig. 2.6. It is to be noticed that the radial interval is of the form $[r, 2r]$. Radial wavelets are defined by their Fourier Transforms as characteristic functions over each set. If the plane is totally covered by

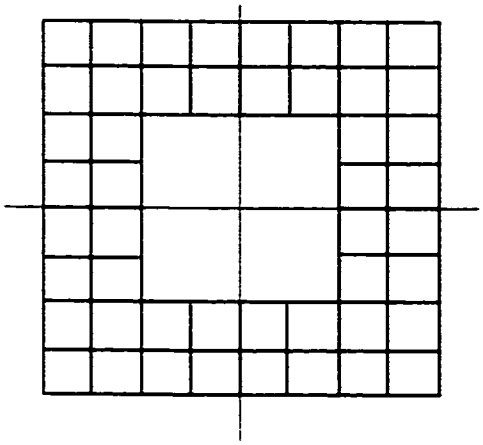

Figure 2.5: Ring of 48 wavelet frame functions in Fourier Space.

L nonoverlapping wedges, the L family of wavelets $\{\psi^{\ell}_{j,k}\}$, with $\ell = 1, \ldots, L$, $j \in \mathbb{Z}$, and $k \in \mathbb{Z}^2$, form an orthonormal basis of $L^2(\mathbb{R}^2)$. This basis is generated by a multiresolution analysis with scaling functions defined as in the case of box wavelets.

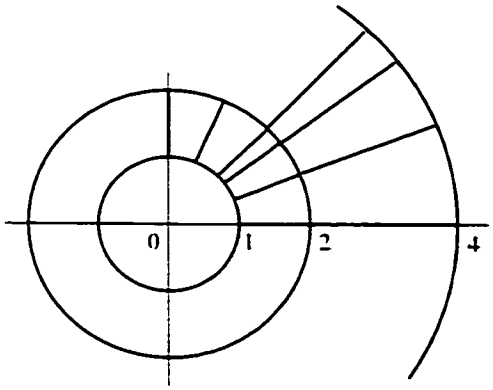

Figure 2.6: Polar frame in Fourier Space.

To have better localization in the x-Space, these functions are tapered with width 2ε on the inside arcs, width 4ε on the outside arcs and width εs, $r \leq s \leq 2r$, on the straight edges. Provided ε is sufficiently small so that tapering overlaps are restricted to immediately adjacent regions, it will be shown below that identity (1.1) is satisfied. By taking a ring sufficiently close to the origin, Remark 1 implies that polar rings form

a tight frame with frame bound 1.

Polar frame wavelets which are almost orthogonal can be associated with a pseudo-multiresolution analysis. In fact, the wavelets ψ^ℓ with Fourier Transform $\widehat{\psi}^\ell$ covering the ℓth tapered wedge can be generated by a scaling function ϕ^ℓ. The scaling function satisfies the identity

$$|\widehat{\phi}^\ell(\xi)|^2 = \sum_{j=1}^{\infty} |\widehat{\psi}^\ell(2^j\xi)|^2.$$

It follows that $\widehat{\phi}^\ell$ satisfies the two-scale equation

$$\widehat{\phi}^\ell(2\xi) = m_0^\ell(\xi)\widehat{\phi}^\ell(\xi)$$

with lowpass filter

$$m_0^\ell(\xi) = \widehat{\phi}^\ell(2\xi) = \sum_{j=2}^{\infty} \widehat{\psi}^\ell(2^j\xi)$$

on a square with sides of length 2π containing the support of $\widehat{\phi}^\ell(2\xi)$, and extended $(2\pi \times 2\pi)$-periodically. The wavelet $\widehat{\psi}^\ell$ satisfies the two-scale equation

$$\widehat{\psi}^\ell(2\xi) = m_1^\ell(\xi)\widehat{\phi}^\ell(\xi)$$

with highpass filter

$$m_1^\ell(\xi) = \chi_{\text{supp}\,\widehat{\psi}^\ell(2\cdot)}(\xi)$$

on a square with sides of length 2π containing the support of $\widehat{\psi}^\ell(2\xi)$, and extended $(2\pi \times 2\pi)$-periodically. The scaling functions and wavelets with adjoining support are not mutually orthogonal. Thus it is only a pseudo-multiresolution analysis (pseudo-MRA). In the terminology of image processing, these are infinite impulse response filters.

The conception of polar frame wavelets is much simplified by using polar coordinates (r, θ) in Fourier Space instead of the Cartesian coordinates (ξ_1, ξ_2). The inverse and

direct transformations are

$$\xi_1 = r\cos\theta, \qquad \xi_2 = r\sin\theta.$$

and

$$r = \sqrt{\xi_1^2 + \xi_2^2}, \qquad \theta = \arctan\left(\frac{\xi_2}{\xi_1}\right)$$

with the appropriate branches of arctan. With the direct transformation, the radial frame wavelets are supported in the semi-infinite strip

$$0 \le r < \infty, \qquad 0 \le \theta \le 2\pi$$

(2π-periodic in θ). The strip is divided into rectangles with vertical sides at $r = 2^j$, $j \in \mathbb{Z}$ and horizontal sides at

$$0 = \theta_0 < \theta_1 < \cdots < \theta_\ell < \ldots < \theta_L = 2\pi.$$

Tapering of characteristic functions of each rectangle is now quite easy. Tapering is done along vertical sides with taper regions of length $2^j\varepsilon$ at $r = 2^j$ and along horizontal sides with taper regions of length $2\varepsilon_\ell$ with $\varepsilon_L = \varepsilon_0$ at $\theta = \theta_L$. It is then clear that the identity (1.1) is satisfied.

The frame wavelets of any horizontal strip can be obtained from a pseudo-MRA. The scaling function $\widehat{\phi}^\ell(r,\theta)$ satisfies the identity

$$|\widehat{\phi}^\ell(r,\theta)|^2 = \sum_{j=1}^{\infty} |\widehat{w}^\ell(2^j r,\theta)|^2.$$

It follows that $\widehat{\phi}^\ell$ satisfies the two-scale equation

$$\widehat{\phi}^\ell(2r,\theta) = m_0^\ell(r,\theta)\widehat{\phi}^\ell(r,\theta)$$

with lowpass filter

$$m_0^\ell(r,\theta) = \widehat{\phi}^\ell(2r,\theta) = \sum_{j=2}^{\infty} \widehat{w}^\ell(2^j r,\theta) = \chi_{\operatorname{supp}\widehat{\phi}^\ell(2\cdot)}(r,\theta)$$

on $[0,1) \times [0,2\pi)$, and extended $(1 \times 2\pi)$-periodically. The wavelet $\widehat{\psi}^\ell$ satisfies the two-scale equation

$$\widehat{\psi}^\ell(2r,\theta) = m_1^\ell(r,\theta)\widehat{\phi}^\ell(r,\theta)$$

with highpass filter

$$m_1^\ell(r,\theta) = e^{i(\xi_1-\xi_2)}\widehat{\psi}^\ell(2r,\theta) = \chi_{\text{supp}\,\widehat{\psi}^\ell(2\cdot)}(r,\theta)$$

on $[0,1) \times [0,2\pi)$, and extended $(1 \times 2\pi)$-periodically.

An almost orthogonal multiresolution analysis can also be obtained by one scaling function $\widehat{\phi}(r,\theta)$ which is a tapered characteristic function of the rectangle

$$0 \le r < \frac{1}{2}, \qquad 0 \le \theta < 2\pi,$$

corresponding to a disk with centre at the origin in x-Space. This function satisfies the two-scale relation

$$\widehat{\phi}(2r,\theta) = m_0(r,\theta)\widehat{\phi}(r,\theta),$$

where

$$m_0(r,\theta) = \widehat{\phi}(2r,\theta)$$

on $0 \le r < 1$, $0 \le \theta < 2\pi$, and extended $(1 \times 2\pi)$-periodically. The wavelet $\widehat{\psi}^\ell(r,\theta)$ satisfies the two-scale relation

$$\widehat{\psi}^\ell(2r,\theta) = m_1^\ell(r,\theta)\widehat{\phi}(r,\theta)$$

where

$$m_1^\ell(r,\theta) = b_1(r)b_2^\ell(\theta)$$

on $0 \le r < 1$, $\theta_\ell \le \theta < \theta_{\ell+1}$, and extended $(1 \times 2\pi)$-periodically. Here $b_1(r)$ is the characteristic function $\chi_{[1/4,\varepsilon-1/2]}(r)$, tapered at the left end with taper region of length ε, and $b_2^\ell(\theta)$ is the characteristic funtion $\chi_{[\theta_\ell,\theta_{\ell+1}]}(\theta)$, tapered at the left and right ends

with taper regions of lengths ε_ℓ and $\varepsilon_{\ell-1}$, respectively. In the Cartesian ξ-plane, this scaling function corresponds to a scaling function whose support is a disk centered at the origin and with radius $\varepsilon + 1/2$.

2.7 General Construction of Pseudo-multiresolution Analyses

Given a ring of functions $\widehat{w}^\ell(\xi)$ as described in Remark 1, define the function $\widehat{\phi}(\xi)$ by the identity

$$|\widehat{\phi}(\xi)|^2 = \sum_{j=1}^{\infty} \sum_{\ell=1}^{L} |\widehat{w}^\ell(2^j\xi)|^2.$$

The function $\widehat{\phi}(\xi)$ with support in the region inside this ring is a scaling function with low pass filter $\widehat{\phi}(2\xi)$ extended periodically. The highpass filter $m_\ell(\xi)$ is the function

$$m_\ell(\xi) = \widehat{w}^\ell(2\xi)/\widehat{\phi}(\xi)$$

restricted to the support of $\widehat{w}^\ell(2\xi)$ and extended periodically. This follows from the facts that

$$\operatorname{supp} \widehat{w}^\ell(2\xi) \subset \operatorname{supp} \widehat{\phi}(\xi)$$

and $\widehat{\phi}(\xi)$ does not vanish on the support of supp $\widehat{w}^\ell(2\xi)$.

2.8 A Numerical Implementation of the Localization Method

Two-dimensional bell functions are produced by the Mathematica Wavelet Explorer by tapering characteristics functions over the unit square with tapering width 1/8 and 1/4 as appropriate to form the first ring of 12 functions. Tapering was done by iterating the sine function twice to get

$$b(t) = \sin\left(\frac{\pi}{2}\sin^2\left(\frac{\pi}{2}\sin^2\left(\frac{\pi}{2}t\right)\right)\right)$$

with the option `Taper -> {Trig[2],epsilon}`. It is easy to see that

$$b(t)^2 + b(1-t)^2 = 1. \quad t \in [0.1],$$

by noting that $\sin((\pi/2)(1-t)) = \cos((\pi/2)t)$ and repeated use of the identity $\cos^2 t = 1 - \sin^2 t$.

Six larger rings are produced by scaling the functions of the first ring. A tapered scaling function is produced from the characteristic function of the central square with sides of length 2 with tapering width 1/8. The support of these functions is a square with centre at the origin and sides of length 287.

These functions are evaluated in the form of tables by Mathematica and exported to MATLAB in the form of matrices:

$$Q_0. \quad Q_1^1. \quad \ldots. \quad Q_1^{12}. \quad \ldots. \quad Q_7^1. \quad \ldots. \quad Q_7^{12}.$$

By construction, the 87×87 matrix Q_7^2 has lower left and upper right elements in position (87.201) and (1.287). The application, therefore, will involve a matrix of size 287×287.

A barely visible singularity along the diagonal segment $S = (40.40) - (80.80)$ of a 287×287 matrix is introduced in the top left corner of the 256×256 matrix of the Barbara image, producing image f shown in Fig. 2.7. The singularity consists of a Hanning filter, defined by the MATLAB command `w=han(n)` from the following equation

$$w(k+1) = 0.5\left(1 - \cos\left(2\pi \frac{k}{n-1}\right)\right). \qquad k = 0.\ldots.n-1.$$

along the diagonal segment $(40.40) - (60.60)$. This function is tapered in the south-west direction, producing a surface S_1, followed by the negative of this surface along the diagonal segment $(60.60) - (80.80)$, producing a surface S_2. These surfaces are

combined in the form

$$S = 128(S_1 - S_2) + 128.$$

The integral of the singularity is zero so that its Discrete Fourier Transform will have zero constant term. Moreover, a smooth circular Gaussian surface centered at (60, 60) of the form

$$G = 100\, e^{-0.005[(r-60)^2-(s-60)^2]}$$

is added to the scarred Barbara image, producing an image f.

The Discrete Fourier Transform F of f is filtered by means of the top right frame, Q_7^2, of the seventh ring to recover the singularity and eliminate the smooth part of the image. Write $k = (k_1, k_2)$. Let the (m, n) element of the matrix of exponentials E_k be

$$(E_k)_{mn} = (E_{k_1,k_2})_{mn} = e^{-2\pi i(m-1,n-1)\cdot(k_1,k_2)/287}. \tag{2.21}$$

Discretizing the scalar product (2.8),

$$\langle \widehat{f}, \widehat{w}_{7,k}^2 \rangle,$$

gives the matrix $C = (c_k)$ of frame coefficients

$$c_k = \sum_{m,n=1}^{287} (F .* E_k .* Q_7^2)(m, n). \qquad k_1 = k_2 = 1, \ldots, 287. \tag{2.22}$$

where .* denotes componentwise matrix multiplication. Note that Q_7^2 is a real matrix. The plot of the absolute value of the entries of C shown in Fig. 2.8 clearly indicates the dominance of the location of the singularity of the image f.

Discretizing the Fourier Transform of the partial sum in (2.7),

$$\sum_{k_1=k_2=40}^{80} \langle \widehat{f}, \widehat{w}_{7,k}^2 \rangle\, \widehat{w}_{7,k}^2(\xi).$$

the matrix $W = (w_k)$ of the Discrete Fourier Transform of the filtered image is obtained, with entries

$$w_k = \left(\sum_{m_1=10}^{80} c_{m_1,m_1} E_{-m_1,-m_1} * Q_7^2 \right)_k . \tag{2.23}$$

where summation is over the diagonal segment S. The image of W is shown in Fig. 2.9.

The Inverse Discrete Fourier Transform of W is shown in Fig. 2.10. Since the Fourier Transform of the smooth component of the image is localized mainly near the origin and the Fourier Transform of the singularity consists mainly of details with high frequency, the properly modulated matrix filter Q_7^2 picks up only the singularity.

Quantization in Fourier Space of the Fast Fourier Transform (FFT) of the image by slices of width 1024 does not affect the results and could be used in reducing computer time and size of storage.

Another example consists in locating singularities and their directions in the zigzag image shown in Fig. 2.11. The frame $\widehat{\psi_{7,k}^2}(\xi)$, with support in the top right corner, picks up the singularities across segments parallel to the main diagonal. The plot of the frame coefficients given by (2.22) is shown in Fig. 2.12. The singularities across segments parallel to the secondary diagonal are similarly picked up by the frame $\widehat{\psi_{7,k}^3}(\xi)$ with support in the top left corner.

In Fig. 2.13, a random noise of intensity 50 has been added to the zigzag figure of intensity 100 (see Fig. 2.11). (Random noise is described in Chapter 4.) The presence of the noise did not prevent the singularities from being picked up by the frame $\widehat{\psi_{7,k}^3}(\xi)$, as shown in Fig. 2.14.

As a numerical application of the smooth polar frame wavelets, a circle of thickness 2 pixels with centre at the origin and radius 50 in x-Space, shown in Fig. 2.15, is localized by a tapered annulus with inner and outer diameters 44 and 287, respectively, in a 287×287 matrix. The figure of frame coefficients (2.22) for the given circle is

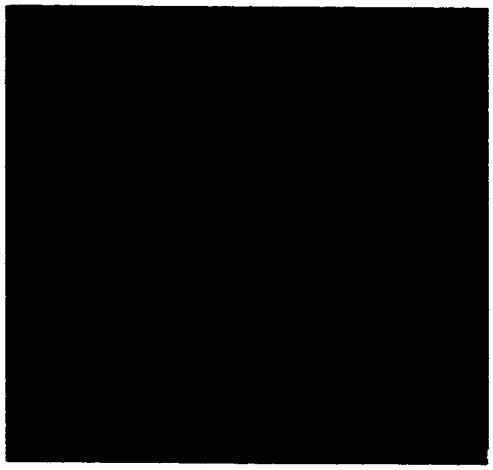

Figure 2.7: Barbara image with added singularity S and Gaussian surface in the top left of the image.

shown in Fig. 2.16.

These examples demonstrate that these frame wavelets are successful at localizing the singularities in images. Hence, they could be used to restore images that have been damaged by a singularity (an error in transmission, for example), by localizing the singularity in Fourier Space so that it may be removed from the image.

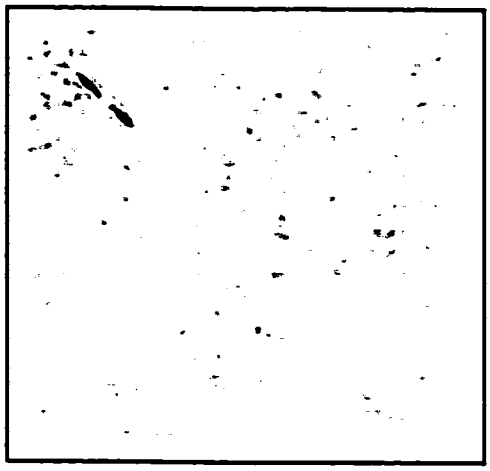

Figure 2.8: Location of the image singularity determined by the frame coefficients (2.22). On a scale from 0 to 1, black is 1 and white is 0. The 2-pixel wide black frame is to delimit the figure.

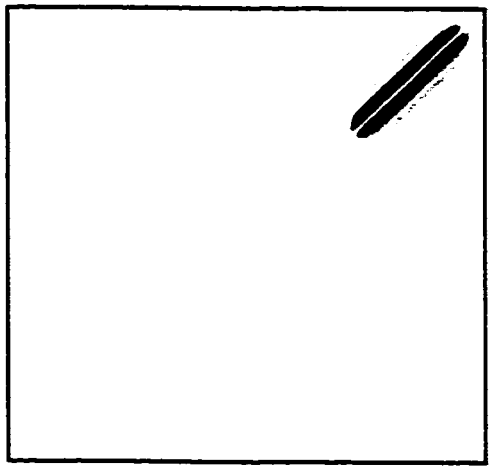

Figure 2.9: Location of the Fourier Transform of the image singularity determined by filtering (2.23). On a scale from 0 to 1, black is 1 and white is 0. The 2-pixel wide black frame is to delimit the figure.

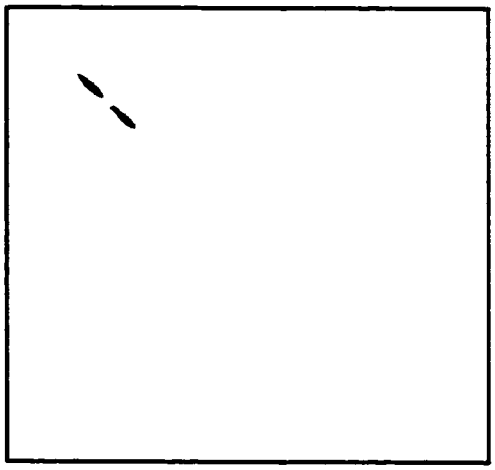

Figure 2.10: Location of the image singularity determined by filtering (2.23). On a scale from 0 to 1, black is 1 and white is 0. The 2-pixel wide black frame is to delimit the figure.

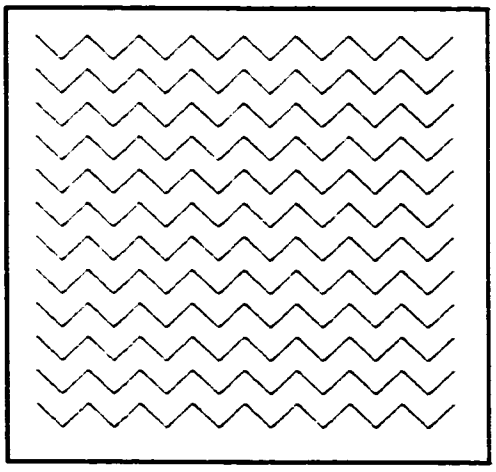

Figure 2.11: The zigzag image. On a scale from 0 to 1, black is 1 and white is 0. The 2-pixel wide black frame is to delimit the figure.

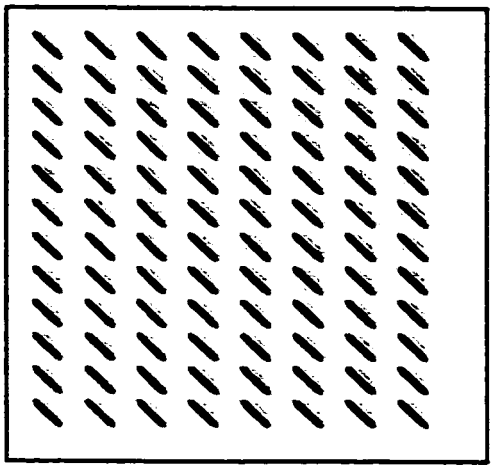

Figure 2.12: Location of the image singularity determined by the frame coefficients (2.22). On a scale from 0 to 1, black is 1 and white is 0. The 2-pixel wide black frame is to delimit the figure.

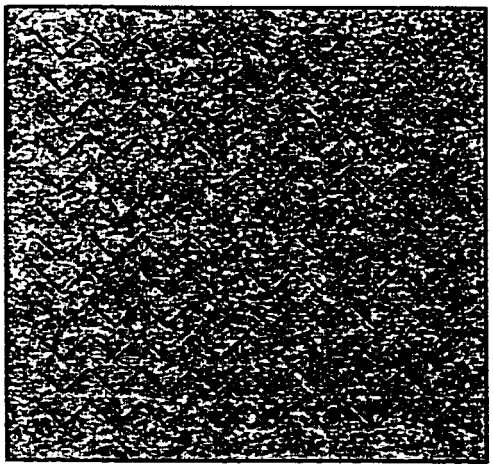

Figure 2.13: The zigzag image with random noise half the intensity of the zigzag curve. On a scale from 0 to 1, black is 1 and white is 0. The 2-pixel wide black frame is to delimit the figure.

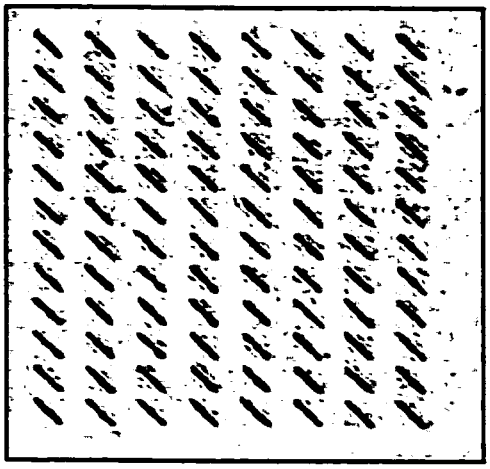

Figure 2.14: Location of the noisy zigzag image singularity determined by the frame coefficients (2.22). On a scale from 0 to 1, black is 1 and white is 0. The 2-pixel wide black frame is to delimit the figure.

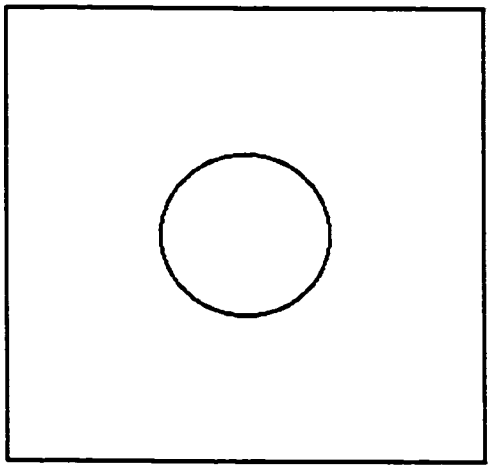

Figure 2.15: Circle of radius 50 in x-Space. On a scale from 0 to 1, black is 1 and white is 0. The 2-pixel wide black frame is to delimit the figure.

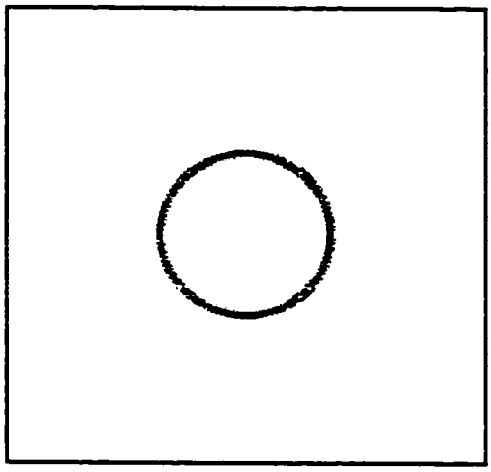

Figure 2.16: Location of the circle image singularity determined by the frame coefficients (2.22). On a scale from 0 to 1, black is 1 and white is 0. The 2-pixel wide black frame is to delimit the figure.

Chapter 3

The Product Filter

The diffusion equation.

$$u_t = u_{x_1 x_1} + u_{x_2 x_2}. \tag{3.1}$$

is of great use in image processing. It has been used to reduce noise in many schemes [1. 2. 3. 4. 5. 6. 11. 13. 14. 34. 35. 40. 43]. The direct application of the equation (by discretizing to a difference equation. for example) to a noisy image will diffuse the noise and smooth the image but may distort edges and details if too much smoothing is allowed. So then. there is a trade-off when applying the simple partial differential equation above – if diffusion is minimal. the noise may remain as a significant distortion to the image: if the diffusion is too strong and the image is smoothed significantly. then details may be lost.

The new alternative proposed is to diffuse minimally. but in many directions. In other words. allow diffusion to proceed without any restrictions with regards to position in the image or the nature of the point (be it a noisy point or an edge point). The more complex schemes discussed in Chapter 1 attempt to preserve edges by minimizing diffusion (or running it backwards in time) in their vicinity and to smooth noise by diffusing more around noisy points. In order to do this. of course. the edge points must be distinguished from the noisy points (typically this involves gradient evaluations).

It is this identification of points in the image that requires the types of PDE's seen in Chapter 1. It was believed that if diffusion were allowed to proceed minimally, but in many directions at once, the diffusion would be strong enough to smooth some of the noise, but not so strong as to ruin image details. A new, simpler method of using the diffusion equation to de-noise images uses a matrix filter that applies the diffusion equation in many directions. No assumptions are made about the nature of the noise or the properties of the image in this new method, with the idea then being to develop a method that would work reasonably well for any image and any type of noise.

To apply the diffusion equation in a specific direction, a change of variable is required. Diffusion in the x_1-direction is governed by $u_t = u_{x_1x_1}$, in the x_2-direction by $u_t = u_{x_2x_2}$. To diffuse in the direction of a line that makes an angle of θ with the x_1-axis, the governing equation would be:

$$u_t = \cos^2\theta\, u_{x_1x_1} + 2\sin\theta\cos\theta\, u_{x_1x_2} + \sin^2\theta\, u_{x_2x_2}.$$

The two-dimensional Fourier Transform of the function $u(x_1, x_2)$ is:

$$\widehat{u}(\xi_1, \xi_2) = \mathcal{F}\{u(x_1, x_2)\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(x_1, x_2)\, e^{-i(\xi_1 x_1 + \xi_2 x_2)}\, dx_1\, dx_2.$$

The Fourier Transform of the PDE would be:

$$\begin{aligned}
\widehat{u}_t &= -\xi_1^2\cos^2\theta\,\widehat{u} - 2\,\xi_1\,\xi_2\,\sin\theta\cos\theta\,\widehat{u} - \xi_2^2\sin^2\theta\,\widehat{u} \\
&= -[\xi_1^2\cos^2\theta + 2\,\xi_1\,\xi_2\,\sin\theta\cos\theta + \xi_2^2\sin^2\theta]\,\widehat{u} \\
&= -[\xi_1\cos\theta + \xi_2\sin\theta]^2\,\widehat{u}.
\end{aligned}$$

The equation is easily solved in Fourier Space. The solution would then be

$$\widehat{u}(\xi_1, \xi_2, t) = \widehat{u}(\xi_1, \xi_2, 0)\exp(-[\xi_1\cos\theta + \xi_2\sin\theta]^2\, t).$$

Also, application of the PDE to an image in Fourier Space would require multiplications, as compared to the finite differences required in x-Space. (See Chapter 6 on

third-order PDE's for examples of the difference equations required to apply a PDE directly to an image.)

Now, if diffusion were to be applied in many directions at once, specified by angles θ_k, the PDE would be:

$$u_t = \sum_k [\cos^2\theta_k\, u_{x_1x_1} + 2\sin\theta_k\cos\theta_k\, u_{x_1x_2} + \sin^2\theta_k\, u_{x_2x_2}].$$

Or, in Fourier Space:

$$\widehat{u}_t = -\left(\sum_k [\xi_1\cos\theta_k + \xi_2\sin\theta_k]^2\right)\widehat{u}.$$

which would have solution

$$\widehat{u}(\xi_1,\xi_2,t) = \widehat{u}(\xi_1,\xi_2,0)\exp\left(-\sum_k [\xi_1\cos\theta_k + \xi_2\sin\theta_k]^2\, t\right).$$

So, if the initial condition, $\widehat{u}(\xi_1,\xi_2,0)$, is taken to be the Fourier Transform of a noisy image, *i.e.*

$$\widehat{u}(\xi_1,\xi_2,0) = \mathcal{F}\{I(x_1,x_2) + N(x_1,x_2)\}.$$

then applying the multi-directional PDE in Fourier Space reduces to matrix multiplication with an appropriately chosen value of t, as will be explained below.

The Fourier Transform of an image, which is represented by a matrix, say of size $m \times n$, will also be an image of the same dimensions, and hence also represented by an $m \times n$ matrix in MATLAB. The origin of the $\xi_1\xi_2$ coordinate system in Fourier Space will be at the centre of the image matrix, more explicitly at position

$$((m+1)/2,(n+1)/2)$$

(which is the exact centre if both m and n are odd), with the ξ_1-axis running downwards and the ξ_2-axis to the right (this corresponds to the matrix coordinate system of MATLAB). And so, the (ξ_1,ξ_2) coordinates of matrix element (i,j) are

$$(\xi_1,\xi_2) = (i-(m+1)/2,\, j-(n+1)/2).$$

The algorithm to apply this multi-directional diffusion equation proceeds in the following manner. The number of directions, p, is chosen (typically 256 - the choice of this number is explained below). Then there will be p angles $\{\theta_k \mid 0 \leq k \leq p-1\}$, where

$$\theta_0 = 0,\ \theta_1 = \pi/p,\ \theta_2 = 2\pi/p,\ \ldots,\ \theta_{p-1} = (p-1)\pi/p.$$

For each θ_k, a matrix with (i, j) elements

$$[(i - (m+1)/2)\cos\theta_k + (j - (n+1)/2)\sin\theta_k]^2$$

is generated. These matrices are summed to produce a matrix with elements

$$\sum_{k=0}^{p-1}[(i - (m+1)/2)\cos\theta_k + (j - (n+1)/2)\sin\theta_k]^2.$$

This matrix is multiplied by the chosen value of t (typically of the order of 10^{-4}, as will be explained below), normalized by dividing by p, the number of directions and negated. This resulting matrix is exponentiated elementwise to produce the matrix with elements

$$\exp\left(-\sum_{k=0}^{p-1}[(i - (m+1)/2)\cos\theta_k + (j - (n+1)/2)\sin\theta_k]^2\, t/p\right).$$

This final matrix is then the product filter matrix for the algorithm.

The Fast Fourier Transform (FFT) (using MATLAB's `fft2` function) of the noisy image is multiplied elementwise by this filter matrix and the Inverse Fourier Transform (MATLAB's `ifft2`) is calculated to produce the smoothed (reduced-noise) image. `fft2` computes the two-dimensional FFT by composing the FFT algorithm with itself. The FFT of each column of the image matrix is calculated, where the kth component of the FFT of a vector x of length N is

$$X(k) = \sum_{j=1}^{N} x(j)\, e^{-2\pi i(j-1)(k-1)/N}.$$

and then the FFT is taken across the rows of the results. Function `fftshift` is used to move the DC to the centre of the matrix (where it should be for an image). `ifft2` takes the two-dimensional Inverse Fast Fourier Transform (IFFT), where

$$x(j) = \frac{1}{N} \sum_{k=1}^{N} X(k)\, e^{2\pi i (j-1)(k-1)/N}.$$

Chapter 4

Properties of the Product Filter

To test the diffusivity of the product filter with respect to the values of the parameter t and the number of directions p, an image of size 256×256 was generated that was all zeros, except for a central element (129, 129), that was set to be 100. The filter was applied to this image for various values of t (0.00001, 0.0001 and 0.001) and p (1, 2, 4, 8, 16, 32, 64, 128, 256 and 512). The maximum intensity value of the processed image was found and used as a measure of the strength of the diffusion. The results are presented in Figure 4.1. It can be seen from the graph of the results, that when $t = 0.00001$, diffusion is minimal (as the maximum value is high) for all values of p. For $t = 0.001$, diffusion is strong (the maximum value is low). When $t = 0.0001$, diffusion is moderate. It will be seen below that t of the order of 10^{-4} is most useful. For all three values of t, the general shape of the curve is the same. There is a strong change between $p = 1$ and $p = 8$ and then the curve levels off. $p = 256$ occurs in this region of levelling off and was chosen as a good compromise between having a large enough value to smooth in many directions versus a small enough value to make the calculations relatively quick. (A discussion of the calculation times required is presented below.) It should be noted that there will be thousands of possible directions in an image of size 256×256 (or larger), when all possible lines through the centre of the image and each

pixel are considered.

Figures 4.2 to 4.5 show the results of the new algorithm applied to the Barbara, Cameraman, Moon and Saturn images. In all cases, the noise was random noise of the form $50\,\mathrm{rand}(m,n) - 25\,\mathrm{rand}(m,n)$ and there were $p = 256$ directions. In MATLAB, `rand(m,n)` generates a matrix of size $m \times n$ where all values are random numbers between 0 and 1 with a uniform distribution (*i.e.* all values are equally likely). This matrix is multiplied by a constant in order to produce the random noise matrix to add to an image. Gaussian and salt and pepper noises are discussed below. Table 4.1 gives some statistical properties for typical random noises used.

Table 4.1: The statistical properties of the typical random noises used.

type of random noise	mean	std deviation	minimum	maximum
$50\,\mathrm{rand}(m,n)$	25.0530	14.46	2.64×10^{-4}	49.9994
$50\,\mathrm{rand}(m,n) - 25\,\mathrm{rand}(m,n)$	12.5317	16.16	−24.8127	49.9572
$100\,\mathrm{rand}(m,n)$	50.0299	28.92	8.39×10^{-4}	99.9984
$100\,\mathrm{rand}(m,n) - 50\,\mathrm{rand}(m,n)$	25.0988	32.19	−49.8361	99.8818

Figures 4.2 to 4.5 show the results for $t = 0.00001$, 0.0001, 0.001 and 0.01. For all four images, it can be seen that with $t = 0.00001$, the algorithm does virtually nothing, while for $t = 0.01$, there is far too much smoothing. The signal-to-noise ratios (SNR's) are given in Table 4.2. More detailed studies, both quantitative and visual, of the effect of the value of t are presented below.

The quantitative study of the value of the t parameter proceeded in the following manner. Two images, a constant and Barbara, were subjected to both random and Gaussian noise and the product filter was applied with several values of t and two values of p. The ratio of the SNR of the processed image to the SNR of the noisy image was calculated. The results are shown in Figures 4.6 to 4.9.

Since Barbara is a 256×256 image with mean intensity 118.8125, the constant

Table 4.2: Signal-to-noise ratios (SNR's) for the new product filter algorithm applied to the four images. Formula (1.8) for SNR is given in Chapter 1.

Figure	Image	Sub-image	t	SNR
4.2	Barbara	(b)	-	40.0945
		(c)	0.00001	42.7242
		(d)	0.0001	51.5011
		(e)	0.001	26.2457
		(f)	0.01	19.1867
4.3	Cameraman	(b)	-	43.2793
		(c)	0.00001	46.3497
		(d)	0.0001	63.1046
		(e)	0.001	45.1310
		(f)	0.01	22.4589
4.4	Moon	(b)	-	21.3041
		(c)	0.00001	26.1063
		(d)	0.0001	43.6330
		(e)	0.001	42.9123
		(f)	0.01	23.8530
4.5	Saturn	(b)	-	18.9383
		(c)	0.00001	21.8166
		(d)	0.0001	37.0732
		(e)	0.001	42.7724
		(f)	0.01	26.4392

image was taken to be 256×256 with all entries equal to 120. in order to have a fair comparison. Figures 4.6 and 4.7 present the results for random noise of the form $50\,\mathrm{rand}(m, n)$ for $p = 8$ and $p = 256$. It can be seen that the shapes of the curves are independent of p. The ratio of SNR increases for the constant image because here the more smoothing that is allowed (as t gets larger). the more the processed image resembles the original (which was smooth). For Barbara. the SNR ratio increases to a peak at approximately $t = 10^{-4}$ and then decreases. So. numerically. the optimal t value is near 10^{-4}.

These calculations were repeated for Gaussian noise of the form $25\,\mathrm{randn}(m, n)$. In

MATLAB. `randn(m,n)` produces a matrix of size $m \times n$. whose elements are normally distributed with mean 0 and variance 1. If randn(m, n) is multiplied by constant C (C randn(m, n)). the result is a matrix with elements with mean 0 and standard deviation C (verified numerically). So. 25 randn(m, n) produces Gaussian white noise with mean 0 and standard deviation 25. Figures 4.8 and 4.9 show the results. which are again independent of p. In the case of Gaussian noise. the SNR ratio for the constant image increases rapidly with t. For Barbara. the same basic shape that was seen with the uniformly distributed random noise is seen with the Gaussian noise – the SNR ratio increases to a maximum at $t = 2 \times 10^{-4}$ and then decreases. So. numerically. $t = 0.0002$ would be seen to be the optimum here. The fact that the SNR ratio values peak at a larger value should not be taken as an indication that the product filter does a better job at removing Gaussian noise than it does with random noise - this judgement needs to be made visually.

A visual study of the value of the t parameter was conducted by varying t over 28 values from 0.00001 to 0.01 and applying the product filter with $p = 256$ directions to the Barbara. Caneraman. Moon and Saturn images with noise of the form 50 rand(m, n). Approximately half of the Barbara study is presented in Figures 4.10 to 4.13. Representative results for the other images are given in Figures 4.14 to 4.19. The SNR values are collected in Table 4.3.

In all cases. it was seen that as t increased. there was more smoothing of the images. For the smaller values of t. the filter does not do anything visibly noticeable. For the larger values of t. there is significant smoothing and loss of detail. The optimal numerical value was found to be around $t = 10^{-4}$. This visual study indicates that the optimum range (for all four images) is $t = 0.0001$ to $t = 0.0005$. In this range. there is some smoothing of the noise without a significant loss of detail. Careful visual inspection of all four images in this range determined that a value of $t = 0.0003$ seems to

be optimal. In this case, optimal means the best compromise between noise reduction, detail preservation and differences for the four images.

Table 4.3: Signal-to-noise ratios (SNR's) for the product filter algorithm applied to the four images in the visual investigation of the parameter t. Formula (1.8) for SNR is given in Chapter 1. The largest value of SNR for each image is indicated in boldface.

parameter t	Barbara	Cameraman	Moon	Saturn
(noisy image)	19.8893	21.4869	10.6686	9.4123
0.00001	20.3775	22.0340	11.4550	9.9323
0.00002	20.7651	22.5092	12.0105	10.3408
0.00003	21.0620	22.9182	12.4053	10.6616
0.00004	21.2788	23.2672	12.6904	10.9146
0.00005	21.4260	23.5620	12.9007	11.1157
0.00006	21.5138	23.8086	13.0589	11.2771
0.00007	**21.5515**	24.0124	13.1801	11.4080
0.00008	21.5475	24.1786	13.2746	11.5152
0.00009	21.5092	24.3118	13.3492	11.6039
0.0001	21.4430	24.4163	13.4087	11.6779
0.0002	20.1207	**24.5233**	13.6321	12.0245
0.0003	18.7483	23.9878	**13.6379**	12.1179
0.0004	17.6938	23.3696	13.5928	**12.1418**
0.0005	16.9139	22.7933	13.5315	12.1385
0.0006	16.3328	22.2798	13.4649	12.1223
0.0007	15.8918	21.8254	13.3973	12.0995
0.0008	15.5504	21.4218	13.3306	12.0731
0.0009	15.2809	21.0610	13.2653	12.0448
0.001	15.0641	20.7359	13.2020	12.0154
0.002	14.0752	18.6140	12.6670	11.7238
0.003	13.6804	17.4289	12.2573	11.4697
0.004	13.4173	16.6229	11.9227	11.2489
0.005	13.2100	16.0180	11.6381	11.0531
0.006	13.0346	15.5367	11.3894	10.8766
0.007	12.8807	15.1386	11.1679	10.7154
0.008	12.7428	14.8003	10.9680	10.5666
0.009	12.6178	14.5070	10.7856	10.4283
0.01	12.5034	14.2487	10.6178	10.2987

A study of the de-noising ability of the product filter with Gaussian noise and

various t values is presented in Figures 4.20 to 4.26. Gaussian noise of the form $25\,\mathrm{randn}(m, n)$ was added to the Moon image. The product filter, with $p = 256$, was applied with five values of t (0.0001 to 0.0005). The SNR values are given in the Figure captions. It is seen that there is more smoothing as t increases, as expected. But, more smoothing, of course, results in blurring of the image details. With $t = 0.0005$ (Fig. 4.26), there is significant blurring, very noticable around the craters. The optimal value of t seems to be $t = 0.0002$ (see Fig. 4.23). With this value of t, there is some de-noising, but still significant detail preservation.

The times required to do standard calculations with the product filter have been measured. The 256×256 Barbara image and the 512×512 Full Barbara image had $50\,\mathrm{rand}(m, n)$ noise added to them and were processed with $t = 0.0003$ and $p = 256$. The times required for the various steps in the algorithm's calculations were measured using MATLAB's `tic` and `toc` commands. The command `tic` starts a stopwatch timer. The step in the calculation is performed and the `toc` command reports the time elapsed since the `tic`. The times are reported in Table 4.4. The times required to apply MATLAB's built-in filters, `averaging` and `median`, to these sizes of images are also given. The total times for the product filter calculations are approximately 20.2 s and 78.7 s, respectively, on a Sun Ultra 5, running at 360 MHz, with 256 MB RAM. It can be seen from the times reported in the Table that the bulk of the calculation time required is to generate the filter matrix. So, if the appropriate size filter matrix already exists, the calculations become incredibly quick - approximately 5 s and 9 s, respectively (which includes display time). So, real-time implementation could be feasible in some applications.

The directionality of the product filter algorithm was also tested. A test image of size 256×256 with sixty-four entries on the main diagonal having intensity 100, all other entries zero, was generated. In the MATLAB coordinate system, this line makes

Table 4.4: Times required for steps in the product filter algorithm applied to the Barbara and Full Barbara images with $50\,\mathrm{rand}(m, n)$ noise. $t = 0.0003$ and $p = 256$. The times are quoted in seconds. The times quoted for the MATLAB filters are the times required to apply the filters to the images.

calculation step	Barbara (256×256)	Full Barbara (512×512)
load image	0.0264	0.5004
display image	1.2044	1.3370
add noise to image	0.0299	0.1729
display noisy image	1.6728	2.6414
take FFT of noisy image	0.1522	0.6605
generate filter matrix	15.3350	69.1373
apply filter	0.0783	0.4719
take IFFT	0.1800	0.7051
display processed image	1.4559	2.7622
calculate and display SNR	0.0772	0.2980
MATLAB's `averaging` filter	0.2799	0.4047
MATLAB's `median` filter	0.5512	0.6180

an angle of $\pi/4$ with the downward pointing x_1-axis. The Fourier Transform of this line is a line perpendicular to it, making an angle of $3\pi/4$ (see Figure 4.27). The product filter was applied (with $t = 0.0003$) in a uni-directional mode (*i.e.* $p = 1$) at these two angles. The filter is applied at these angles in Fourier Space. At an angle of $\pi/4$, there is not much diffusion as the diffusion direction runs perpendicularly to the line in Fourier Space, or parallel to the line in x-Space. The maximum value of the processed image is 99.5871. At an angle of $3\pi/4$, the direction of smoothing is parallel to the direction of the line in Fourier Space (perpendicular in x-Space) and there is more smoothing, as the maximum value of the processed image is 37.4769. And so, the direction of smoothing can be controlled.

In the case of the Saturn image (of size 328×438), most of the structure of the image is approximately parallel to the diagonal running from the bottom left to the top right corner, making an angle of $\phi = \arctan(328/438) = 0.6428$ radians relative

to the bottom of the image. In the MATLAB coordinate system, this angle would correspond to $\theta = \phi + \pi/2 = 2.2136$ radians relative to the downward pointing x_1-axis. The Saturn image, with noise of the form $50\,\mathrm{rand}(m,n) - 25\,\mathrm{rand}(m,n)$ and $t = 0.0003$, was tested with two uni-directional filters, one at angle θ, another at angle $\theta + \pi/2$ (see Figure 4.28). It can be seen that the first filter (applied at angle θ) gives a better result, both visually and with regards to SNR values as it diffuses less, being perpendicular to the dominant features (in Fourier Space). The uni-directional PDE algorithm is extremely quick, taking approximately 5 s. So, in cases where the image shows a predominant angle (like the Saturn image), a uni-directional PDE, while certainly faster, may reduce noise well enough for certain applications.

Numerically, the optimal range for the t parameter has been found to be 0.0001 to 0.0005. The visual investigation agrees and confirms that this range is optimal. The value $t = 0.0003$ is then used as the standard for calculations. The standard for the number of directions was chosen to be $p = 256$ in order to include many directions, but still have an efficient algorithm. (It was seen that there is no real gain in increasing p from 256 to 512, for example.) The algorithm has been seen to be relatively quick and does some noise reduction while managing to preserve image details reasonably well. The direction of diffusion can also be controlled in the algorithm. More investigation into the de-noising ability is presented in Chapter 5.

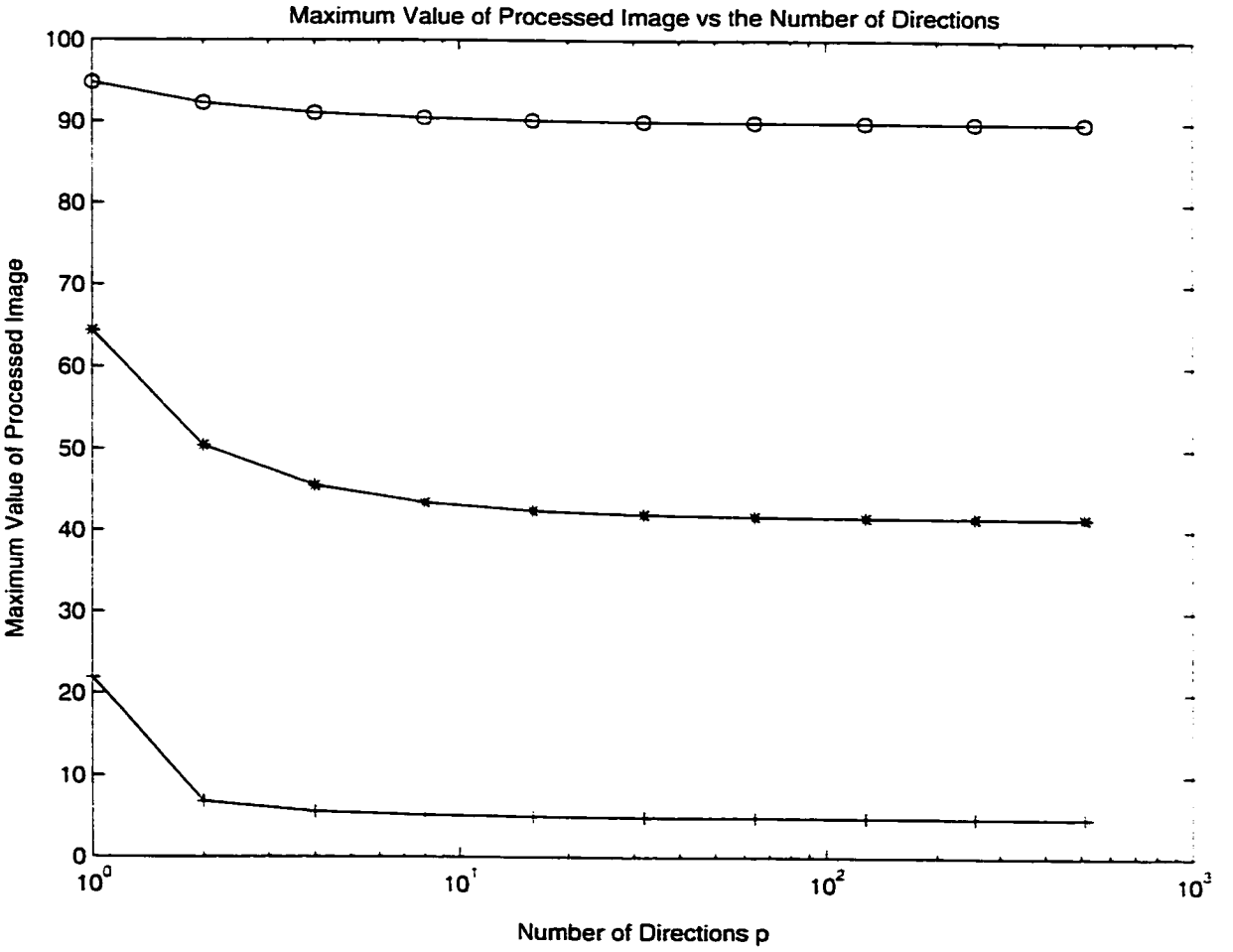

Figure 4.1: A graph of the maximum intensity value of the single peak image processed with the product filter versus the number of directions used. The values of t were: 0.00001 (o), 0.0001 (*) and 0.001 (+).

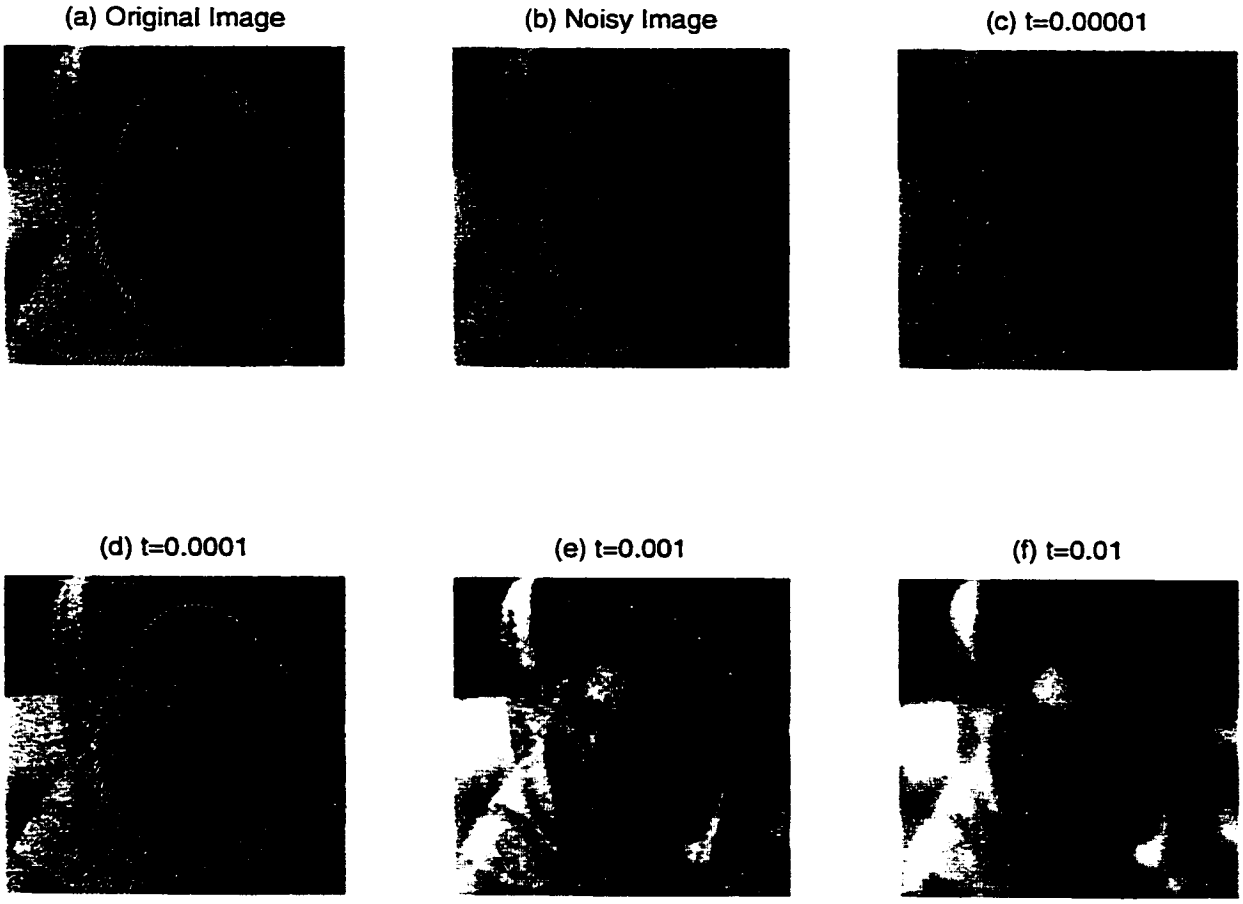

Figure 4.2: The Barbara image with random noise. $50\,\mathrm{rand}(m, n) - 25\,\mathrm{rand}(m, n)$. for varying values of t and $p = 256$. (a) is the original (pre-noise) image. (b) is the image with the noise added. (c) to (f) are the processed images with the varying values of t shown above each sub-image.

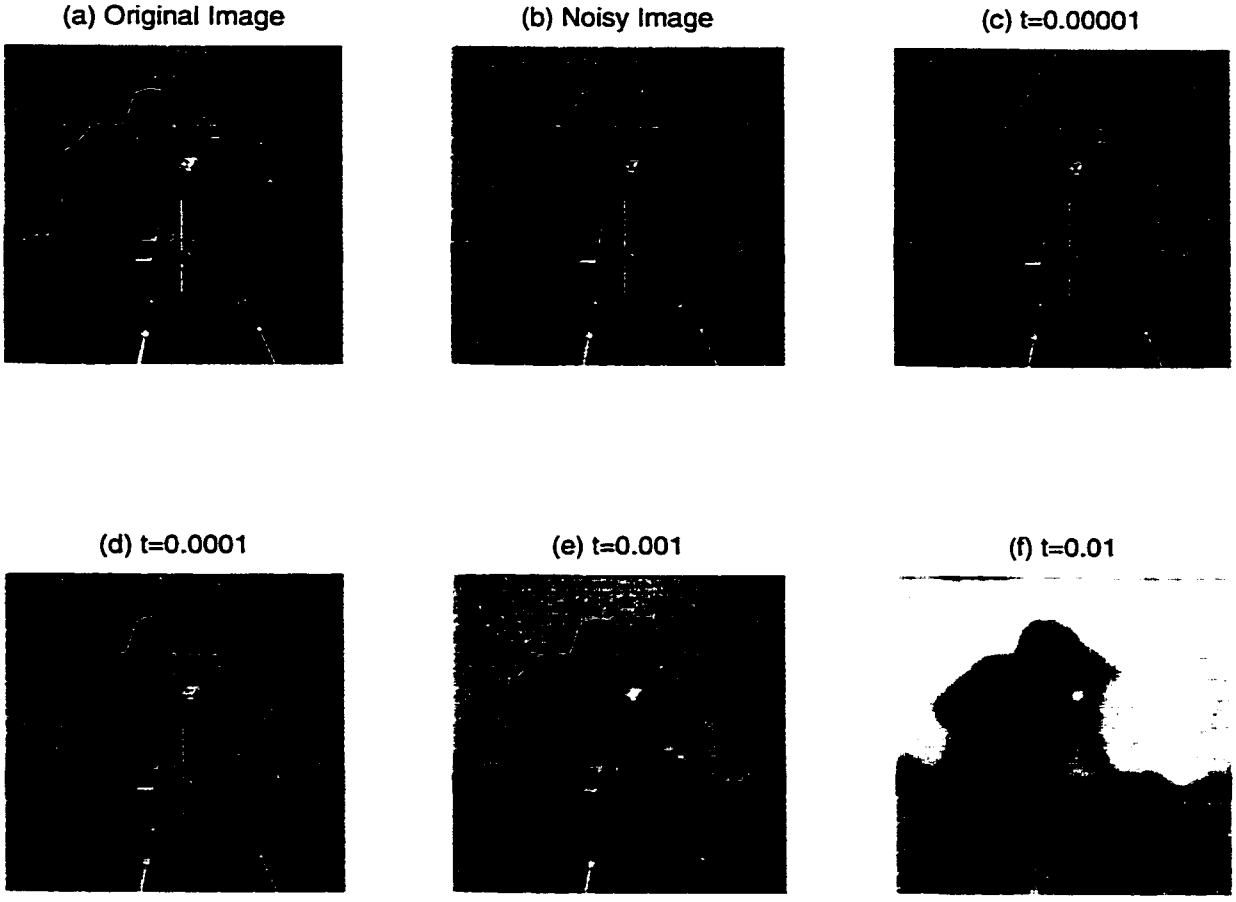

Figure 4.3: The Cameraman image with random noise. $50\,\mathrm{rand}(m,n) - 25\,\mathrm{rand}(m,n)$. for varying values of t and $p = 256$. (a) is the original (pre-noise) image. (b) is the image with the noise added. (c) to (f) are the processed images with the varying values of t shown above each sub-image.

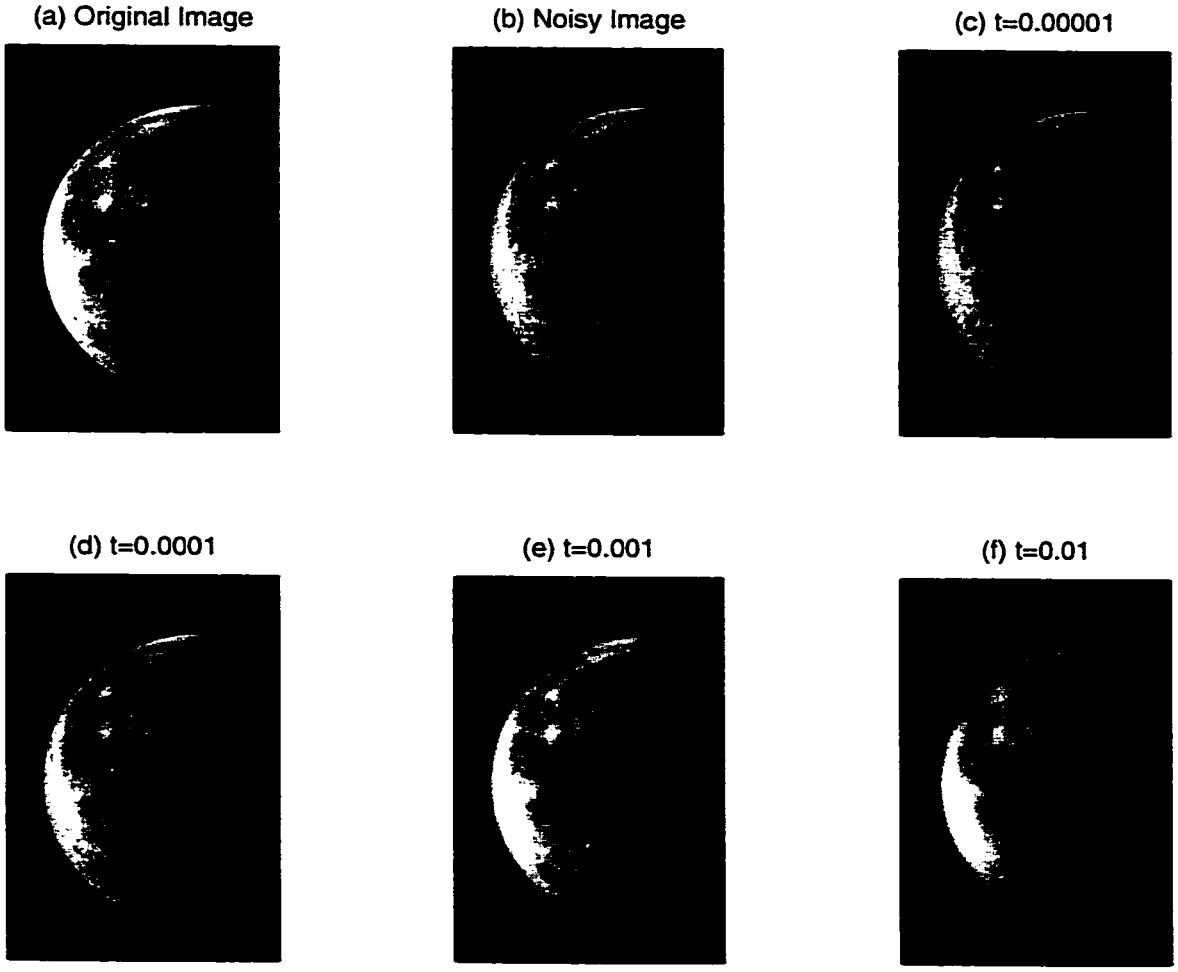

Figure 4.4: The Moon image with random noise. $50\,\mathrm{rand}(m, n) - 25\,\mathrm{rand}(m, n)$. for varying values of t and $p = 256$. (a) is the original (pre-noise) image. (b) is the image with the noise added. (c) to (f) are the processed images with the varying values of t shown above each sub-image.

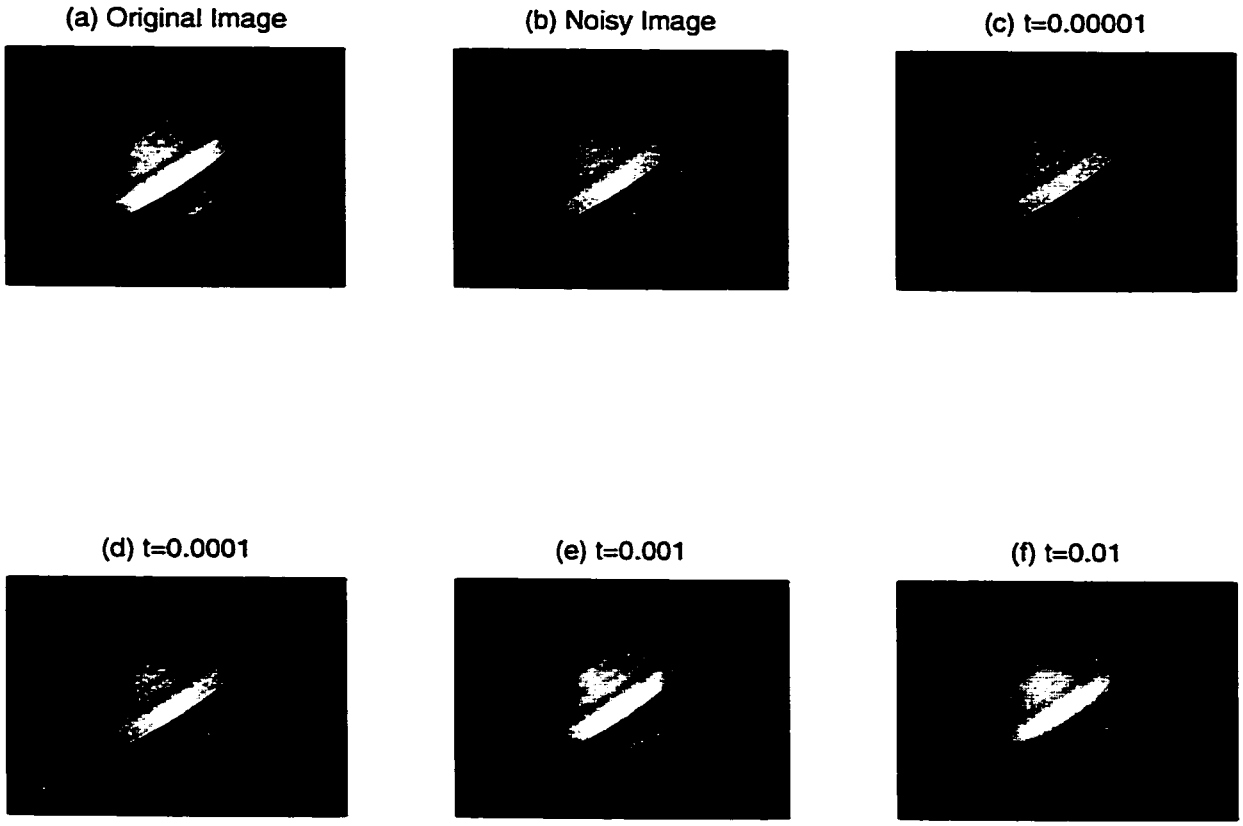

Figure 4.5: The Saturn image with random noise, $50\,\mathrm{rand}(m,n) - 25\,\mathrm{rand}(m,n)$, for varying values of t and $p = 256$. (a) is the original (pre-noise) image. (b) is the image with the noise added. (c) to (f) are the processed images with the varying values of t shown above each sub-image.

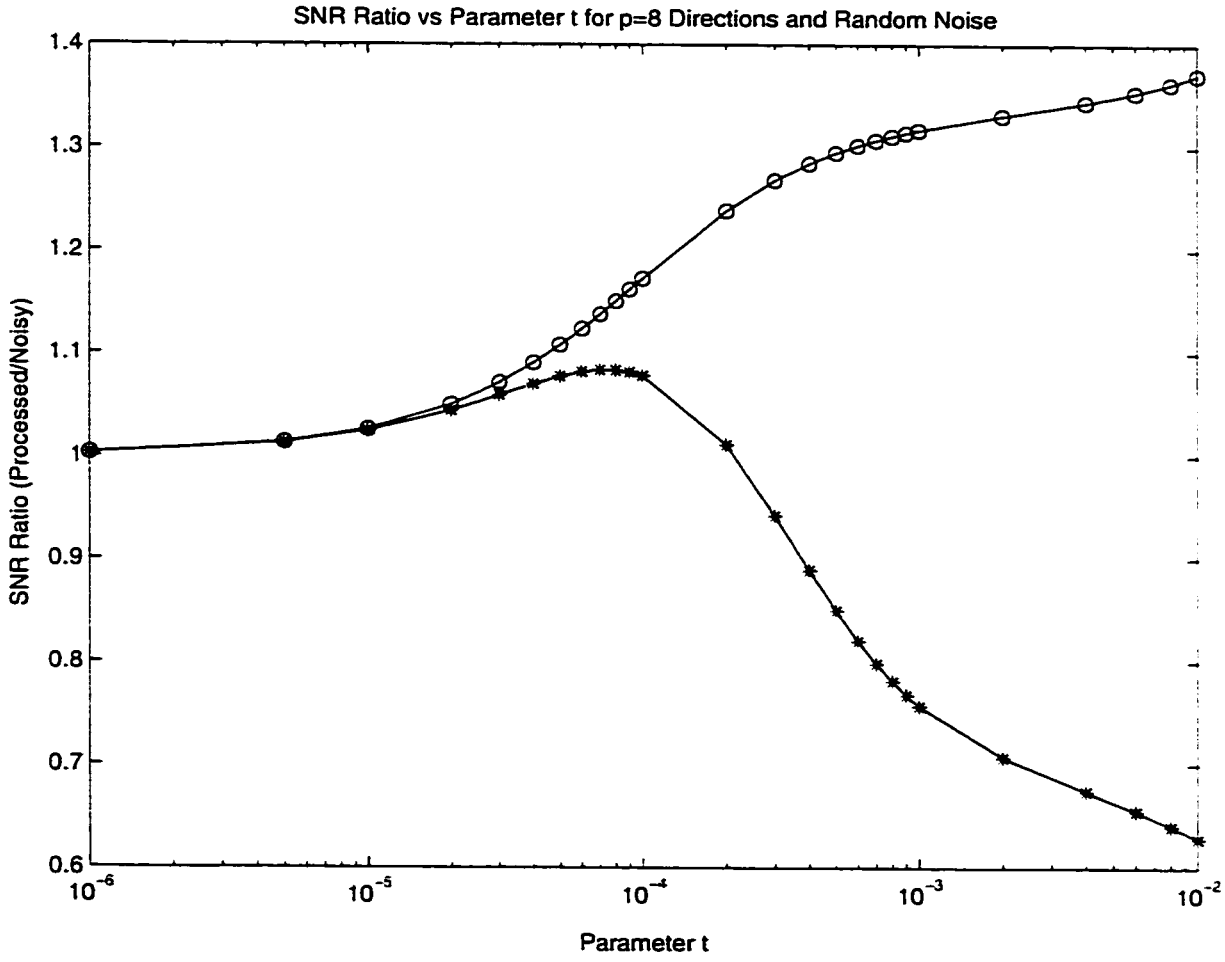

Figure 4.6: A graph of the ratio of the SNR of the processed image to the SNR of the noisy image with $p = 8$ directions and random noise of the form $50\,\mathrm{rand}(m, n)$ for the constant image ($\circ$) and Barbara ($*$).

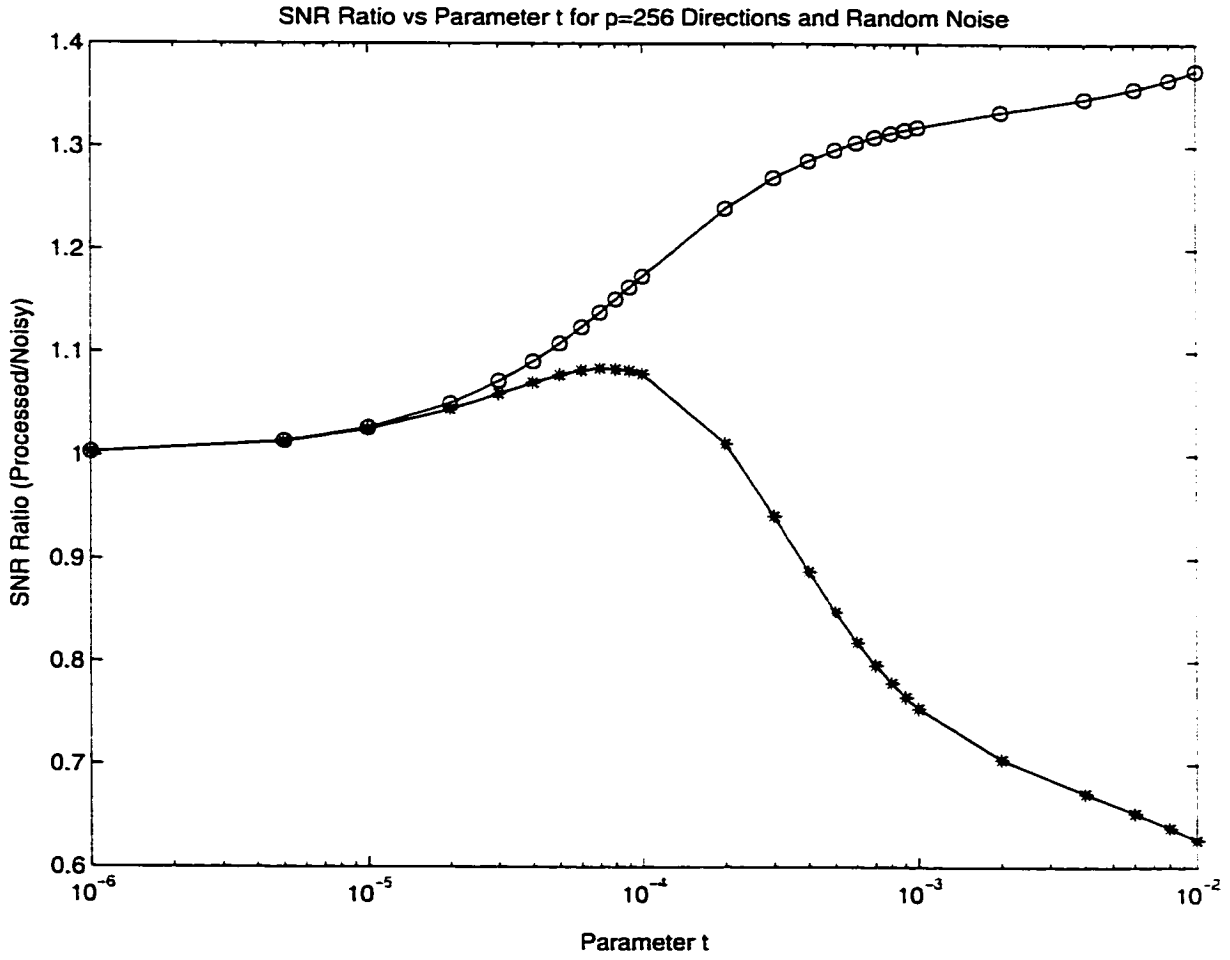

Figure 4.7: A graph of the ratio of the SNR of the processed image to the SNR of the noisy image with $p = 256$ directions and random noise of the form $50\,\mathrm{rand}(m, n)$ for the constant image ($\circ$) and Barbara ($*$).

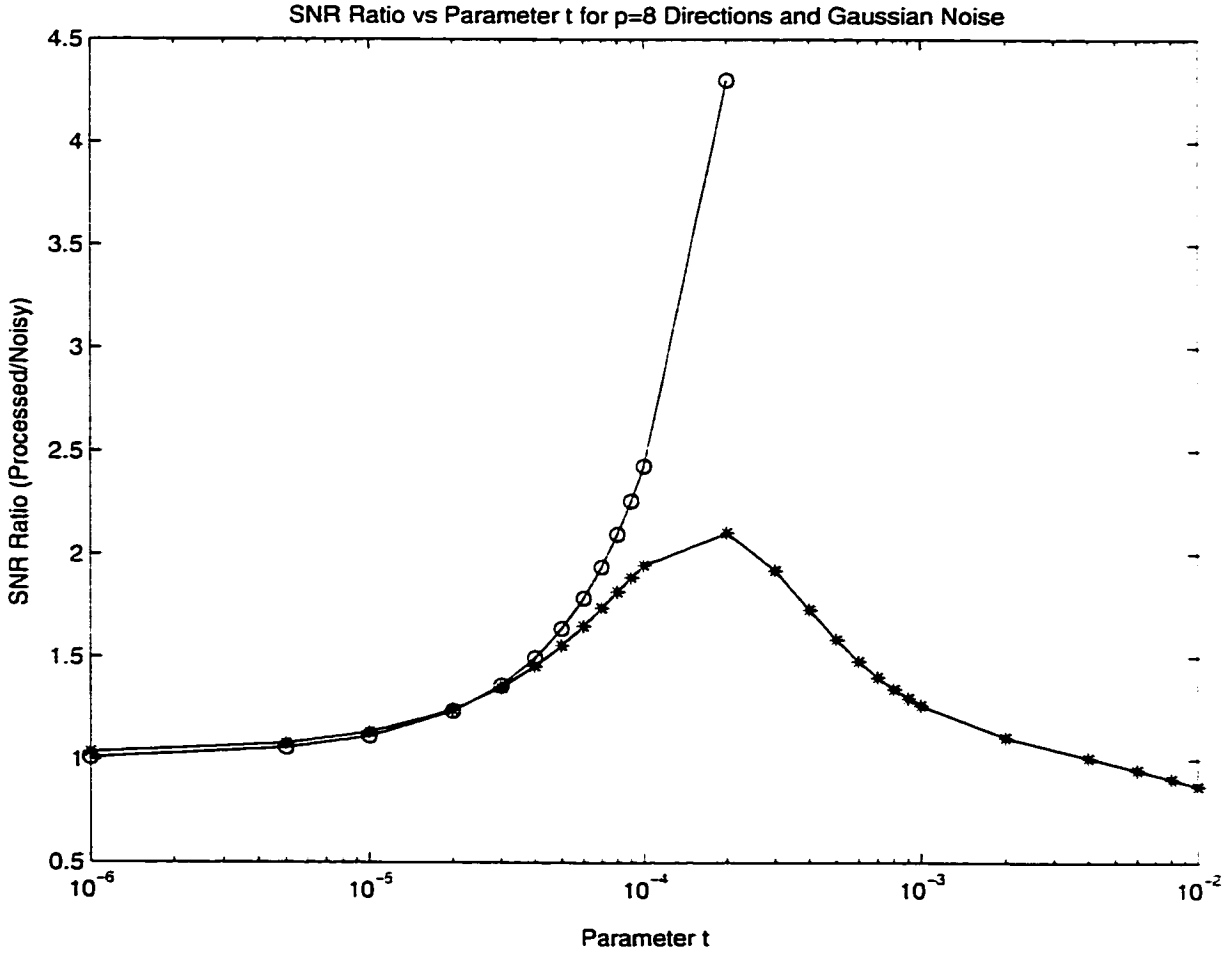

Figure 4.8: A graph of the ratio of the SNR of the processed image to the SNR of the noisy image with $p = 8$ directions and Gaussian noise of the form $25\,\mathrm{randn}(m, n)$ for the constant image (o) and Barbara (*).

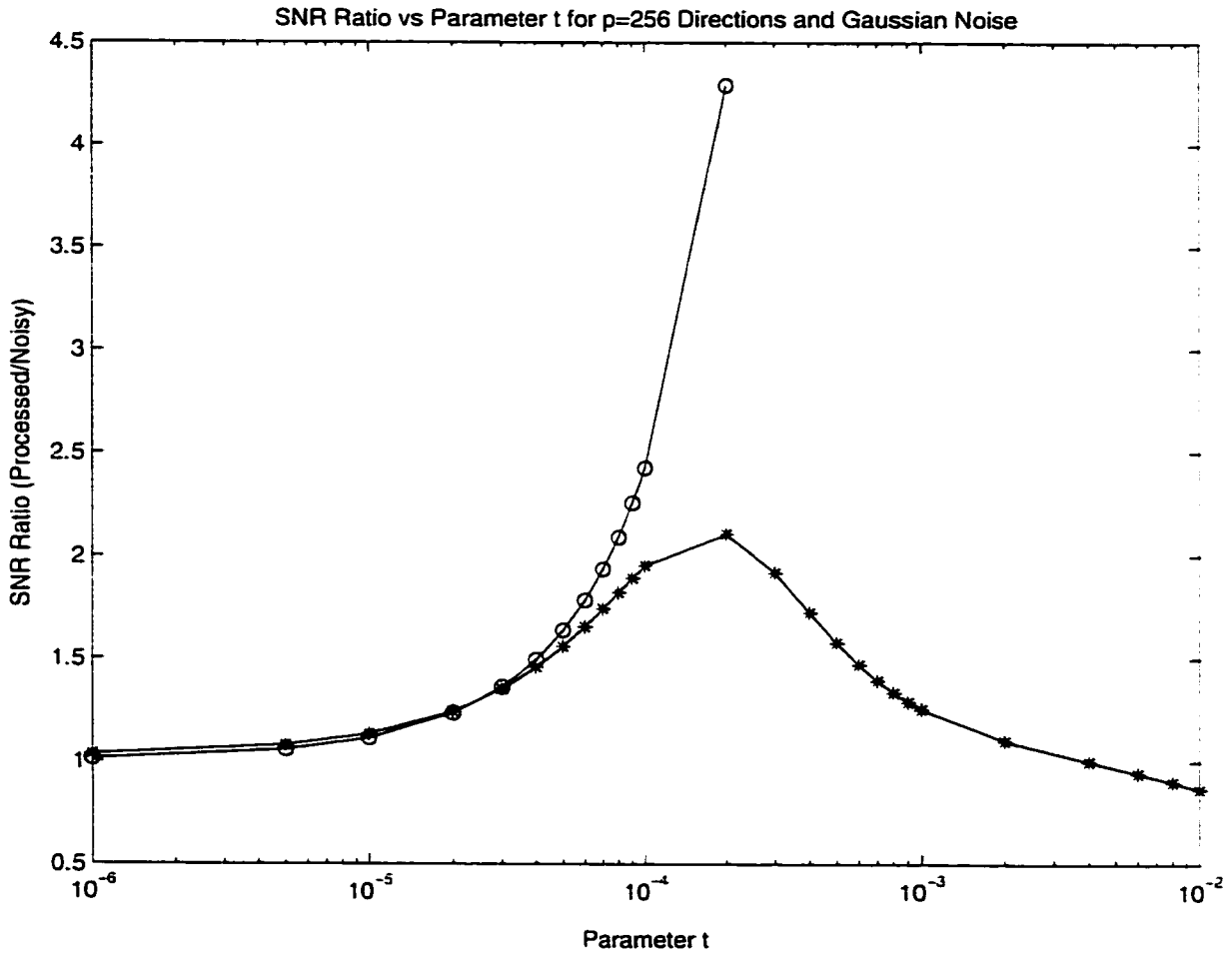

Figure 4.9: A graph of the ratio of the SNR of the processed image to the SNR of the noisy image with $p = 256$ directions and Gaussian noise of the form 25 randn(m, n) for the constant image (o) and Barbara ($*$).

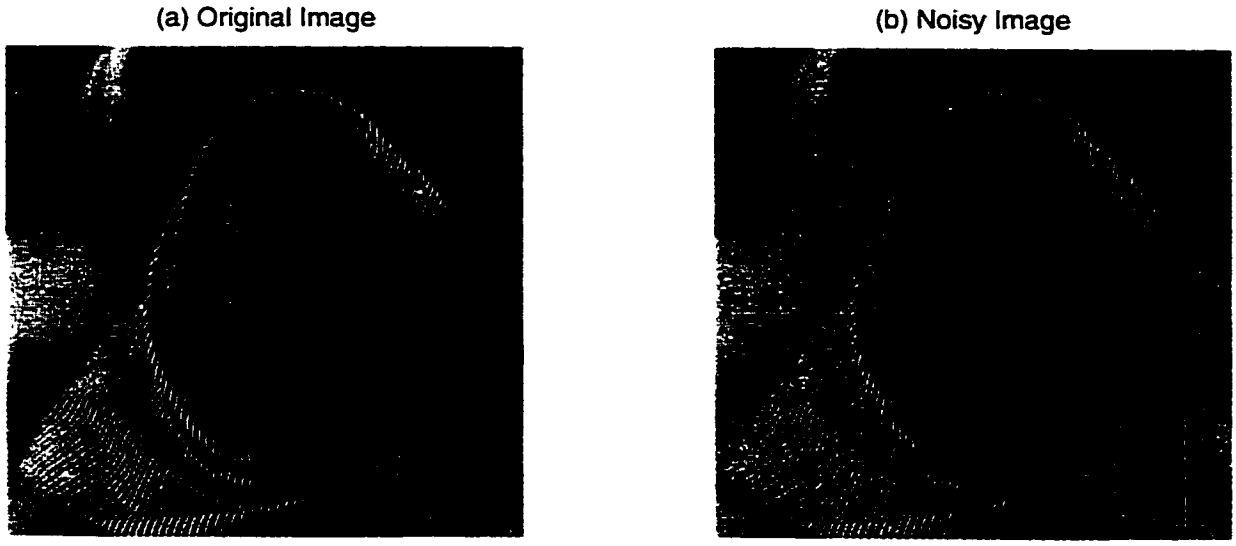

Figure 4.10: The Barbara image with random noise of the form $50\,\mathrm{rand}(m, n)$ used for the visual investigation of the t parameter. (a) is the original image. (b) is the noisy image (SNR value in Table 4.3).

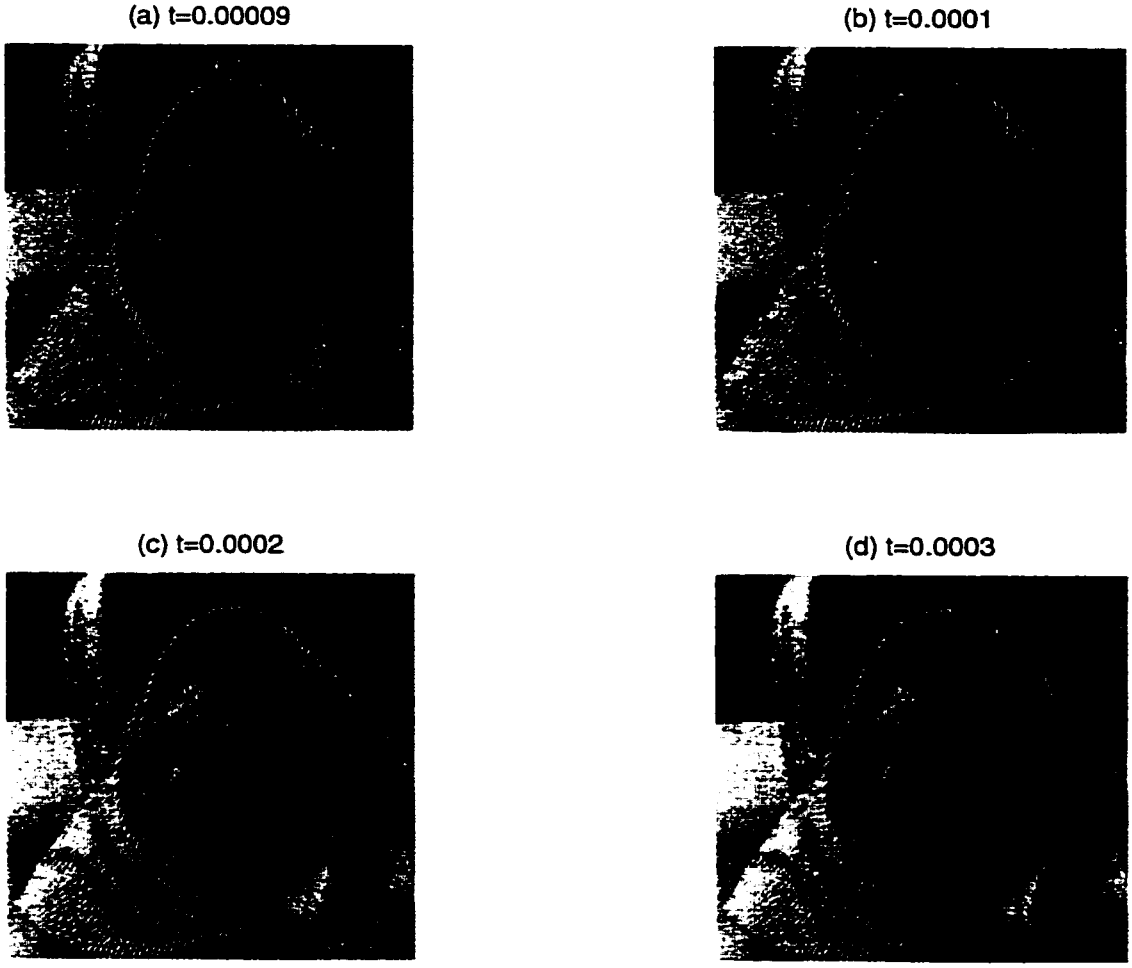

Figure 4.11: Processed Barbara images for the visual investigation of the t parameter. The corresponding t values are given above each sub-image. The original noisy image is shown in Figure 4.10. SNR values are in Table 4.3.

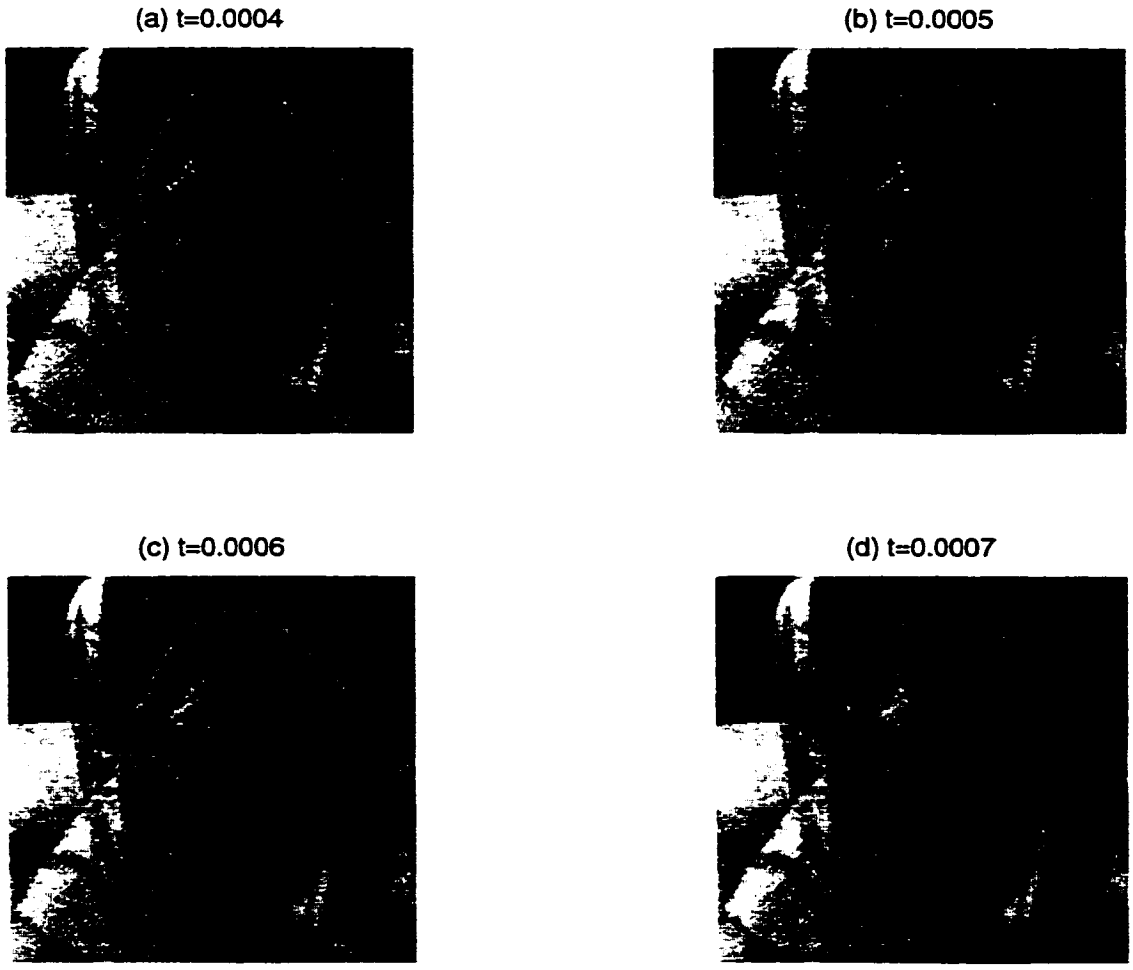

Figure 4.12: Processed Barbara images for the visual investigation of the t parameter. The corresponding t values are given above each sub-image. The original noisy image is shown in Figure 4.10. SNR values are in Table 4.3.

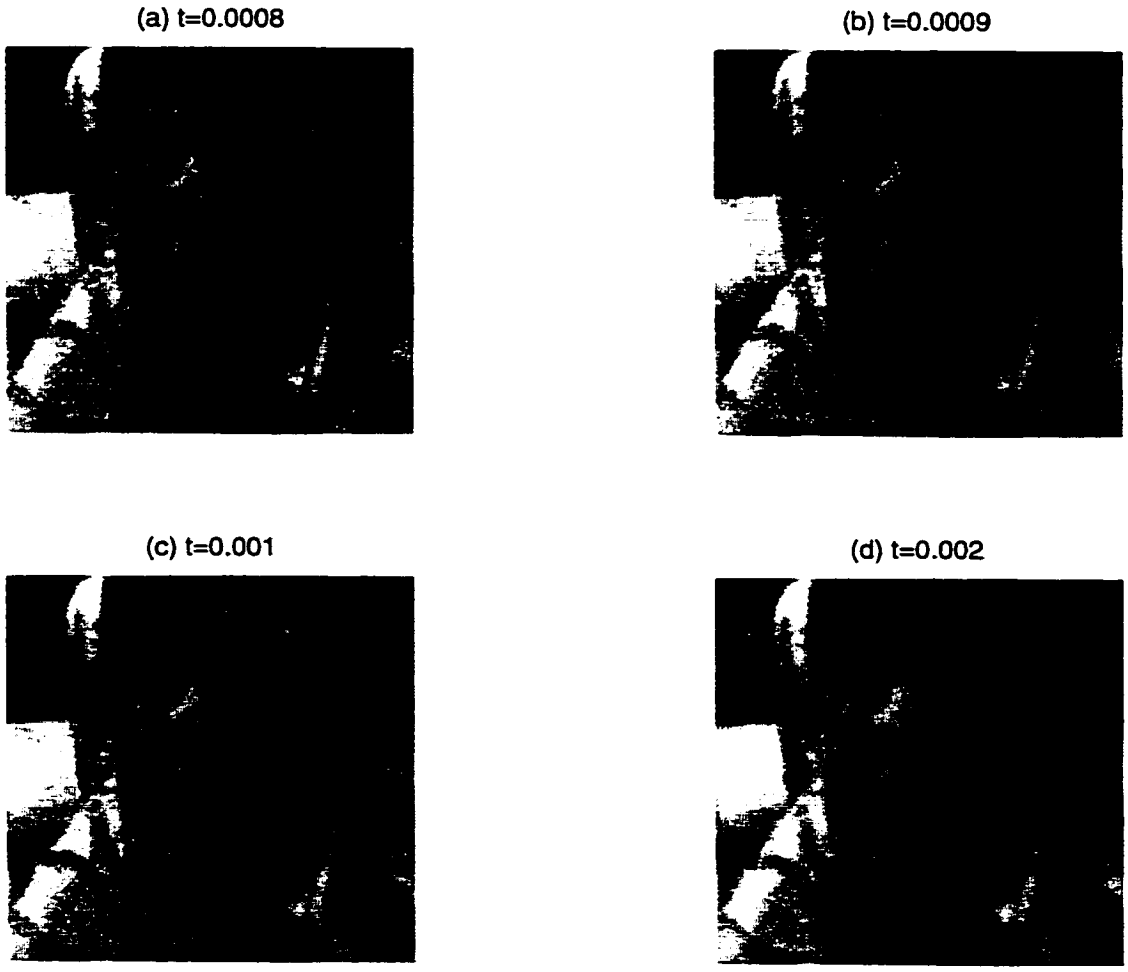

Figure 4.13: Processed Barbara images for the visual investigation of the t parameter. The corresponding t values are given above each sub-image. The original noisy image is shown in Figure 4.10. SNR values are in Table 4.3.

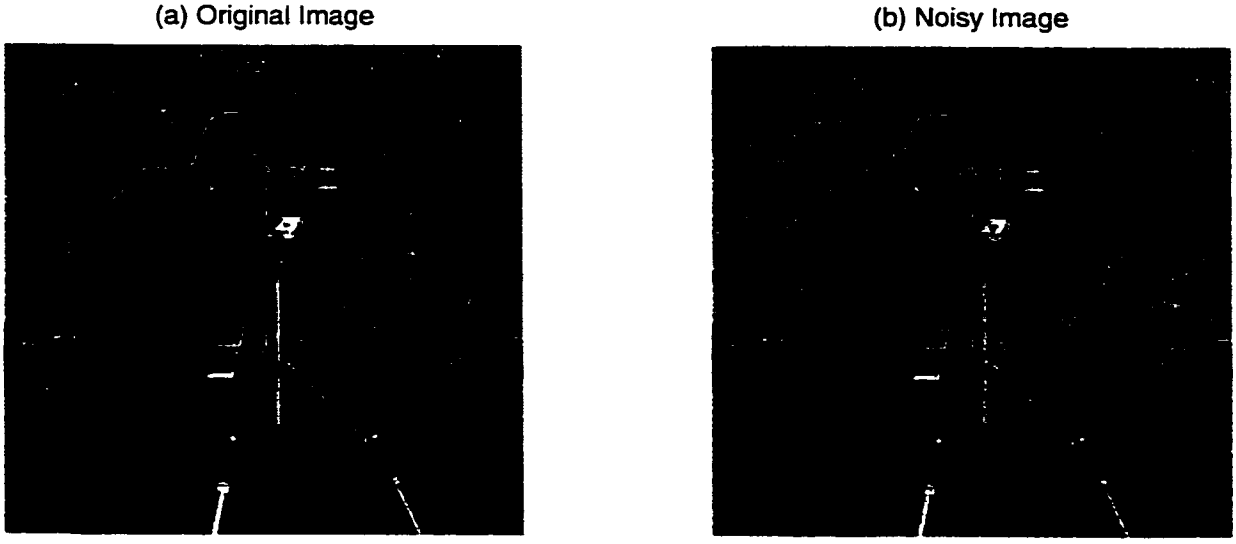

Figure 4.14: The Cameraman image with random noise of the form 50 rand(m, n) used for the visual investigation of the t parameter. (a) is the original image. (b) is the noisy image (SNR value in Table 4.3).

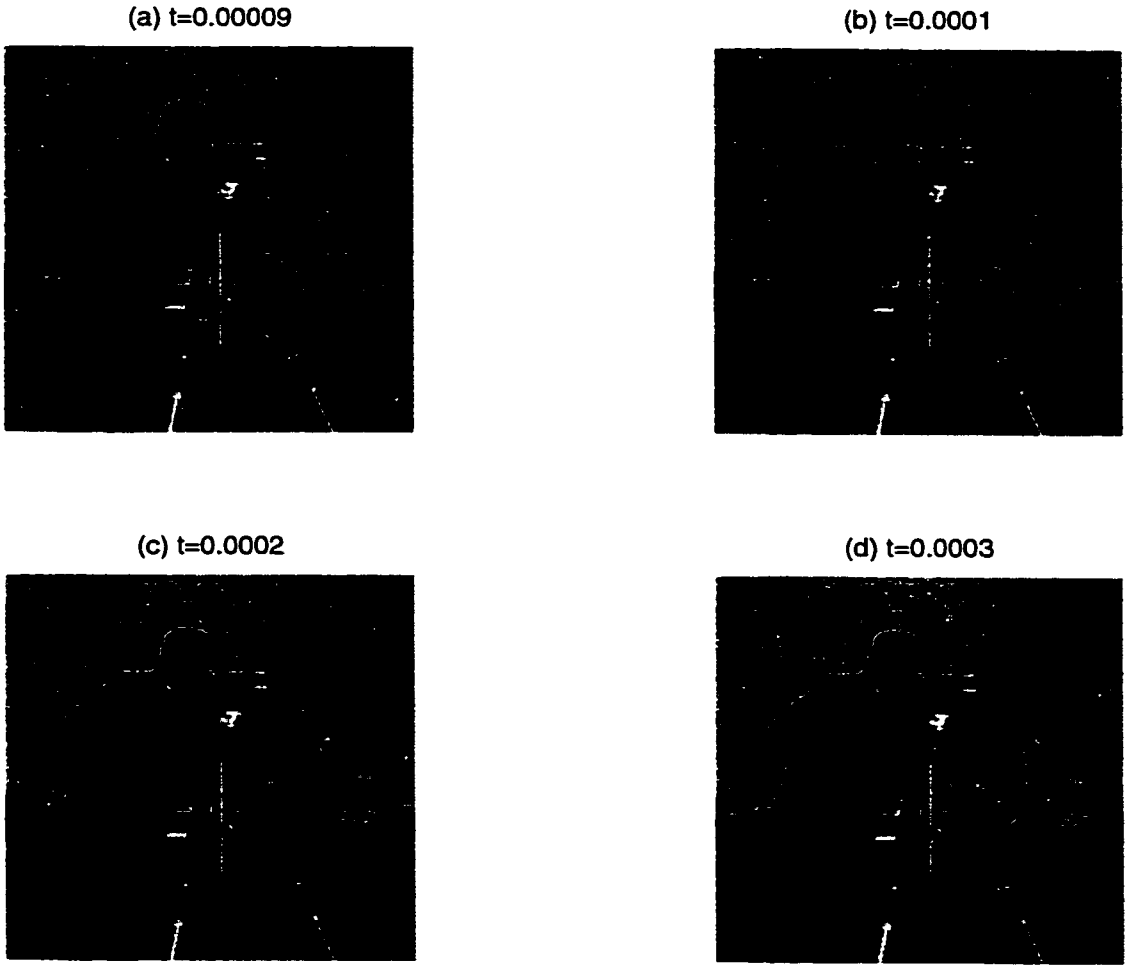

Figure 4.15: Processed Camerman images for the visual investigation of the t parameter. The corresponding t values are given above each sub-image. The original noisy image is shown in Figure 4.14. SNR values are in Table 4.3.

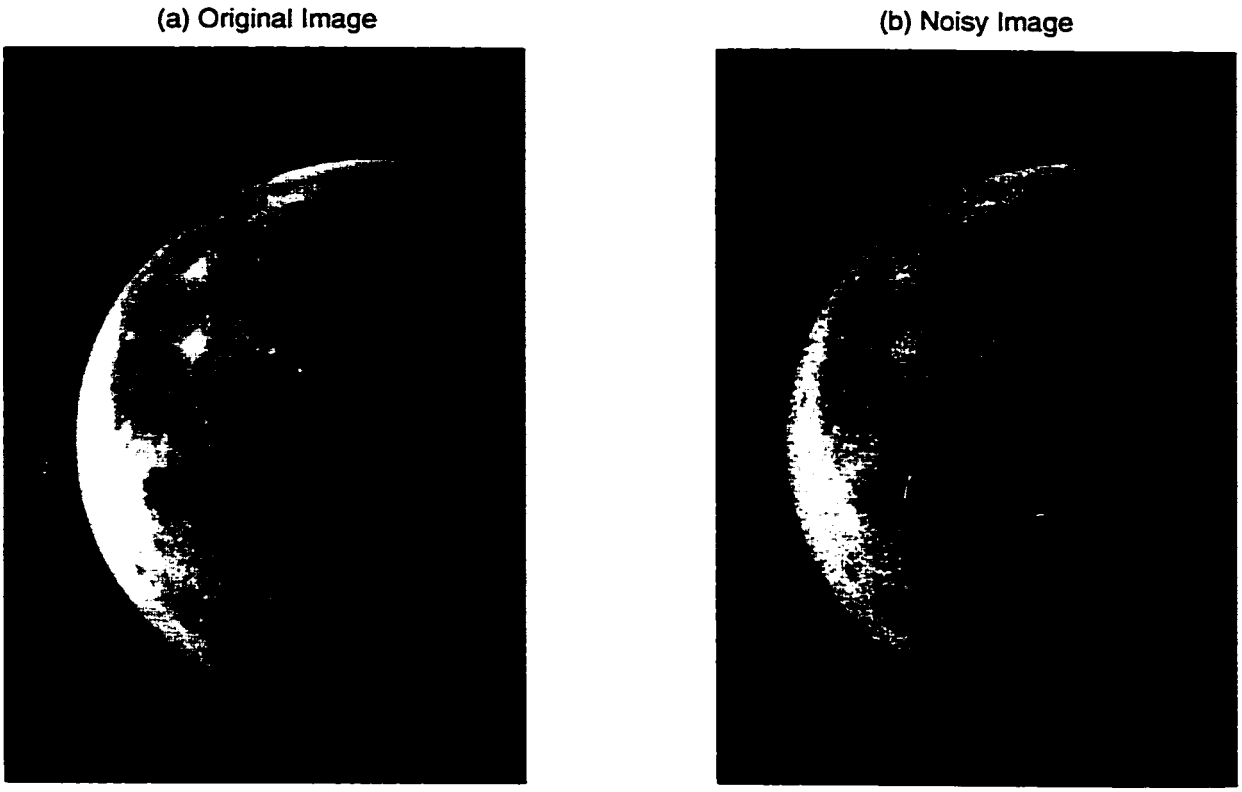

Figure 4.16: The Moon image with random noise of the form $50\,\mathrm{rand}(m, n)$ used for the visual investigation of the t parameter. (a) is the original image. (b) is the noisy image (SNR value in Table 4.3).

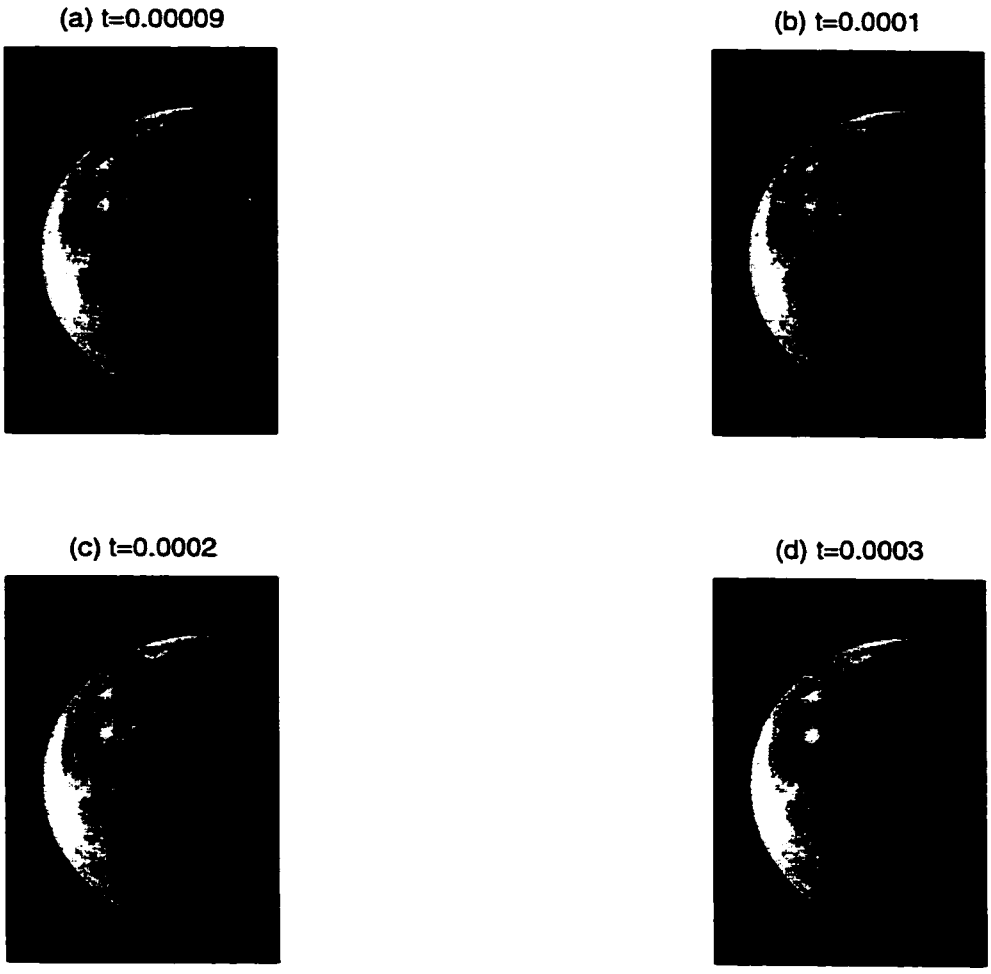

Figure 4.17: Processed Moon images for the visual investigation of the t parameter. The corresponding t values are given above each sub-image. The original noisy image is shown in Figure 4.16. SNR values are in Table 4.3.

Figure 4.18: The Saturn image with random noise of the form $50\,\mathrm{rand}(m, n)$ used for the visual investigation of the t parameter. (a) is the original image. (b) is the noisy image (SNR value in Table 4.3).

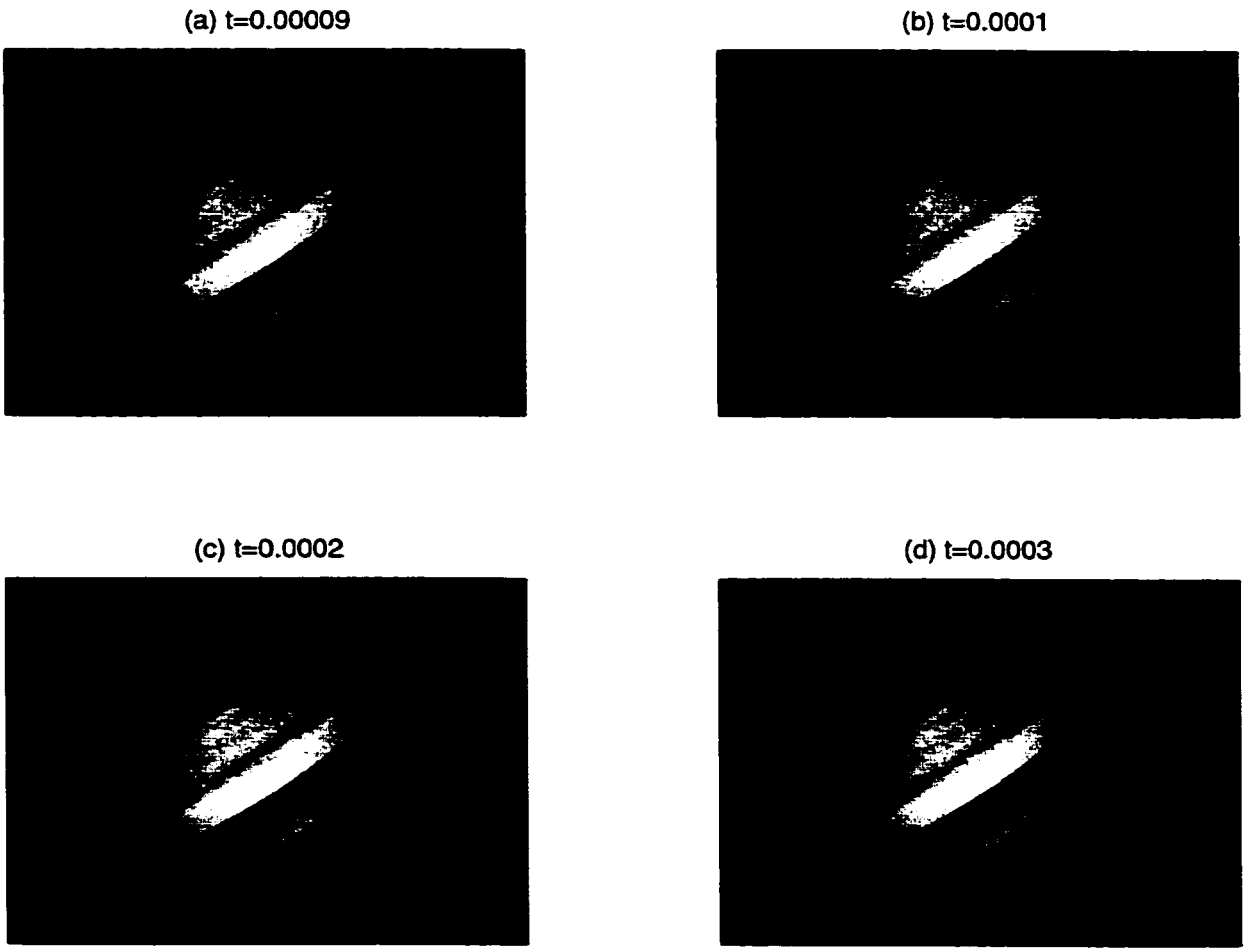

Figure 4.19: Processed Saturn images for the visual investigation of the t parameter. The corresponding t values are given above each sub-image. The original noisy image is shown in Figure 4.18. SNR values are in Table 4.3.

Figure 4.20: The original Moon image.

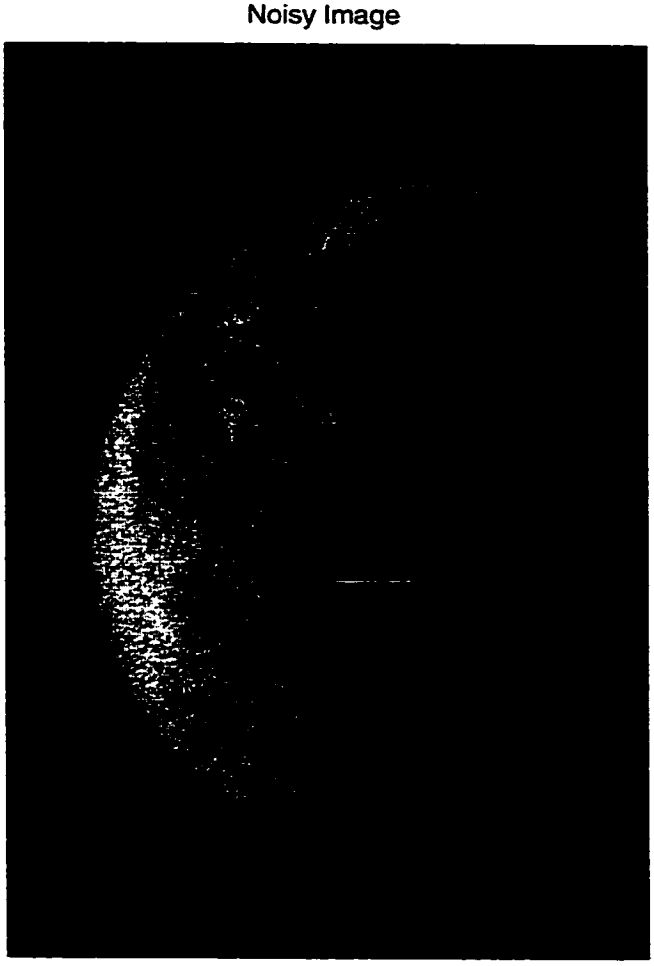

Figure 4.21: The Moon image with Gaussian noise of the form 25 randn(m, n). SNR = 14.1444. The original image is Figure 4.20.

Figure 4.22: Results of product filter applied to the noisy image from Figure 4.21 with $p = 256$ and $t = 0.0001$. SNR = 103.6984.

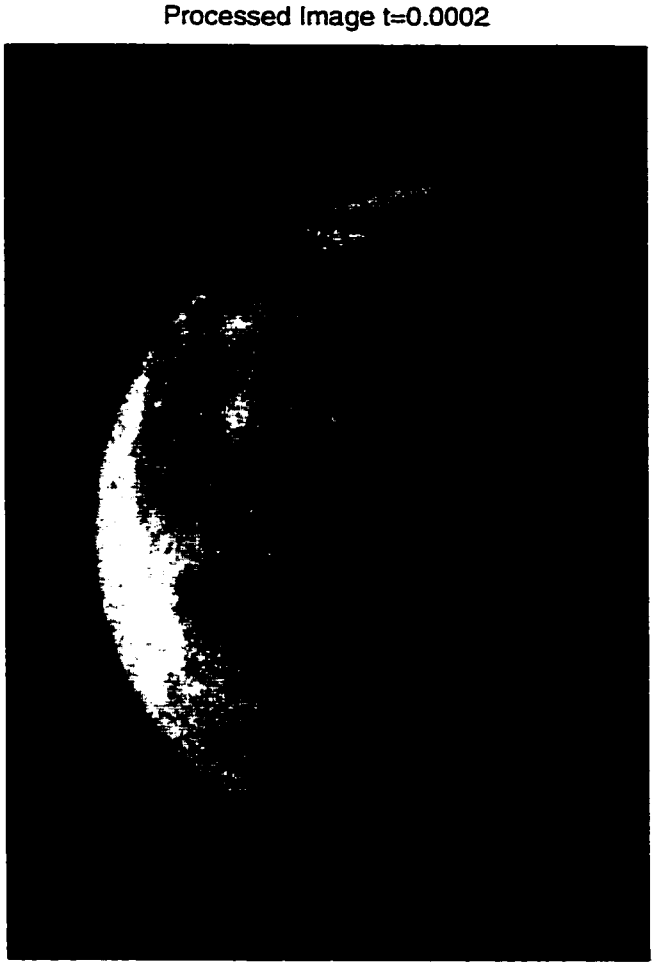

Figure 4.23: Results of product filter applied to the noisy image from Figure 4.21 with $p = 256$ and $t = 0.0002$. SNR = 176.6331.

Figure 4.24: Results of product filter applied to the noisy image from Figure 4.21 with $p = 256$ and $t = 0.0003$. SNR $= 209.7790$.

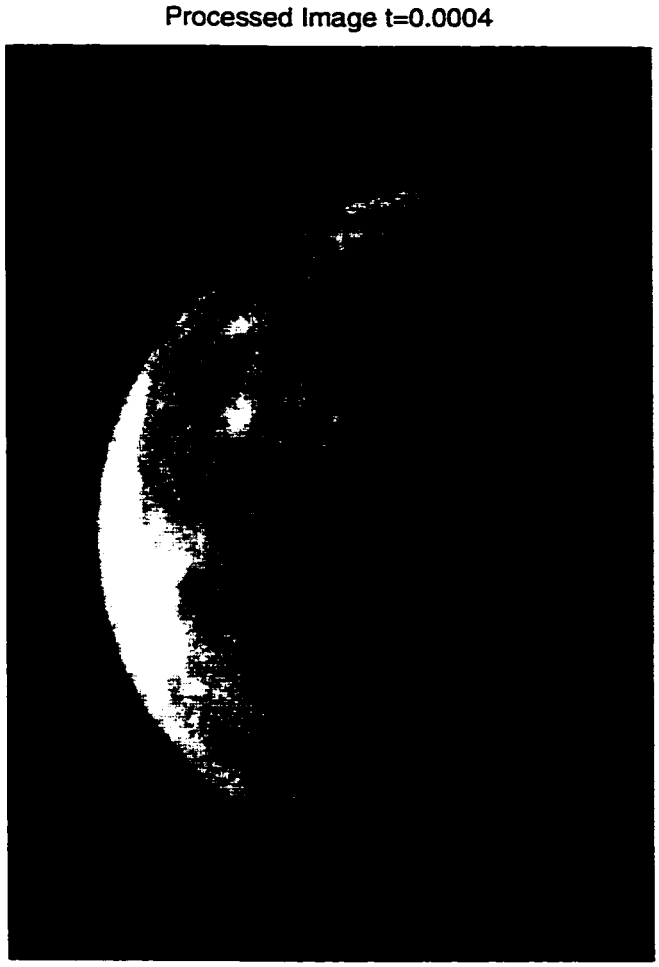

Figure 4.25: Results of product filter applied to the noisy image from Figure 4.21 with $p = 256$ and $t = 0.0004$. SNR = 217.3510.

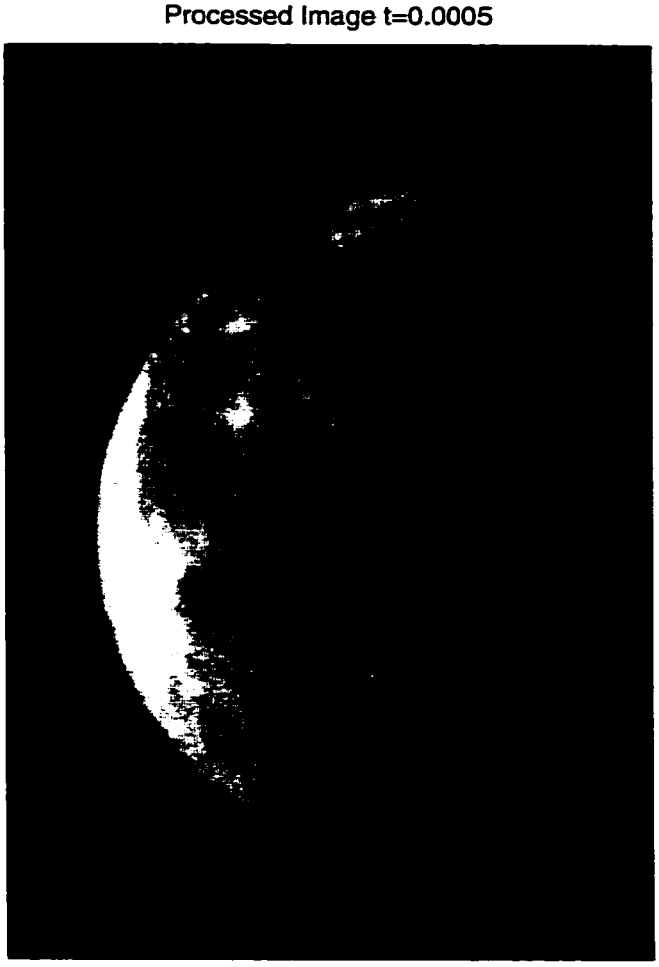

Figure 4.26: Results of product filter applied to the noisy image from Figure 4.21 with $p = 256$ and $t = 0.0005$. SNR = 212.9275.

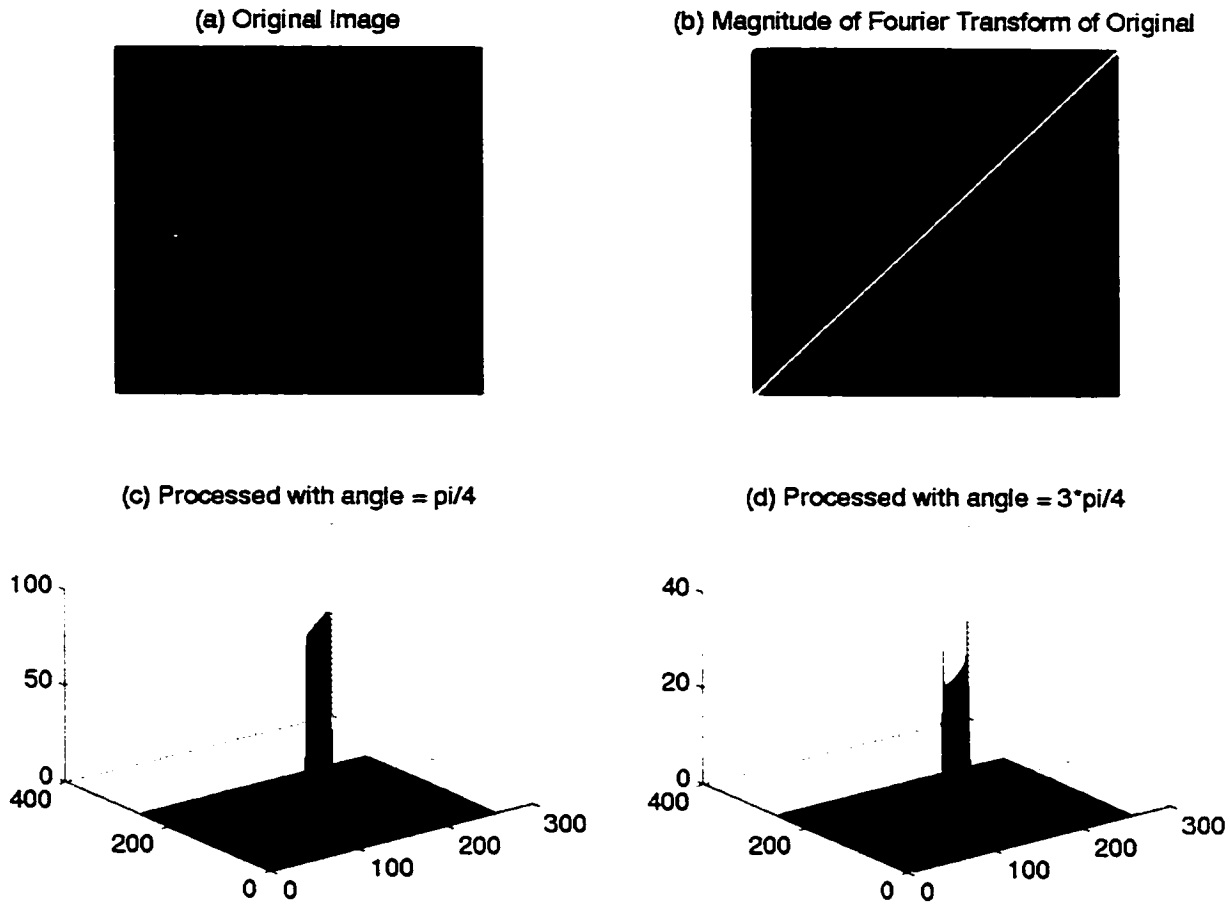

Figure 4.27: Two uni-directional filters applied to a diagonal line image. (a) is the original diagonal line image (intensity 100). (b) is the magnitude of the Fourier Transform of the original image. (c) is the result when angle $\pi/4$ is used (maximum value is 99.5871). (d) is the result when angle $3\pi/4$ is used (maximum value is 37.4769).

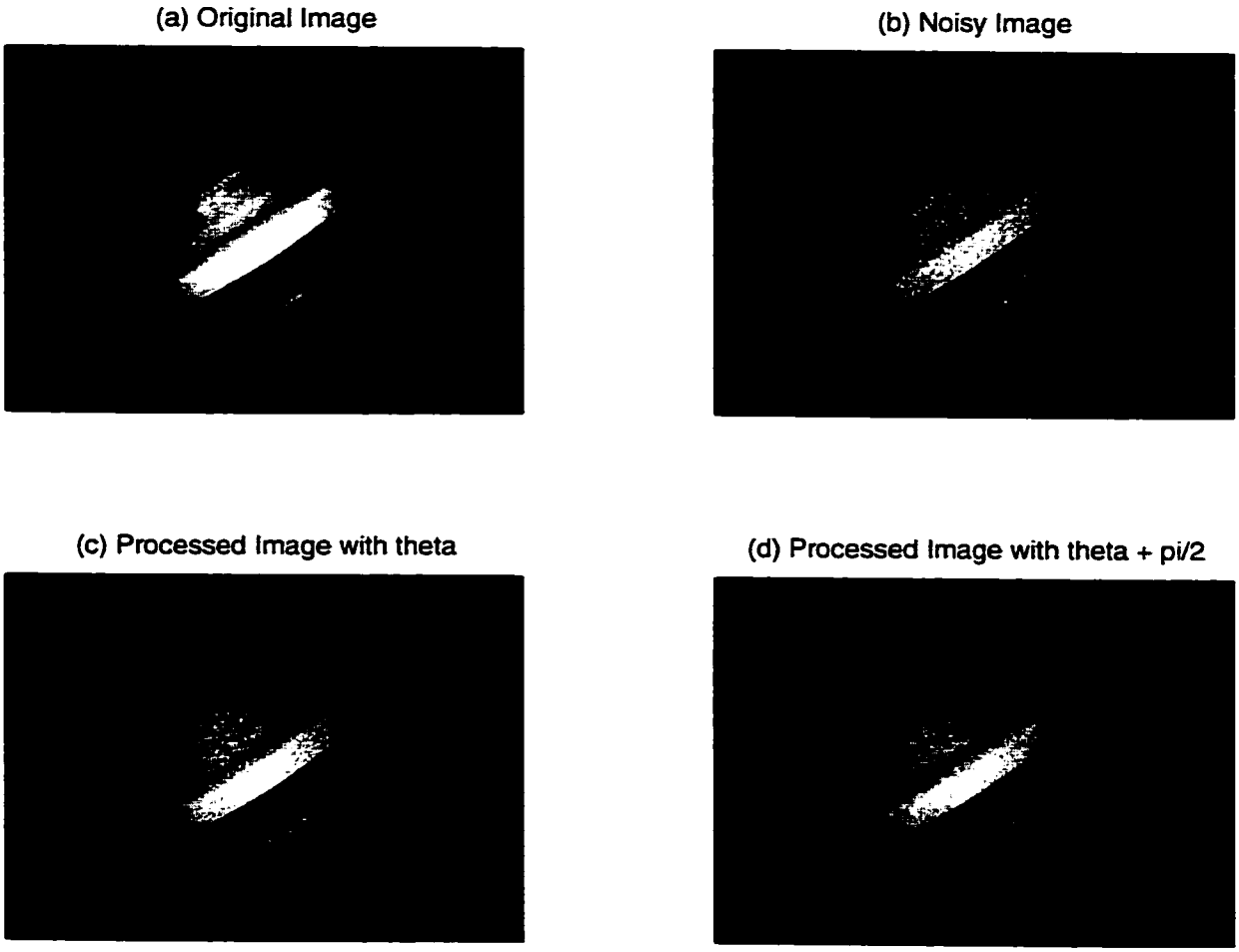

Figure 4.28: The Saturn image with two uni-directional filters. The noise is of the form $50\,\mathrm{rand}(m,n) - 25\,\mathrm{rand}(m,n)$ and $t = 0.0003$. The angle $\theta = 2.2136$ radians. (a) is the original image. (b) is the noisy image. SNR = 18.9807. (c) shows the results of the filter at angle θ. SNR = 37.2455. (d) shows the results of the filter at angle $\theta + \pi/2$. SNR = 33.9752.

Chapter 5

Comparing the Product Filter

The performance of the product filter algorithm in de-noising was evaluated and tested against other techniques, including MATLAB's built-in filters (`averaging` and `median`) and two wavelet-based methods. The product filter is also tested with respect to its ability to restore a quantized image.

Figures 5.1 to 5.14 present results of the new product filter algorithm, compared with the results of MATLAB's `averaging` filter. The `averaging` filter is invoked using the `fspecial('average')` command. It replaces a pixel's intensity value with the average of the intensities over a 3×3 neighbourhood centred on the pixel. This results in a smoothing of the image. In all Figures, $t = 0.0003$ and $p = 256$ and the noise was random (but of varying intensities). Signal-to-noise ratios are given in the captions.

Figures 5.1 to 5.12 present the results with the four images, Barbara (Figs. 5.1 to 5.3), the Cameraman (Figs. 5.4 to 5.6), the Moon (Figs. 5.7 to 5.9) and Saturn (Figs. 5.10 to 5.12). For Figures 5.1, 5.4, 5.7 and 5.10, the noise was of the form $50\,\mathrm{rand}(m, n)$. For Figures 5.2, 5.5, 5.8 and 5.11, the noise was more intense, $100\,\mathrm{rand}(m, n)$. A noise of the form $100\,\mathrm{rand}(m, n) - 50\,\mathrm{rand}(m, n)$ was used for Figures 5.3, 5.6, 5.9 and 5.12. In all cases, it can be seen that the new algorithm does a comparable job of reducing the noise when compared with MATLAB. However, it

seems to do a better job at preserving details and the SNR values are typically higher.

Figures 5.13 and 5.14 present a different example. The image is part of the Barbara image with noise (of unknown type and intensity) already added (this image is used in examples in MATLAB). In Figure 5.13, both the new product filter algorithm and MATLAB's `averaging` filter are applied. MATLAB seems to do a superior job on the noise reduction, however, it also smoothes the image a great deal. The product filter algorithm, while not removing as much noise, does a far superior job at preserving the details and has a much higher SNR (615 versus 30). In Figure 5.14, random noise of intensity 50 (*i.e.* $50\,\mathrm{rand}(m, n)$) was added to the already noisy image. The same result is seen - MATLAB may do a better job with the noise, but the new algorithm preserves more details and has a higher SNR.

Figures 5.15 to 5.20 present results of the new algorithm applied to images with salt and pepper noise, with $t = 0.0003$ and $p = 256$. In salt and pepper noise, random pixels are set to black or white (the extremes of the intensity range). The user controls the percentage of the image to corrupt. Salt and pepper noise is invoked using the command `imnoise(X,'salt & pepper',r)`, where X is the image to corrupt and r is the percentage of pixels to effect. Figures 5.15 to 5.18 involve the Barbara image. Figures 5.19 and 5.20 are for the Cameraman image. Figure 5.15 shows the results of the new product filter algorithm and MATLAB's `median` filter applied to Barbara with 5% salt and pepper noise. The `median` filter is invoked using the command `medfilt2`. It works in a similar way to the `averaging` filter, but replaces the pixel's intensity with the median over the 3×3 neighbourhood. The median (the *middle* value) is less sensitive to extreme values than the mean, making it it more appropriate for use with salt and pepper noise, where the effect of the noise is to produce extreme values. Obviously, MATLAB's `median` filter does a far superior job at removing the noise. Though, it should be noted that the product filter algorithm does a better

job at preserving details. Figure 5.16 compares the new algorithm with MATLAB's **averaging** filter for the same 5% salt and pepper noise. Again, the product filter does a better job at preserving the details. Figure 5.17 compares the algorithm with MATLAB's **median** filtering for 10% salt and pepper noise. MATLAB's filter does a better job with the noise, but does smooth over some of the details. Figure 5.18 shows the comparison for 10% salt and pepper noise with **averaging** filtering. Again, the product filter algorithm does a better job with detail preservation. Figures 5.19 and 5.20 show the Cameraman with 5% salt and pepper noise compared to MATLAB's **median** filter (Fig. 5.19) and **averaging** filter (Fig. 5.20). So, for salt and pepper noise, MATLAB's **median** filter is clearly superior to the new product filter algorithm in noise reduction.

The application of the product filter to an artificial/geometric image (called zigzag) is shown in Figures 5.21 to 5.23. The lines in the image are of intensity 120 (approximately equal to the mean of the Barbara image, 118.8125) on a background of zeros. The image's size is 256×256. The product filter, with $t = 0.0003$ and $p = 256$, was applied under three different noise conditions. The results were compared with MATLAB's **averaging** filter. SNR values are given in the Figure captions. Figure 5.21 shows the results with noise of the form $50\,\mathrm{rand}(m, n)$. The noise is $50\,\mathrm{rand}(m, n) - 25\,\mathrm{rand}(m, n)$ in Figure 5.22 and $25\,\mathrm{randn}(m, n)$ in Figure 5.23. The results are all very similar - the product filter and MATLAB do a comparable job of noise reduction: the SNR values for the product filter algorithm are slightly higher and the product filter seems to do a slightly better job at preserving the lines (*i.e.* they look sharper).

Wavelets have proven useful in the reduction of noise in digital images [8, 19, 32, 33, 36, 38, 39]. Figures 5.24 to 5.27 show a comparison of the product filter algorithm with a wavelet technique of Jean-Marc Lina of the Centre de Recherches Mathématiques of the Université de Montréal. The Full Barbara image, of size 512×512 (see Fig. 5.24).

is used for this example. Noise of unknown intensity and distribution was added to produce the noisy image in Figure 5.25. The product filter was applied with $p = 256$ and $t = 0.0003$ (see Fig. 5.26). The wavelet technique of Lina (see Fig. 5.27) appears to do a better job with the noise reduction, has a much higher SNR and preserves some of the details well (Barbara's clothes, for example). The product algorithm appears to do a better job preserving some of the other details (Barbara's eyes and the books, for example).

The simplest way of combining the product filter with any wavelet technique would be to apply the product filter to the output of the wavelet technique, to see if any improvement can be made. Figure 5.28 shows the results of applying the product filter (with $p = 256$) to the output of Lina's wavelet technique (Fig. 5.27) with $t = 0.0001$. The result is that more detail is lost and nothing is gained. The product filter was also applied with $t = 0.0002$ (SNR = 48.8085) and $t = 0.0003$ (SNR = 46.4962). As t increases, the results look worse - there is more blurring and more detail is lost.

Another simple way of combining the product filter with wavelets is to exploit the fact that the wavelet decomposition divides the image into an approximation and details. The details are divided into horizontal, vertical and diagonal detail sub-images, each of which has dominant directions that could be exploited by the directional diffusion possible with the product filter. The product filter can be tuned, via the number of directions and the angles of those directions, to diffuse in specific directions. Figure 5.29 shows the results of a simple experiment conducted with the Daubechies 3 wavelet [17]. Noise of the type $50\,\mathrm{rand}(m, n)$ was added to the Barbara image. The two-level wavelet decomposition of the noisy image was performed. In Figure 5.29(c), the product filter was applied in the preferential directions to the detail sub-images (one each for the horizontal and vertical and two for the diagonal) with $t = 0.0003$. The reconstructed image is relatively poor - a lot of detail has been lost and there are

some blocking effects. In Figure 5.29(d). the product filter was applied (with $p = 256$) to the approximation image as well as the details (with their prefential directions). The processed image is smoother and more detail is retained. However. there is no advantage to this technique as nothing is gained in image quality and calculation complexity is significantly increased over simple application of the product filter directly to the noisy image.

A recent wavelet technique of Portilla *et al* [36] is quite successful at removing Gaussian noise from Barbara. Gaussian noise of mean 0 and standard deviation 25 was added to the Barbara image (as was done in [36]) to produce the noisy image. The results are shown in Figure 5.30. The SNR of the noisy image was 26.6409: the PSNR was 20.1735 dB (compared to 20.17 dB in [36]). Formula (1.9) for PSNR is given in Chapter 1. The product filter was applied with $t = 0.0003$ and $p = 256$ to produce the processed image. The SNR of the processed image is 51.7922: the PSNR is 23.0606 dB. The wavelet technique of Portilla *et al* produced a PSNR of 28.57 dB and a superior image [36].

The last set of experiments conducted with the product filter concerned quantized images. When an image is quantized. the effect can be similar to noise. as the intensity values are affected. The Barbara image was quantized and the product filter applied with $p = 256$ and five t values (0.0001 to 0.0005) for three quantizations. In each case. the intensity values are set to the midpoint of the appropriate level. The Barbara image has intensity values ranging from 0 to 254. Figure 5.31 shows the intensity values of the image. It can be seen that certain values are more likely than others. and thus. the image cannot be considered to be of a truly random nature. Representative results are shown in Figures 5.32 and 5.33. In all cases. it was seen that the effect of quantization is not random either. as the difference between the original image and the quantized image actually resembles the image. The effect of the product filter is to smooth

the quantized image, improving the appearance slightly, but not actually undoing the effects of quantization. The SNR values for this study are given in Table 5.1.

Table 5.1: The SNR results for the product filter applied to quantized images. The results for 8 quantization levels are shown in Figures 5.32 and 5.33. Formula (1.8) for SNR is given in Chapter 1.

t parameter	16 levels	8 levels	4 levels
(quantized)	205.7274	45.1290	10.6750
0.0001	119.6928	44.1139	11.7285
0.0002	73.3049	36.9073	11.4629
0.0003	54.9345	31.8877	11.0500
0.0004	45.7668	28.6494	10.6963
0.0005	40.4421	26.4854	10.4189

The product filter has been found to be comparable in de-noising ability to MATLAB's `averaging` filter and inferior to the wavelet techniques. However, the product filter does preserve image details reasonably well.

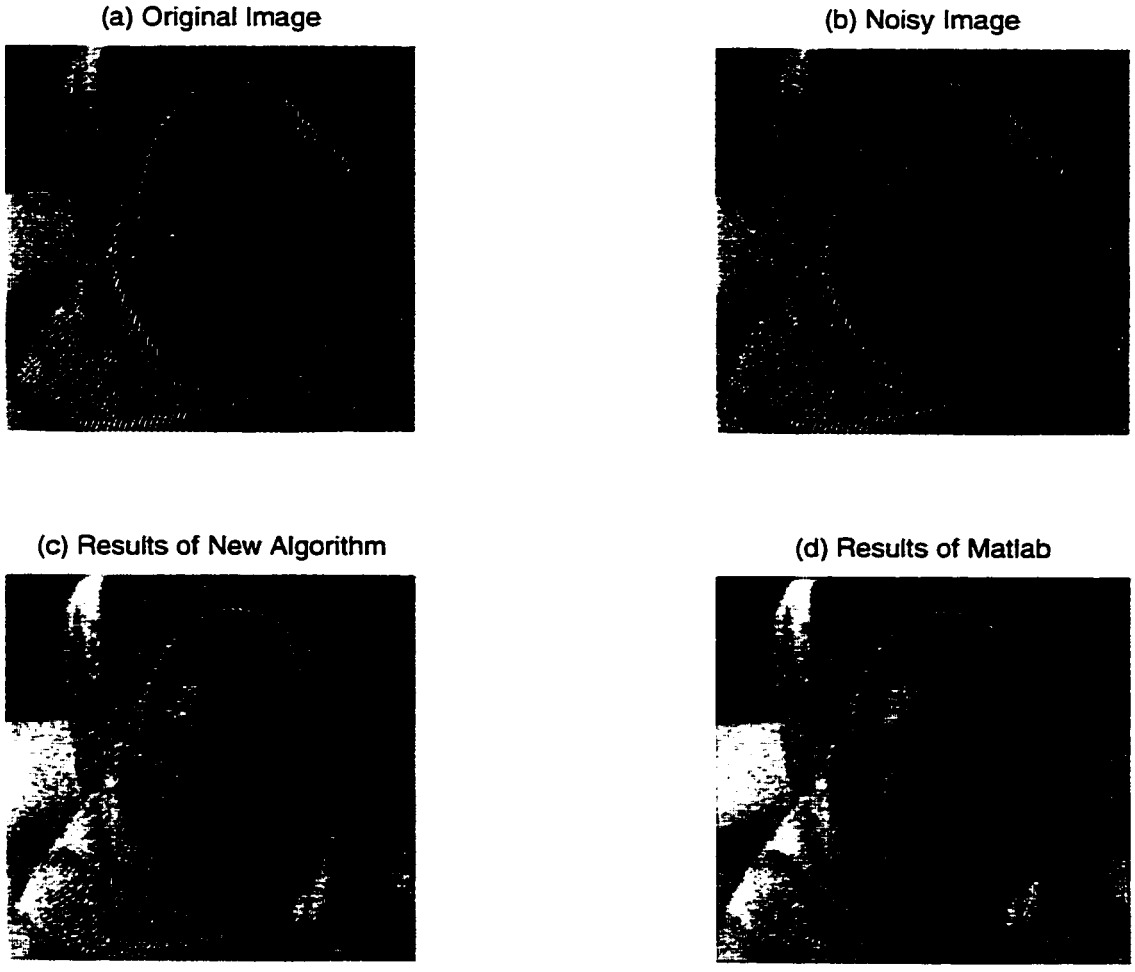

Figure 5.1: A comparison of the new product filter with MATLAB's **averaging** filter. $t = 0.0003$. $p = 256$ and the noise is $50\,\mathrm{rand}(m, n)$. (a) is the original image. (b) is the noisy image. SNR = 19.9288. (c) shows the results of the new product filter. SNR = 18.7798. (d) shows the results of MATLAB. SNR = 15.6814.

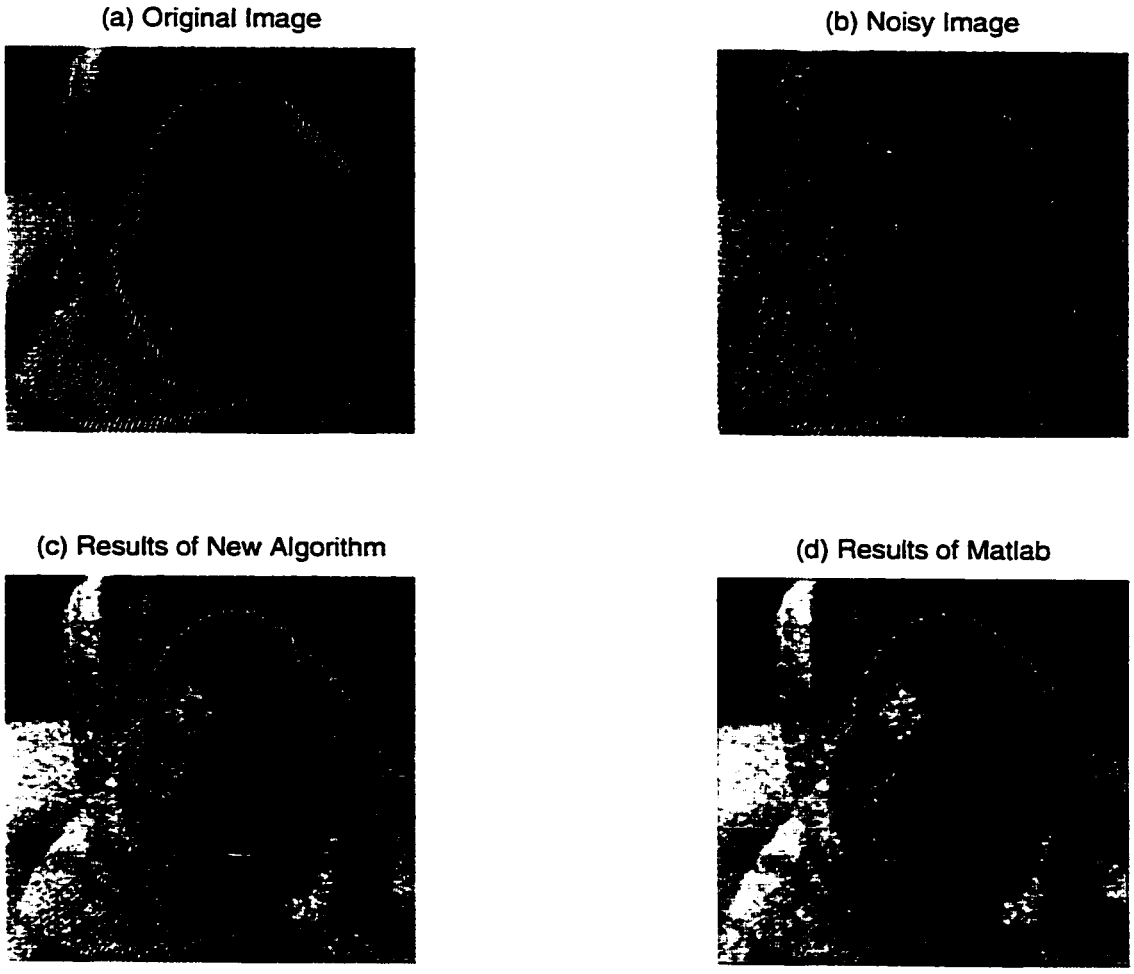

Figure 5.2: A comparison of the new product filter with MATLAB's **averaging** filter. $t = 0.0003$, $p = 256$ and the noise is $100\,\mathrm{rand}(m, n)$. (a) is the original image. (b) is the noisy image, SNR = 5.0098. (c) shows the results of the new product filter. SNR = 5.8448. (d) shows the results of MATLAB. SNR = 5.6238.

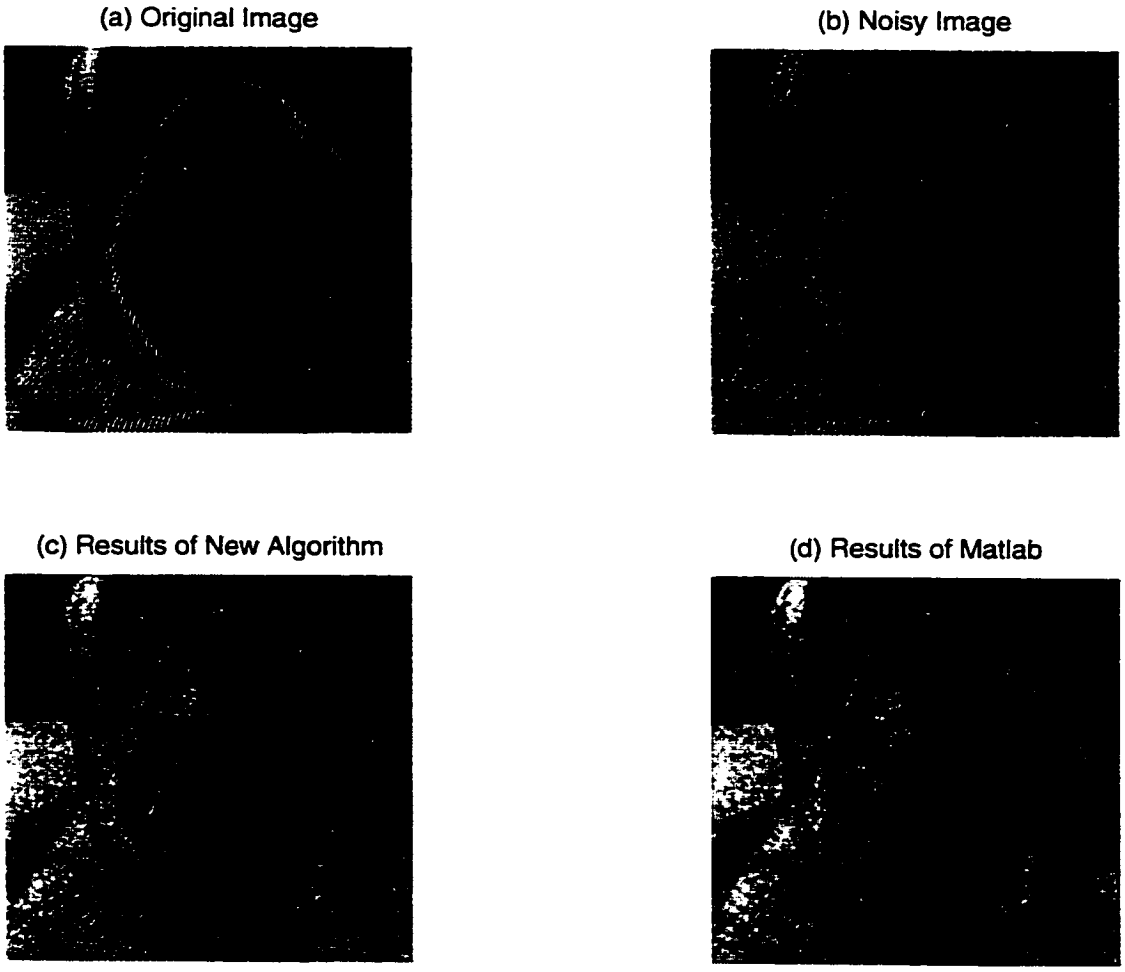

Figure 5.3: A comparison of the new product filter with MATLAB's **averaging** filter. $t = 0.0003$. $p = 256$ and the noise is $100\,\mathrm{rand}(m, n) - 50\,\mathrm{rand}(m, n)$. (a) is the original image. (b) is the noisy image. SNR = 9.9567. (c) shows the results of the new product filter. SNR = 16.3794. (d) shows the results of MATLAB. SNR = 14.4540.

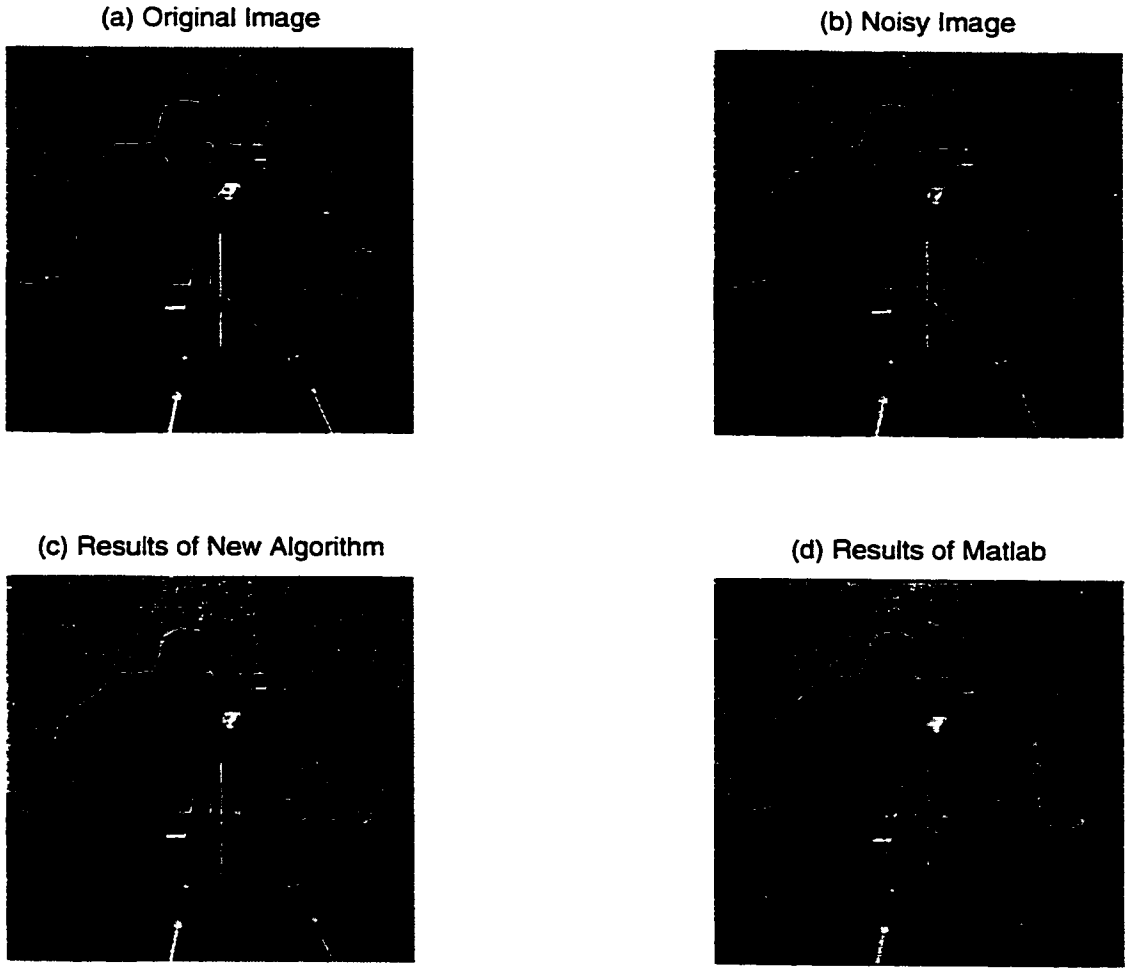

Figure 5.4: A comparison of the new product filter with MATLAB's averaging filter. $t = 0.0003$. $p = 256$ and the noise is $50\,\mathrm{rand}(m.n)$. (a) is the original image. (b) is the noisy image. SNR = 21.7465. (c) shows the results of the new product filter. SNR = 24.3076. (d) shows the results of MATLAB. SNR = 22.1840.

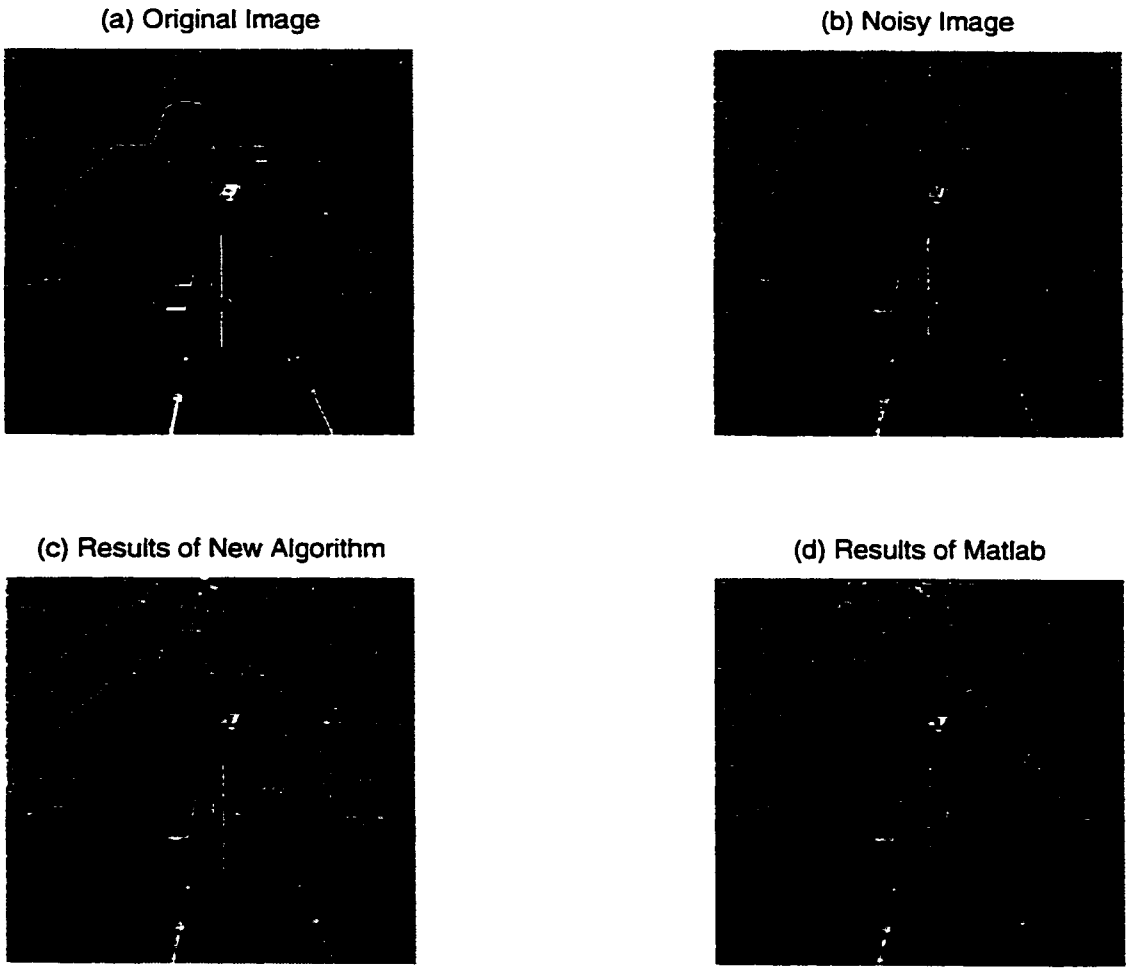

Figure 5.5: A comparison of the new product filter with MATLAB's **averaging** filter. $t = 0.0003$, $p = 256$ and the noise is $100\,\mathrm{rand}(m, n)$. (a) is the original image. (b) is the noisy image, SNR = 5.3681. (c) shows the results of the new product filter. SNR = 6.5761. (d) shows the results of MATLAB. SNR = 6.5777.

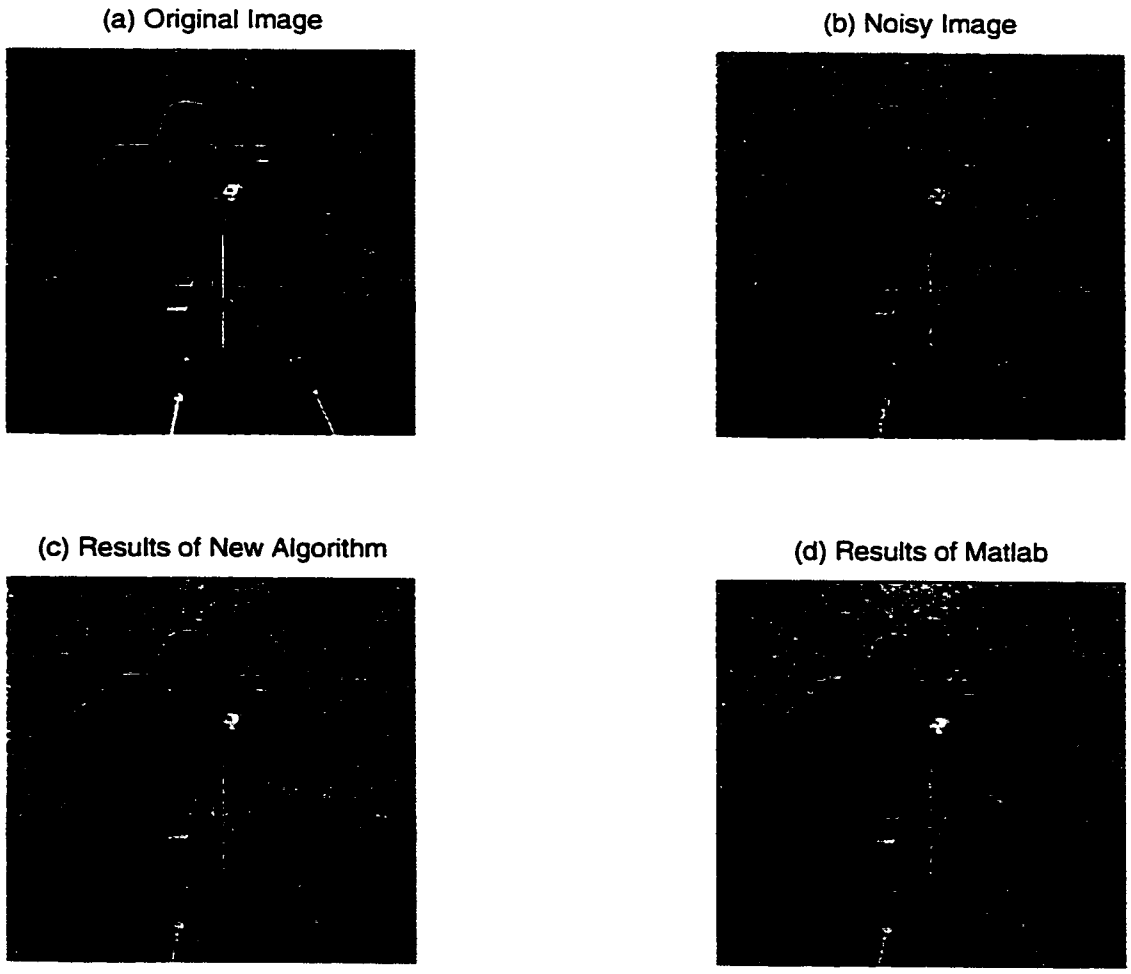

Figure 5.6: A comparison of the new product filter with MATLAB's **averaging** filter. $t = 0.0003$. $p = 256$ and the noise is $100\,\mathrm{rand}(m, n) - 50\,\mathrm{rand}(m, n)$. (a) is the original image. (b) is the noisy image. SNR = 10.8747. (c) shows the results of the new product filter. SNR = 20.7148. (d) shows the results of MATLAB. SNR = 19.9638.

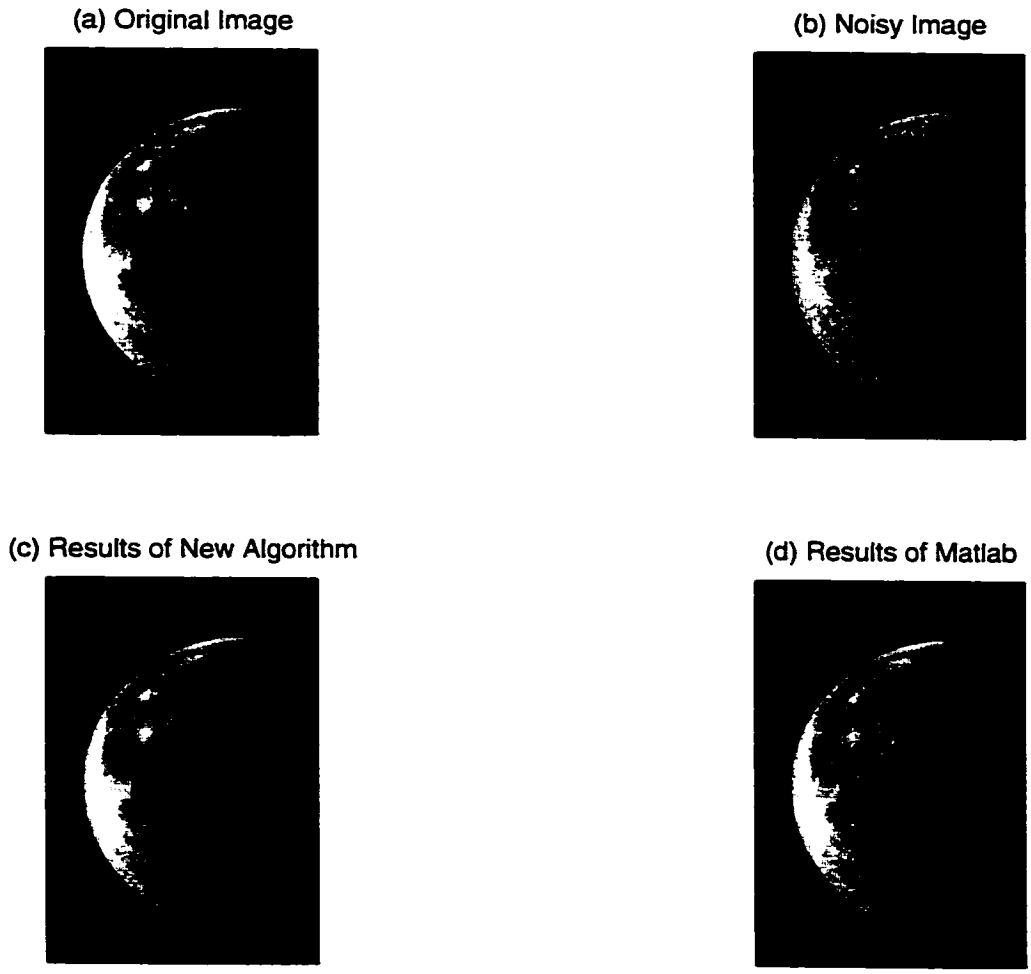

Figure 5.7: A comparison of the new product filter with MATLAB's averaging filter. $t = 0.0003$. $p = 256$ and the noise is $50\,\mathrm{rand}(m.n)$. (a) is the original image. (b) is the noisy image. SNR = 10.6216. (c) shows the results of the new product filter. SNR = 13.5580. (d) shows the results of MATLAB. SNR = 13.5281.

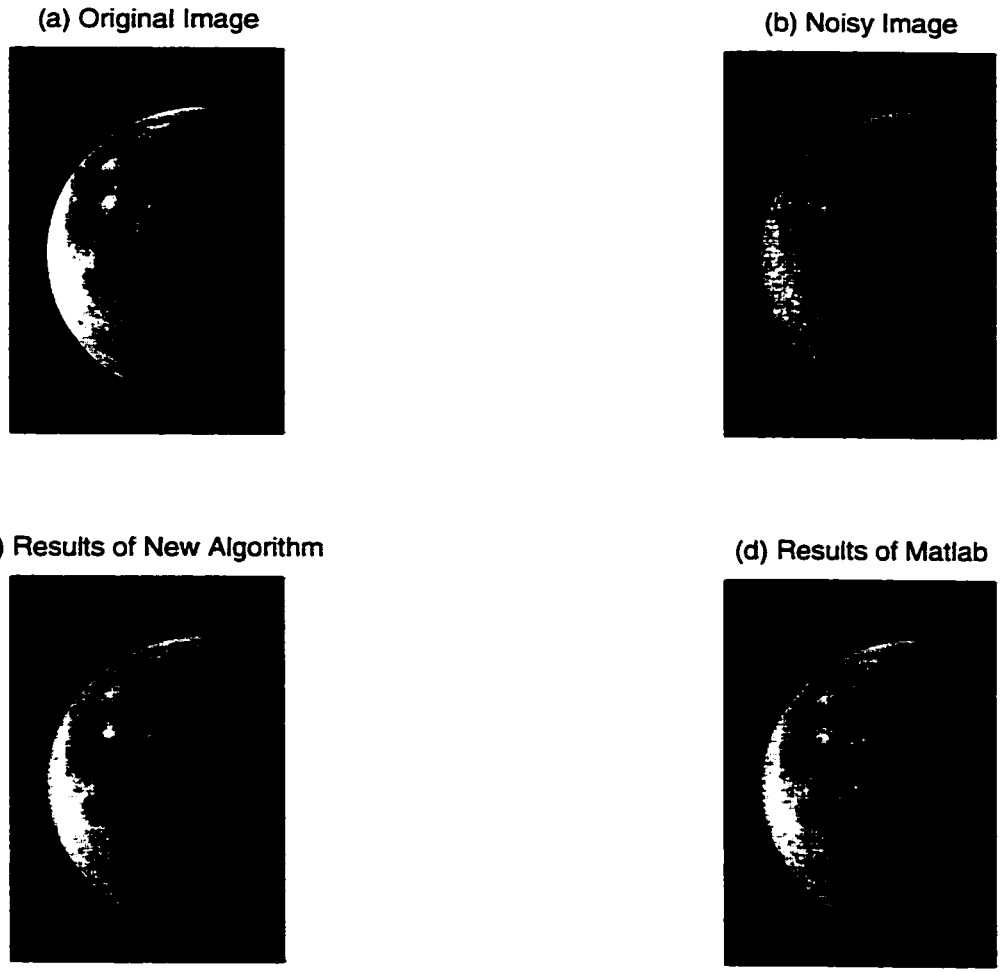

Figure 5.8: A comparison of the new product filter with MATLAB's **averaging** filter. $t = 0.0003$, $p = 256$ and the noise is $100\,\mathrm{rand}(m, n)$. (a) is the original image. (b) is the noisy image. SNR = 2.6674. (c) shows the results of the new product filter. SNR = 3.4714. (d) shows the results of MATLAB. SNR = 3.4355.

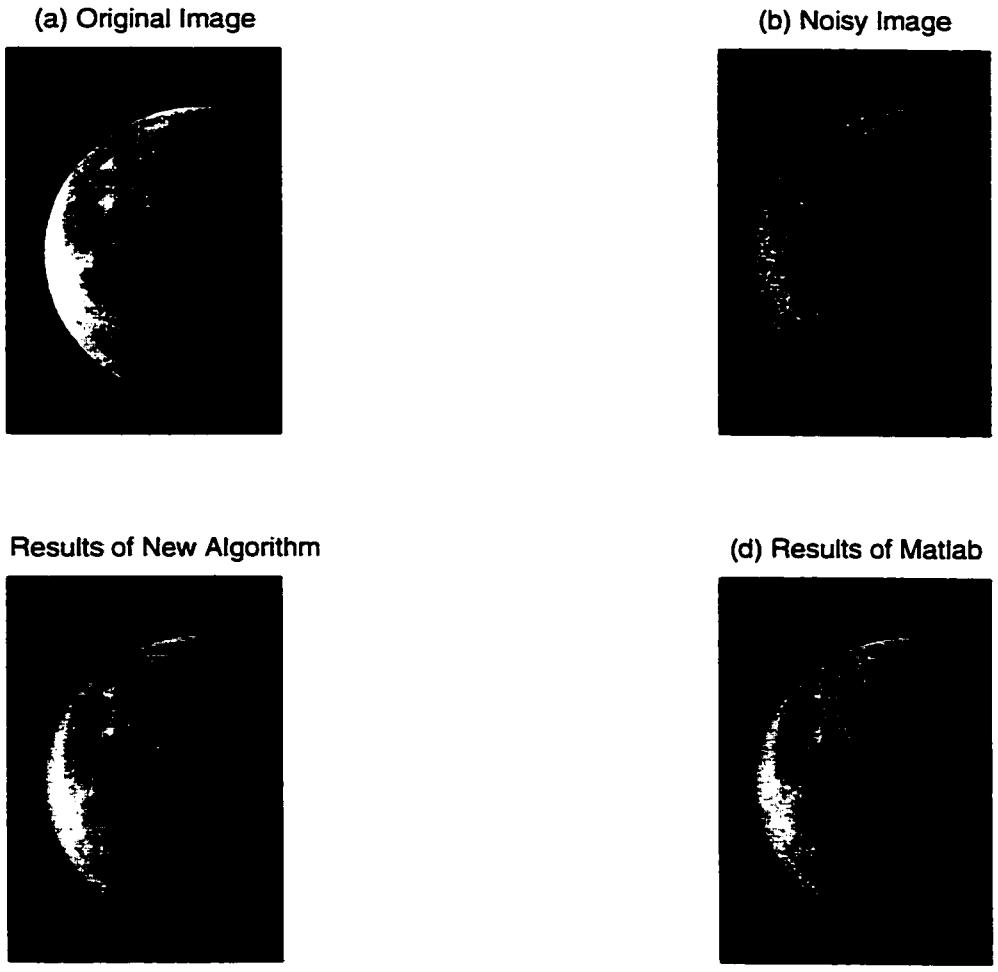

Figure 5.9: A comparison of the new product filter with MATLAB's **averaging** filter. $t = 0.0003$, $p = 256$ and the noise is $100\,\mathrm{rand}(m, n) - 50\,\mathrm{rand}(m, n)$. (a) is the original image. (b) is the noisy image. SNR = 5.3437. (c) shows the results of the new product filter. SNR = 12.8224. (d) shows the results of MATLAB. SNR = 11.9916.

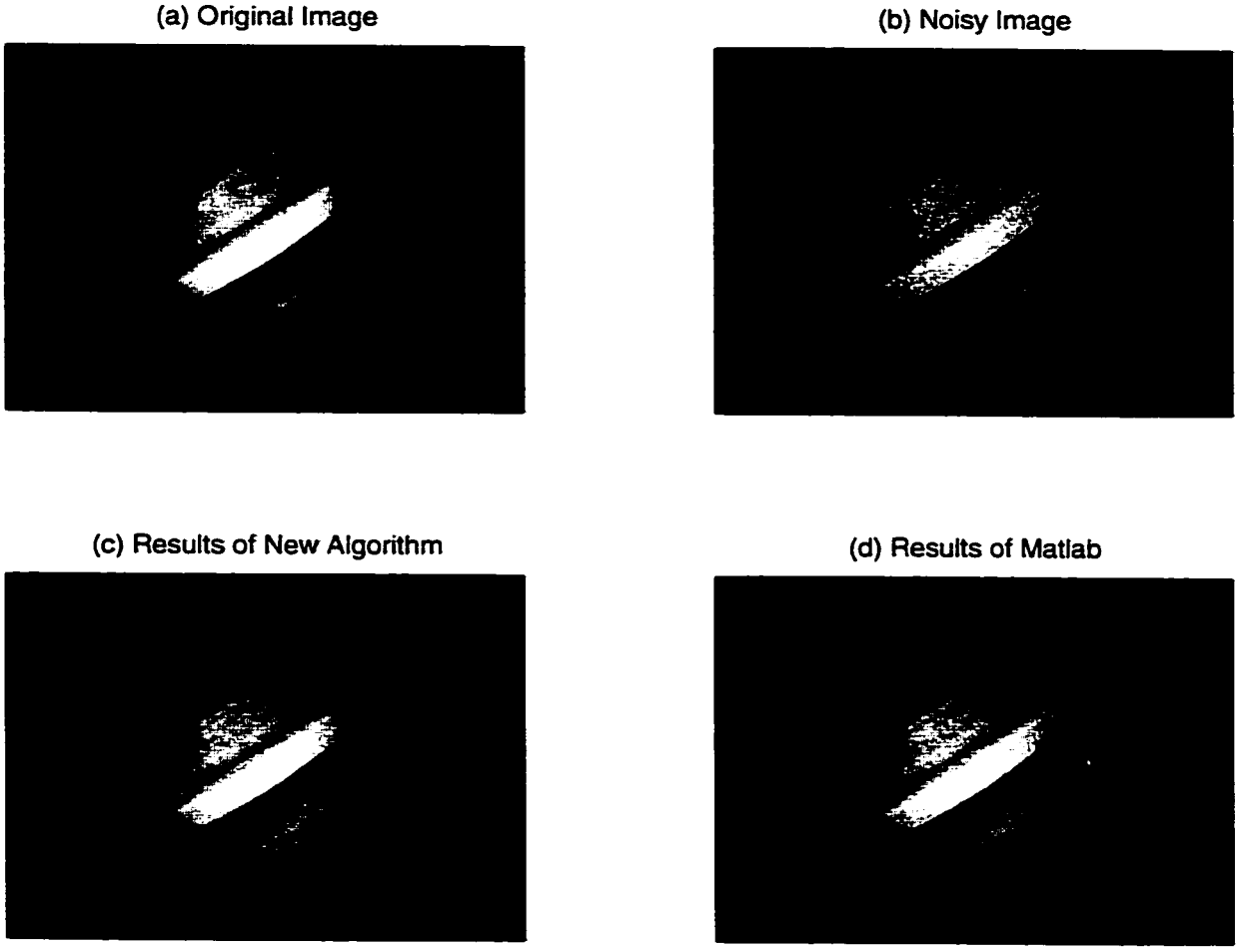

Figure 5.10: A comparison of the new product filter with MATLAB's averaging filter. $t = 0.0003$, $p = 256$ and the noise is $50\,\mathrm{rand}(m, n)$. (a) is the original image. (b) is the noisy image, SNR = 9.4653. (c) shows the results of the new product filter. SNR = 12.1977. (d) shows the results of MATLAB. SNR = 12.1605.

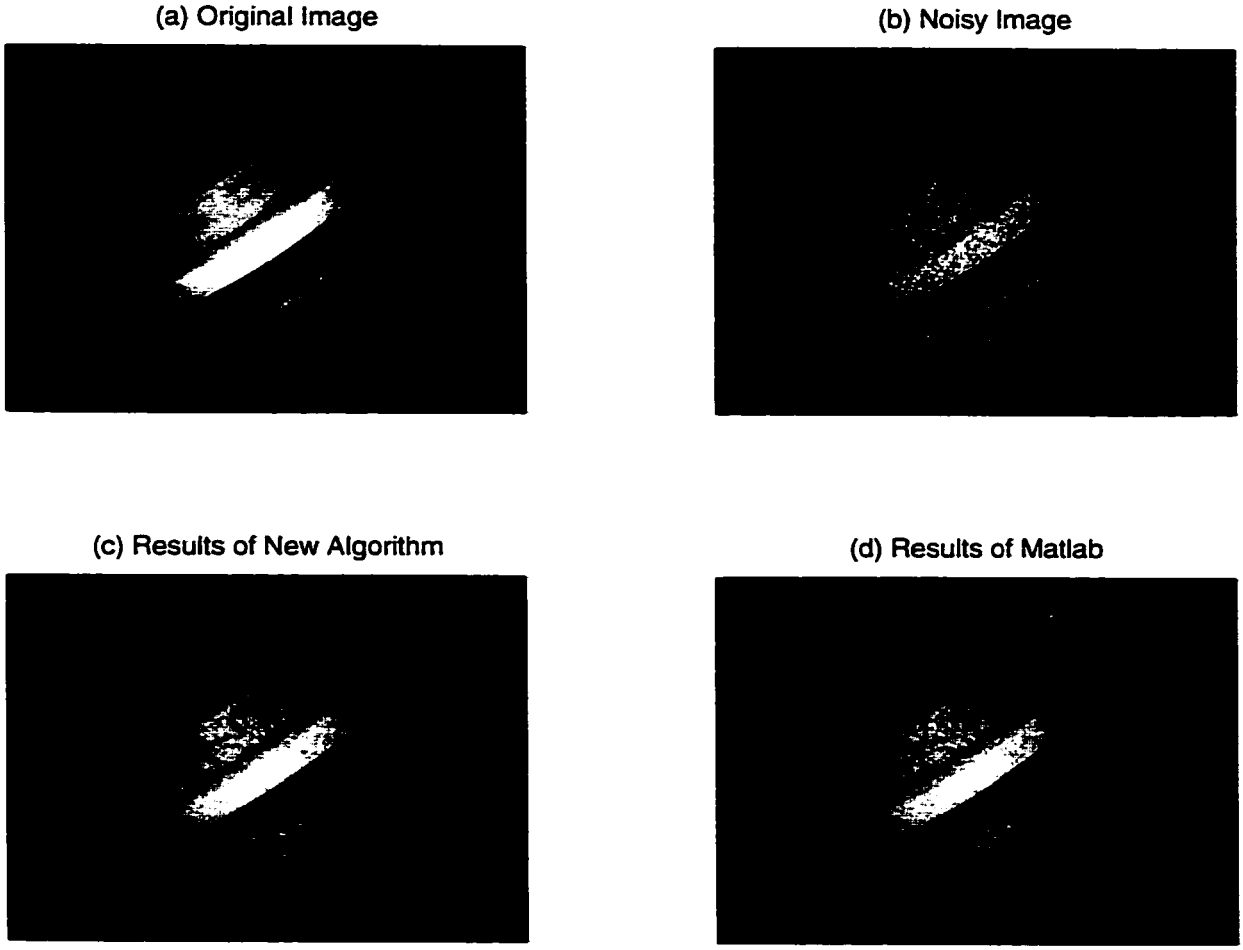

Figure 5.11: A comparison of the new product filter with MATLAB's averaging filter. $t = 0.0003$. $p = 256$ and the noise is $100\,\mathrm{rand}(m, n)$. (a) is the original image. (b) is the noisy image. SNR = 2.3580. (c) shows the results of the new product filter. SNR = 3.0627. (d) shows the results of MATLAB. SNR = 3.0448.

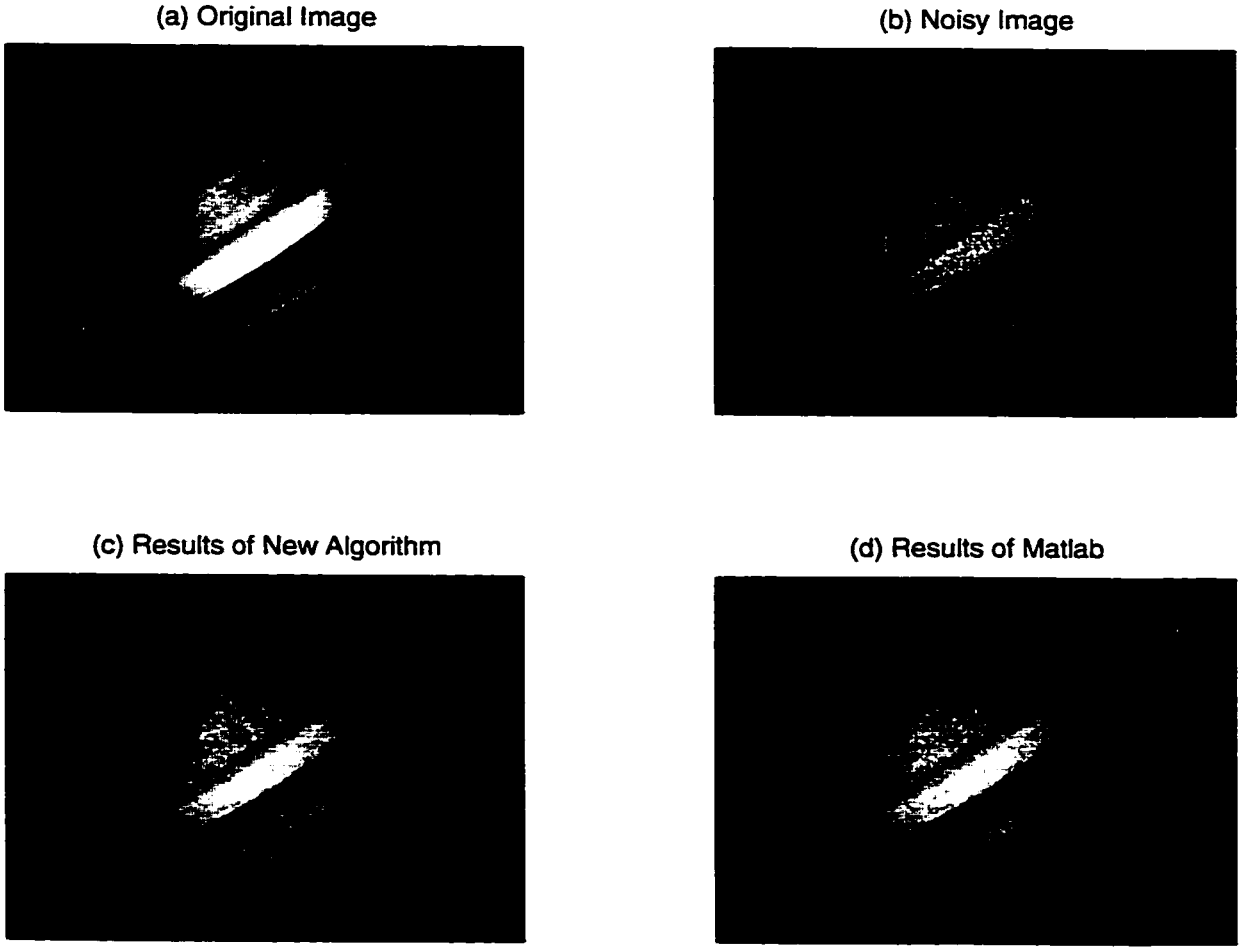

Figure 5.12: A comparison of the new product filter with MATLAB's **averaging** filter. $t = 0.0003$, $p = 256$ and the noise is $100\,\mathrm{rand}(m, n) - 50\,\mathrm{rand}(m, n)$. (a) is the original image. (b) is the noisy image. SNR = 4.7220. (c) shows the results of the new product filter. SNR = 11.1907. (d) shows the results of MATLAB. SNR = 10.6792.

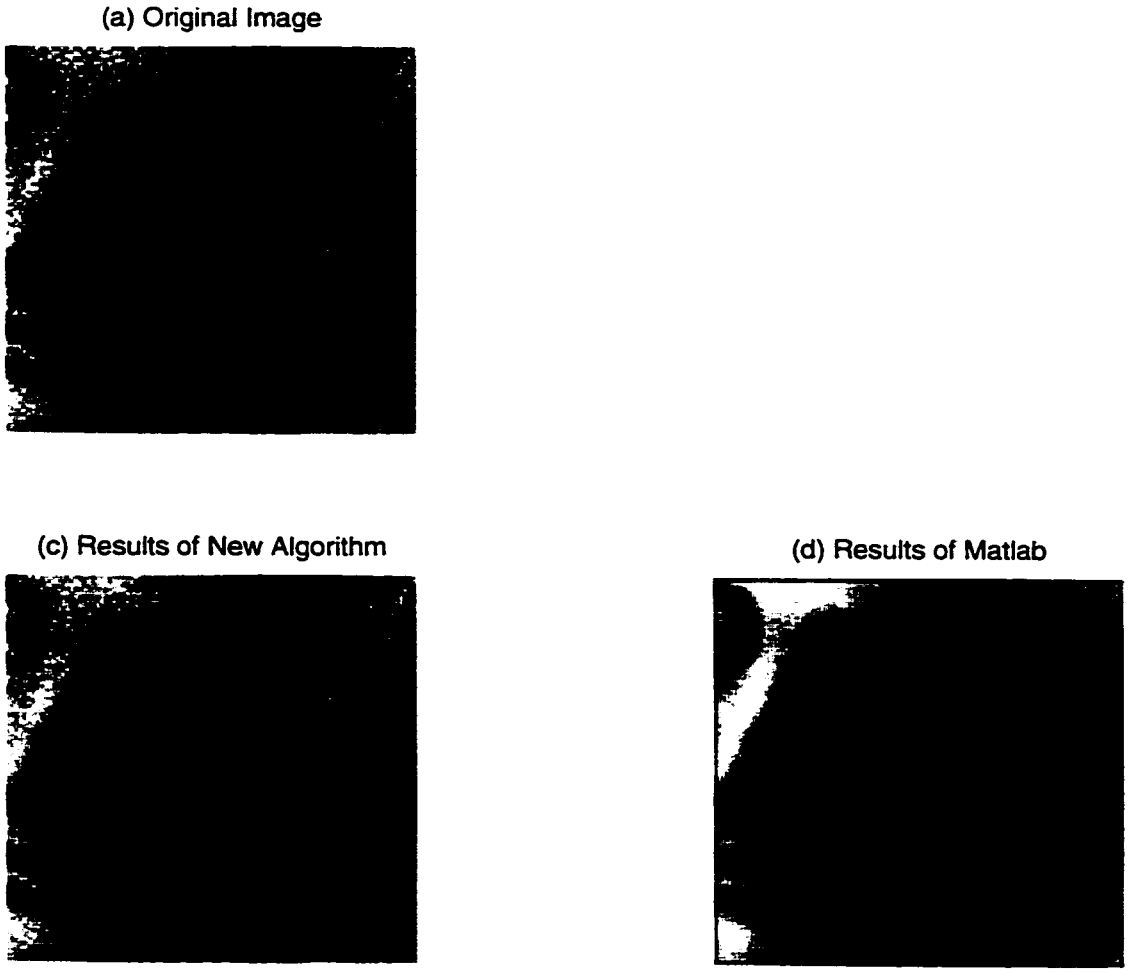

Figure 5.13: A comparison of the new product filter algorithm with MATLAB's **averaging** filter. $t = 0.0003$ and $p = 256$. (a) is the original noisy image. (Sub-image (b) would be the same as (a) in this example and is not shown.) (c) shows the results of the new product filter. SNR = 615.1433. (d) shows the results of MATLAB. SNR = 30.2710.

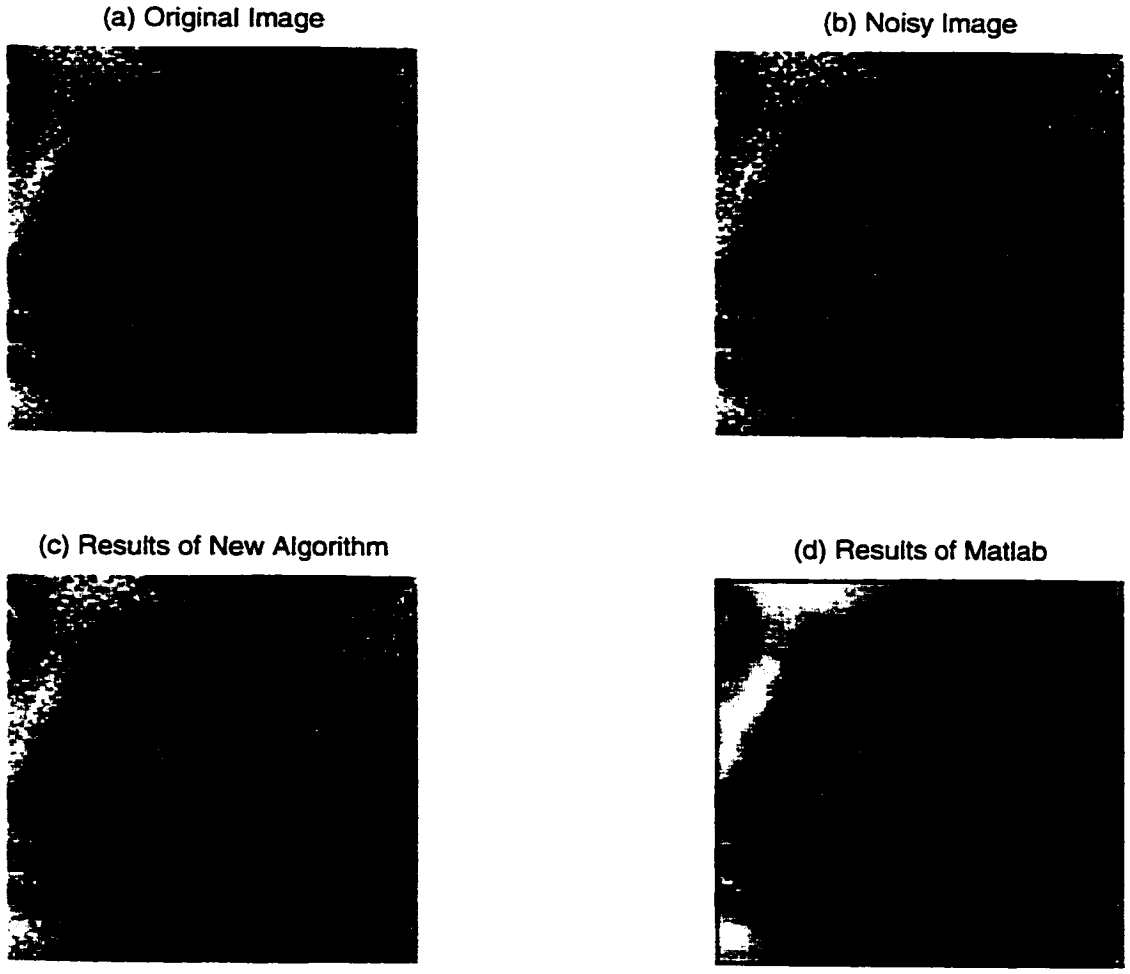

Figure 5.14: A comparison of the new product filter algorithm with MATLAB's averaging filter. $t = 0.0003$ and $p = 256$. (a) is the original noisy image. (b) shows the image with random noise of intensity 50 added. SNR = 13.9918 (c) shows the results of the new product filter. SNR = 14.9311. (d) shows the results of MATLAB. SNR = 12.1096.

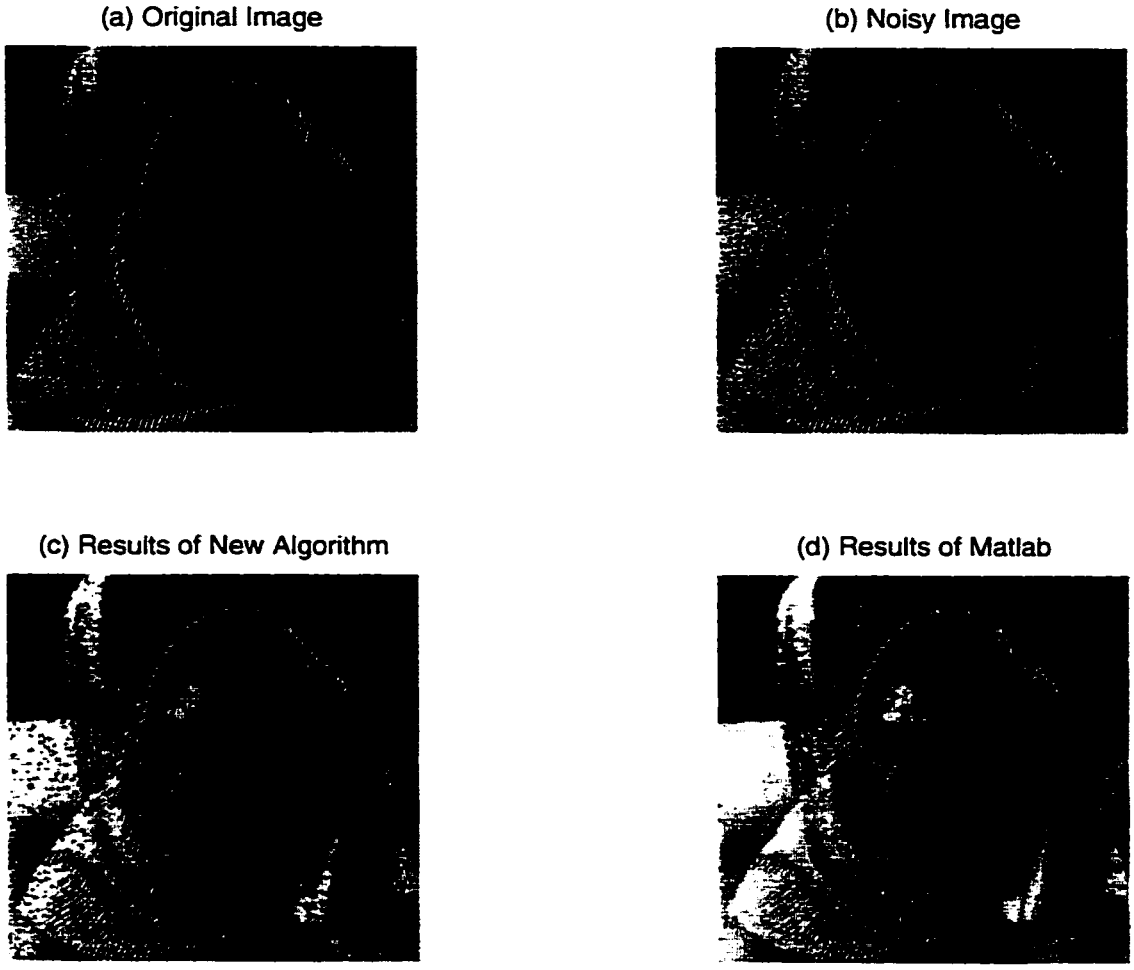

Figure 5.15: A comparison of the new product filter algorithm with MATLAB's `median` filter for 5% salt and pepper noise. with $t = 0.0003$ and $p = 256$. (a) is the original image. (b) shows the image with 5% salt and pepper noise added. SNR = 19.8656. (c) shows the results of the product filter. SNR = 40.9008. (d) shows the results of MATLAB. SNR = 36.1251.

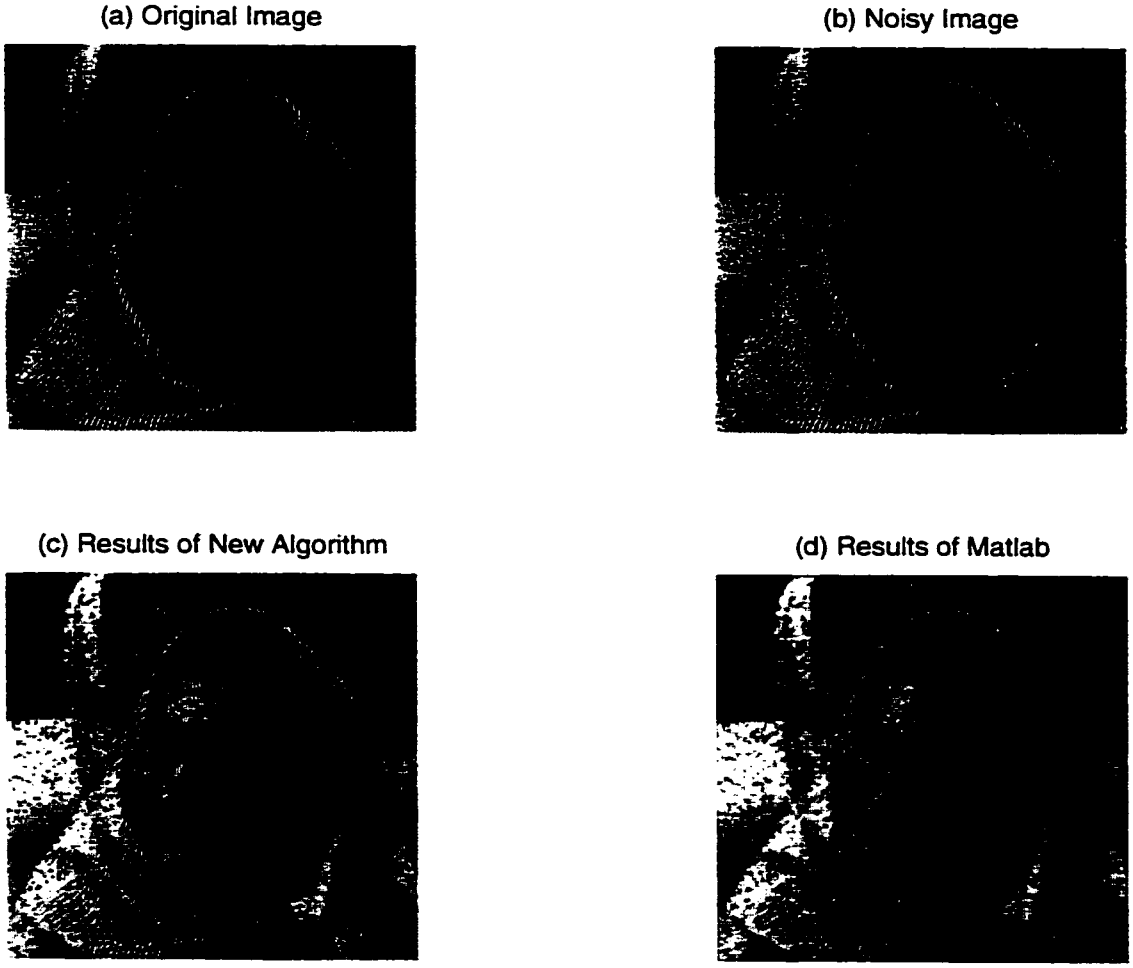

Figure 5.16: A comparison of the new product filter algorithm with MATLAB's averaging filter for 5% salt and pepper noise. with $t = 0.0003$ and $p = 256$. (a) is the original image. (b) shows the image with 5% salt and pepper noise added. SNR = 20.3063. (c) shows the results of the product filter. SNR = 41.3073. (d) shows the results of MATLAB. SNR = 28.9028.

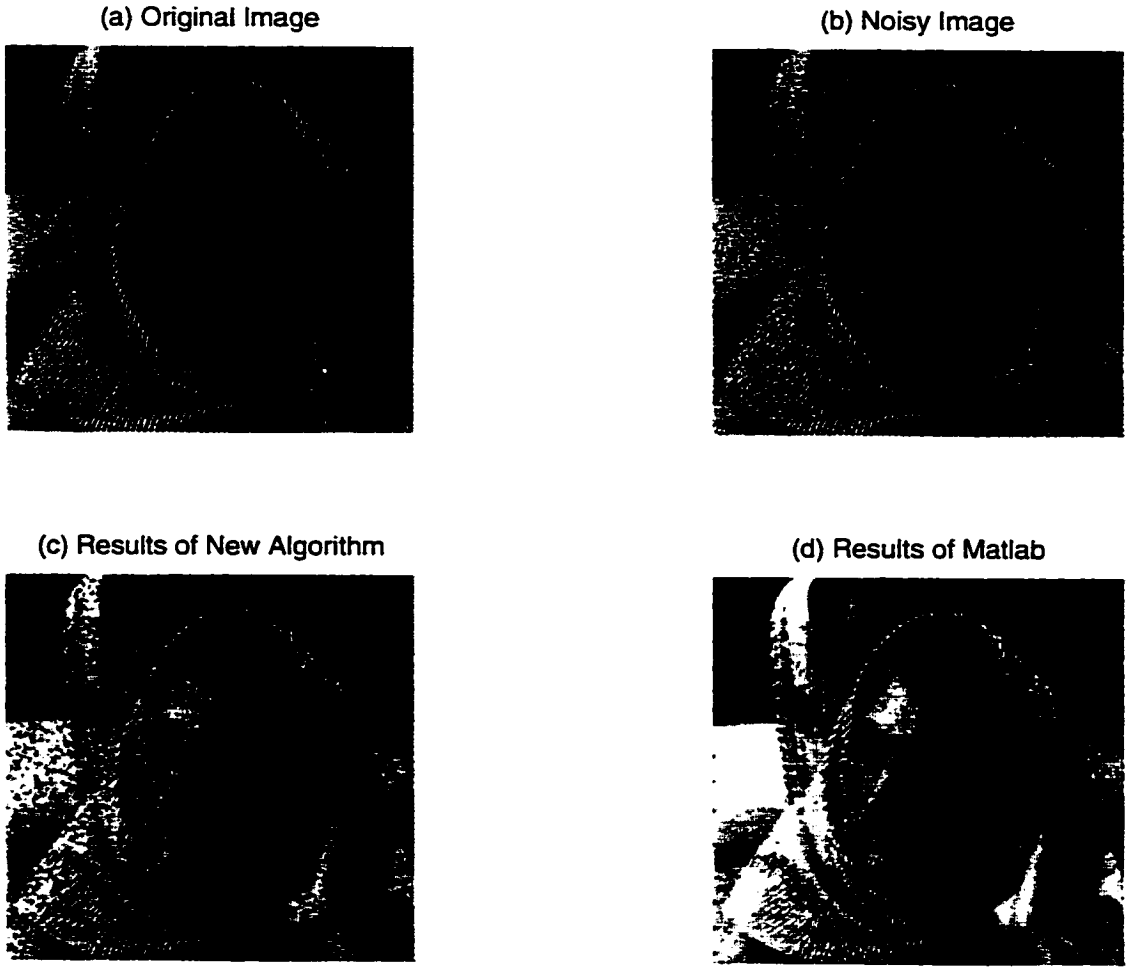

Figure 5.17: A comparison of the new product filter algorithm with MATLAB's `median` filter for 10% salt and pepper noise. with $t = 0.0003$ and $p = 256$. (a) is the original image. (b) shows the image with 10% salt and pepper noise added. SNR = 10.0473. (c) shows the results of the product filter. SNR = 25.5365. (d) shows the results of MATLAB. SNR = 31.0094.

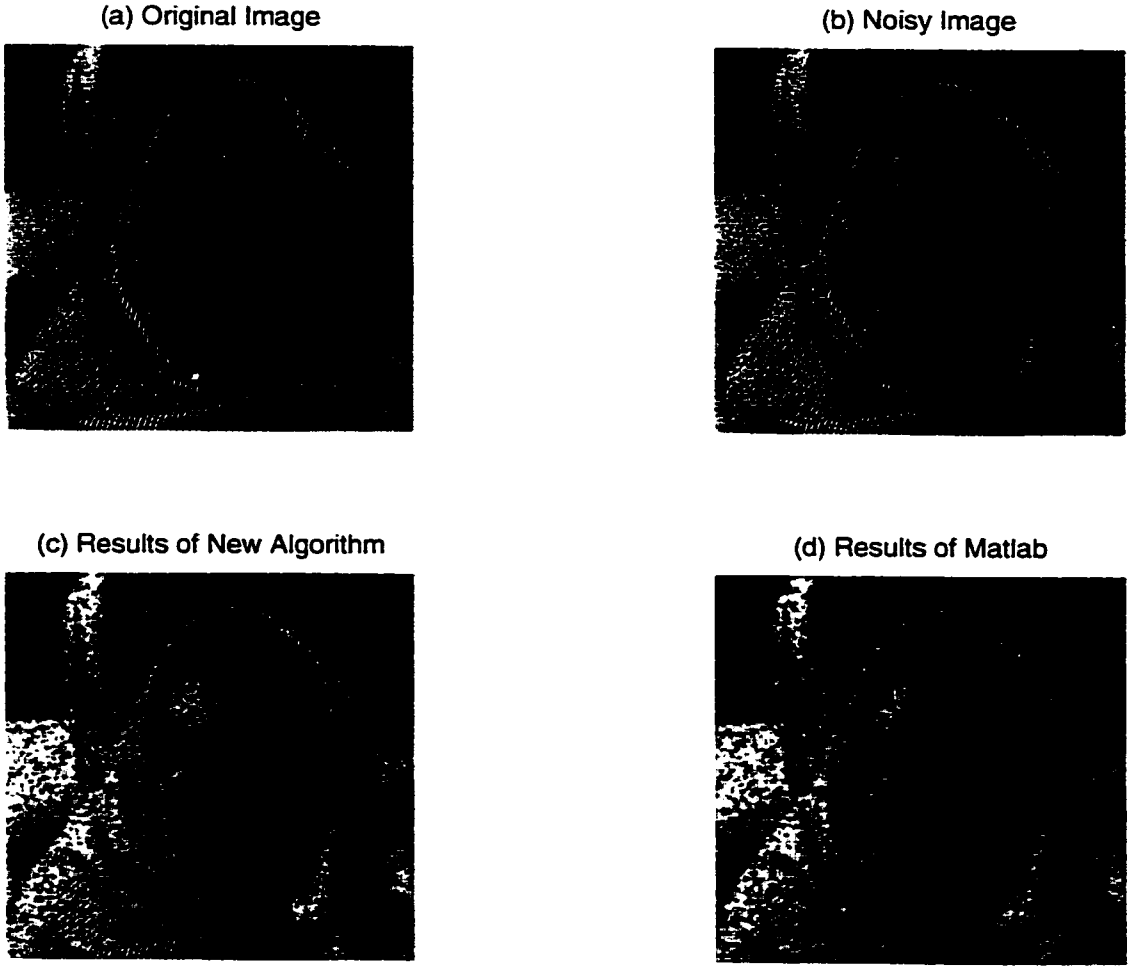

Figure 5.18: A comparison of the new product filter algorithm with MATLAB's averaging filter for 10% salt and pepper noise. with $t = 0.0003$ and $p = 256$. (a) is the original image. (b) shows the image with 10% salt and pepper noise added. SNR = 9.9598. (c) shows the results of the product filter. SNR = 25.2802. (d) shows the results of MATLAB. SNR = 21.0220.

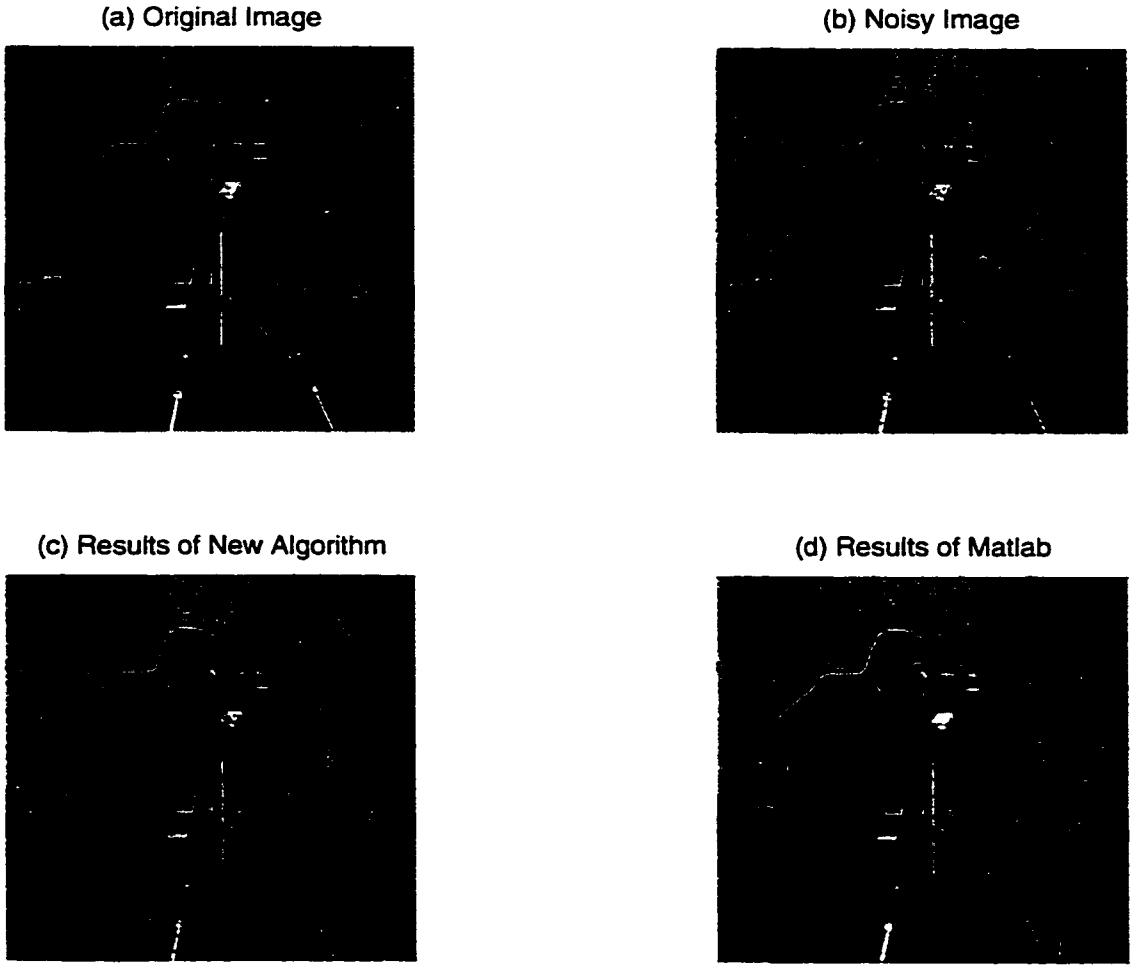

Figure 5.19: A comparison of the new product filter algorithm with MATLAB's `median` filter for 5% salt and pepper noise. with $t = 0.0003$ and $p = 256$. (a) is the original image. (b) shows the image with 5% salt and pepper noise added. SNR = 20.4459. (c) shows the results of the product filter. SNR = 65.5020. (d) shows the results of MATLAB. SNR = 116.3877.

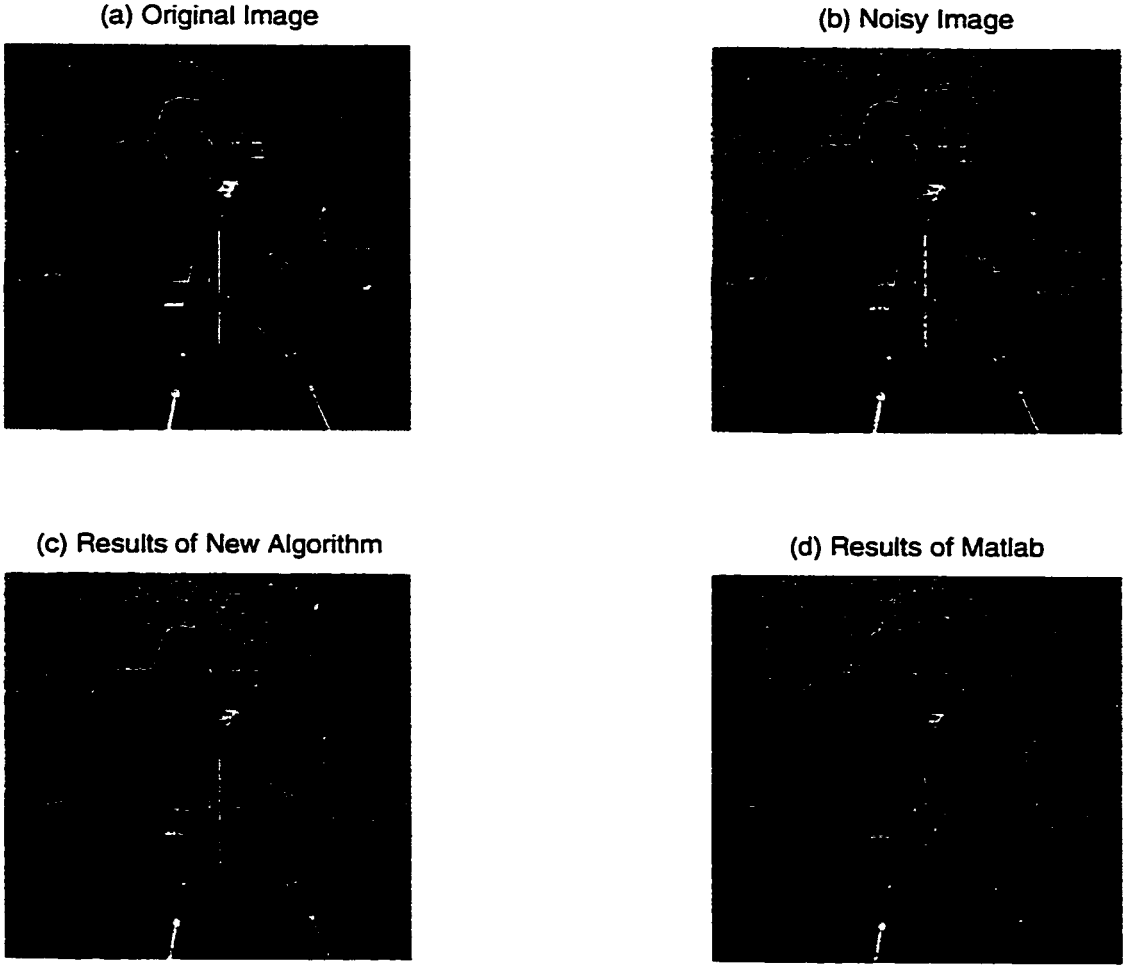

Figure 5.20: A comparison of the new product filter algorithm with MATLAB's **averaging** filter for 5% salt and pepper noise. with $t = 0.0003$ and $p = 256$. (a) is the original image. (b) shows the image with 5% salt and pepper noise added. SNR = 19.8365. (c) shows the results of the product filter. SNR = 64.0569. (d) shows the results of MATLAB. SNR = 51.0473.

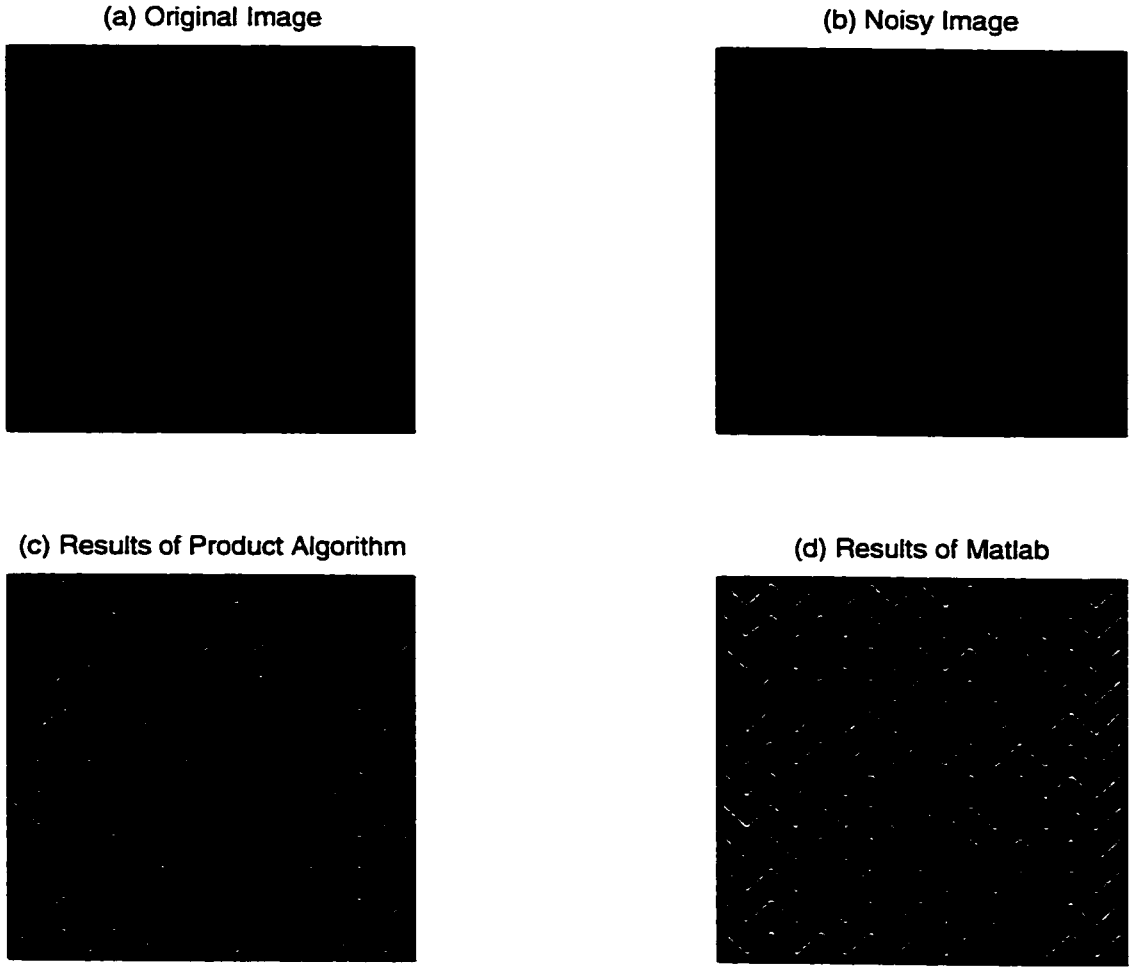

Figure 5.21: A comparison of the product filter algorithm with MATLAB's **averaging** filter for the zigzag image with $t = 0.0003$. $p = 256$ and noise of the form $50\,\mathrm{rand}(m, n)$ (a) is the original image. (b) is the noisy image. SNR = 0.7814. (c) shows the results of the product filter. SNR = 0.6706. (d) shows the results of MATLAB. SNR = 0.6409.

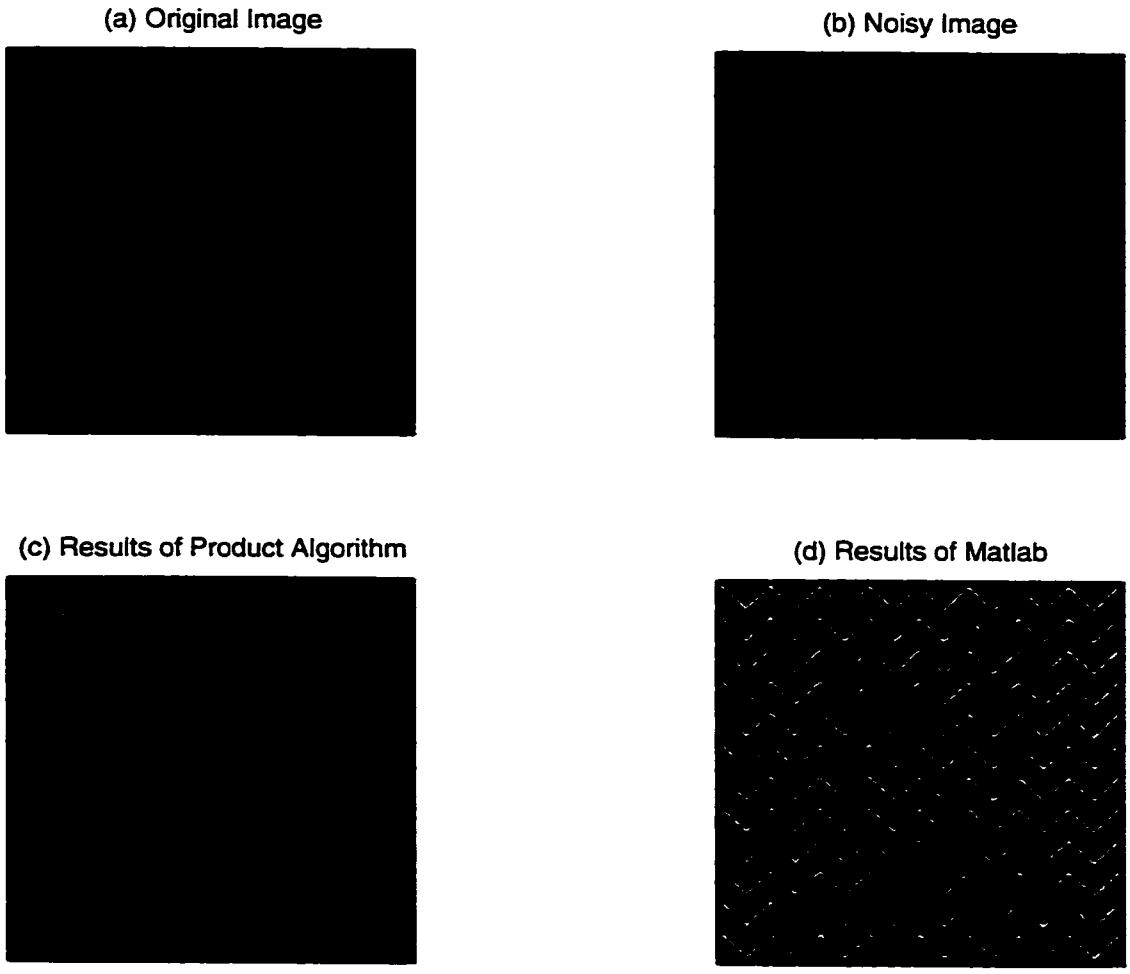

Figure 5.22: A comparison of the product filter algorithm with MATLAB's averaging filter for the zigzag image with $t = 0.0003$. $p = 256$ and noise of the form $50\,\mathrm{rand}(m, n) - 25\,\mathrm{rand}(m, n)$ (a) is the original image. (b) is the noisy image. SNR = 1.5559. (c) shows the results of the product filter. SNR = 1.2716. (d) shows the results of MATLAB. SNR = 1.1659.

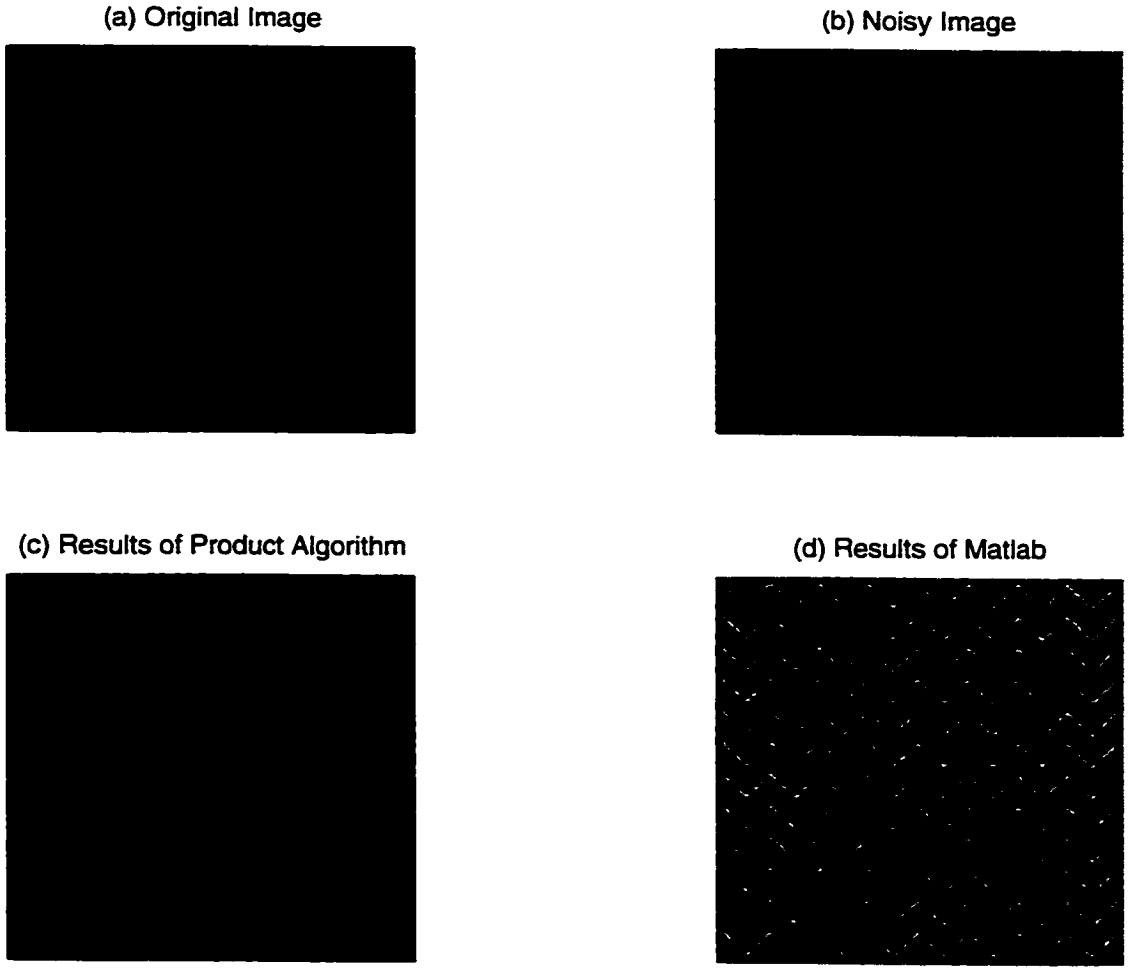

Figure 5.23: A comparison of the product filter algorithm with MATLAB's averaging filter for the zigzag image with $t = 0.0003$. $p = 256$ and noise of the form 25 randn(m, n) (a) is the original image. (b) is the noisy image. SNR = 1.0399. (c) shows the results of the product filter. SNR = 1.5716. (d) shows the results of MATLAB. SNR = 1.4662.

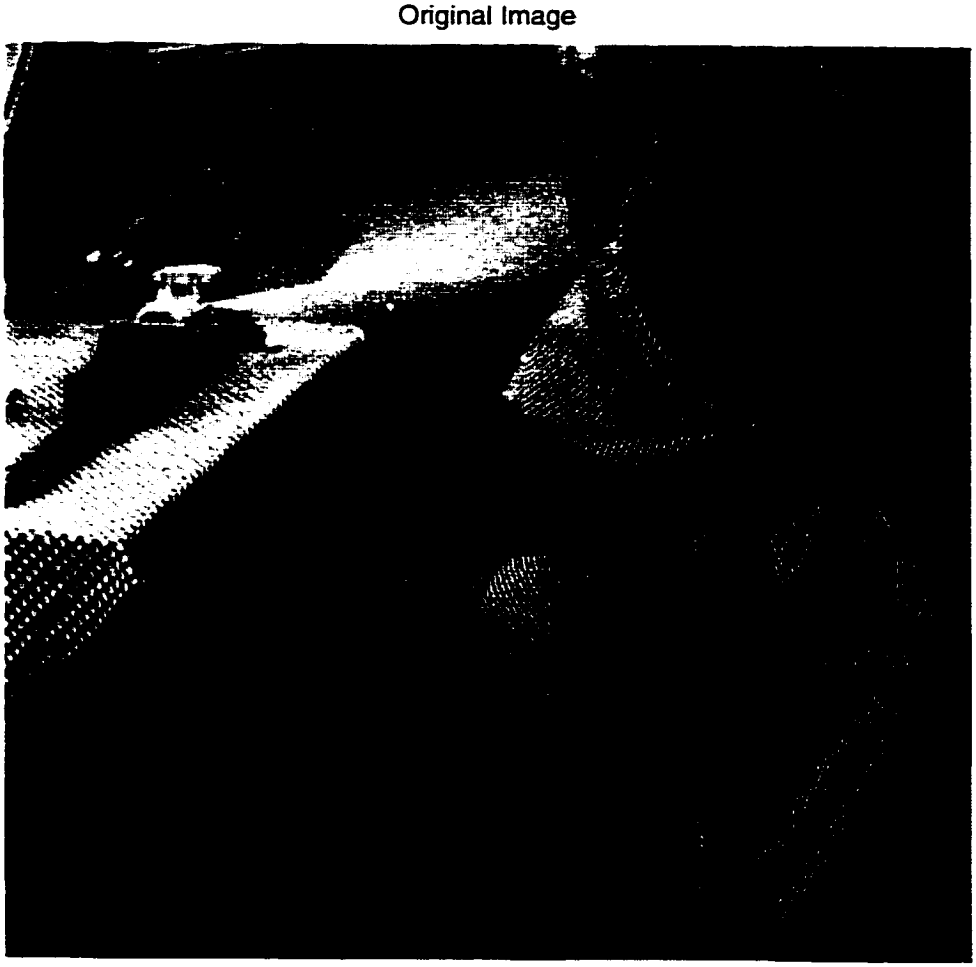

Figure 5.24: The Full Barbara image of size 512×512.

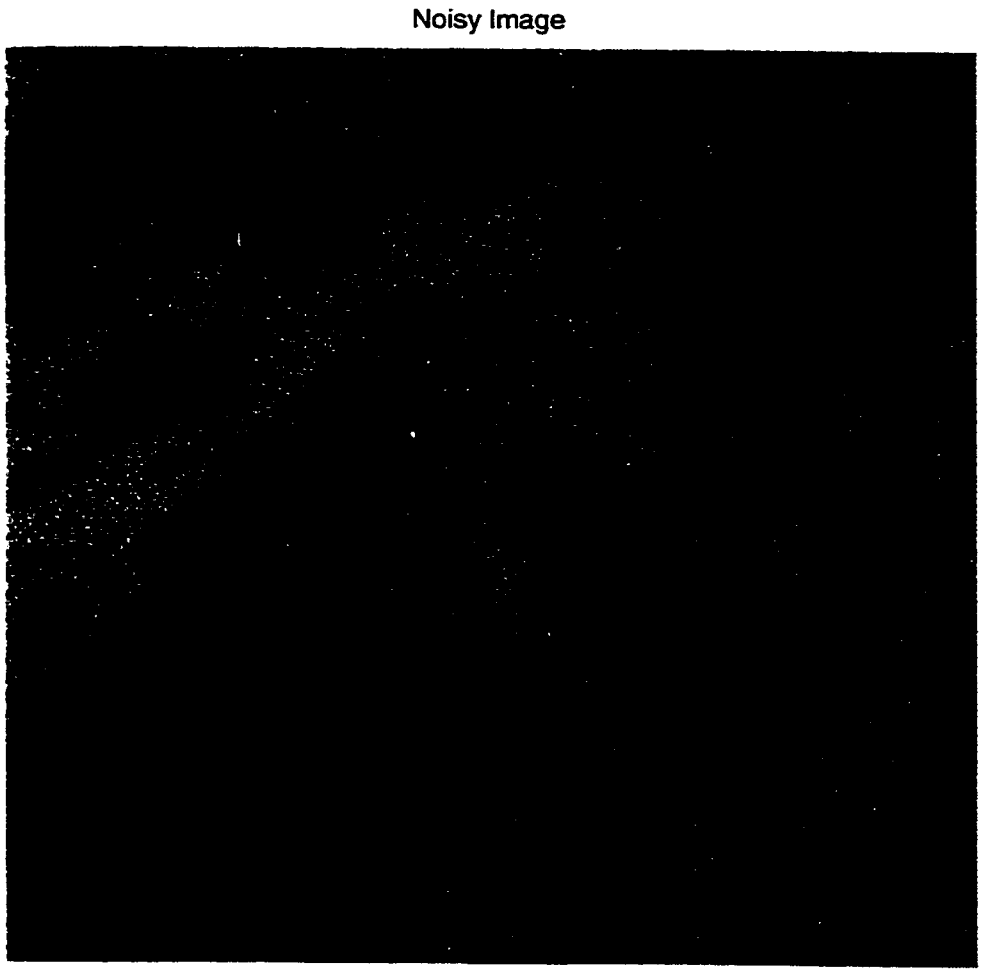

Figure 5.25: The Full Barbara image of size 512×512 with noise of unknown type and distribution. SNR = 10.5562. The original image is shown in Figure 5.24.

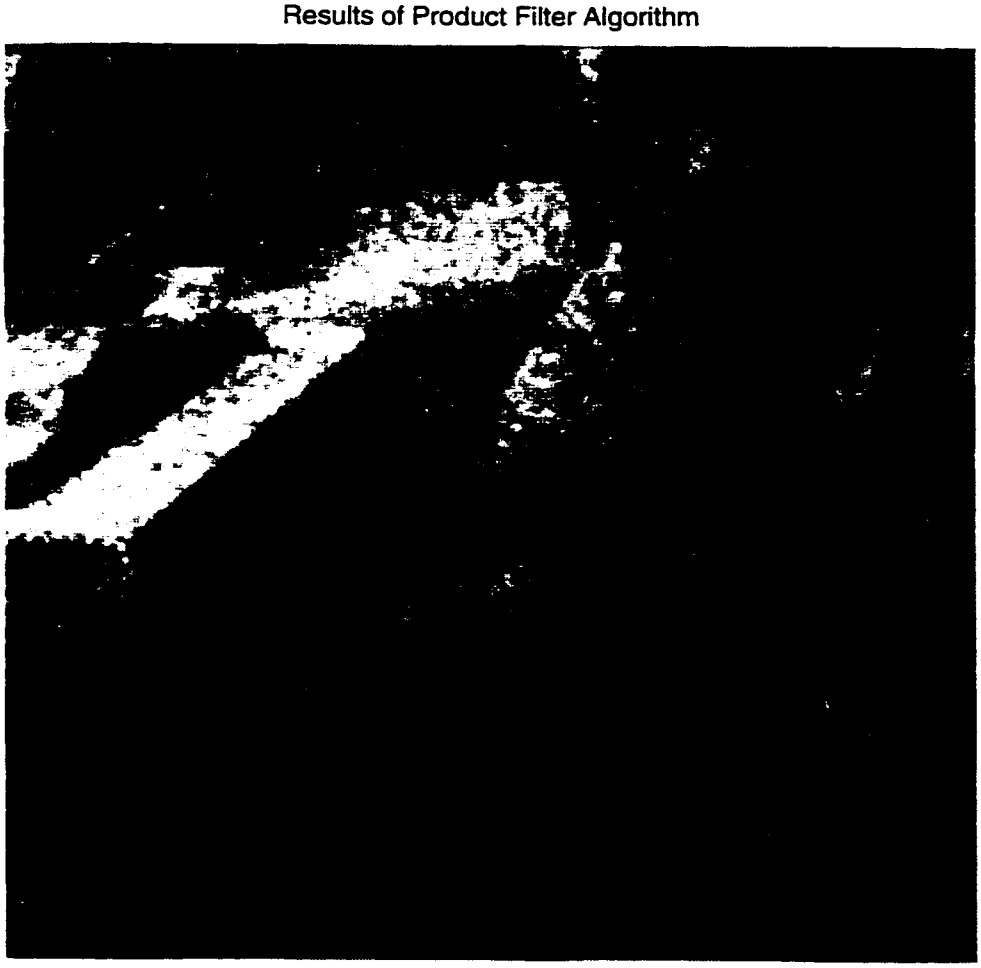

Figure 5.26: Results of the product filter algorithm applied to the noisy Full Barbara image from Figure 5.25. with $t = 0.0003$ and $p = 256$. SNR = 12.0218.

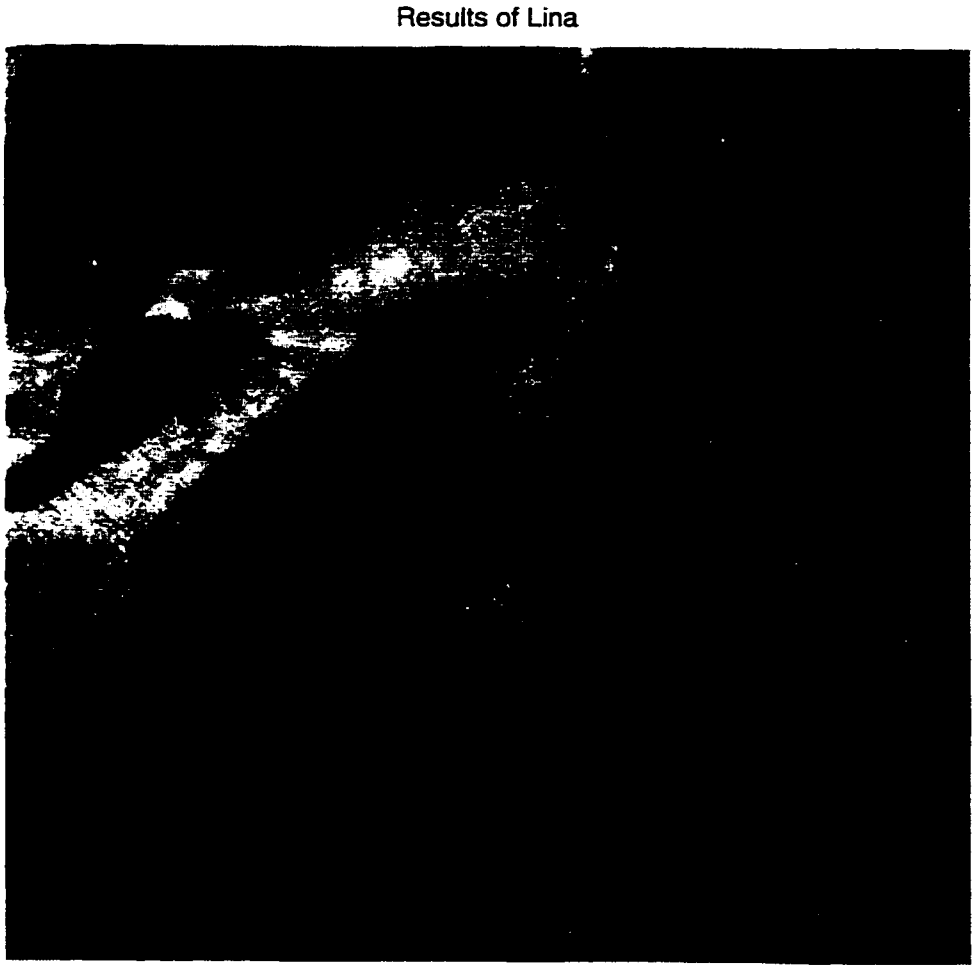

Figure 5.27: Results of the wavelet technique of Lina applied to the noisy Full Barbara image from Figure 5.25. SNR = 66.3294.

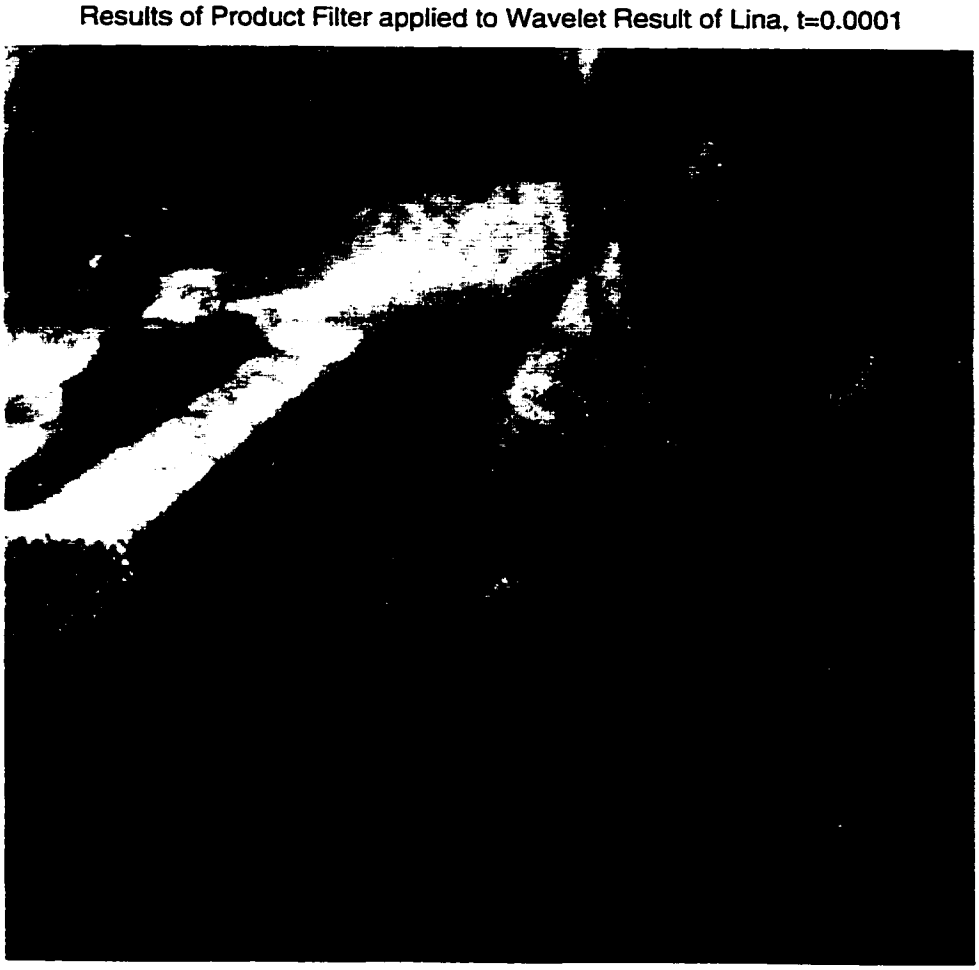

Figure 5.28: Results of the product filter algorithm applied to the Full Barbara wavelet results of Lina from Figure 5.27 with $p = 256$ and $t = 0.0001$. SNR = 53.7721.

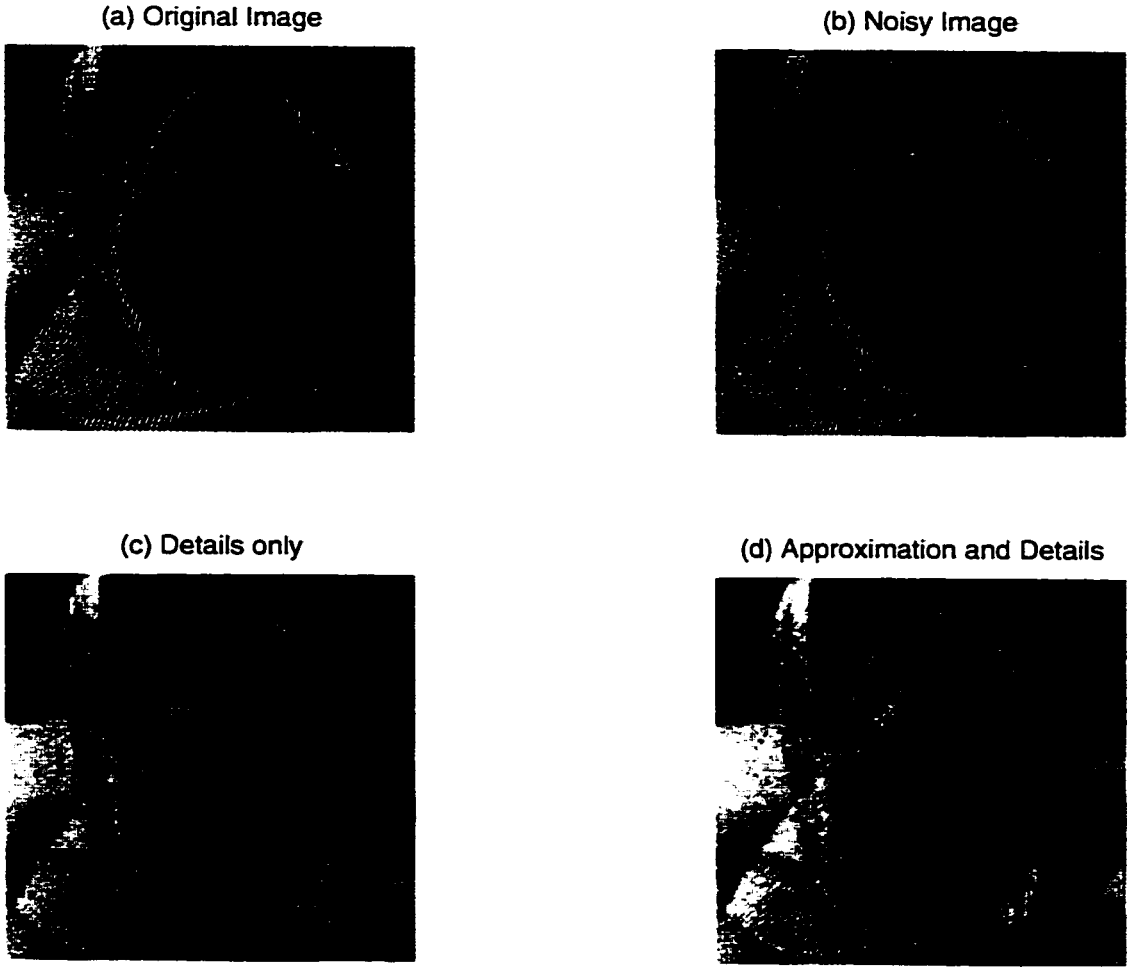

Figure 5.29: The product filter applied to the sub-images of the two-level Daubechies 3 decomposition. (a) is the original image. (b) is the noisy image with 50 rand(m, n) noise. SNR = 19.9012. (c) shows the results of the product filter applied to the detail sub-images only. SNR = 13.0223. (d) shows the results of the product filter applied to the approximation and detail sub-images. SNR = 17.0830.

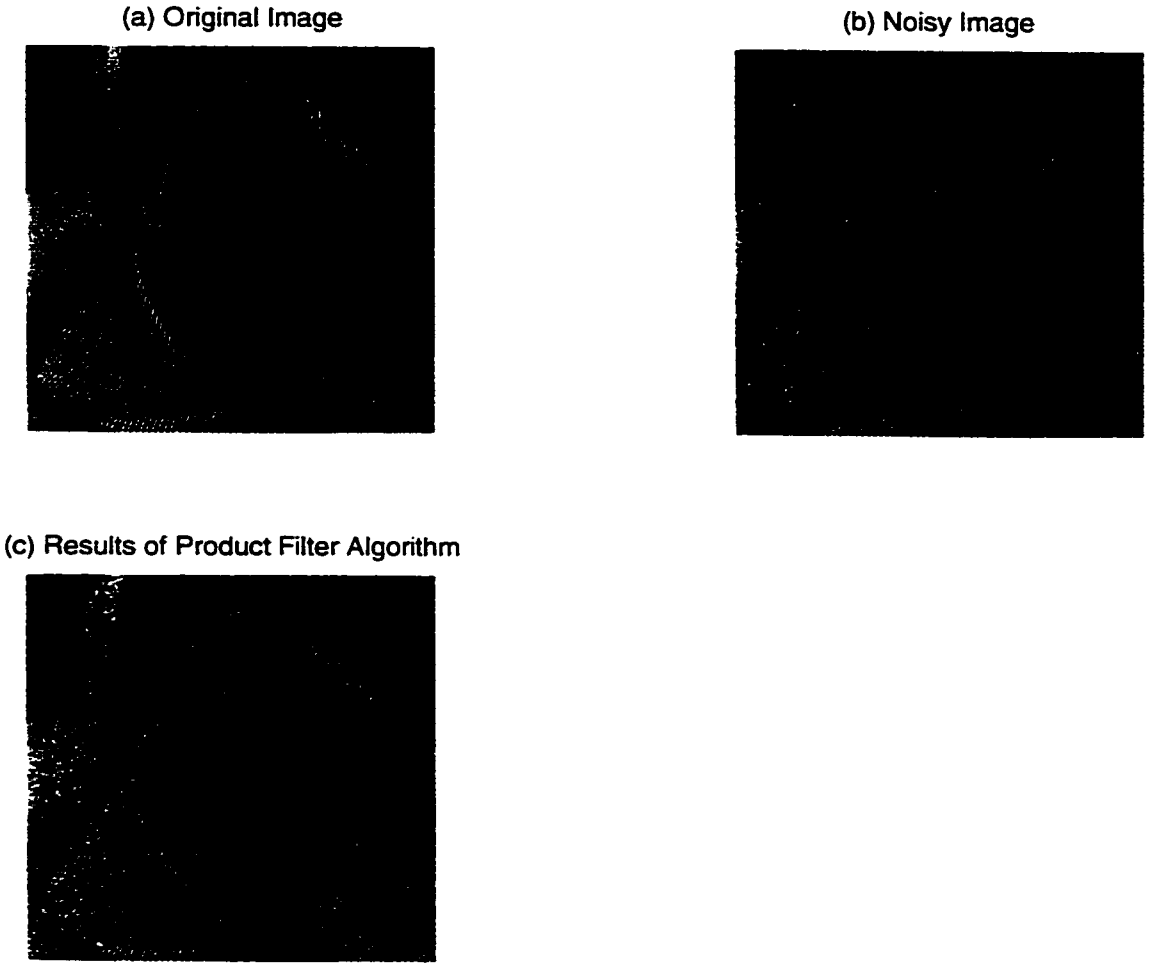

Figure 5.30: A comparison of the product filter with a wavelet technique of Portilla *et al* [36]. (a) is the original image. (b) is the image with Gaussian noise of the form $25\,\mathrm{randn}(m, n)$ added; SNR = 26.6409 and PSNR = 20.1735 dB. (c) shows the results of the product filter, applied with $t = 0.0003$ and $p = 256$; SNR = 51.7922 and PSNR = 23.0606 dB. Portilla *et al* achieved a PSNR of 28.57 dB, with superior image quality.

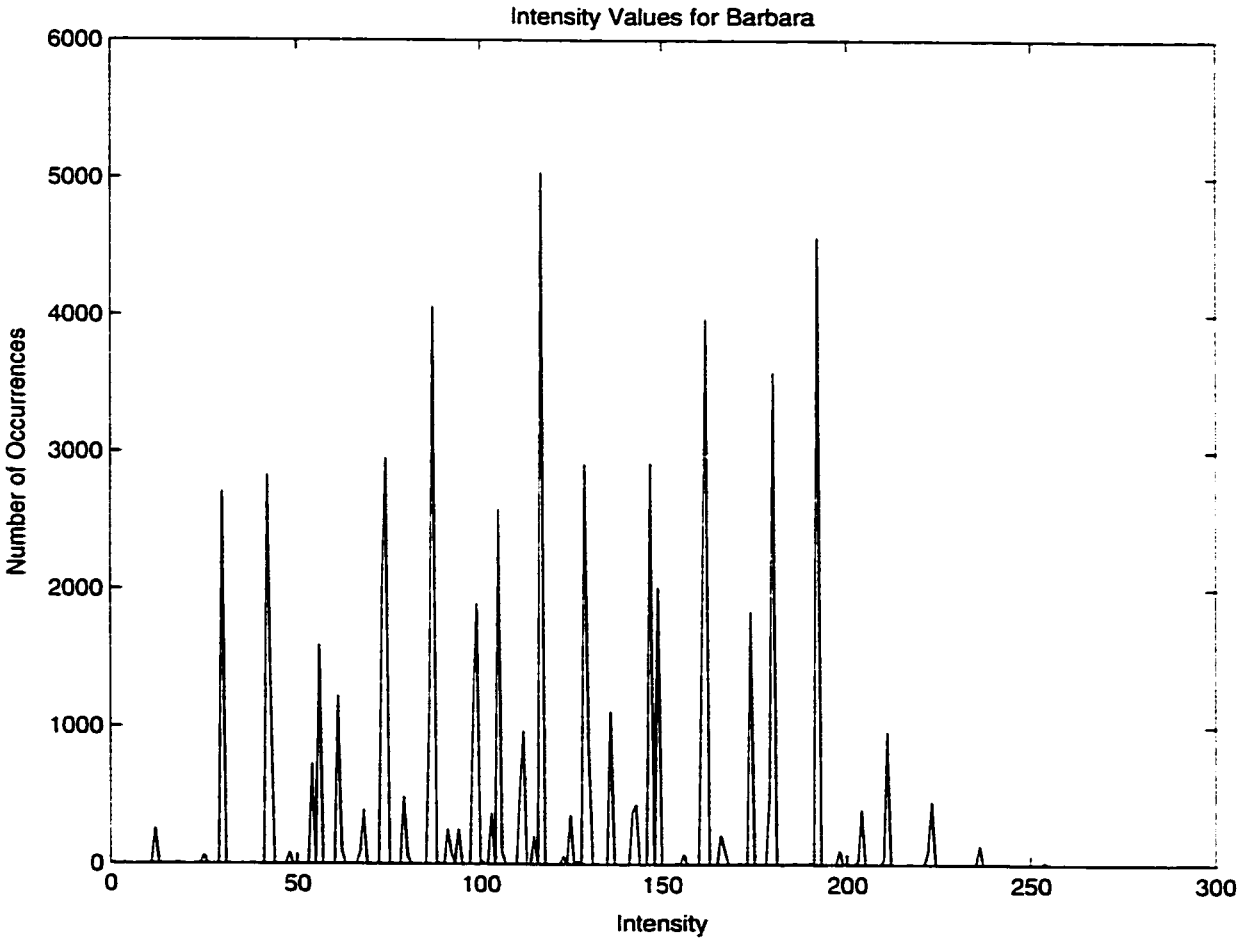

Figure 5.31: Graph of the number of occurrences of the intensity values of the Barbara image. The range is 0 to 254. with a mean of 118.8125.

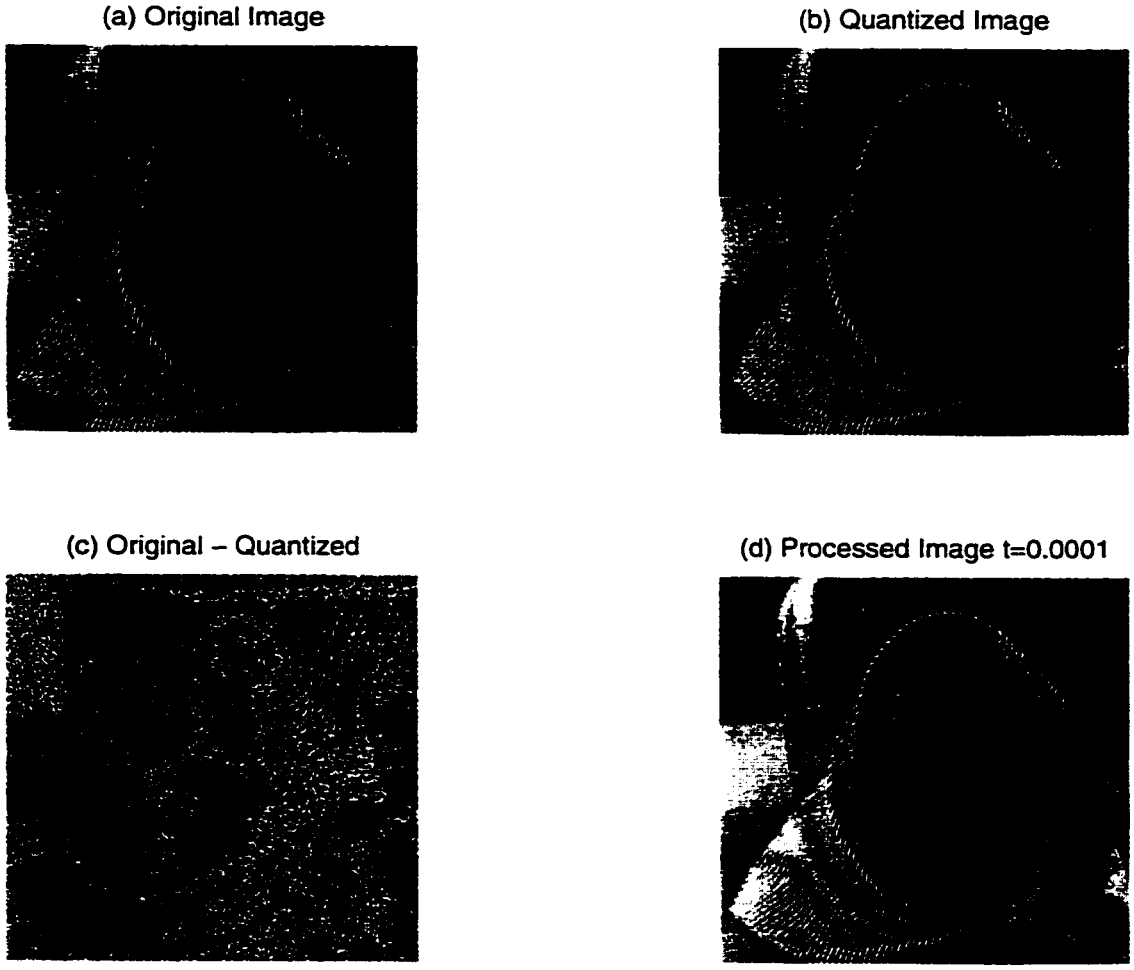

Figure 5.32: The Barbara image quantized into 8 levels. (a) is the original image. (b) is the quantized image, SNR = 45.1290. (c) shows the difference between the original and quantized images. (d) shows the results of the product filter with $p = 256$ and $t = 0.0001$, SNR = 44.1139.

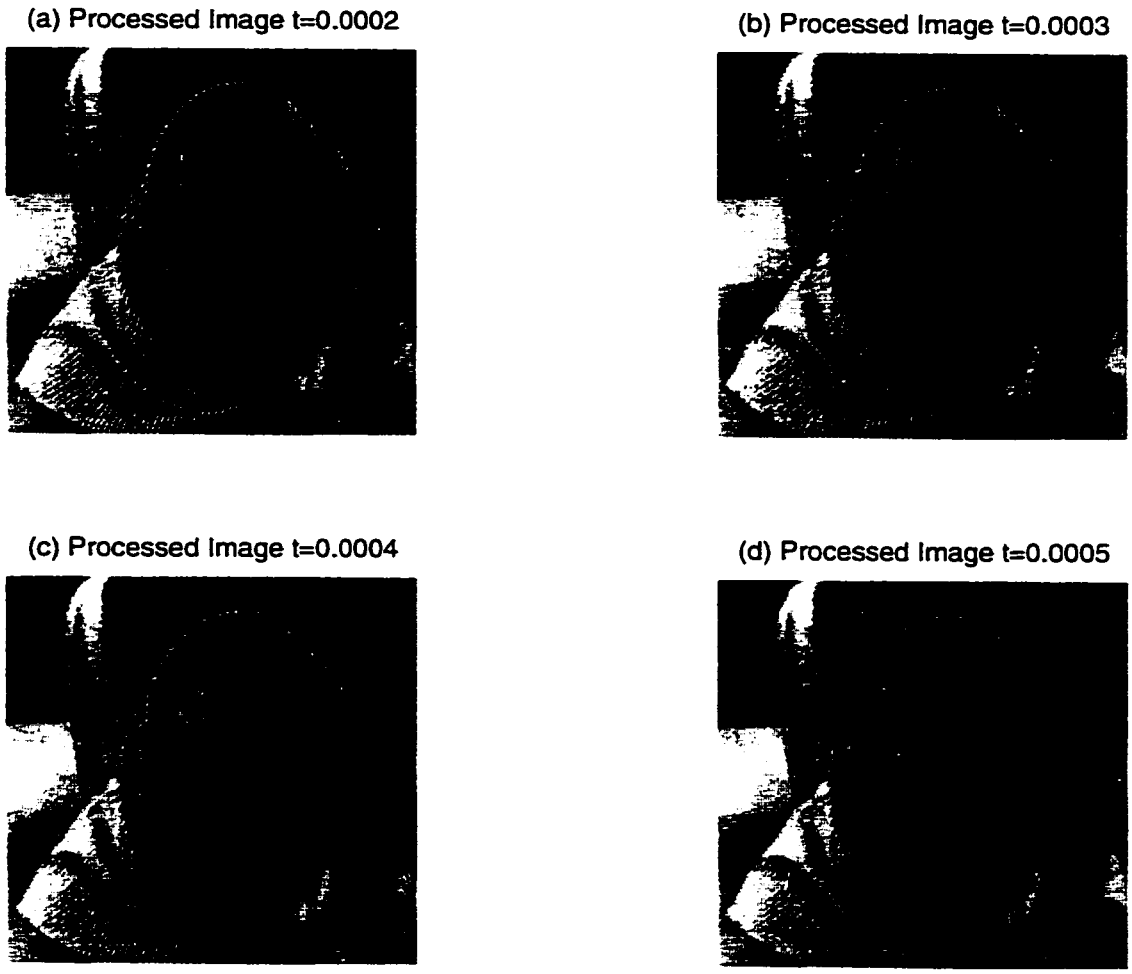

Figure 5.33: Results of the product filter applied to the quantized image from Figure 5.32(b). In all cases. $p = 256$. (a) shows the results of the product filter with $t = 0.0002$. SNR = 36.9073. (b) shows the results of the product filter with $t = 0.0003$. SNR = 31.8877. (c) shows the results of the product filter with $t = 0.0004$. SNR = 28.6494. (d) shows the results of the product filter with $t = 0.0005$. SNR = 26.4854.

Chapter 6

Third-Order PDE's

The usefulness of the diffusion equation.

$$u_t = u_{x_1x_1} + u_{x_2x_2}.$$

in image processing is well known [1, 2, 3, 4, 5, 6, 11, 13, 14, 19, 34, 35, 40, 43]. Application of the diffusion equation to a noisy image will result in a reduction of the noise, as the noise is diffused (smoothed) away. What has not been thoroughly investigated is the use of higher-order partial differential equations. A study of the practicality of PDE's of the form

$$u_t + a\,u_{x_1} + b\,u_{x_2} + c\,u_{x_1x_1} + d\,u_{x_2x_2} + e\,u_{x_1x_2} + f\,u_{x_1x_1x_1} + g\,u_{x_2x_2x_2} = 0.$$

where the coefficients (a to g) are constants, in the reduction of noise in images was undertaken. The rationale behind this study was the thought that the dispersive nature of these PDE's might prove useful in the reduction of noise – they might be able to disperse the transient noise away, while leaving the underlying image intact. In other words, the possibility of tuning the dispersion was studied. Different frequencies in the solution of a dispersive PDE will travel at different speeds. It is this attribute of the solution that could potentially be used to effect the noise reduction.

The PDE above was discretized in the following manner. A superscript is used to represent the time variable, so $u^k(i,j)$ represents $u(x_1, x_2, t)$ at the node $(x_1, x_2, t) =$

(i, j, k). For the time derivative,

$$u_t(i,j) = \frac{\partial u}{\partial t} \approx \frac{u^{n-1}(i,j) - u^n(i,j)}{\Delta t}$$

was used [37]. For the space derivatives, the following difference equations (all of order 4) were used:

$$\begin{aligned}
u^n_{x_1}(i,j) &\approx \frac{1}{12\Delta x_1}[-u^n(i+2,j) + 8u^n(i+1,j) - 8u^n(i-1,j) + u^n(i-2,j)] \\
u^n_{x_2}(i,j) &\approx \frac{1}{12\Delta x_2}[-u^n(i,j+2) + 8u^n(i,j+1) - 8u^n(i,j-1) + u^n(i,j-2)] \\
u^n_{x_1x_1}(i,j) &\approx \frac{1}{12(\Delta x_1)^2}[-u^n(i+2,j) + 16u^n(i+1,j) - 30u^n(i,j) \\
&\quad + 16u^n(i-1,j) - u^n(i-2,j)] \\
u^n_{x_2x_2}(i,j) &\approx \frac{1}{12(\Delta x_2)^2}[-u^n(i,j+2) + 16u^n(i,j+1) - 30u^n(i,j) \\
&\quad + 16u^n(i,j-1) - u^n(i,j-2)] \\
u^n_{x_1x_2}(i,j) &\approx \frac{1}{144\Delta x_1\Delta x_2}[u^n(i+2,j+2) - 8u^n(i+2,j+1) + 8u^n(i+2,j-1) \\
&\quad - u^n(i+2,j-2) - 8u^n(i+1,j+2) + 64u^n(i+1,j+1) \\
&\quad - 64u^n(i+1,j-1) + 8u^n(i+1,j-2) + 8u^n(i-1,j+2) \\
&\quad - 64u^n(i-1,j+1) + 64u^n(i-1,j-1) - 8u^n(i-1,j-2) \\
&\quad - u^n(i-2,j+2) + 8u^n(i-2,j+1) - 8u^n(i-2,j-1) \\
&\quad + u^n(i-2,j-2)] \\
u^n_{x_1x_1x_1}(i,j) &\approx \frac{1}{8(\Delta x_1)^3}[-u^n(i+3,j) + 8u^n(i+2,j) - 13u^n(i+1,j) \\
&\quad + 13u^n(i-1,j) - 8u^n(i-2,j) + u^n(i-3,j)] \\
u^n_{x_2x_2x_2}(i,j) &\approx \frac{1}{8(\Delta x_2)^3}[-u^n(i,j+3) + 8u^n(i,j+2) - 13u^n(i,j+1) \\
&\quad + 13u^n(i,j-1) - 8u^n(i,j-2) + u^n(i,j-3)].
\end{aligned}$$

The formulas for u_{x_1}, u_{x_2}, $u_{x_1x_1}$ and $u_{x_2x_2}$ were found in the literature [25]; those for $u_{x_1x_2}$, $u_{x_1x_1x_1}$ and $u_{x_2x_2x_2}$ were derived. Since the images that these PDE's were applied to were of size 256×256, the values $\Delta x_1 = \Delta x_2 = 1/256$ were used.

The time evolution of the PDE is described by:

$$u^{n+1}(i,j) = u^n(i,j) - \Delta t\,[a\,u^n_{x_1}(i,j) + b\,u^n_{x_2}(i,j) + c\,u^n_{x_1x_1}(i,j) + d\,u^n_{x_2x_2}(i,j) + e\,u^n_{x_1x_2}(i,j) + f\,u^n_{x_1x_1x_1}(i,j) + g\,u^n_{x_2x_2x_2}(i,j)].$$

The PDE is applied to the image by taking the image $\{(x_1, x_2, I(x_1, x_2))\}$ as the initial data, *i.e.* $u^0(i,j) = I(i,j)$, and applying the above difference scheme. The boundary of the image was dealt with in a very simple way - the difference scheme was not applied to a three-pixel wide border around the outside edges of the image. So, in the calculations, i and j were restricted to be between 4 and 253.

The original intention for this investigation was to study the application of all of the PDE's of the form above, where each of the coefficients varied from -10 to 10, in steps of 0.1. However, this plan was too ambitious and had to be scaled-down. Several hundreds of the PDE's above were applied to the Barbara and Cameraman images by varying the values of the coefficients (a to g) over many ranges (typically in the range -5 to 5, with steps of 0.2, 0.5 or 1), for various values of Δt (typically from $(\Delta x_1)^3/16$ to $(\Delta x_1)^3$) and differing numbers of iterations of the above difference equation (typically from 1 to 15). Δt was kept small in order to ensure the numerical stability of the results. Stability was tested experimentally and the range quoted above was found to be well within the interval of stability.

Because of the impracticality of monitoring the results of hundreds of PDE's applied to images and visually comparing the results, a numerical measuring method was used. The single SNR value is not entirely reliable, so three numerical measures were devised. Since the original (pre-noise) image was known, represented by a matrix A, its norm (in MATLAB, `norm(A)` is the largest singular value of A) could be calculated and its largest element determined. In MATLAB, `max(A)` will give the largest elements in each column of the matrix A. `max(max(A))` will give the largest of these, which is the largest element

of the matrix. Also, the average absolute difference between each element's intensity and the intensity values of its eight nearest neighbours was calculated (represented by avg(A)). After each PDE was applied to the noisy image, these three values were calculated for the resulting processed image, represented by the matrix u. Then the ratios were taken with the values for the original image, to define the following three values:

$$\begin{aligned} \text{unorm} &= \text{norm}(u)/\text{norm}(A) \\ \text{umax} &= \max(\max(u))/\max(\max(A)) \\ \text{uavg} &= \text{avg}(u)/\text{avg}(A). \end{aligned}$$

It was thought that the closer these values were to one, the better the processed image should be. These ratios would be close to one when the processed image had a norm, largest element and average nearest neighbour difference very similar to the original (pre-noise) image. In that case, it was thought that the processed image would better resemble the original. It was also thought that these three ratios might prove more insightful than a single SNR value, as they measure three different things and hence provide three avenues of comparison between the processed and original images. However, visual comparison is still the best measurement of relative image quality. If any of the three ratio values were less than 1, the reciprocal was used for comparisons. The PDE's that gave the lowest values of the sum and product of the three ratios were identified. Consistently, the *best* PDE's were the ones with $f = g = 0$ and $c = d < 0$, *i.e.* equations of the diffusion type. It was also found that PDE's with the third-order terms, $u_{x_1x_1x_1}$ and $u_{x_2x_2x_2}$, are not particularly useful in de-noising images.

Figures 6.1 to 6.4 present examples of these calculations, using a *good* second-order partial differential equation

$$u_t + u_{x_1} + u_{x_2} - u_{x_1x_1} - u_{x_2x_2} - u_{x_1x_2} = 0.$$

and a *bad* third-order partial differential equation

$$u_t + u_{x_1} + u_{x_2} + u_{x_1x_1} + u_{x_2x_2} + u_{x_1x_2} - u_{x_1x_1x_1} - u_{x_2x_2x_2} = 0.$$

Here, *good* and *bad* refer to the de-noising ability of the PDE. Both of these PDE's were applied to the Barbara and Cameraman images with random noise of intensity 50 and with salt and pepper noise affecting 5% of the pixels. In all of the calculations, $\Delta x_1 = \Delta x_2 = 1/256$ (since both of the images are 256×256). $\Delta t = (\Delta x_1)^3$ (for stability) and there was one iteration only. Figure 6.1 presents Barbara with random noise; Figure 6.2 is Barbara with salt and pepper; Figure 6.3 is the Cameraman with random noise and Figure 6.4 is the Cameraman with salt and pepper. Numerical results from these Figures are presented in Table 6.1.

Table 6.1: A numerical comparison of the results of second and third order PDE's applied to the Barbara and Cameraman images. The ratios unorm, umax and uavg are defined in the text above. Formula (1.8) for SNR is given in Chapter 1.

Figure	Image	PDE Order	unorm	umax	uavg	SNR
6.1	Barbara	2	1.1976	1.1719	1.2776	20.1552
	(SNR $\approx$ 20)	3	1.2018	3.1574	5.0799	1.5055
6.2	(SNR $\approx$ 20)	2	0.9510	0.9956	1.3810	21.1242
		3	0.9569	4.0481	6.4435	0.9095
6.3	Cameraman	2	1.1931	1.1846	2.1472	21.6694
	(SNR $\approx$ 21.5)	3	1.1952	3.0940	8.1899	2.9241
6.4	(SNR $\approx$ 20)	2	0.9508	0.9958	1.9770	20.9097
		3	0.9542	3.5866	11.2715	1.2204

It is clear from the Figures and the numerical results in the Table that the third-order equation does more harm than good. Instead of de-noising the image (by smoothing, as the second-order PDE does), it actually makes the image poorer. Numerically, the ratios unorm, umax and uavg are better (*i.e.* closer to 1) and the signal-to-noise ratios, SNR, are much higher for the second-order PDE than those for the third-order.

Visually, the second-order PDE clearly does a superior job at both reducing (smoothing) the noise and preserving the image detail, whereas the third-order PDE seems to amplify the noise and more detail is lost. However, the third-order PDE appears to enhance some texture structures. The texture of the scarf in the Barbara image seems to be more pronounced in the images that were processed by the third-order PDE than those processed by the second-order PDE.

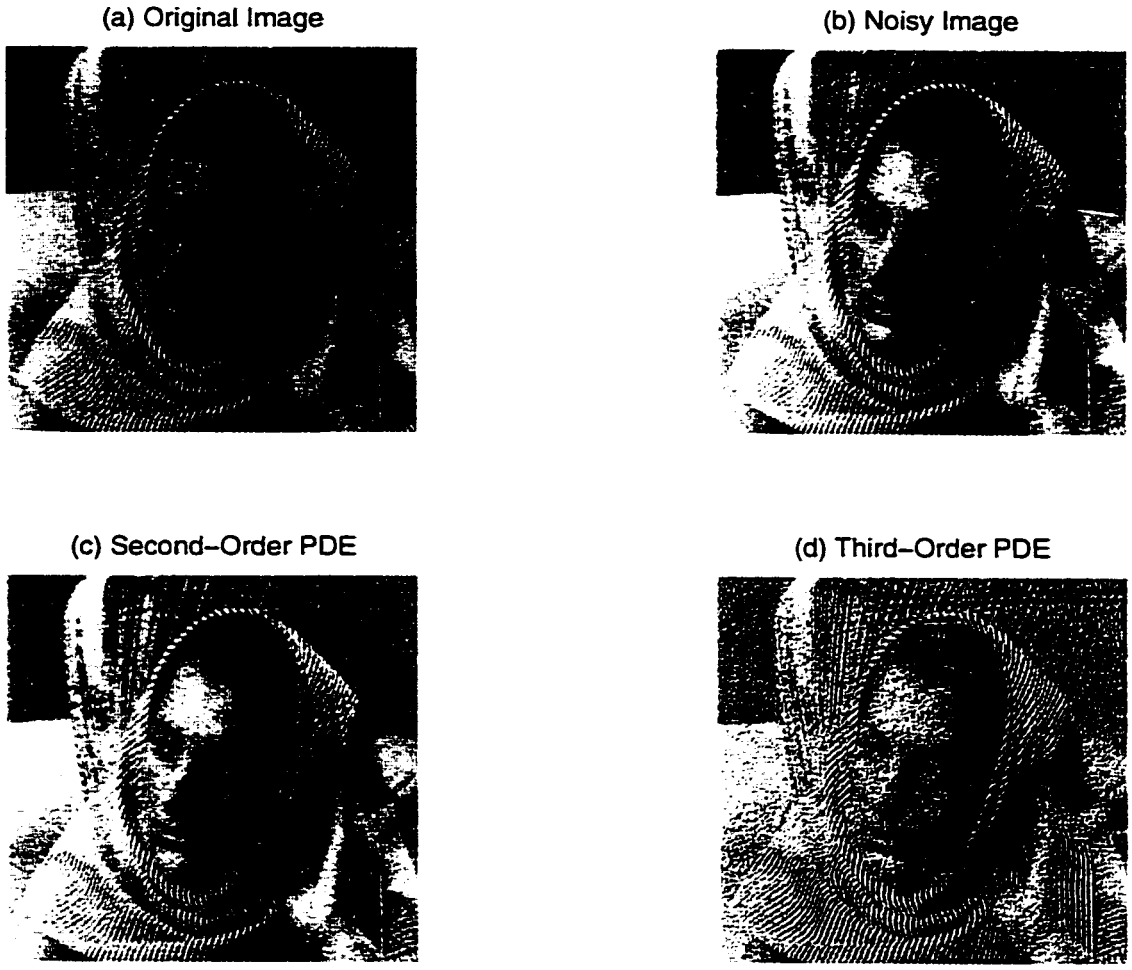

Figure 6.1: The Barbara image with random noise. $50\,\mathrm{rand}(m.n)$. (a) is the original (pre-noise) image. (b) is the image with the noise added. (c) is the result of the second-order PDE applied to (b). (d) is the result of the third-order PDE applied to (b).

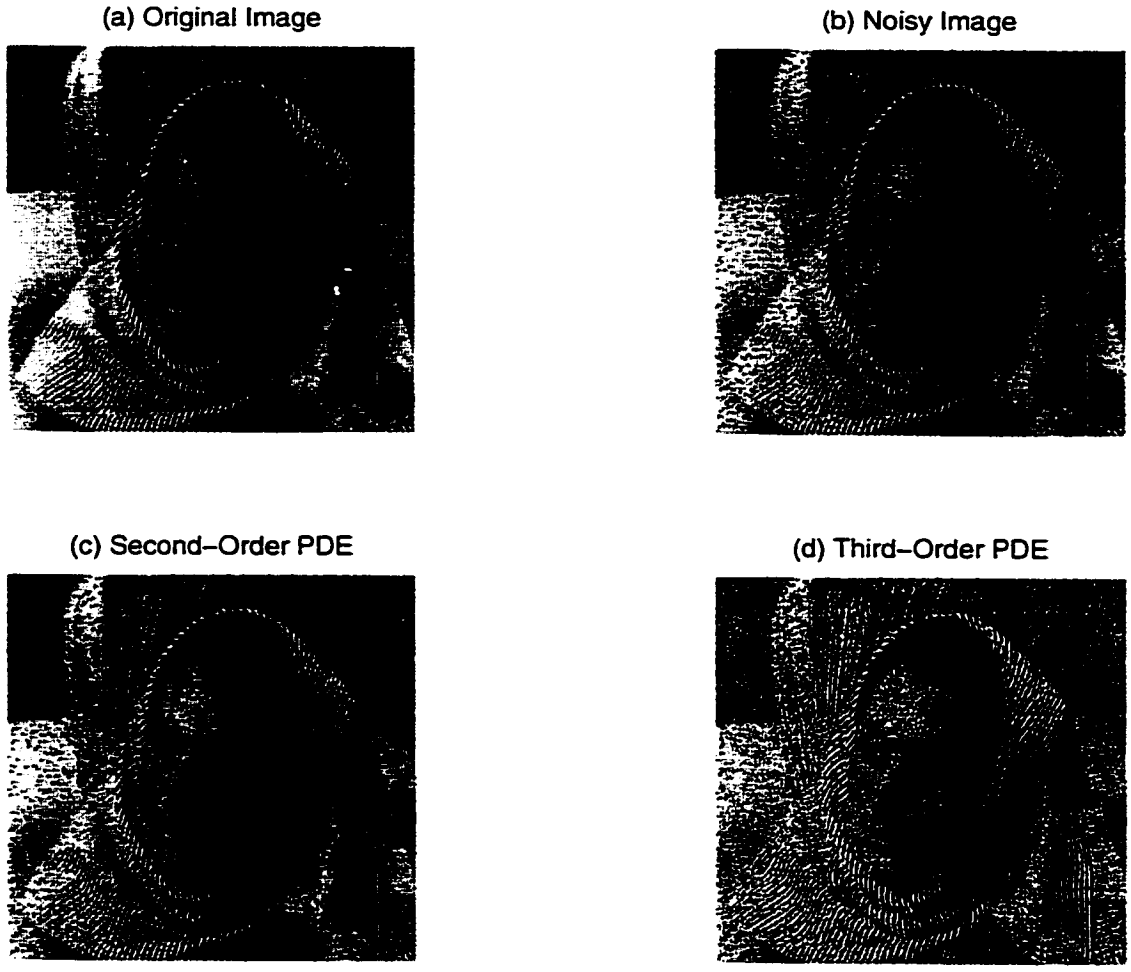

Figure 6.2: The Barbara image with 5% salt and pepper noise. (a) is the original (pre-noise) image. (b) is the image with the noise added. (c) is the result of the second-order PDE applied to (b). (d) is the result of the third-order PDE applied to (b).

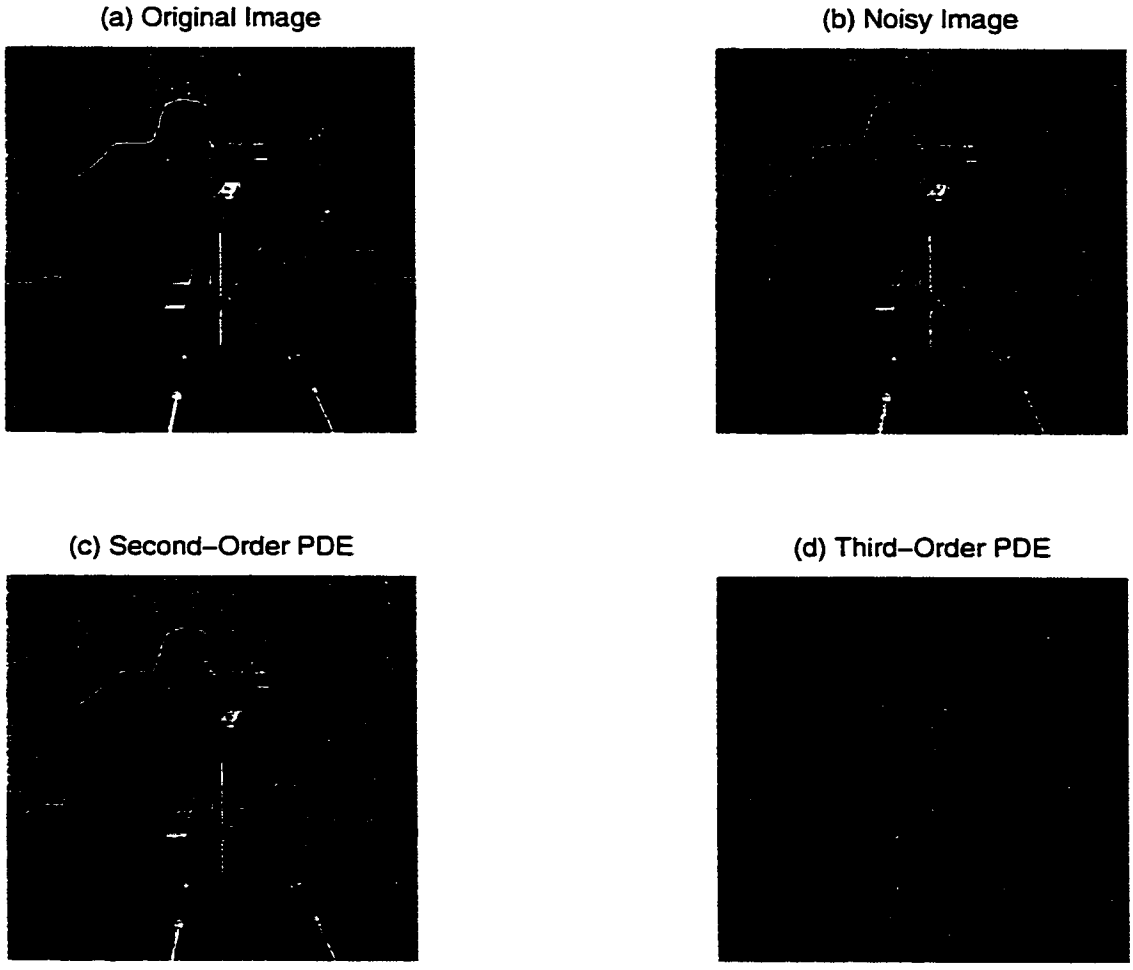

Figure 6.3: The Cameraman image with random noise. 50 rand(m, n). (a) is the original (pre-noise) image. (b) is the image with the noise added. (c) is the result of the second-order PDE applied to (b). (d) is the result of the third-order PDE applied to (b).

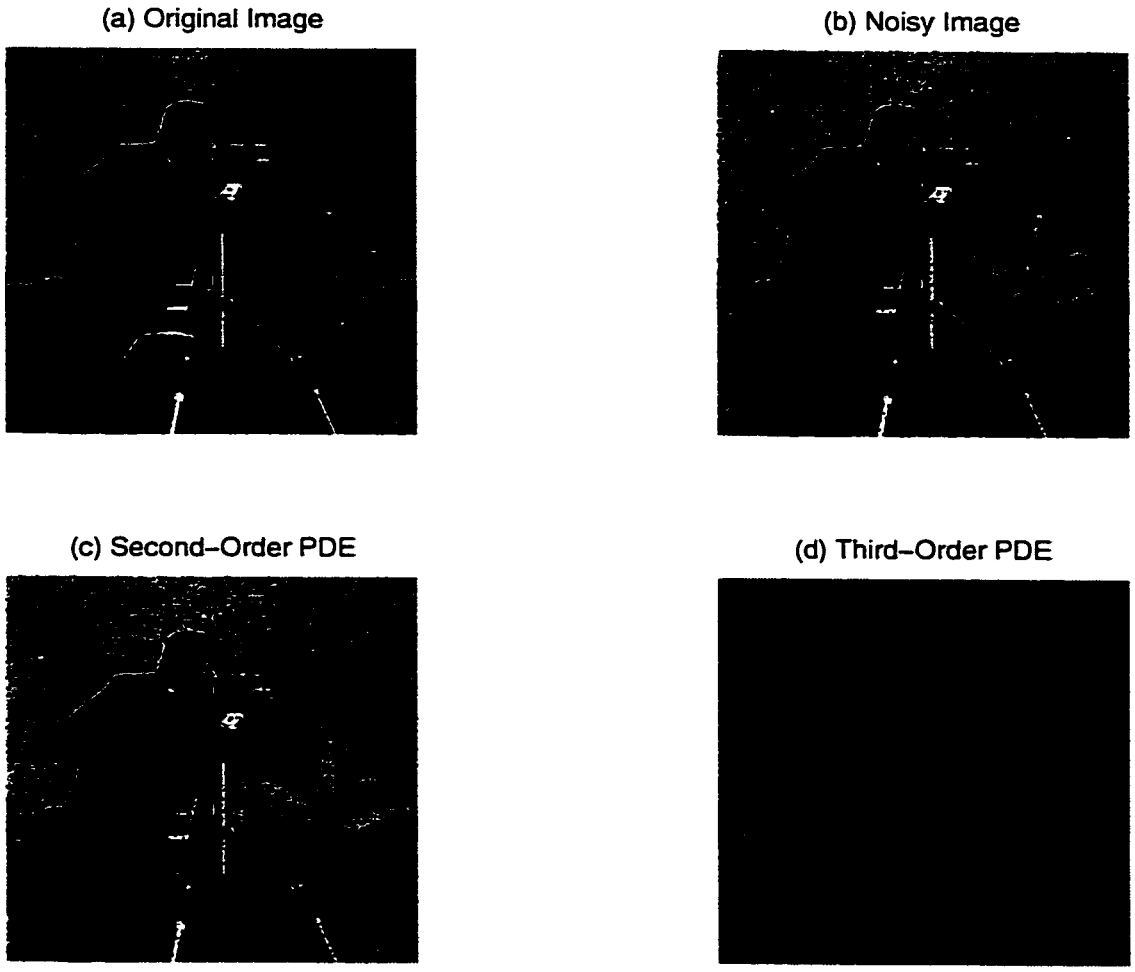

Figure 6.4: The Cameraman image with 5% salt and pepper noise. (a) is the original (pre-noise) image. (b) is the image with the noise added. (c) is the result of the second-order PDE applied to (b). (d) is the result of the third-order PDE applied to (b).

Chapter 7

Conclusions

The tight frame wavelets constructed in Chapter 2 were found to be successful in localizing singularities in images. Once a singularity has been localized, it may be effectively removed from the image. The localization technique could then be used in image restoration.

The product filter algorithm method for reducing noise is quick, simple to implement, does smooth noise and preserves image details. In de-noising ability, the product filter is comparable to MATLAB's **averaging** filter (but superior to MATLAB in detail preservation). The wavelet techniques were found to be superior to the product filter in noise reduction.

The third-order PDE's were found to be detrimental in de-noising as they enhance the noise, rather than reduce it. But, the third-order PDE's do seem to enhance some texture structures.

Bibliography

[1] S. T. Acton. *Multi-grid anisotropic diffusion.* IEEE Trans. Image Processsing. **7**(3) (March 1998) 280–291.

[2] L. Alvarez. F. Guichard. P.-L. Lions and J.-M. Morel. *Axiomes et équations fondamentales du traitement d'images (Analyse multiéchelle et E.D.P.).* C.R. Acad. Sci. Paris. **315** série I (1992) 135–138.

[3] L. Alvarez. F. Guichard. P.-L. Lions and J.-M. Morel. *Axiomatisation et nouveaux opérateurs de la morphologie mathématique.* C.R. Acad. Sci. Paris. **315** série I (1992) 265–268.

[4] L. Alvarez. F. Guichard. P.-L. Lions and J.-M. Morel. *Axioms and fundamental equations of image processing.* Arch. Rational Mech. Anal.. **123** (1993) 199–257.

[5] L. Alvarez. P.-L. Lions and J.-M. Morel. *Image selective smoothing and edge detection by nonlinear diffusion II.* SIAM J. Numer. Anal.. **29**(3) (1992) 845–866.

[6] L. Alvarez and L. Mazorra. *Signal and image restoration using shock-filters and anisotropic diffusion.* SIAM J. Numer. Anal.. **31**(2) (1994) 590–605.

[7] R. Ashino. C. Heil. M. Nagase and R. Vaillancourt. *Multiwavelets. pseudodifferential operators and microlocal analysis.* in Proceedings of the International Con-

ference on Geometry. Analysis and Applications. 21-24 August 2000. Varanasi. India. R. S. Pathak ed.. World Scientific. Singapore. pp. 293–302.

[8] R. Ashino. C. Heil. M. Nagase and R. Vaillancourt. *Microlocal filtering with multiwavelets.* Computers Math. Applic.. **41**(1-2) (2001) 111–133.

[9] J. J. Benedetto and O. M. Treiber. *Wavelet frames: Multiresolution analysis and extension principles.* in Wavelet Transforms and Time-Frequency Signal Analysis. Ed. L. Debnath. Birkhäuser. Boston. 2001.

[10] J. J. Benedetto. *The theory of multiresolution analysis and applications to filter banks.* Applied Computational Harm. Anal. **5** (1998) 389–427.

[11] M. J. Black. G. Sapiro. D. H. Marimont and D. Heeger. *Robust anisotropic diffusion.* IEEE Trans. Image Processsing. **7**(3) (March 1998) 421–432.

[12] A. P. Calderón and R. Vaillancourt. *On a class of bounded pseudo-differential operators.* Proc. Nat. Acad. Sci. USA **69** (1972) 1185–1187.

[13] V. Casselles. J.-M. Morel. G. Sapiro and A. Tannenbaum. *Introduction to the special issue on partial differential equations and geometry-driven diffusion in image processing.* IEEE Trans. Image Processsing. **7**(3) (March 1998) 269–273.

[14] F. Catté. P.-L. Lions. J.-M. Morel and T. Coll. *Image selective smoothing and edge detection by nonlinear diffusion.* SIAM J. Numer. Anal.. **29**(1) (1992) 182–193.

[15] F. Cao. *Approximation d'équations d'évolution paraboliques géométriques par des schémas invariants. une approche axiomatique de l'interpolation: applications au traitement d'images.* Ph.D. thesis. École Normale Supérieure de Cachan. 2000.

[16] I. Daubechies. A. Grossmann and Y. Meyer. *Painless nonorthogonal expansions.* J. Math. Phys. **27**. No. 5 (1986) 1271–1283.

[17] I. Daubechies. *Ten lectures on wavelets.* SIAM. Philadelphia. PA. 1992.

[18] S. J. Desjardins and R. Vaillancourt. *Image de-noising by a multi-directional diffusion equation.* to be published in Mathematical Reports of the Academy of Science of the Royal Society of Canada.

[19] F. L. Fontaine and S. Basu. *Wavelet-based solution to anisotropic diffusion equation for edge detection.* Intl. J. Imaging Systems and Technology. **9**(5 Special Issue SI) (1998) 356–368.

[20] M. Frazier. G. Garrigós. K. Wang and G. Weiss. *A characterization of functions that generate wavelets and related expansion.* The Journal of Fourier Analysis and Applications. 1997. Special Issue (1997) 883–906.

[21] M. Frazier and B. Jawerth. *A discrete transform and decomposition of distribution spaces.* J. Funct. Anal.. **93** (1990). 34-170.

[22] M. Frazier. B. Jawerth and G. Weiss. *Littlewood–Paley theory and the study of function spaces.* CBM. **79**. Amer. Math. Soc.. Providence RI. 1991.

[23] W. T. Freeman and E. H. Adelson. *The design and use of steerable filters.* IEEE Trans. Pattern Anal. Mach. Intel.. **13**(9) (Sept. 1991) 891–906.

[24] K. O. Friedrichs. *Pseudo-differential operators. An introduction.* Notes prepared with the assistance of R. Vaillancourt. revised edition. April 1970. Courant Institute of Mathematical Sciences. New York University. New York. 1970

[25] C. F. Gerald and P. O. Wheatley. *Applied numerical analysis.* 5th edition. Addison-Wesley. Reading MA. 1994.

[26] K. Gröchenig. *Foundations of time-frequency analysis.* Birkhäuser. Boston. 2001.

[27] D. Han and D. R. Larson. *Frames, bases and group representations.* Memoirs of the Amer. Math. Soc., **147**, No. 697, (Sept. 2000).

[28] C. Heil and D. F. Walnut. *Continuous and discrete wavelet transforms.* SIAM Rev., **31**(4) (1989), 628–666.

[29] E. Hernández and G. Weiss. *A first course on wavelets.* CRC Press, Boca Raton, FL, 1996.

[30] J. S. Lim. *Two-dimensional signal and image processing.* Prentice Hall PTR, Englewood Cliffs NJ 07632, 1990.

[31] F. Malgouyres. *Augmentation de résolution d'images digitales: théorie variationnelle et applications.* Ph.D. thesis, École Normale Supérieure de Cachan, 2000.

[32] S. Mallat. *Applied mathematics meets signal processing.* Doc. Math. J. DMV Extra Volume ICM I (1998) 319–338.

[33] S. Mallat. *A wavelet tour of signal processing.* 2nd edition, Academic Press, San Diego, CA, 1999.

[34] P. Perona. *Orientation diffusions.* IEEE Trans. Image Processsing, **7**(3) (March 1998) 457–467.

[35] P. Perona and J. Malik. *Scale-space and edge detection using anisotropic diffusion.* IEEE Trans. Pattern Anal. Mach. Intel., **12**(7) (July 1990) 629–639.

[36] J. Portilla, V. Strela, M. J. Wainwright and E. P. Simoncelli. *Adaptive Wiener denoising using a Gaussian scale mixture model in the wavelet domain.* Proc. 8th IEEE Int. Conf. Image Processing, Thessaloniki, Greece, Oct. 2001.

[37] G. Strang, *Introduction to applied mathematics*. Wellesly-Cambridge Press. Wellesley MA 02181. 1986.

[38] G. Strang and T. Nguyen. *Wavelets and filter banks*. rev. ed.. Wellesly-Cambridge Press. Wellesley MA 02181. 1997.

[39] V. Strela. P. N. Heller. G. Strang. P. Topiwala and C. Heil. *The application of multiwavelet filterbanks to image processing*. IEEE Trans. Image Processsing. **8**(4) (April 1999) 548–563.

[40] F. Torkamani-Azar and K. E. Tait. *Image recovery using the anisotropic diffusion equation*. IEEE Trans. Image Processing. **5**(11) (Nov. 1996) 1573–1578.

[41] R. Vaillancourt. *A Strong form of Yamaguti and Nogi's stability theorem for Friedrichs' scheme*. Publ. RIMS. Kyoto. *5* (1969) 113–117.

[42] R. Vaillancourt. *On the stability of Friedrichs' scheme and the modified Lax-Wendoff scheme*. Math. Comput. **24** (1970) 767–770.

[43] J. Weickert. B. M. ter Haar Romeny and M. A. Viergever. *Efficient and reliable schemes for nonlinear diffusion filtering*. IEEE Trans. Image Processsing. **7**(3) (March 1998) 398–410.