WAVELET AND RIDGELET TRANSFORMS FOR PATTERN RECOGNITION AND DENOISING

Guangyi Chen

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Doctor of Philosophy
Concordia University
Montréal, Québec, Canada

April 2004

Your file Votre référence
ISBN: 0-612-90380-X
Our file Notre référence
ISBN: 0-612-90380-X

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Guangyi Chen**
Entitled: **Wavelet and Ridgelet Transforms for Pattern Recognition and Denoising**

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (Computer Science)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. V. Ramachandran, Chair

Dr. J.-M. Lina, External Examiner

Dr. W. Lynch, External to Program

Dr. C.Y. Suen, Examiner

Dr. S.P. Mudur, Examiner

Dr. T.D. Bui, Thesis Co-Supervisor

Dr. A. Krzyzak, Thesis Co-Supervisor

Approved by _____
for Dr. T.D. Bui, Graduate Program Director

APR 23 2004
2004

Dr. N. Esmail, Dean
Faculty of Engineering & Computer Science

Abstract

Wavelet and Ridgelet Transforms for Pattern Recognition and Denoising

Guangyi Chen

The application of wavelet and ridgelet transforms in pattern recognition is still in its infancy; while their use in denoising has been a very hot topic in recent years. The aim of this thesis is to study these two important problems. In the area of pattern recognition, we develop a handwritten numeral recognition descriptor using multiwavelets and neural networks. We perform multiwavelet orthonormal shell expansion on the contour to get several resolution levels and the average. Then we use the shell coefficients as features to input into a feed-forward neural network to recognize the handwritten numerals. We also present two novel descriptors for feature extraction by using ridgelets. Fourier spectrum and wavelet cycle-spinning are used to achieve rotational invariance. The descriptors are very robust to noise even when the noise level is high. Experimental results show that the new descriptors are excellent choices for pattern recognition.

In the area of denoising, we first study multiwavelet thresholding by incorporating neighbouring coefficients. Experimental results show that this approach outperforms neighbour single wavelet denoising for some standard test signals and real life images. Then, we propose a wavelet image thresholding scheme by incorporating neighbouring coefficients. Experimental results show that translation invariant (TI) denoising with neighbour dependency is better than *VisuShrink* and the TI denoising method developed by Yu et al. Finally, we propose to use Simulated Annealing to find both the customized wavelet filters and the customized threshold for the given noisy image at the same time. The results we obtained are promising compared to other results published in the literature.

Acknowledgements

I would like to thank my supervisors, Dr. T. D. Bui and Dr. A. Krzyżak, for their financial support, advice, and encouragement during this work. Without their help, this work could not have been completed. Thank you very much! Dr. Bui introduced me to the field of wavelets back in 1996 when I was doing my Master's thesis at Concordia University. I have been working on this fascinating topic since.

I would further like to express my gratitude to my sponsors, Natural Sciences and Engineering Research Council of Canada (NSERC), Fonds Quebecois de la Recherche sur la Nature et les Technologies (FQRNT) of Quebec, and Concordia University, for their generous fellowships during my graduate study at Concordia University.

Finally, special thanks go to my wife, my son and my daughter for their understanding.

Contents

[1]This work was published in the *Pattern Recognition* Journal [14]. We believe this is the first published paper using multiwavelet transforms in pattern recognition.

[2]This work was published in *IEEE Signal Processing Letters* [12].

List of Figures

List of Tables

Chapter 1

Preliminaries

In this Chapter, we will review some basic aspects of the Fourier transform, single wavelet transform, multiwavelet transform, ridgelet transform, and neural network.

1.1 The Fourier Transform

The Fourier transform's utility lies in its ability to analyse a signal in the time domain for its frequency content. The transform works by first transforming a function in the time domain into a function in the frequency domain. The signal can then be analysed for its frequency content because the Fourier coefficients of the transformed function represent the contribution of each sine and cosine function at each frequency. An inverse Fourier transform does just what you expect, transform data from the frequency domain into the time domain.

1.1.1 The Continuous Fourier Transform

Let $f(x)$ be a continuous function of a real variable in $L^1(R)$. The Fourier transform of $f(x)$ is defined by the equation

$$\hat{f}(u) = \int_{-\infty}^{+\infty} f(x) e^{-iux} dx \tag{1}$$

Given $\hat{f}(u)$, $f(x)$ can be obtained by the inverse Fourier Transform

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{f}(u) e^{iux} du \tag{2}$$

The Fourier transform can be easily extended to a function $f(x, y)$ of two variables:

$$\hat{f}(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) e^{-i(ux+vy)} dx dy \tag{3}$$

and

$$f(x, y) = \frac{1}{4\pi^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \hat{f}(u, v) e^{i(ux+vy)} du dv \tag{4}$$

where u and v are the frequency variables.

1.1.2 The Discrete Fourier Transforms

The discrete Fourier transform(DFT) estimates the Fourier transform of a function from a finite number of its sample points. The sample points are supposed to be typical of what the signal looks like at all other times. Suppose that a continuous function $f(x)$ is discretized into a sequence of $\{f(x_0), f(x_0 + \triangle x), \ldots, f(x_0 + (n-1)\triangle x)\}$ by taking n samples $\triangle x$ apart. We may define

$$f(x) = f(x_0 + x\triangle x)$$

where x now assumes the discrete values 0, 1, ..., $n-1$. With this notation in mind, the discrete Fourier transform pair that applies to sampled functions is given by

$$\hat{f}(u) = \frac{1}{n} \sum_{x=0}^{n-1} f(x) e^{-iux/n} \tag{5}$$

for $u = 0, 1, 2, \ldots, n-1$, and

$$f(x) = \sum_{u=0}^{n-1} \hat{f}(u) e^{iux/n} \tag{6}$$

for $x = 0, 1, 2, \ldots, n-1$.

In the two-variable case the discrete Fourier transform pair is given by the equations

$$\hat{f}(u,v) = \frac{1}{n^2} \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} f(x,y) e^{-i(ux+vy)/n} \tag{7}$$

for $u = 0, 1, 2, \ldots, n-1$, $v = 0, 1, 2, \ldots, n-1$, and

$$f(x,y) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \hat{f}(u,v) e^{i(ux+vy)/n} \tag{8}$$

for $x = 0, 1, 2, \ldots, n-1$, $y = 0, 1, 2, \ldots, n-1$.

1.1.3 The Fast Fourier Transform

To approximate a function by samples, and to approximate the Fourier integral by the discrete Fourier transform, requires applying a matrix whose order is the number of sample points n. Since multiplying a matrix by a vector costs on the order of $O(n^2)$ arithmetic operations, the problem gets quickly worse as the number of sample points increases. However, if the samples are uniformly spaced, then the Fourier transform matrix can be factored into a product of just a few sparse matrices, and the resulting factors can be applied to a vector in a total of order $O(n \log_2 n)$ arithmetic operations. This is the so-called Fast Fourier Transform(FFT).

1.1.4 Some Properties of the Fourier Transform

In this section we review the properties of the Fourier transform which will be of value in subsequent discussions.

Translation: The translation properties of the Fourier transform pair with one variable or two variables are given by

$$f(x)e^{iu_0 x/n} \Longleftrightarrow \hat{f}(u - u_0) \tag{9}$$

$$f(x - x_0) \Longleftrightarrow \hat{f}(u)e^{-iux_0/n} \tag{10}$$

and

$$f(x, y)e^{i(u_0x+v_0y)/n} \Longleftrightarrow \hat{f}(u - u_0, v - v_0) \tag{11}$$

$$f(x - x_0, y - y_0) \Longleftrightarrow \hat{f}(u, v)e^{-i(ux_0+vy_0)/n} \tag{12}$$

It is interesting to note that a shift in $f(x)$ or $f(x, y)$ does not affect the magnitude of its Fourier transform since

$$|\hat{f}(u)e^{-iux_0/n}| = |\hat{f}(u)| \tag{13}$$

$$|\hat{f}(u, v)e^{-i(ux_0+vy_0)/n}| = |\hat{f}(u, v)| \tag{14}$$

This property is especially useful in deriving invariant features in pattern recognition.

Rotation: If $f(x, y)$ is rotated by an angle θ_0, then $\hat{f}(u, v)$ is rotated by the same angle. Similarly, rotating $\hat{f}(u, v)$ causes $f(x, y)$ to be rotated by the same angle.

1.2 The Single Wavelet Transform

In this section we briefly review the most important properties of the wavelet transform and the wavelet based multiresolution decompositions. More detailed information about wavelet transform can be found in [2], [21], [28], [29], [48]-[52]. Wavelets are functions that satisfy certain mathematical requirements. The very name *wavelet* comes from the requirement that they should integrate to zero, "waving" above and below the x-axis. The diminutive connotation of wavelet suggest the function has to be well localised. Other requirements are technical and needed mostly to insure quick and easy calculation of the direct and inverse wavelet transform.

The orthonormal basis of compactly supported wavelets of $L^2(R)$ is formed by the

dilation and translation of a single function $\psi(x)$

$$\psi_{j,k}(x) = 2^{-\frac{j}{2}}\psi(2^{-j}x - k),$$

where j, $k \in Z$. The function $\psi(x)$ has a companion, the scaling function $\phi(x)$, and these functions satisfy the following relations:

$$\phi(x) = \sqrt{2}\sum_{k=0}^{L-1} h_k \phi(2x - k), \tag{15}$$

$$\psi(x) = \sqrt{2}\sum_{k=0}^{L-1} g_k \phi(2x - k), \tag{16}$$

where h_k and g_k are called low-pass and high-pass filter coefficients respectively, and

$$g_k = (-1)^k h_{L-k-1}, \quad k = 0, \ldots, L-1$$

$$\int_{-\infty}^{+\infty} \phi(x)dx = 1.$$

The filter coefficients are assumed to satisfy the orthogonality relations:

$$\sum_n h_n h_{n+2j} = \delta(j), \tag{17}$$

and

$$\sum_n h_n g_{n+2j} = 0. \tag{18}$$

for all j, where $\delta(0) = 1$ and $\delta(j) = 0$ for $j \neq 0$.

The vanishing moments property simply means that the basis functions are chosen to be orthogonal to the low degree polynomials, namely, if the set of functions $\{\psi(x - k)\}_{k \in Z}$ is an orthonormal basis of W_0, then

$$\int_{-\infty}^{+\infty} \psi(x)x^m dx = 0, \quad m = 0, \ldots, M-1. \tag{19}$$

The number of coefficients L in (15) and (16) may be related to the number

of vanishing moments M. However, no matter what conditions are imposed, L is always even. It should be mentioned that a wavelet with higher vanishing moments is desirable in real applications such as image compression and denoising.

1.2.1 Multiresolution Analysis

The wavelet basis induces a multiresolution analysis on $L^2(R)$, i.e., the decomposition of the Hilbert space $L^2(R)$, into a chain of closed subspaces

$$\cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \cdots \tag{20}$$

such that

1. $\cap_{j\in Z} V_j = \{0\}$ and $\cup_{j\in Z} V_j$ is dense in $L^2(R)$
2. For any $f \in L^2(R)$ and any $j \in Z$, $f(x) \in V_j$ if and only if $f(2x) \in V_{j-1}$
3. For any $f \in L^2(R)$ and any $k \in Z$, $f(x) \in V_0$ if and only if $f(x-k) \in V_0$
4. There exists a function $\psi \in V_0$ such that $\{\psi(x-k)\}_{k\in Z}$ is an orthogonal basis of V_0.

Let us define the subspaces W_j as an orthogonal complement of V_j in V_{j-1},

$$V_{j-1} = V_j \oplus W_j \tag{21}$$

and represent the space $L^2(R)$ as a direct sum

$$L^2(R) = \bigoplus_{j\in Z} W_j \tag{22}$$

Selecting the coarsest scale J, we may replace the chain of the subspaces (20) by

$$V_J \subset \cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \cdots \tag{23}$$

and obtain

$$L^2(R) = V_J \bigoplus_{j \leq J} W_j \tag{24}$$

If there is a finite number of scales then, without loss of generality, we set $j = 0$ to be the finest scale and consider

$$V_J \subset \cdots \subset V_2 \subset V_1 \subset V_0, \quad V_0 \subset L^2(R) \tag{25}$$

On each fixed scale j, the wavelets $\{\psi_{j,k}(x)\}_{j \in Z}$ form an orthonormal basis of W_j and the functions $\{\phi_{j,k}(x) = 2^{-\frac{j}{2}}\phi(2^{-j}x - k)\}_{j \in Z}$ form an orthonormal basis of V_j. The coefficients $H = \{h_k\}_{k=0}^{L-1}$ and $G = \{g_k\}_{k=0}^{L-1}$ are quadrature mirror filters. Once the filter H has been chosen, it completely determines the functions ψ and ϕ. Let us define the function

$$m_0(\mu) = \frac{1}{\sqrt{2}} \sum_{k=0}^{L-1} h_k e^{ik\mu}. \tag{26}$$

then the function $m_0(\mu)$ satisfies the equation

$$|m_0(\mu)|^2 + |m_0(\mu + \pi)|^2 = 1. \tag{27}$$

for the coefficients h_k.

1.2.2 The Fast Wavelet Transform

Daubechies [29] has discovered that the wavelet transform can be implemented with a specially designed pair of Finite Impulse Response(FIR) filters called a "Quadrature Mirror Filter"(QMF) pair. The output of the QMF filter pair are down-sampled by a factor of two, that is, every other output sample of the filter is kept, the others are discarded. The low-frequency filter output is fed into another identical QMF filter pair. This operation can be repeated as a pyramid algorithm, yielding a group of signals that divides the spectrum of the original into octave bands with successively coarser measurements in time as the width of each spectral band narrows and decreases in

frequency.

The fast wavelet transform is actually more computationally efficient than the Fast Fourier Transform. As we know, a FFT of length n (where n is an integral power of 2) takes on the order of $O(n \log_2 n)$ operations. A fast wavelet transform of length n requires approximately $O(n)$ operations - the best efficiency possible.

1.2.3 Some Wavelet Families

There are many kinds of wavelets. One can choose between smooth wavelets, compactly supported wavelets, wavelets with simple mathematical expressions, wavelets with simple associated filters, etc. In this section we list some of the most frequently used wavelets. Figure 1 shows these wavelets reproduced from WAVELAB developed by D. L. Donoho.

The **Haar** filter is discontinuous, and can be considered a Daubechies-2. Its scaling filter is

$$H = (1/\sqrt{2}, 1/\sqrt{2}).$$

The **Daubechies-4** filter has its advantage on its most compact support of 4 and its orthogonality. The size 4 is indeed shortest even span in which the second derivatives are computable. Its scaling filter is

$$H = (0.482962913145,\ \ 0.836516303738,\ \ 0.224143868042,\ \ -0.129409522551).$$

The **Coiflet** filters are designed to give both the mother and father wavelets 2, 4, 6, 8, or 10 vanishing moments. Here we only test the 6 vanishing moment case. Its scaling filter is

$$H = (0.038580777748,\ \ -0.126969125396,\ \ -0.077161555496,\ \ 0.607491641386,\ \ 0.745687558934,\ \ 0.226584265197).$$

The **Symmlet-8** is the least asymmetric compactly-supported wavelets with 8 vanishing moments. Its scaling filter is

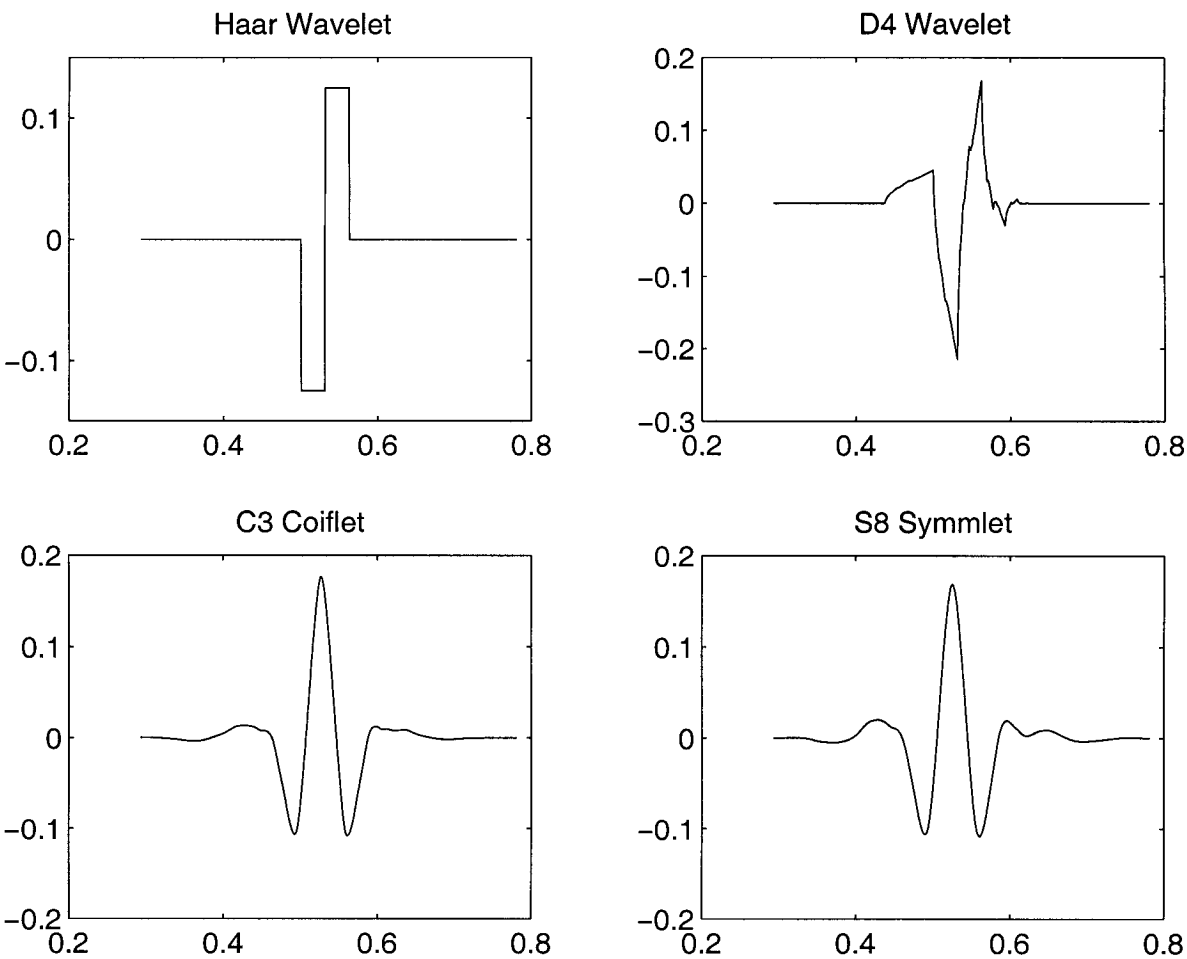

Figure 1: Some single wavelet families frequently used in the literature

$$H = (-0.107148901418,\ -0.041910965125,\ 0.703739068656,\ 1.136658243408,$$
$$0.421234534204,\ -0.140317624179,\ -0.017824701442,\ 0.045570345896).$$

1.3 The Multiwavelet Transform

Multiwavelets are also very powerful in many real applications. In this section, we will give a short introduction to multiwavelet transform. Multiwavelets are generalization of single wavelets. Multiwavelet basis uses translations and dilations of $M \geq 2$ scaling functions $\{\phi_k(x)\}_{1\leq k\leq M}$ and M mother wavelet functions $\{\psi_k(x)\}_{1\leq k\leq M}$. If we write $\Phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_M(x))^T$ and $\Psi(x) = (\psi_1(x), \psi_2(x), \ldots, \psi_M(x))^T$, then we have

$$\Phi(x) = \sqrt{2}\sum_{k=0}^{L-1} H_k \Phi(2x-k), \tag{28}$$

and

$$\Psi(x) = \sqrt{2}\sum_{k=0}^{L-1} G_k \Phi(2x-k). \tag{29}$$

where $\{H_k\}_{0\leq k\leq L-1}$ and $\{G_k\}_{0\leq k\leq L-1}$ are $M\times M$ filter matrices. The scaling functions $\phi_i(x)$ and associated wavelets $\psi_i(x)$ are constructed so that all the integer translations of $\phi_i(x)$ are orthogonal, and the integer translations and the dilations of factor 2 of $\psi_i(x)$ form an orthonormal basis for $L^2(R)$.

As an example, for $M = 2$, $L = 4$, we give the most commonly used multiwavelets developed by Geronimo, Hardin and Massopust [36]. Let

$$H_0 = \begin{pmatrix} 3/10 & 2\sqrt{2}/5 \\ -\sqrt{2}/40 & -3/20 \end{pmatrix}, \quad H_1 = \begin{pmatrix} 3/10 & 0 \\ 9\sqrt{2}/40 & 1/2 \end{pmatrix},$$

$$H_2 = \begin{pmatrix} 0 & 0 \\ 9\sqrt{2}/40 & -3/20 \end{pmatrix}, \quad H_3 = \begin{pmatrix} 0 & 0 \\ -\sqrt{2}/40 & 0 \end{pmatrix},$$

and

$$G_0 = \begin{pmatrix} -\sqrt{2}/40 & -3/20 \\ -1/20 & -3\sqrt{2}/20 \end{pmatrix}, \quad G_1 = \begin{pmatrix} 9\sqrt{2}/40 & -1/2 \\ 9/20 & 0 \end{pmatrix},$$

$$G_2 = \begin{pmatrix} 9\sqrt{2}/40 & -3/20 \\ -9/20 & 3\sqrt{2}/20 \end{pmatrix}, \quad G_3 = \begin{pmatrix} -\sqrt{2}/40 & 0 \\ 1/20 & 0 \end{pmatrix}.$$

then the two functions $\phi_1(x)$ and $\phi_2(x)$ can be generated via (28). Similarly, the two mother wavelet functions $\psi_1(x)$ and $\psi_2(x)$ can be constructed by (29). Let V_J be the closure of the linear span of $2^{J/2}\phi_l(2^J x - k), l = 1, 2;\ k \in Z$. With the above constructions, it has been proved that $\phi_l(x - k),\ l = 1, 2;\ k \in Z$ form an orthonormal basis for V_0, and moreover the dilations and translations $2^{j/2}\psi_l(2^j x - k)$, $l = 1, 2;\ j, k \in Z$ form an orthonormal basis for $L^2(R)$ [31]. In other words, the spaces $V_j,\ j \in Z$, form an orthogonal multiresolution analysis of $L^2(R)$. The two scaling functions $\phi_1(x)$ and $\phi_2(x)$ are supported in $[0, 1]$ and $[0, 2]$, respectively. They are also symmetric and Lipschitz continuous. This is impossible to achieve for single orthogonal wavelets.

The original signal $f(t)$ should be first discretized into the vector $\{f_i\}_{1 \le i \le 2^J}$, where $n = 2^J$ = signal length, and prefiltered before it can be used as input of the discrete multiwavelet transform (DMWT). While single wavelet transform has one input stream of size $n \times 1$ which is provided by $\{f_i\}_{1 \le i \le 2^J}$, multiwavelet transform requires input that consists of M streams each of size $n \times 1$. Therefore a method of mapping the data $\{f_i\}_{1 \le i \le 2^J}$ to the multiple streams has to be developed. This mapping process is called preprocessing and is done by a prefilter or a repeated signal filter [34]. We will return to the prefilter later in this section.

For the sake of clarity, we use $\{S_k^0\}_{0 \le k \le 2^J - 1}$ to denote the multiple streams obtained by applying a prefilter to the original discretized signal $\{f_i\}_{1 \le i \le 2^J}$. Also, we use S_k^j and D_k^j, which are periodic in k with period 2^{j-1}, to represent the low-pass and high-pass coefficients. The forward and inverse DMWT can be recursively calculated

by:

$$S_k^{j+1} = \sqrt{2} \sum_{n=0}^{L-1} H_n S_{n+2k}^j, \tag{30}$$

$$D_k^{j+1} = \sqrt{2} \sum_{n=0}^{L-1} G_n S_{n+2k}^j, \tag{31}$$

and

$$S_{2k+p}^j = \sqrt{2} \sum_{n=0}^{L/2-1} (H_{2n+p}^T S_{n+k}^{j+1} + G_{2n+p}^T D_{n+k}^{j+1}), \tag{32}$$

for $j = 0, 1, \ldots, J-1$; $p = 0, 1$; $k = 0, 1, \ldots$ It is noted that Eq.(32) is different from Eq.(3.7) of [78]. A simple verification can show that the inverse DMWT in Eq.(3.7) of [78] is incorrect. For example, let us consider reconstructing S_0^j from $\{S_k^{j+1}\}_k$ and $\{D_k^{j+1}\}_k$. Before passing through the synthesis filters, we have to upsample (i.e. insert zeros) $\{S_k^{j+1}\}_k$ and $\{D_k^{j+1}\}_k$ first. Since the upsampled version is 0 in every other sample, S_0^j can only be the sum of the form

$$S_0^j = \sqrt{2}(H_0^T S_0^{j+1} + G_0^T D_0^{j+1} + H_2^T S_1^{j+1} + G_2^T D_1^{j+1} + \cdots),$$

that is, with coefficients of H_0, H_2, H_4, ... and similarly with G. However, the equation in [78] has terms with H_0, H_1, H_2, H_3, ...

Because a given signal consists of one stream but the DMWT algorithm requires that the input data be multiple streams, a method of mapping the data to the multiple streams has to be developed. This mapping process is called preprocessing and is done by a *prefilter* Q [34]. A *postfilter* P just does the opposite, i.e., mapping the data from multiple streams into one stream. Thus, with $M = 2$,

$$S_k^0 = \sum_n Q_n \begin{pmatrix} f_{2(n+k)+1} \\ f_{2(n+k)+2} \end{pmatrix}.$$

The postfilter P that accompanies the prefilter Q satisfies $PQ = I$, where I is the

identity filter. So, if one applies a prefilter, DMWT, inverse DMWT and postfilter to any sequence the output will be identical to the input. The commonly used prefilters are:

Identity prefilter

$$Q_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Xia prefilter

$$Q_0 = \begin{pmatrix} 2+\sqrt{2}/10 & 2-\sqrt{2}/10 \\ \sqrt{2}+3/20 & \sqrt{2}-3/20 \end{pmatrix}.$$

Minimal prefilter

$$Q_0 = \begin{pmatrix} 2\sqrt{2} & -\sqrt{2} \\ 1 & 0 \end{pmatrix}.$$

Repeated signal prefilter is different from the definition above. It is defined by

$$S_k^0 = f_{k+1} \begin{pmatrix} \sqrt{2} \\ 1 \end{pmatrix}.$$

It has been shown by Downie and Silverman [34] that the repeated signal filter is very good for denoising purposes.

Multiwavelets have some advantages in comparison to single ones. For example, such features as short support, orthogonality, symmetry, and higher order of vanishing moments, are known to be important in signal processing. A single wavelet cannot possess all these properties at the same time, but multiwavelets can. It is confirmed that multiwavelets can give better results than the single wavelets in image compression and denoising [67].

As an alternative, complex wavelets can also achieve short support, orthogonality, symmetry and maximum vanishing moments at the same time. They have been

proved to be advantageous in many applications. One good example is their applications in image denoising.

1.4 The Ridgelet Transform

As we know, ridgelets have been recently developed for image processing applications ([33], [63], [35], [6], [7], [8], [30]). For each $a > 0$, each $b \in R$ and each $\theta \in [0, 2\pi)$, the bivariate ridgelet $\psi_{a,b,\theta} : R^2 \rightarrow R$ is defined as

$$\psi_{a,b,\theta} = a^{-1/2}\psi((x_1 cos\theta + x_2 sin\theta - b)/a)$$

A ridgelet is constant along the lines $x_1 cos\theta + x_2 sin\theta = constant$. Transverse to these ridges it is a wavelet. Given an integrable bivariate image $f(x_1, x_2)$, we can define its ridgelet coefficients as

$$R(a, b, \theta) = \int \psi_{a,b,\theta} f(x_1, x_2) dx_1 dx_2$$

Fig. 2 shows a ridgelet function and its scaled, shifted and rotated versions.

The ridgelet transform can be represented in terms of the Radon transform. The Radon transform of an image $f(x_1, x_2)$ is defined as

$$RA(\theta, t) = \int f(x_1, x_2)\delta(x_1 cos\theta + x_2 sin\theta - t) dx_1 dx_2$$

where δ is the Dirac distribution. So the ridgelet transform is precisely the application of a 1D wavelet transform to the slices of the Radon transform where the angular variable θ is constant and t is varying. Ridgelets are different from wavelets in a sense that ridgelets exhibit very high directional sensitivity and are highly anisotropic. A fast ridgelet transform can be performed in the Fourier domain. First the 2D FFT is computed. Then it is interpolated along a number of straight lines equal to the selected number of projections. Each line passes through the center of the

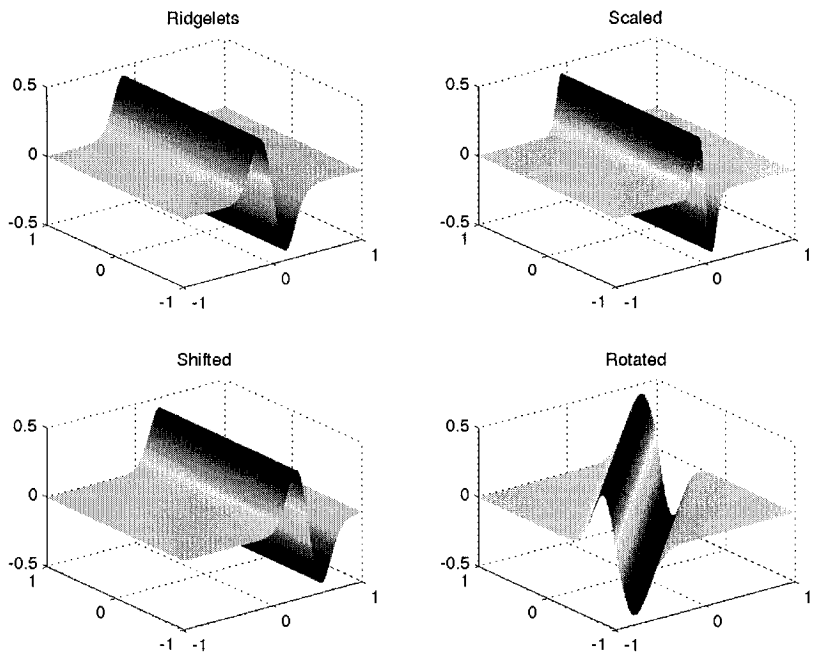

Figure 2: The ridgelets used by Candes and others.

2D frequency space, with a slope equal to the projection angle, and a number of interpolation points equal to the number of rays per projection. After the 1D inverse FFT along each interpolated ray we get the ridgelet transform coefficients. Fig. 3 illustrates the steps of this fast transformation.

1.5 Neural Networks

The term neural network is used to describe various topologies of highly interconnected simple processing elements (neurons) that offer an alternative to traditional approaches to computing. Neural networks offer important new approaches to information processing because of their adaptability and ability to learn as well as their massive parallelism. Neural network research has matured greatly since the perceptron of 1950's. This maturation has three resources: an advancement in mathematical theories, development of new computer tools, and increased understanding of neurobiology. The limits of today's computing devices such as inadequacies in storage, speed,

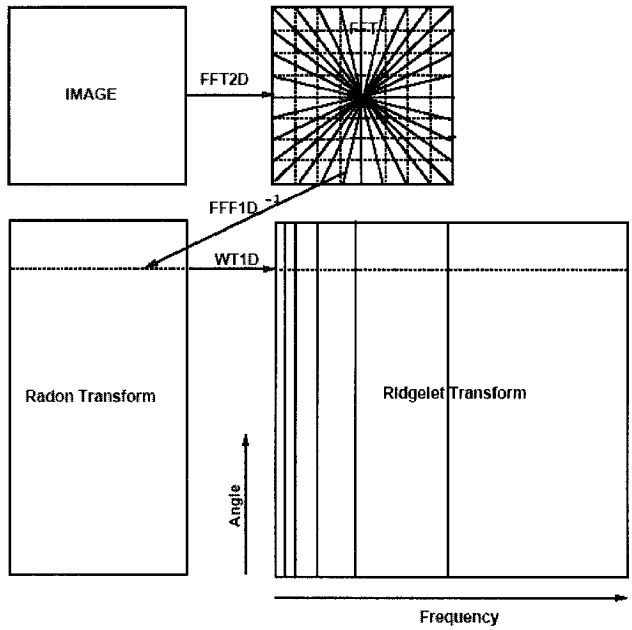

Figure 3: The fast ridgelet transform proposed by Candes [64].

and user flexibility have restricted present neural network efforts. There have been significant demonstrations of neural network capabilities in speech, signal processing, control, OCR, and robotics, etc.

Neural networks are fundamentally different from a traditional computer. Neural networks do not sequentially execute instructions nor do they contain memory for storing operation instructions or data. Neural networks respond in parallel to a set of inputs. Neural networks are more concerned with transformations than algorithms and procedures. Neural networks do not contain only one or a few complicated computational devices, but are comprised of a large number of simple devices often doing little more than computing weighted sums. Neural networks are not replacements for traditional computers, but are an entirely different class of computational devices that are capable of qualitatively different tasks.

Neural networks provide a completely new and unique way to look at information processing. Procedural programming requires a procedural statement to calculate the solution to the problem, i.e. an algorithm. Expert systems require a statement of

the solution to the problem whose symbolic elements can be linked by the inference engine to achieve the goal or solution. Neural networks require a statistically relevant set of training examples of the desired mapping.

Neural networks have either fixed weights or adaptable (adjustable) weights. Learning laws are used to adjust the weights, which represent the interconnection strengths. There are two types of learning, namely *supervised* and *unsupervised*. *Supervised* learning occurs when the neural network is supplied with both the input values and correct output values, and the neural network adjusts its weights based upon the error of the computed output. *Unsupervised* learning occurs when the neural network is only provided with the input values, and the network adjusts the weights based solely on the input values and the current network output. Neural networks have two phases to their operation: training of the network and the recall or computation phase. Neural networks learn from experience, generalize from previous examples to new ones and abstract essential characteristics from inputs containing relevant data.

1.5.1 Backpropagation Neural Network

Multilayer perceptrons are feedforward, nonrecurrent networks with one or more layers of nodes between the input and output layers called hidden layers. They overcome many of the limitations of a single layer perceptrons. The capabilities stem from the nonlinearities used within nodes. The activation function attenuates large signals and amplifies small signals. Training assumes that each input vector is paired with a target vector representing the desired output. Before training starts, weights are initialized to small random numbers. During training, the network selects a training pair from the training data set and calculate the output. Error between the net output and the desired output is calculated. Weights are adjusted according to the error backpropagated from the output layer to the previous layers. After training, the network can be used to generalize the output of a new unseen input pattern.

There are various issues in designing and training a neural network, including

1. Selecting the number of neurons for input, hidden and output layers.

2. Selecting the number of hidden layers.

3. Selecting the fan-in and fan-out of the hidden units.

4. Selecting the number of training pairs or the complexity of the mapping from input to output.

1.5.2 Convolutional Network

Convolutional network was developed by LeCun et al. [24] - [26] for handwritten digit recognition. In convolutional network, each unit takes its input from a local receptive field from the previous layer, forcing it to extract a local feature. Furthermore, units located at different places on the image are grouped in planes, called feature maps, within which units are constrained to share a single set of weights. This makes the operation performed by a feature map shift-invariant, and equivalent to a convolution, followed by squashing functions. The weight-sharing technique greatly reduced the number of free parameters.

There are several versions of convolutional networks, including LeNet 1, LeNet 4, LeNet 5, etc. LeNet 1 has six layers with each layer extracting invariant features or making a down-sampling from the previous layer. For each feature map, a set of weights are constrained to be the same. In order to make optimal use of the large size of the training set, LeNet 4 was introduced so that it has more free parameters in the network. Experimental results show that LeNet 4 gives much higher recognition rate for handwritten digits. LeNet 5 has more feature maps and a larger fully-connected layer. It uses a distributed representation to encode the categories at the output layer. LeNet 5 produces the state-of-the-art recognition rate and it is used in many banks in the United States. Figure 4 illustrates the network structure of a convolutional network.

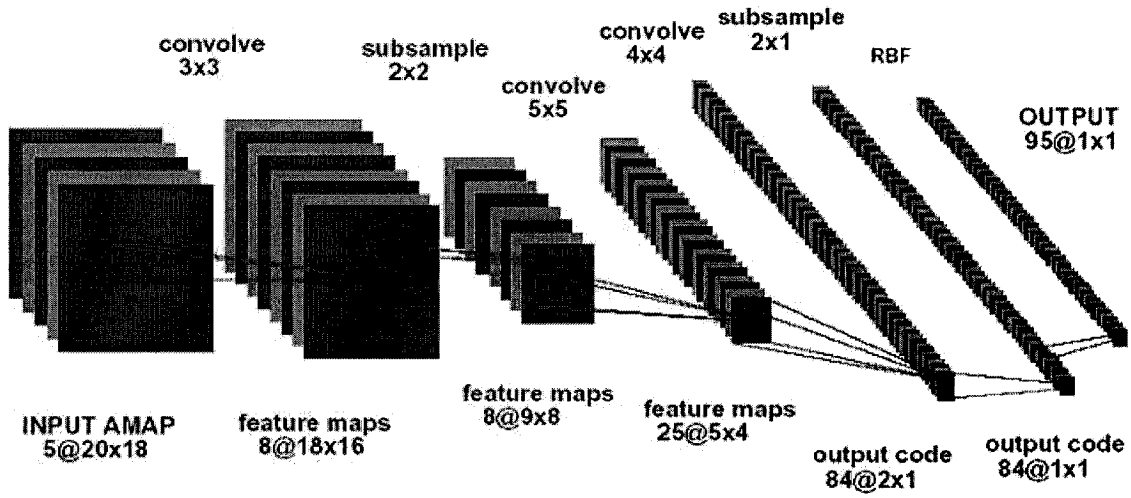

Figure 4: The network structure of a convolutional network [24].

1.6 Translation and Scale Normalization

In this section, we review some of the most frequently used normalization techniques to eliminate translation and scale variance. To achieve translation invariance, we can use the regular moments of an image. Recall that the regular moments m_{pq} are defined as $m_{pq} = \int_{-\infty}^{+\infty} x_1^p x_2^q f(x_1, x_2) dx_1 dx_2$. Translation invariance can be achieved by transforming the image so that its first order moments, m_{01} and m_{10}, are both equal to zero. Let $\bar{x}_1$ and $\bar{x}_2$ be the centroid location of the original image, then

$$\bar{x}_1 = \frac{m_{10}}{m_{00}}, \quad \bar{x}_2 = \frac{m_{01}}{m_{00}}.$$

therefore, we can transform the image $f(x_1, x_2)$ to $f(x_1 + \bar{x}_1, x_2 + \bar{x}_2)$.

There are three commonly used methods to achieve scale-invariance. First, it can be accomplished by scaling the image such that its zeroth order moment m_{00} is equal to a predetermined value β [42]. By calculation, we can choose the scaling factor $a = \sqrt{\frac{\beta}{m_{00}}}$. Thus scale invariance is achieved by transforming the original image function $f(x_1, x_2)$ into a new function $f(\frac{x_1}{a}, \frac{x_2}{a})$ with $a = \sqrt{\frac{\beta}{m_{00}}}$. Second, we can achieve scale invariance by setting $a = \max_{f(x_1,x_2)\neq 0} \sqrt{(x_1 - \bar{x}_1)^2 + (x_2 - \bar{x}_2)^2}$, the longest

distance from $(\bar{x}_1, \bar{x}_2)$ to a point (x_1, x_2) on the pattern. This method, however, is rarely used in noisy environment since it is very sensitive to noise. The third method is to normalise the image so that the average radius r_0 for the image pixels is a quarter of the input grid dimension; i.e., we have

$$r_0 = \frac{1}{m_{00}} \sum_{j=1}^{n} \sum_{k=1}^{n} f(x_{1,j}, x_{2,k}) \sqrt{x_{1,j}^2 + x_{2,k}^2},$$

$$a = \frac{n}{4r_0}.$$

In summary, an image function $f(x_1, x_2)$ can be normalised to be scale and translation invariant by transforming it into $f(\frac{x_1}{a} + \bar{x}_1, \frac{x_2}{a} + \bar{x}_2)$, with $(\bar{x}_1, \bar{x}_2)$ being the centroid of $f(x_1, x_2)$ and a suitable scaling factor a. When $(\frac{x_1}{a} + \bar{x}_1, \frac{x_2}{a} + \bar{x}_2)$ does not correspond to a grid location, we can interpolate it from the values of the four nearest grid locations around it.

1.7 Outline of the Thesis

In this thesis, I propose different approaches to find good features for pattern recognition and different methods for signal/image denoising by using wavelets and ridgelets. In Chapter 2, we develop a handwritten numeral recognition descriptor using multiwavelets and neural networks. We first trace the contour of the numeral, then normalize and resample the contour so that it is translation- and scale-invariant. We then perform multiwavelet orthonormal shell expansion on the contour to get several resolution levels and the average. Finally, we use the shell coefficients as features to input into a feed-forward neural network to recognize the hand-written numerals. The main advantage of the orthonormal shell decomposition is that it decomposes a signal into multiresolution levels, but without down-sampling. Wavelet transforms with down-sampling can give very different coefficients when the input signal is shifted. This is the main limitation of wavelet transforms in pattern recognition. For the shell

expansion, we prefer multiwavelets to scalar wavelets because we have two coordinates x and y for each point on the contour. If we extract features from x and y separately, just as Wunsch et al. did [77], then we may not get the best features. In addition, we know that multiwavelets have advantages over scalar wavelets, such as short support, orthogonality, symmetry and higher order of vanishing moments. These properties allow multiwavelets to outperform scalar wavelets in some applications, e.g. signal denoising [3]. We conducted experiments and found that it is feasible to use multiwavelet features in handwritten numeral recognition.

In Chapter 3, we present two novel descriptors for feature extraction by using ridgelets, wavelet cycle-spinning, and Fourier features. Ridgelets have been developed recently and have many advantages over wavelets in applications to image processing. However, those ridgelets have been developed over a square domain. For pattern recognition, ridgelets over a square cannot be used to extract invariant features. To avoid this difficulty, we have successfully implemented ridgelets on a disk. For the first descriptor, we combine this new implemented ridgelets with Fourier transform to extract rotation-invariant features for pattern recognition. For the second descriptor, we extract rotation-invariant features by using ridgelets, wavelet cycle-spinning, and Fourier transform. The two descriptors are very robust to Gaussian noise even when the noise level is very high. Experimental results show that the two descriptors are excellent choices for pattern recognition.

Multiwavelets give better results than single wavelets for signal denoising ([67], [34], [3]). In Chapter 4, we study multiwavelet thresholding by incorporating neighbouring coefficients. Experimental results show that this approach is better than the conventional approach which only uses the term-by-term multiwavelet denoising. Also, it outperforms neighbour single wavelet denoising for some standard test signals and real life images. This is an extension to Cai and Silverman's work [5].

The denoising of a natural image corrupted by Gaussian noise is a classical problem in signal or image processing. Donoho and his coworkers at Stanford pioneered a wavelet denoising scheme by thresholding the wavelet coefficients arising from the

standard discrete wavelet transform. This work has been widely used in science and engineering applications. However, this denoising scheme tends to kill too many wavelet coefficients that might contain useful image information. In Chapter 5, we propose one wavelet image thresholding scheme by incorporating neighbouring coefficients for both TI and non-TI cases. This approach is valid because a large wavelet coefficient will probably have large wavelet coefficients as its neighbours. Experimental results show that our algorithm is better than *VisuShrink* and the TI image denoising method developed by Yu et al. [81]. We also investigate different neighbourhood sizes and find that a size of 3×3 or 5×5 is the best among all window sizes.

Image denoising by means of wavelet transforms has been an active research topic for many years. For a given noisy image, which kind of wavelet and what threshold we use should have significant impact on the quality of the denoised image. In Chapter 6, we use Simulated Annealing to find the customized wavelet filters and the customized threshold corresponding to the given noisy image at the same time. Also, we consider a small neighbourhood around the customized wavelet coefficient to be thresholded. Experimental results show that our approach is better than *VisuShrink*, *NeighShrink* and the *wiener2* filter that is available in Matlab Image Processing Toolbox.

Finally, Chapter 7 gives the conclusions of current work and proposes the future work in the areas of pattern recognition and denoising.

Chapter 2

Contour-Based Handwritten Numeral Recognition using Multiwavelets and Neural Networks[1]

2.1 Introduction

Handwritten numeral recognition is an important problem of optical character recognition (OCR) [74]. Among all existing techniques, one important approach is to extract the outer contour of the handwritten numeral. Since the contour is periodic, it is well suited for Fourier-based methods. Zahn and Roskies [82] defined the cumulative angular function $\phi(l)$ as the net amount of angular bend between the starting point and the point with arc length l. They normalized $\phi(l)$ so that it is periodic. Suppose the Fourier expansion is

$$\phi(t) = \mu_0 + \sum(a_k \cos kt + b_k \sin kt)$$

[1]This work was published in the *Pattern Recognition* Journal [14]. We believe this is the first published paper using multiwavelet transforms in pattern recognition.

They defined rotation- and mirror-invariant features $A_k = \sqrt{a_k^2 + b_k^2}$ and rotation-invariant features $F_{kj} = j * \alpha_k - k * \alpha_j$, where $\alpha_k = tan^{-1}(b_k/a_k)$. However, they warned that α_k is unreliable when A_k is very small. Therefore, F_{kj} may be unreliable under this condition. Granlund [38] generated a complex-valued function $u(t)$ for the contour. Suppose the Fourier coefficients a_n is defined as

$$a_n = \frac{1}{T}\int_0^T u(t)e^{-jn2\pi t/T}dt$$

then he defined scale- and rotation-invariant features

$$b_n = \frac{a_{1+n}a_{1-n}}{a_1^2}$$

and

$$D_{mn} = \frac{a_{1+m}^{n/k}}{a_{1-n}^{m/k}}$$

where k is the greatest common divisor of m and n. He also defined features that are scale-invariant, but depend on rotation.

Spline curve approximation is also frequently used on contour. Paglieroni [55] represented contours in the spatial domain by far fewer B-spline control points than the contour samples. Under proper conditions, there exists a fast transform from contours to these control points. It was shown that discriminant analysis between pairs of normalized and similarly modelled contours can be efficiently performed directly from control points. Taxt et al. [69] approximated the contour by a parametric spline curve. The curvatures are calculated and the values measured at regular intervals of these derived spline curves were used as descriptors in a statistical classification scheme. The features are translation-invariant by nature, but are dependent on rotation. Sekita et al. [57] used splines to approximate the contour and defined the breakpoints as the maximum points of curvature functions of the contours. The contours between adjacent breakpoints are extracted in the form of directed curves which compose the features of a character. Yang et al. [79] calculate the curvature of the

contours of handwritten numerals by means of high order B-splines to recognize similiar handwritten numerals. The features extracted are rotation-invariant. Artificial neural network and support vector machines are used in the classifier.

Recently wavelet descriptor has been used for printed handwritten character recognition. Wunsch et al. [77] proposed to use wavelet features in combination with feed-forward neural networks. Because they employed the wavelet transform with down-sampling, the wavelet features can get quite different coefficients even if the contour is shifted very little. In addition, they extracted wavelet features from x and y coordinates independently.

In this Chapter, we present a novel shape descriptor for the recognition of handwritten numerals. The descriptor is derived from the multiwavelet shell expansion of an object's contours. The motivation to use multiwavelet basis is threefold. First, multiwavelets provide a localized frequency representation, which can reflect local properties much better than Fourier based method. Second, orthonormal multiwavelets provide a natural hierarchical multiresolution representation, and there is substantial evidence that the human visual system use similar mutiscale representations. Third and more importantly, the contour is represented by two streams of data $(x,y)^T$, so it is natural to use multiwavelet transform instead of the scalar one. Since we decompose the contour into orthonormal multiwavelet shell, the bad behaviour of down-sampling in pattern recognition can be avoided. To make it more clear, for wavelet transform with down-sampling we can get quite different wavelet coefficients even if we only shift the input signal a few sample points. On the contrary, the orthonormal shell decomposition does not have this problem. In fact, we have successfully used scalar orthonormal shell and Fourier transform to extract invariant features in [4]. We trained the neural network with 4000 handwritten numerals. The test dataset consists of 2000 handwritten numerals. The handwritten numeral databases are from the Centre for Pattern Recognition and Machine Intelligence at Concordia University (CENPARMI). Experimental results show that our proposed method is better than the wavelet neural network method in [77].

The Chapter is organized as follows. Section 2.2 explains what an orthonormal shell is. Section 2.3 presents the orthonormal multiwavelet neural network descriptor. Section 2.4 shows some experimental results.

2.2 The Orthonormal Shell Expansion

In this section, we generalize the orthonormal shell expansion, developed in [56] for the scalar wavelet, to the multiwavelet case. Basically speaking, a shell is a multiresolution wavelet decomposition of the original signal where no down-sampling is performed. That means we get the same number of wavelet coefficients for every decomposition scale. It should be mentioned that the coefficients of an orthogonal multiwavelet expansion are not shift-invariant. However, if all the multiwavelet coefficients of n circulant shifts of the streams are computed, we may use them when shift invariance is important. Based on this observation, the notion of a shell (much more redundant than a frame) was introduced in [56] to obtain a redundant but shift-invariant family of functions.

In our applications, there is always the finest and the coarsest scales of interest and therefore the number of scales is finite and we can consider only shifts by multiples of some fixed unit. Assuming that the finest scale is described by the n-dimensional subspace V_0 and consider only circulant shifts in V_0. Let V_{j_0} be the subspace describing the coarsest scale ($1 \leq j_0 \leq J$) where $n = 2^J$, and let $\Psi_{j,k}(t) = 2^{-\frac{j}{2}}\Psi(2^j(t-k))$ and $\Phi_{j,k}(t) = 2^{-\frac{j}{2}}\Phi(2^{-j}(t-k))$. Therefore, the functions $\{\Psi_{j,k}\}_{1\leq j\leq j_0, 0\leq k\leq 2^{J-j}-1}$ and $\{\Phi_{j_0,k}\}_{0\leq k\leq 2^{J-j_0}-1}$ generate the coefficients S_k^j and D_k^j:

$$S_k^j = \int F(t)\Phi_{j,k}(t)dx, \tag{33}$$

and

$$D_k^j = \int F(t)\Psi_{j,k}(t)dx. \tag{34}$$

where $j = 1, 2, \ldots, j_0$ regulates scale and $k = 0, 1, \ldots, 2^{J-j} - 1$ regulates shift. Here

$F(t)$ is the original input multi-stream functions. The coefficients $\{D_k^j\}_{1\le j\le j_0, 0\le k\le 2^{J-j}-1}$ are known as orthonormal wavelet coefficients.

In the case of orthonormal shell decomposition, Saito and Beylkin defined the functions $\{\Psi_{j,k}(t)\}_{1\le j\le j_0, 0\le k\le 2^J-1}$ and $\{\Phi_{j_0,k}(t)\}_{0\le k\le 2^J-1}$ as a shell of the orthonormal wavelets for shifts in V_0. As a consequence, the coefficients $\{D_k^j\}_{1\le j\le j_0, 0\le k\le 2^J-1}$ and $\{S_k^{j_0}\}_{0\le k\le 2^J-1}$ are called the orthonormal shell coefficients. Clearly the set of coefficients in the orthonormal shell is much more abundant and overly redundant compared to the set of coefficients in the orthogonal multiwavelet transform. However, this redundancy is needed for our shift invariant property.

Assuming that the orthonormal multiwavelet coefficients of the finest scale $\{S_k^0\}_{0\le k\le n-1}$ are given as an original signal stream and let us consider the function $F = \sum_{k=0}^{n-1} S_k^0 \Phi_{0,k}$. The orthonormal shell coefficients of this function F are obtained from the filters $H = \{H_l\}_{0\le l\le L-1}$ and $G = \{G_l\}_{0\le l\le L-1}$

$$S_k^j = \sqrt{2} \sum_{l=0}^{L-1} H_l S_{k+2^{j-1}l}^{j-1}, \tag{35}$$

and

$$D_k^j = \sqrt{2} \sum_{l=0}^{L-1} G_l S_{k+2^{j-1}l}^{j-1}. \tag{36}$$

for $j = 1, \ldots, j_0, \quad k = 0, \ldots, 2^{J-j} - 1$. The complexity of (35) and (36) is $O(n \log n)$.

It is easy to show that the recurrence relations (35) and (36) compute the orthonormal wavelet coefficients of all circulant shifts of the function F. For D_k^j, the first scale is:

$$D_k^1 = \sqrt{2} \sum_{l=0}^{L-1} G_l S_{k+l}^0 = \sqrt{2} \sum_{l=0}^{L-1} G_l S_{2k+l}^0 + \sqrt{2} \sum_{l=0}^{L-1} G_l S_{2k+1+l}^0 = D_{2k}^1 + D_{2k+1}^1. \tag{37}$$

for $k = 0, \ldots, \frac{n}{2} - 1$.

It is clear that the sequence $\{D_{2k}^1\}$ contains all the orthonormal multiwavelet coefficients that appear if $F(x)$ is circularly shifted by even numbers and the sequence

$\{D^1_{2k+1}\}$ contains all the orthonormal multiwavelet coefficients for odd shifts.

Similarly, at the j-th scale

$$D^j_{2^j k+m} = \sqrt{2} \sum_{l=0}^{L-1} G_l S^{j-1}_{2^{j-1}(2k+l)+m}, \tag{38}$$

and

$$S^j_{2^j k+m} = \sqrt{2} \sum_{l=0}^{L-1} H_l S^{j-1}_{2^{j-1}(2k+l)+m}. \tag{39}$$

for $k = 0, 1, \ldots, 2^{J-j}$, $m = 0, 1, \ldots, 2^j - 1$.

The sequences $\{D^j_{2^j k}\}$, $\{D^j_{2^j k+1}\}$, $\ldots$, $\{D^j_{2^j k+2^j-1}\}$ contain the orthonormal multiwavelet coefficients of the j-th scale of the signal shifted by $0, 1, \ldots, 2^j - 1$, respectively. Therefore, the set $\{D^j_k\}_{1 \le j \le j_0, 0 \le k \le 2^{J-j_0}-1}$ and $\{S^{j_0}_k\}_{0 \le k \le 2^{J-j_0}-1}$ contain all the coefficients of the orthonormal wavelet expansion of $F(t)$, $F(t+1)$, $\ldots$, $F(t+n-1)$. This set of coefficients defines the orthonormal shell decomposition. The diagram for computing these coefficients is illustrated in Figure 5.

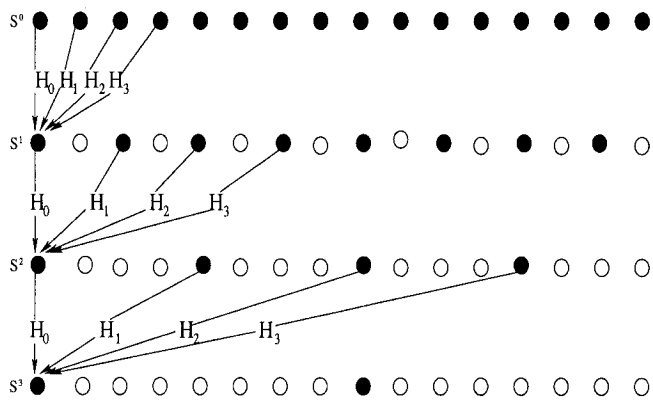

Figure 5: A scheme illustrating the algorithm for expanding a signal into multiresolution scales using the filter $H = (H_0, H_1, H_2, H_3)$.

2.3 Orthonormal Multiwavelet Neural Network Descriptor

Feature selection is the critical step in the recognition process, and what distinguishes OCR methodologies from each other are the types of features selected for representation. For a good feature extraction algorithm, it is desirable to have the properties of invariance in terms of translation, rotation, and scale. In handwritten recognition, however, the property of rotation invariance is not necessary, on the contrary it may lead to confusion of characters like '6' and '9'.

In this section, we introduce a descriptor which uses an orthonormal multiwavelet and a neural network to recognize handwritten numerals. In order to get invariant features, we have to normalize the handwritten numeral and its contour. Translation-invariance can be easily achieved by moving the coordinate system origin to the centroid of the handwritten numeral. For the starting point of the contour, we adopt the normalization method used in [77]. Suppose O is the upper-left corner of the minimal bounding rectangle of the handwritten numeral, we select the starting point as the point on the handwritten numeral that has the shortest distance from O. After finding the starting point, we can trace the contour by following the outer contour in the clock-wise direction. In order to eliminate the size difference, we have to resample the contour so that the total number of contour points is fixed, say 64. We also need to size-normalize the contour so that the handwritten numeral falls into a unit circle. This finishes the normalization stage. We perform orthonormal multiwavelets shell expansion on this normalized contour and input the multiwavelets coefficients or the average to a feed-forward neural network.

The algorithm can be summarized as follows:

1. Find the starting point and trace the outer contour of the handwritten numeral.

2. Resample the contour to a fixed number of points and scale the contour so that it falls into the unit circle.

3. Apply orthonormal multiwavelet shell expansion to the contour $(x,y)^T$.

4. Feed the multiwavelet coefficients or the average into neural networks.

Figure 6 gives an example of this algorithm applied to a handwritten numeral. The numeral is displayed in binary format in (a). The contour of the numeral with a staring point is shown in (b). The x and y coordinates of the normalized contour are showed in (c) and (d), respectively. The orthonormal multiwavelet expansions of the contour are shown in (e) and (f). Here the orthonormal multiwavelet expansions are applied to $(x, y)^T$ together instead of x and y separately. It should be mentioned that in this figure -1, -2, -3 mean multiwavelet levels D^1, D^2, D^3, respectively, whereas -4 means the average S^3.

The main advantage of the orthonormal shell decomposition is that it decomposes a signal into multiresolution levels without down-sampling. As we know, the wavelet transform with down-sampling can produce significantly different coefficients even if the input signal is shifted only slightly. This is the main limitation of wavelet transforms in pattern recognition. For the shell expansion, we prefer multiwavelets to scalar wavelets because we have two coordinates x and y for each point on the contour. If we extract features from x and y separately, just as Wunsch et al. did [77], then we may not get the best features. In addition, we know that multiwavelets have advantages over scalar wavelets, such as short support, orthogonality, symmetry and higher order of vanishing moments. These properties allow multiwavelets to outperform scalar wavelets in some applications (e.g. signal denoising [3]).

In theory, the matching process can be performed from coarse to fine resolution. For each resolution, we input the features of the handwritten numeral into the neural network and have three decisions to make:

1. Accept the handwritten numeral as a specific pattern if the minimum distance between the network output and the desired output for every class is below a predefined threshold ϵ_1.

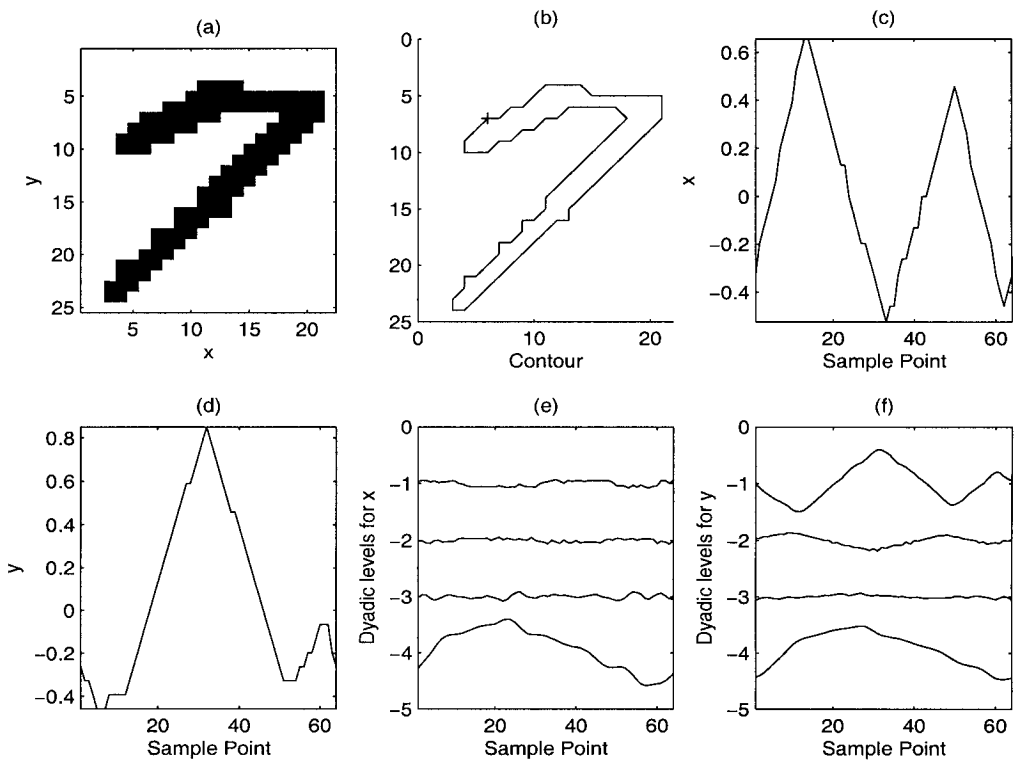

Figure 6: An illustration of the orthonormal multiwavelet descriptor: (a) the handwritten numeral in binary format, (b) the contour of the numeral with a starting point, (c) the x coordinate of the normalized contour, (d) the y coordinate of the normalized contour, (e) the first component of the orthonormal multiwavelet expansion, (f) the second component of the orthonormal multiwavelet expansion.

2. Reject the handwritten numeral if the distance is above a predefined threshold ϵ_2 .

3. Mark the handwritten numeral whose distance is between ϵ_1 and ϵ_2 as to be determined and begin the next iteration.

If the handwritten numeral is accepted or rejected, the matching process is then terminated. If the handwritten numeral is undetermined, we continue the matching process to the next finer resolution, but only those entries that are marked as "to be determined" are used. Even though the above mentioned multi-resolution approach is similar to human simultaneous interpretation of visual information, it is not trivial to apply it in real applications. There are two aspects to consider. First, we have to determine a threshold in order to provide guidelines for acceptance, rejection, and "to be determined". There is no way to determine an optimal threshold mathematically. One has to choose it experimentally. Second, significant features are lost at very low resolutions and it is likely that for very high resolutions the intraclass variance will become larger because of the deformation of the pattern and the accumulation errors during the transformations. Therefore, it is desirable to use only intermediate resolutions during the classification phase.

Multilayer feed-forward neural networks [39] have been used in OCR for many years. These networks can be treated as feature extractors and classifiers. In our work, the network has 10 output nodes, one for each of the ten digit classes '0' - '9' (Figure 7). Each node in a layer has full connections from the nodes in the previous layer and the proceeding layer. During the training phase, connection weights are learned. The output at a node is a function of the weighted sum of the connected nodes at the previous layer. We can consider a feed-forward neural network as constructing decision boundaries in a feature space. As the number of layers and nodes increases, the flexibility of the classifier increases by allowing more and more complex decision boundaries. Neural networks can also perform hierarchical feature extraction. Each node sees a window in the previous layer and combines the low-level features in

this window into a higher level feature. So the higher the layer, the more global the features that are extracted. A typical example of this kind of network is the convolution networks whose performance is very good in recognizing handwritten characters [24], [25], [26].

2.4 Experimental Results

In order to evaluate the performance of our proposed recognition system, we use a 3-layer feed-forward neural network in our experiments. The number of nodes in each layer is given by $40 \times 20 \times 10$. The input layer is not given here because it depends on the input feature size. Our experiments are performed on the CENPARMI handwritten numeral database. This database contains 6000 unconstrained handwritten numerals originally collected from dead letter envelopes by the U.S. postal service at different locations. The numerals in the database are stored in bi-level format. We use 4000 numerals for training and 2000 for testing. A sample of 100 numerals from the database is shown in Figure 8. Some of the numerals are very difficult to recognize even with human eyes. One thing we want to mention is that it would be best if our proposed descriptor can be applied to a database which contains both numerals and characters. However, since some characters have isolated parts, e.g. 'i' and 'j', it is not a good idea to use contours to represent them. We restrict the application of our proposed descriptor to handwritten numeral recognition. We train the neural network up to 50000 epochs for all the experiments conducted in this section. For every handwritten numeral in the database, we apply the above-mentioned orthonormal multiwavelet shell expansion on the contour and input the multiwavelet coefficients into the neural network. Since the numeral centroid is used as the reference point of the system, the translation invariance property is obviously satisfied. We normalize the number of points on the contour to 64. The orthonormal multiwavelet expansion is performed up to level 3, i.e., we have multiwavelet coefficients levels D^1, D^2, D^3 and the average S^3. The main idea of our proposed technique is to represent the

object contour by fine-to-coarse approximation. We implemented a multi-classifier in our experiments. For level D^3 we input it into the neural network to get a decision. Also, we get a decision for level S^3. If the the two decisions are the same, then the numeral is classified. Otherwise, we use features S^3 and D^3 together to input into the neural network and make the final decision. We get 92.20% recognition rate for this multiclassifier.

We compare our method with the scalar wavelet neural network method introduced by Wunsch et al. [77], where the scalar wavelet transform is applied to the contour coordinates x and y separately and then different levels of wavelet coefficients are fed into the neural network. Here, the wavelet transform uses down-sampling for every decomposition level, so the number of wavelet coefficents is reduced by half for every successive level. On the other hand, the orthonormal shell wavelet expansion has the same number of wavelet coefficients for every level. In our experiment we use the intermediate level 3 of the wavelet coefficients as features to input into the neural network. For the same data set and neural network, the method in [77] only gets 85.25% recognition rate. It is clear that our proposed method obtains higher recognition rate. The main problem with Wunsch's method is that scalar wavelets with down-sampling are used. A wavelet transform with down-sampling can cause quite different wavelet coefficients. That is not good for pattern recognition. Also, a scalar wavelet does not have properties as good as a multiwavelet transform.

As mentioned in [77], the limitation of a contour-based method is that the contour may not be closed or it may be composed of isolated parts. This is true for any methods based on the outer contours of the objects. The positive side of this approach is that we reduce the feature space from 2D to 1D which saves a lot of space and processing time. Basically, this saving can be approximated as from $O(n^2)$ to $O(n)$. On the other hand, we sacrifice some recognition rate because of this feature reduction. We can see this by looking at the recognition rate of this section.

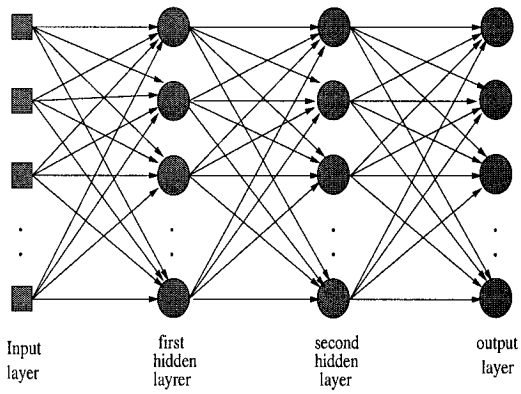

Figure 7: A neural network with two hidden layers

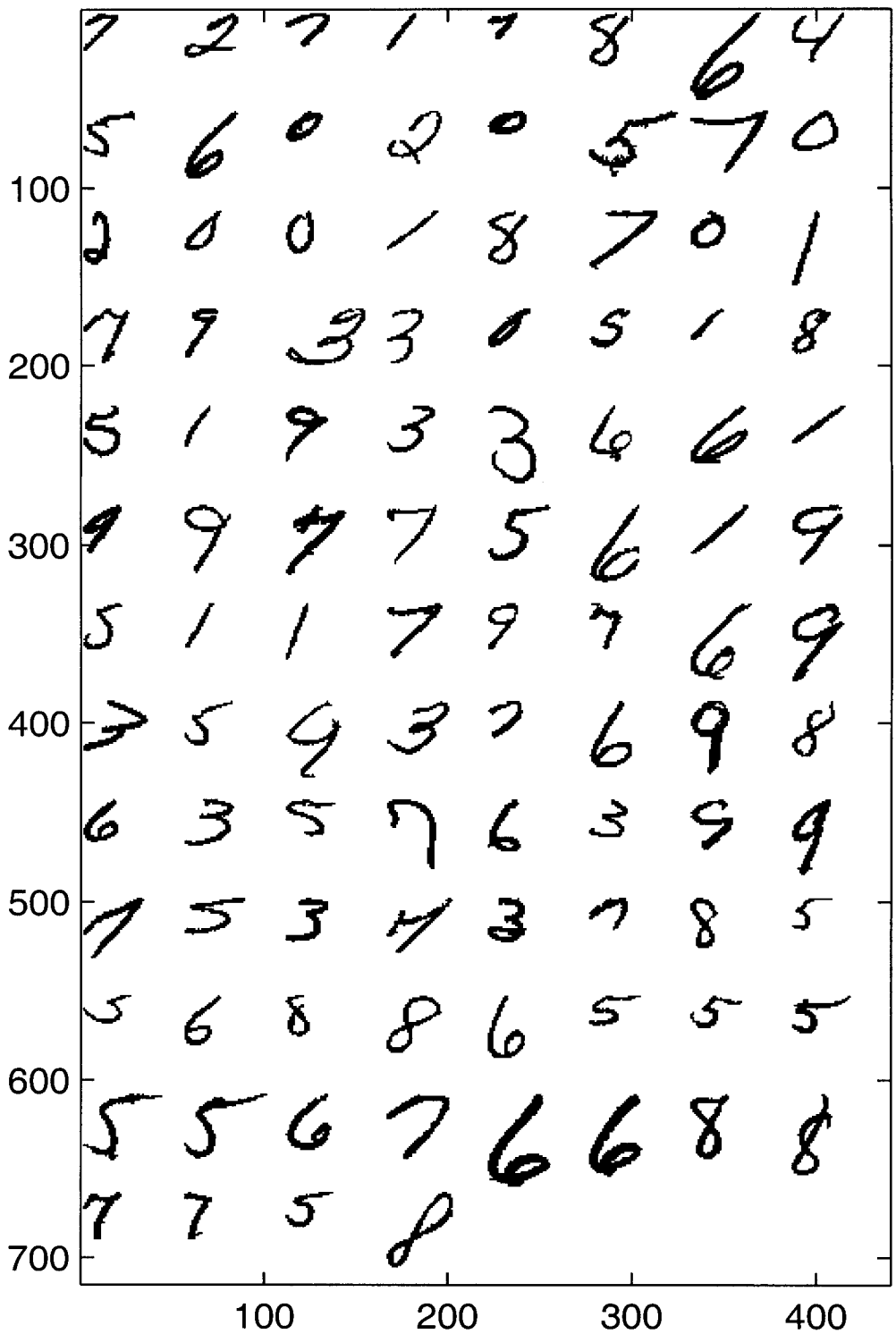

Figure 8: A sample of 100 handwritten numerals in the CENPARMI database.

Chapter 3

Rotation-invariant Pattern Recognition using Ridgelets

3.1 Introduction

Feature extraction is a crucial step in pattern recognition. In general, good features must satisfy the following requirements. First, *intraclass variance* must be small, which means that features derived from different samples of the same class should be close (e.g., numerically close if numerical features are selected). Secondly, the *interclass* separation should be large, i.e., features derived from samples of different classes should differ significantly. Furthermore, features should be independent of the size, orientation, and location of the pattern. This independence can be achieved by processing or by extracting features that are translation-, rotation-, and scale-invariant.

The Fourier transform has been a powerful tool for pattern recognition ([1], [38], [46], [75], [82], [44], [47]). One important property of the Fourier transform is that a shift in the time domain causes no change for the spectrum magnitude. This can be used to extract invariant features in pattern recognition. Translation invariance of a 2D pattern can be achieved by taking the spectrum magnitude of the 2D Fourier transform of the pattern. Rotation invariance can be done by performing 1D Fourier

transform along the angle direction in polar coordinates. Scale invariance can be accomplished by taking the *logarithm* of the polarized image and then conducting a 1D Fourier transform along the radius direction.

Wavelet transforms have proved to be very popular and effective in pattern recognition. Here we briefly review some of the previous works on pattern recognition by using wavelets. Bui et al. [4] proposed an invariant descriptor by using an orthonormal shell and the Fourier transform. The descriptor is invariant to translation, rotation, and scaling. Chen and Bui [13] invented an invariant descriptor by using a combination of the Fourier transform and a wavelet transform. They polarize the pattern first, and then perform a 1D wavelet transform along the radius direction and 1D Fourier transform along the angle direction. A more elaborate version of these two descriptors and their experiments is available in [11]. Yang et al. [80] proposed a novel approach for image recognition based on nonlinear wavelet approximation. They showed that the nonlinear wavelet approximation contains much more information of the original image than the linear wavelet approximation. Lee et al. [45] proposed a scheme for multiresolution recognition of unconstrained handwritten numerals using wavelet transform and a simple multilayer cluster neural network. The wavelet features of handwritten numerals at two decomposition levels are fed into the multilayer cluster neural network. Wunsch and Laine [77] gave a descriptor by extracting wavelet features from the outer contour of the handwritten characters and feeding the features into neural networks. Their experiments were done for handprinted characters. In Chapter 2 we developed a descriptor by using multiwavelets and neural networks. The multiwavelet features are also extracted from the outer contour of the handwritten numerals and fed into neural networks. This descriptor gives higher recognition rate than the one given in [77] for handwritten numeral recognition. Khalil and Bayoumi investigated to use wavelet modulus maxima for invariant 2D pattern recognition [41]. Tieng and Boles used wavelet zero-crossing to recognize 2D patterns [71]. Tao et al. proposed a technique for feature extraction by using wavelet and fractal [68]. Shen and Ip presented a set of wavelet moment

invariants for the classification of seemingly similar objects with subtle differences [60]. Tieng and Boles considered wavelet-based affine invariant pattern recognition in [72] and [73].

Recently ridgelet transform has been successfully proposed to analyze digital images ([33], [63], [35], [6], [7], [8], [30]). Unlike wavelet transforms, the ridgelet transform processes data by first computing integrals over different orientations and locations. A ridgelet is constant along the lines $x_1 cos\theta + x_2 sin\theta = constant$. Transverse to these ridges it is a wavelet. Ridgelet transform could be successfully applied in invariant pattern recognition. However, no known paper has considered this new transform for pattern recognition. This is because the ridgelet transform defined on a square is not suitable for extracting invariant features. In order to extract rotation-invariant features, we implement our ridgelet transform on a unit disk. This means we discard all those pixels that are outside the outer unit circle of the disk and consider only the region that is within the unit circle. Our implementation is quite similiar to the methods described in the literature, but we work on a disk. Details on our ridgelet implementation will follow later.

In this Chapter, we present two novel descriptors for feature extraction by using ridgelets, wavelet cycle-spinning, and Fourier transform. We extract ridgelet coefficients by doing the following. First, we normalize the pattern so that it is translation- and scale-invariant. Second, we discard all those pixels of the pattern that fall outside the circle containing the pattern. This means we only work on the region that is within the circle. Third, we project the pattern on each slice segment that passes through the center of the circle and ends at the boundary of the circle. These slice segments are equally spaced in angle. This is the Radon transform, but it works on a disk rather than a square. Fourth, we perform 1D wavelet transform along each Radon slice so that we get our ridgelet transform coefficients. In order to achieve rotation invariance, we do it in the following two ways. First, we conduct 1D Fourier transform along the angle direction so that the rotation variance can be eliminated. Second, we apply wavelet cycle-spinning in the angle direction and take the Fourier

spetra for each subband. Experimental results show that these two descriptors are excellent choices for pattern recognition, and they are also robust to noise.

The organization of the Chapter is as follows. In Section 3.2, we explain wavelet cycle-spinning and how to match the features of a pattern in the cycle-spinning tree. In Section 3.3, we propose two rotation-invariant descriptors by using ridgelets, wavelet-cycle-spinning, and Fourier transform. In Section 3.4, we conduct some experiments on a database of Chinese characters and get very high recognition rate. Noisy data are also considered in our experiments.

3.2 Wavelet Cycle-Spinning

The cycle-spinning table, also called translation-invariant (TI) table, is for discrete scalar wavelet transform [31] and multiwavelet transform [3]. The cycle-spinning table of a signal is an $n \times (J+1)$ matrix S with $J = \log_2 n$. Every row i of the matrix is partitioned into $n/2^{i-1}$ blocks. These blocks contain all the wavelet coefficients at scale i for different shifts. The first row of the matrix is used to store the low pass coefficients. This matrix is dynamically filled during each resolution scale. The fill-in of the cycle-spinning table is fulfilled by a series of decimation and filtering operations. Let g and h stand for the usual down-sampling high pass and low pass operations of the wavelet theory. Also let R_h stand for circular shift by h. Set $S_{0,0} = f$, the 1D signal, and initialize

$$D_{1,0} = gR_0S_{0,0}; \quad D_{1,1} = gR_1S_{0,0},$$

$$S_{1,0} = hR_0S_{0,0}; \quad S_{1,1} = hR_1S_{0,0}.$$

then, the recursive equations follow

$$D_{j+1,2k} = gR_0S_{j,k}; \quad D_{j+1,2k+1} = gR_1S_{j,k},$$

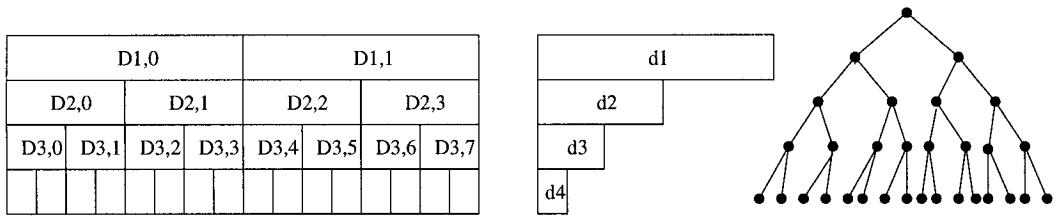

Figure 9: Wavelet cycle-spinning table and the tree for calculating the distance between the unknown pattern and the pattern database.

$$S_{j+1,2k} = hR_0S_{j,k}; \quad S_{j+1,2k+1} = hR_1S_{j,k}.$$

for $j = 1, \ldots, J-1$ and $k = 0, 1, \ldots, 2^{j-1} - 1$.

Fig. 9 illustrates how we can match the features of the unknown pattern to the cycle-spinning feature tree in the feature database. As we know, the wavelet cycle-spinning table contains the wavelet coefficients of a signal for all the shifts. For a signal with given shift, we can perform 1D wavelet transform and calculate the Fourier spectrum of the wavelet subbands $\{d_J, d_{J-1}, \cdots, d_3, d_2, d_1\}$. Because the Fourier spectrum is symmetric, we only keep half of the coefficients. This can make the processing time faster. In order to calculate the distance between the spectrum features of the unknown pattern and the cycle-spinning table of the pattern database, we take the difference between d_i and every $D_{i,k}$ for $0 \leq k < 2i$ in the ith row of the cycle-spinning table. If we represent every wavelet subband as a node, then we can denote the wavelet cycle-spinning table as a tree. Every node in the tree has two children except the leaves. The distance can be evaluated from bottom up. For every node, we take the minimum distance of the two children and add it to the distance of the current node. Finally at the top, we get the distance between the unknown pattern and the wavelet cycle-spinning table.

3.3 Rotation-invariant Pattern Recogniton using Ridgelets

Projection histograms have been successfully used in many applications such as segmenting characters, words, and text lines, as well as detecting the orientation of an image. Basically, the horizontal projection $h(x_i)$ is defined as the number of pixels that have the same x-coordinate x_i. Similarly, the vertical projection $v(y_i)$ is defined as the number of pixels that have the same y-coordinate y_i. Even though these projections are very useful in many applications, they have such limitations as very sensitive to rotation and writing style. Also, some important information is lost due to the projections.

In this section, we propose two novel descriptors that overcome all these limitations and have rotationally invariant properties. As we know, ridgelets have been recently developed for image processing applications. However, so far ridgelets have only been investigated on a square image. This makes it difficult to extract invariant features by using ridgelets over a square. We successfully overcome this issue by implementing ridgelets on a disk containing the pattern image. To be more specific, we discard all those pixels of the pattern that are outside the outer circle of the disk and consider only the region that is within the circle. The ridgelet transform of the pattern image can be done by applying 1D wavelet transform on each slice of the Radon transform. The slices of the Radon transform can be obtained by summing all the intensity values of those pixels that are within the circle surrounding the pattern to be recognized and on the line that is perpendicular to the ridge. The length of our ridge is the same in the horizontal and vertical directions compared to ordinary ridgelet transform. However, it is shorter for all other orientations especially in the diagonal directions. Note that the variable r in the Radon domain $R(r, \theta)$ is discretized using the same dimension as the diameter of the circle, and we select twice the dimension for θ. Due to the orientation changes of the pattern image, we arrange the Radon slices in counter-clockwise direction in terms of θ. Therefore, the same slice is saved twice but

in reverse order, one in the normal order for orientation θ ($0 \leq \theta < \pi$) and the other one for $\theta + \pi$. After storing the slices in this way, we can only get circularly shifted rows for the Radon slice matrix no matter how the pattern image is rotated.

In order to eliminate the rotational variance, we can perform 1D Fourier transform along the θ direction and get the spectrum magnitude. Therefore, we have successfully extracted rotation-invariant features from the pattern image. The translation- and scaling-invariance can be achieved by the normalization techniques mentioned before.

The steps of our first invariant Ridgelet-Fourier descriptor (*Descriptor A*) can be listed as follows:

1. Normalize the pattern $f(x_1, x_2)$ so that it is translation- and scale-invariant. The pattern is represented as an $n \times n$ matrix.

2. Discard all those pixels that fall outside the circle with center at $(n/2, n/2)$ and radius $n/2$.

3. Project the pattern within the circle onto different orientation slices so that we get the Radon transform coefficients. The orientation slices are line segments that pass through the center of the circle and end at the two intersection points of the line with the circle. The orientation angles of the slices are ordered in counter-clockwise direction with incremental angle step π/n.

4. Apply 1D wavelet transform on each Radon slice to get the ridgelet coefficients $R(r, \theta)$.

5. Perform 1D Fourier transform along the θ direction and take the spectrum magnitude to obtain the invariant feature $FR(r, \theta)$.

6. Use the resulting invariant features to query the pattern feature database at different resolution scales.

Fig. 10 shows the steps of out Ridgelet-Fourier descriptor in the following order: a 2D pattern, the Radon transform coefficients, the ridgelet transform coefficients, and the invariant Fourier spectrum magnitude.

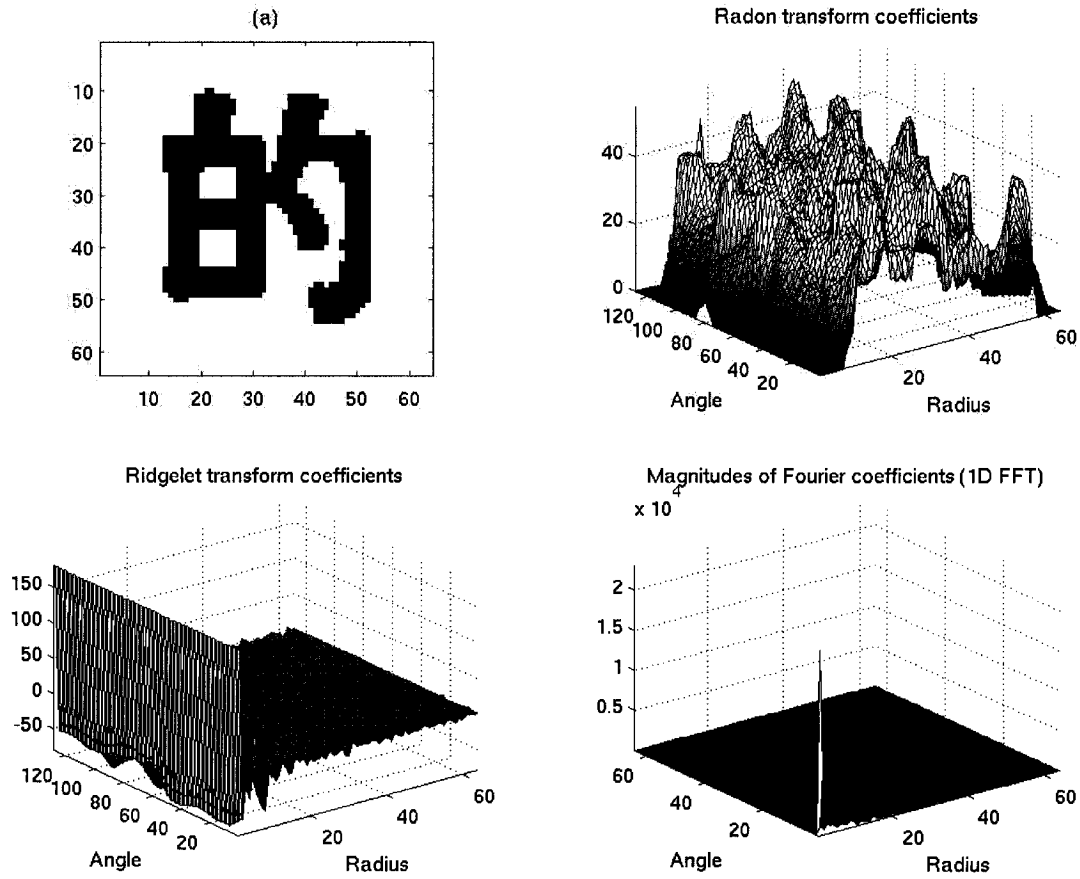

Figure 10: The steps of our Ridgelet-Fourier descriptor A.

The other way to achieve rotational invariance can be done by performing wavelet cycle-spinning along the angle direction and get the Fourier spectrum magnitude of each wavelet subband. These features are stored in the feature database as cycle-spinning tables for each pattern in the pattern database. For the unknown pattern, we conduct 1D wavelet transform along the angle direction and get the Fourier spectrum magnitude of each wavelet subband. We can guarantee that these subband features will match exactly one branch of the cycle-spinning table tree. The distance between the unknown pattern and the patterns in the pattern database can be done following the idea in the previous section. The only difference is that we work on multiple streams of data at the same time instead of a 1D signal.

The steps of operations for the patterns in the pattern database can be described as follows:

1. Normalize the pattern so that it is translation- and scale-invariant.
2. Discard all those pixels that are outside the surrounding circle with center $(n/2, n/2)$ and radius $n/2$.
3. Project the pattern in different orientations to get the Radon transform coefficients.
4. Perform 1D wavelet transform in the Radon domain along the radius direction to obtain the ridgelet coefficients.
5. Conduct 1D wavelet cycle-spinning along the angle direction and get the Fourier spectrum of every wavelet subband.
6. Save these features into the feature database.

The steps of our rotation-invariant descriptor for the unknown pattern can be described as follows:

1. Normalize the unknown pattern so that it is translation- and scale-invariant.

2. Discard all those pixels that are outside the surrounding circle with center $(n/2, n/2)$ and radius $n/2$.

3. Project the pattern in different orientations to get the Radon transform coefficients.

4. Perform 1D wavelet transform in the Radon domain along the radius direction to obtain the ridgelet coefficients.

5. Conduct 1D wavelet transform along the angle direction and get the Fourier spectrum of every wavelet subband.

6. Use the resulting features to query the pattern feature database at different resolution scales.

We call this descriptor as *Descriptor B*.

The main advantages of our proposed descriptors are that we project the pattern onto all different orientation slices. Therefore, important features are extracted in the Radon slices. The wavelet transform on each Radon slice also gives us multiresolution representation in the feature space. In addition, the Fourier spectrum is a very popular tool to eliminate translation/shift variance. All these good properties join in our proposed descriptors, making it a very successful choice in invariant feature extraction. We conduct some experiments on a database of printed Chinese characters by using our proposed descriptor and find that it is an excellent choice for pattern recognition. Note that our proposed descriptors work for not only Chinese characters, but also any other 2D pattern recognition problems.

3.4 Experimental Results

In order to test the efficiency of our new descriptors, we use the same set of 85 printed Chinese characters in our experiment as in [4], [13], [11], and [83]. Fig. 11 shows the database of our 85 printed Chinese characters. Each original Chinese

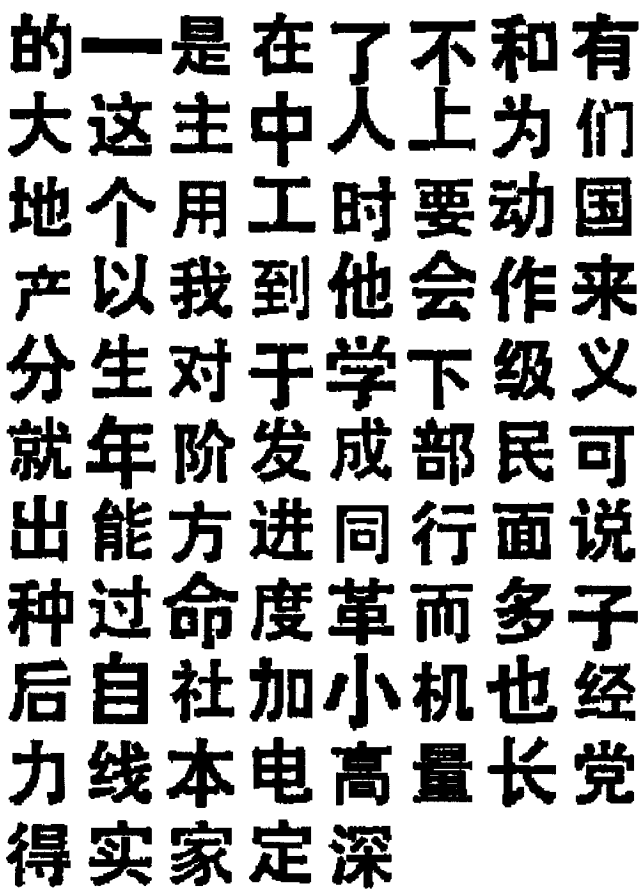

Figure 11: The Chinese character database used.

character is represented by 64×64 pixels. The size of the pattern in the Radon domain is 128×64, and so is the number of ridgelet coefficients $R(r, \theta)$. We conduct experiments for *Descriptor A* and *Descriptor B* separately.

3.4.1 Descriptor A

For this descriptor, the size of the invariant features $FR(r, \theta)$ is 64×64. This is because the spectrum of 1D Fourier transform is symmetric. So we only keep half of the Fourier coefficients. Because translation will not change the relative position of the centre of mass of the character, our major concern is the performance of the system on rotation and scaling. For each character, we tested ten rotation angles and

five scaling factors. The ten rotation angles are 30^o, 60^o, 90^o, 120^o, 150^o, 180^o, 210^o, 230^o, 240^o and 270^o, and the five different scaling factors are 0.4, 0.6, 0.8, 1.0 and 1.2. Fig. 12 shows a combination of rotation and scaling factors for the printed Chinese character "zai".

We use four kinds of wavelet transforms in our experiments, namely, Haar, Daubechies-4, Coiflet-6, and Symmlet-8. Table 1 gives the recognition rates for different rotation angles and scaling factors. The invariant features used here are d_3 and d_4 subbands. The distance metric L_1 is also used in this table. We find the Daubechies-4 obtains the best recognition rates for all rotation angles and scaling factors. It is clear that the choice of a wavelet makes a difference in the recognition process.

The wavelet coefficients of an image have multi-resolution representation of the original image. The coarse resolution wavelet coefficients normally represent the global shape of the image, while the fine resolution coefficients represent the details of the image. Due to noise introduced in the original image and the errors accumulated in the process of computation, the detail coefficients are becoming less important than the intermediate scale coefficients. Also, the low frequency wavelet subbands have lost important information of the original image. Therefore, it is desirable to use intermediate scale wavelet coefficients as robust features in the classification phase. We test the descriptor by using Daubechies-4 wavelet and the ridgelet coefficients at different resolution scales. Table 2 show the experimental results of our *Descriptor A* with regard to recognition rates by using features at different scales. It is clear that intermediate scales d_3 and d_4 carry the most significant invariant features and achieve the highest recognition rates for all orientation angles and scaling factors. The high frequency subbands d_1 and d_2 are very sensitive to noise and accumulation errors. Also, the average s_6 and the low frequency subbands d_6, d_5 have lost important information. Therefore, all these subbands are not very good for pattern recognition. On the contrary, the intermediate frequency subbands d_3 and d_4 are our best choices.

We also test the noise tolerance and sensitivity of our proposed descriptor. The noisy images with different orientations are generated by adding white noise to the

noise-free images. The signal to noise ratio (SNR) is defined as

$$\text{SNR} = \frac{\sqrt{\sum_{i,j}(f_{i,j} - avg(f))^2}}{\sqrt{\sum_{i,j}(n_{i,j} - avg(n))^2}}$$

where f is the noise-free image, n is the added white noise, and $avg(f)$ is the average value of the image f. Fig. 13 shows some noisy patterns for SNR = 20, 15, 10, 5, 4, 3, 2, 1, and 0.5, respectively. We test our descriptor for SNR = 20, 15, 10, 5, 4, 3, 2, 1 and 0.5, and for rotation angle = 30^o, 60^o, 90^o, 120^o, 150^o, 180^o, 210^o, 240^o, and 270^o. Daubechies-4 wavelet is used in this experiment. The features d_3 and d_4 are used in the classification phase. Also, two different distance metrics, L_1 and L_2, are employed. The results are tabulated in Table 3 and Table 4, respectively. Our experiments verify that our proposed descriptor A is very much robust to white noise. Also, L_1 metric gives slightly better results than L_2 metric. However, the two distance metrics do not make too much difference.

We compare the recognition rates of our descriptor A with those of [4], [13], and [11]. In [4] an invariant descriptor using orthonormal shell and the Fourier transform was proposed. The descriptor is invariant to translation, rotation, and scaling. In [13] we described an invariant descriptor by using a combination of Fourier transform and wavelet transform. The pattern was first polarized, then a 1D wavelet transform along the radius direction and 1D Fourier transform along the angle direction are performed. A more elaborate version of these two descriptors and their experiments is available in [11]. We use the same database for our proposed Ridgelet-Fourier descriptor and for [4], [13], and [11]. Our descriptor A consistently gets higher recognition rates for all combinations of rotation angles and scaling factors. For noisy pattern recognition, the results are shown in Table 3 and Table 4, and Table 5 for our descriptor and the one used in [13], respectively. It is clear that our descriptor A is much better than the descriptor described in [13]. For $SNR = 0.5$, it is even difficult for humans to recognize the noisy patterns, however note that the proposed descriptor A performs very well. This indicates that our descriptor A is very robust to noise even when the

Different Wavelets	*Scaling Factor*	*Rotation*								
		30^o	60^o	90^o	120^o	150^o	180^o	210^o	240^o	270^o
Haar	1.2	98.82	98.82	98.82	98.82	97.65	97.65	98.82	98.82	98.82
	1.0	100	100	100	100	100	100	100	100	100
	0.8	100	100	100	100	100	100	100	100	100
	0.6	100	100	100	98.82	98.82	100	98.82	100	100
	0.4	92.94	95.29	95.29	88.24	88.24	95.29	91.76	89.41	97.65
Daubechies-4	1.2	100	100	100	100	100	100	100	100	100
	1.0	100	100	100	100	100	100	100	100	100
	0.8	100	100	100	100	100	100	100	100	100
	0.6	100	100	100	100	100	100	100	100	100
	0.4	97.65	100	100	97.65	95.29	98.82	96.47	97.65	98.82
Coiflet-6	1.2	100	100	98.82	98.82	98.82	95.29	97.65	100	100
	1.0	100	100	100	100	100	100	100	100	100
	0.8	100	100	100	100	100	100	100	100	100
	0.6	100	100	100	98.82	98.82	100	98.82	100	100
	0.4	97.65	100	97.65	97.65	95.29	98.82	95.29	96.47	98.82
Symmlet-8	1.2	100	100	97.65	97.65	95.29	97.65	97.65	97.65	97.65
	1.0	100	100	100	100	100	100	100	100	100
	0.8	100	100	100	100	100	100	100	100	100
	0.6	100	100	100	100	100	100	98.82	100	100
	0.4	98.82	98.82	97.65	97.65	96.47	98.82	95.29	96.47	98.82

Table 1: The recognition rates for different wavelets.

noise level is very high.

We also tested our descriptor under occlusion. For every pattern in the pattern database, we cut off the upper left-hand side corner and set the pixel values to zero. Fig. 14 shows an original Chinese character and a series of more and more partially occluded images. We produce the occluded characters for the whole pattern database, and test the recognition rates of our proposed descriptor for all combinations of rotation and scaling. The ridgelet-Fourier works well for low to intermediate levels of occlusions and it breaks down when we trim more than 50% of the pixels of the original image. Table 6 gives the recognition rates for the occlusion case (c) in Fig. 14. It is clear that the proposed descriptor is quite robust to occlusion.

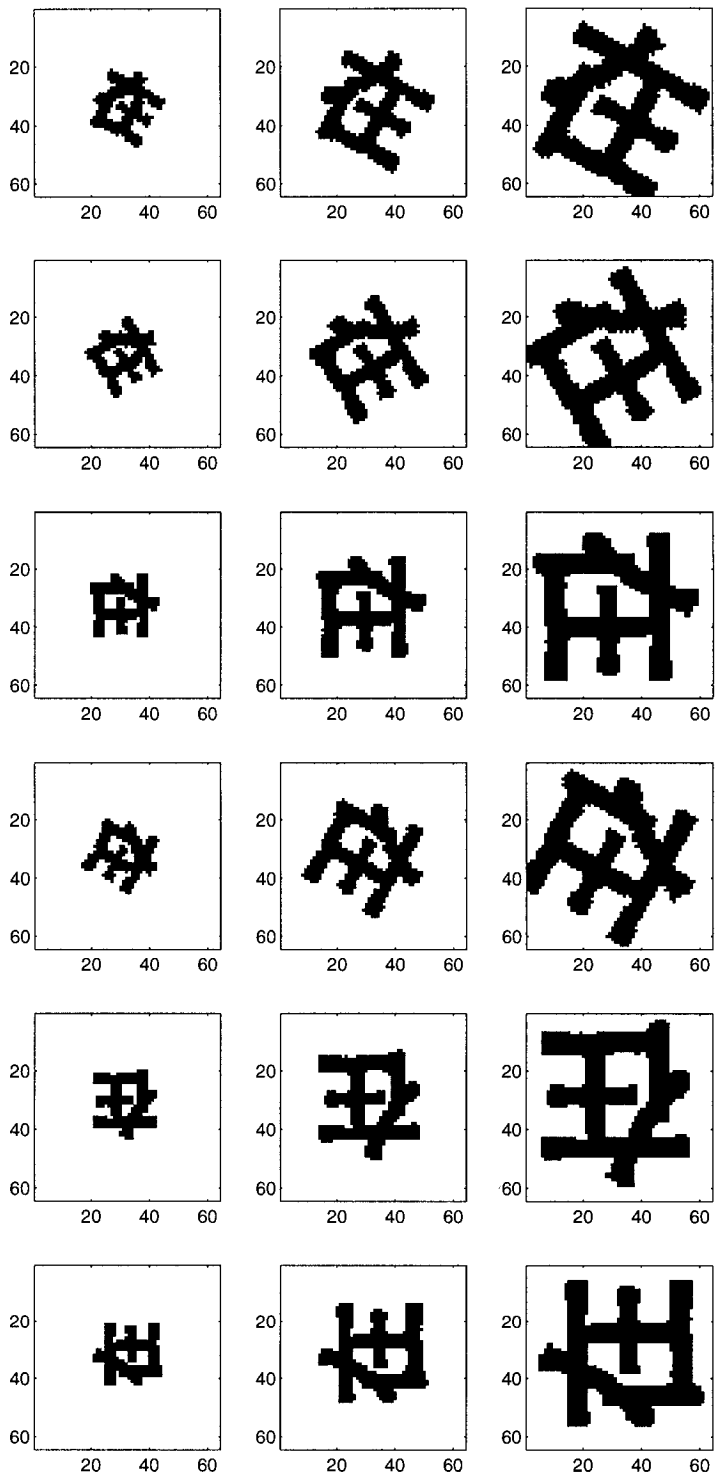

Figure 12: A combination of rotation and scaling factors for character "zai"

Wavelet features	*Scaling Factor*	*Rotation*								
		30^o	60^o	90^o	120^o	150^o	180^o	210^o	240^o	270^o
d_2,d_3,d_4	1.2	100	100	100	100	100	100	100	100	100
	1.0	100	100	100	100	100	100	100	100	100
	0.8	100	100	100	100	100	100	100	100	100
	0.6	100	100	100	100	100	100	98.82	100	100
	0.4	96.47	100	100	96.47	96.47	97.65	94.12	98.82	98.82
d_2,d_3	1.2	100	100	100	100	100	98.82	98.82	100	100
	1.0	100	100	100	100	100	100	100	100	100
	0.8	100	100	100	100	100	100	100	100	100
	0.6	100	100	100	100	100	100	98.82	100	100
	0.4	96.47	97.65	97.65	94.12	92.94	95.29	90.59	90.59	97.65
d_3,d_4	1.2	100	100	100	100	100	100	100	100	100
	1.0	100	100	100	100	100	100	100	100	100
	0.8	100	100	100	100	100	100	100	100	100
	0.6	100	100	100	100	100	100	100	100	100
	0.4	97.65	100	100	97.65	95.29	98.82	96.47	97.65	98.82
d_3,d_4,d_5	1.2	100	100	100	100	100	100	100	100	100
	1.0	100	100	100	100	100	100	100	100	100
	0.8	100	100	100	100	100	100	100	100	100
	0.6	100	100	100	100	100	100	100	100	100
	0.4	95.29	95.29	100	95.29	92.94	96.47	94.12	95.29	98.82
d_3,d_4,d_5,d_6	1.2	100	100	100	100	100	100	100	100	100
	1.0	100	100	100	100	100	100	100	100	100
	0.8	100	100	100	100	100	100	100	100	100
	0.6	100	100	100	100	98.82	100	98.82	100	100
	0.4	94.12	90.59	95.29	92.94	87.06	91.76	85.88	90.59	95.29
d_4,d_5,d_6	1.2	100	100	100	100	100	98.82	100	100	98.82
	1.0	100	100	100	100	100	100	100	100	100
	0.8	100	100	100	100	100	100	100	100	100
	0.6	100	97.65	100	98.82	95.29	100	96.47	94.12	100
	0.4	78.82	87.06	78.82	71.76	69.41	70.59	67.06	71.76	81.18
d_4,d_5	1.2	100	100	100	100	98.82	98.82	98.82	98.82	98.82
	1.0	100	100	100	100	100	100	100	100	100
	0.8	100	100	100	100	100	100	100	100	100
	0.6	100	98.82	100	98.82	97.65	100	98.82	100	100
	0.4	94.12	90.59	90.59	88.24	85.88	85.88	83.53	87.06	88.24

Table 2: The recognition rates for different rotation and scaling factors. Daubechies-4 is used in this table.

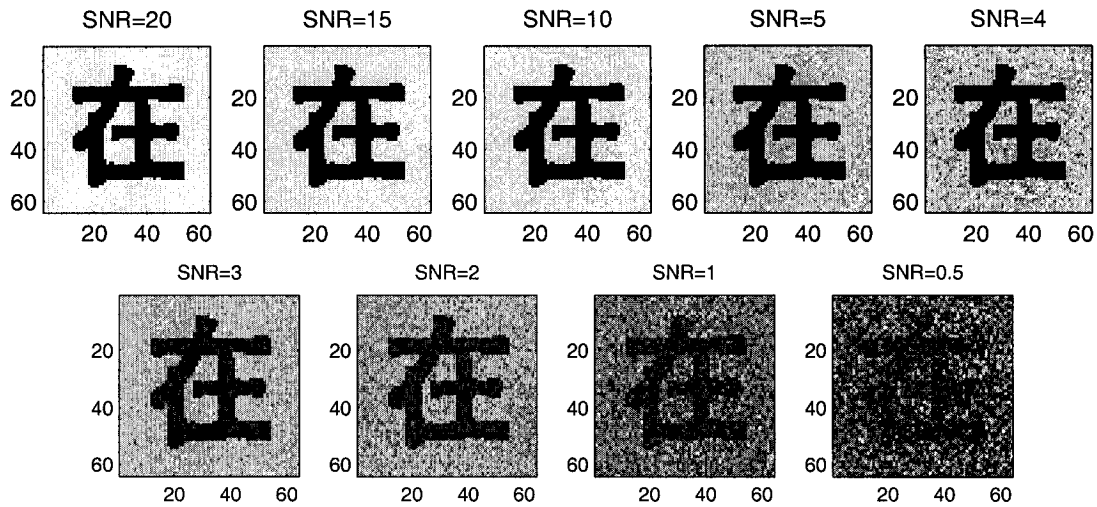

Figure 13: The noisy patterns with SNR = 20, 15, 10, 5, 4, 3, 2, 1, and 0.5, respectively.

Different	*Rotation*								
SNR	30^o	60^o	90^o	120^o	150^o	180^o	210^o	240^o	270^o
20	100	100	100	100	100	100	100	100	100
15	100	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100	100
5	100	100	100	100	100	100	100	100	100
4	100	100	100	100	100	100	100	100	100
3	100	100	100	100	100	100	100	100	100
2	100	100	100	100	100	100	100	100	100
1	100	100	100	100	100	100	100	100	100
0.5	98.82	100	100	100	100	100	100	98.82	100

Table 3: The recognition rates for different SNR's using L_1 distance by using features d_3 and d_4.

Different	*Rotation*								
SNR	30^o	60^o	90^o	120^o	150^o	180^o	210^o	240^o	270^o
20	100	100	100	100	100	100	100	100	100
15	100	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100	100
5	100	100	100	100	100	100	100	100	100
4	100	100	100	100	100	100	100	100	100
3	100	100	100	100	100	100	100	100	100
2	100	100	100	100	100	100	100	100	100
1	100	100	100	100	100	100	98.82	100	100
0.5	100	100	100	100	100	98.82	100	97.65	100

Table 4: The recognition rates for different SNR's using L_2 distance by using features d_3 and d_4.

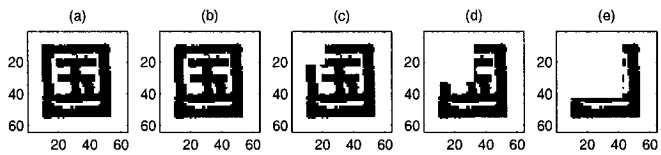

Figure 14: The original and occluded patterns.

3.4.2 Descriptor B

We give the memory requirement of this descriptor here. The wavelet cycle-spinning table has a dimension of $128 \times 64 \times 9$. Since the spectrum of 1D Fourier transform is symmetric, we only keep half of the Fourier spectrum of every wavelet subband. Therefore, the size of the final invariant features in the feature database is $64 \times 64 \times 9$.

Our major concern in our experiments is the performance of the system on rotation. For each character, we tested ten rotation angles 30^o, 60^o, 90^o, 120^o, 150^o, 180^o, 210^o, 230^o, 240^o and 270^o. We use two intermediate wavelet subbands, d_3 and d_4, for the wavelet transform in the process of obtaining ridgelet coefficients. We use three wavelet subbands, D_4, D_5 and D_6, for wavelet cycle-spinning. Daubechies-4 wavelet is used in this experiment. For all these rotation angles, we get 100% recognition rate without errors.

We also test the performance of our proposed descriptor B on noisy data. The

Different	*Rotation*								
SNR	30^o	60^o	90^o	120^o	150^o	180^o	210^o	240^o	270^o
20	100	100	100	100	100	100	100	100	100
15	100	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100	100
5	100	100	100	100	100	100	100	100	100
4	100	100	100	100	100	100	100	100	100
3	100	100	100	100	100	100	100	100	100
2	100	100	100	97.65	97.65	98.82	100	98.82	100
1	90.59	89.41	89.41	85.88	84.71	87.06	87.06	84.71	89.41
0.5	60.00	55.29	56.47	44.71	50.59	48.24	45.88	47.06	60.00

Table 5: The recognition rates of [13] for different SNR's.

Scaling	*Rotation*								
Factor	30^o	60^o	90^o	120^o	150^o	180^o	210^o	240^o	270^o
1.2	97.65	97.65	96.47	97.65	97.65	95.29	97.65	97.65	96.47
1.0	100	100	100	100	100	100	100	100	100
0.8	100	100	100	97.65	98.82	98.82	98.82	98.82	98.82
0.6	98.82	98.82	97.65	96.47	96.47	95.29	98.82	98.82	98.82
0.4	95.29	95.29	95.29	90.59	88.24	82.35	88.24	91.76	90.59

Table 6: The recognition rates of the proposed descriptor for the occlusion case (c) in Fig. 14.

Different	*Rotation*								
SNR	30^o	60^o	90^o	120^o	150^o	180^o	210^o	240^o	270^o
20	100	100	100	100	100	100	100	100	100
15	100	100	100	100	100	100	100	100	100
10	100	100	100	100	100	100	100	100	100
5	100	100	100	100	100	100	100	100	100
4	100	100	100	100	100	100	100	100	100
3	100	100	100	100	100	100	100	100	100
2	100	100	100	100	100	100	100	100	100
1	100	100	100	98.82	98.82	100	100	100	100
0.5	97.65	97.65	100	88.24	94.12	97.65	90.59	89.41	100

Table 7: The recognition rates for different SNR's.

noisy images with different orientations are generated by adding white noise to the noise-free images. Daubechies-4 wavelet is used in this experiment. The same features are used as before. The results are listed in Table 7. It is clear that our proposed descriptor is very robust to white noise.

We also compare the recognition rates of our proposed descriptor with those of [4], [13], and [11]. Chen did some experiments in [11] by using the descriptor given in [13] and got about 97% recognition rate when $SNR = 2$. By using our proposed descriptor and setting $SNR = 2$, we obtain 100% recognition rate for all the rotation angles. In addition, we get nearly 100% recognition rate for $SNR = 1$ and nearly 95% for $SNR = 0.5$. This indicates that our proposed descriptor B is very robust to Gaussian white noise even when the noise level is high.

By looking at the recognition rates for algorithm A and B, we find that algorithm A performs better than algorithm B when the noise level is high. However, when the nosie level is low or moderate, the two algorithms have similiar performance.

Chapter 4

Signal Denoising using neighbouring coefficients[2]

4.1 Introduction

Let $g(t)$ be the noise-free signal and $f(t)$ the signal corrupted with white noise $z(t)$, i.e., $f(t) = g(t) + \sigma z(t)$, where $z(t)$ has a normal distribution $N(0,1)$. The basic procedure to denoising is transforming the signal into wavelet domain, thresholding the wavelet coefficients, and then performing the inverse wavelet transform. Traditional approaches use a term-by-term method that does not consider the influence of other wavelet coefficients on the wavelet coefficient being thresholded.

Many authors have proposed denoising schemes that consider the influence of other wavelet coefficients in addition to the current coefficient. Cai and Silverman [5] gave a new thresholding scheme by taking the immediate neighbour coefficients into account. They claimed that this approach gives better results over the traditional term-by-term approach for both translation invariant (TI) and non-TI single wavelet denoising. Simoncelli and Adelson [62] proposed a bayesian wavelet coring approach by incorporating the higher-order statistical regularity present in the point statistics

[2]This work was published in *IEEE Signal Processing Letters* [12].

of subband representation. Crouse et al. [23] developed a framework for statistical signal processing based on wavelet-domain hidden markov models (HMM). The framework enables us to concisely model the non-Gaussian statistics of individual wavelet coefficients and capture statistical dependencies between coefficients. Sendur and Selesnick [58] developed a bivariate shrinkage function for denoising. Their results showed that the estimated wavelet coefficients depend on the parent coefficients. The smaller the parent coefficients, the greater the shrinkage. Mihcak et al. [54] performed an approximate maximum *a posteriori* (MAP) estimation of the variance for each coefficient, using the observed noisy data in a local neighbourhood. Then an approximate minimum mean squared error estimation procedure is used to restore the noisy image coefficients.

In this Chapter we extend Cai and Silverman's idea to the multiwavelet case. Multiwavelets have been developed by using translates and dilates of more than one mother wavelet functions ([36], [67], [78]). In multiwavelet transform, the low-pass and high-pass filters are two sets of $M \times M$ matrices. We need to perform a prefilter on the 1-D input signal in order to convert the 1-D input signal to multiple stream data. Also, we need to do a postfilter to transform from the multiple stream data into a 1-D signal. This postfiltered signal is equal to the original signal if no thresholding is performed. Multiwavelets are known to have several advantages over single wavelets such as short support, orthogonality, symmetry, and higher order of vanishing moments. It is claimed that multiwavelet denoising outperforms single wavelet denoising in [67], [34] and [3]. The experimental results in this Chapter show that by using neighbouring coefficients we get smaller mean-square-errors (MSE) for both TI and non-TI multiwavelet denoising. Also, we find that neighbour multiwavelet denoising outperforms neighbour single wavelet denoising for some signals, e.g. *Blocks* and *Doppler*, and real life images.

The organisation of this Chapter is as follows. Section 4.2 explains how we can incorporate neighbouring multiwavelet coefficients into signal denoising. Section 4.3 shows some experimental results.

4.2 Incorporating Neighbouring Coefficients in Multiwavelet Denoising

The basic motivation of neighbour thresholding is that: if the current coefficient contains some signal, then it is likely that the two neighbour coefficients also do. For this reason, at each location we threshold every coefficient by using the coefficient at that location as well as the coefficients of the two neighbours. Cai et al [5] proposed the following thresholding scheme for single wavelet denoising. If $S_{j,k}^2 = d_{j,k-1}^2 + d_{j,k}^2 + d_{j,k+1}^2$ is less than or equal to λ^2, then we set the wavelet coefficient $d_{j,k}$ to zero. Otherwise, we set it to

$$d_{j,k} = d_{j,k}(1 - \lambda^2 / S_{j,k}^2)$$

where $\lambda = \sqrt{2\sigma^2 \log n}$, λ is the variance of the Gaussian noise and n is the length of the signal. This threshold λ was proposed by Donoho and his coworkers ([31], [32]) and it is shown to be an optimal threshold. It should be mentioned that if $d_{j,k}$ is at the left (right) boundary of level j wavelet coefficients, we omit the first (last) term in $S_{j,k}^2$.

For multiwavelet denoising, we have multi-streams of multiwavelet coefficients $D_{j,k}$. Suppose we apply the discrete multiwavelet transform with an appropriate prefilter to a noisy function, then we get M stream coefficients of the form, $D_{j,k} = D_{j,k}^* + E_{j,k}$, where $E_{j,k}$ has a multivariate normal distribution $N(0, V_j)$. The matrix V_j is the covariance matrix for the error term which depends on the resolution scale j. Using the standard transform $\theta_{j,k} = D_{j,k}^T V_j^{-1} D_{j,k}$, we obtain a positive scalar value which in the absence of any signal component will have a χ_M^2 distribution, where M is the number of elements in $D_{j,k}$. It is these $\theta_{j,k}$ values that are thresholded, and the coefficients vectors can then be adapted accordingly. More detailed explanation on the optimal threshold and the χ_M^2 distribution of $\theta_{j,k}$ is given in [34].

We can use a robust covariance estimation method to estimate V_j directly from

the observed coefficients [40]. The pseudo code for robust covariance estimation can be listed as follows:

$$
\begin{aligned}
&\#\text{define mad}(y)\ 1.4826 * \text{median}(\text{abs}(y - \text{median}(y))) \\
&\text{float } a_1, a_2, b_1, b_2, V_j[2][2]; \\
&a_1 \quad = \quad 1.0/\text{mad}\ (row_1); \\
&a_2 \quad = \quad 1.0/\text{mad}\ (row_2); \\
&b_1 \quad = \quad \text{mad}\ (a_1 * row_1 + a_2 * row_2); \\
&b_2 \quad = \quad \text{mad}\ (a_1 * row_1 - a_2 * row_2); \\
&V_j[1][1] \quad = \quad 1/(a_1 * a_1); \\
&V_j[2][2] \quad = \quad 1/(a_2 * a_2); \\
&V_j[1][2] \quad = \quad (b_1 - b_2)/((b_1 + b_2) * a_1 * a_2); \\
&V_j[2][1] \quad = \quad V_j[1][2];
\end{aligned}
$$

In the first row, y is a row vector and we give the definition of mad(y). We use this definition to calculate a_1, a_2, b_1, and b_2 by replacing y with row_1, row_2, etc. Here row_1 and row_2 are the two rows of multiwavelet coefficients at scale j that contains $D_{j,k}$.

Applying the highpass and lowpass multiwavelet filters will spread the dependence between multiwavelet vector coefficients. A large multiwavelet coefficient will probably have large coefficients as its neighbours. Therefore, incorporating neighbour multiwavelet coefficients in the thresholding step should also work for signal denoising. Let r be a non-negative integer and $S^r_{j,k} = \theta^r_{j,k-1} + \theta^r_{j,k} + \theta^r_{j,k+1}$, then the neighbour multiwavelet thresholding formula is given by

$$
\hat{D}_{j,k} = \begin{cases} D_{j,k}(1 - \frac{\mu^r}{S^r_{j,k}}) & \text{if } S^r_{j,k} \geq \mu^r \\ 0 & \text{otherwise} \end{cases}
$$

where μ is given by $2 \log n$. Since μ is a threshold for the multiwavelet coefficients $\theta_{j,k}$, we know that μ^r is a threshold for $\theta^r_{j,k}$. If we want to incorporate the immediate

neighbour coefficients into the thresholding formula, we can threshold the above defined $S^r_{j,k}$ using μ^r as a threshold. When $r = 1$ and the coefficients are not correlated, $S^r_{j,k}$ will have a χ^2_{3M} distribution in the absence of any signal component. This means a larger μ should be chosen for neighbour multiwavelet denoising [34]. However, since the multiwavelet coefficients may be correlated, it is not easy to analyze it theoretically. In our experiments, μ is set to the same value $2\log n$ for different r. We also find that $r = 2$ always performs better than $r = 1$. So we prefer $r = 2$ when using neighbour multiwavelet denoising.

We can give the neighbour multiwavelet denoising algorithm as follows:

1. Use a prefilter to change the original 1-D noisy signal into multiple stream data.
2. Perform forward multiwavelet transform (TI or non-TI) on this multiple data.
3. Apply the thresholding scheme by using neighbouring multiwavelet coefficients.
4. Perform inverse multiwavelet transform (TI or non-TI) on the thresholded multiple stream data.
5. Apply a postfilter to the denoised multiple stream data to get the denoised 1-D signal.

4.3 Experimental Results

In our experiments we use the standard test signals given in [22]: *Blocks*, *Bumps*, *HeaviSine*, and *Doppler*. Gaussian white noise is added to the signals so that the root-signal-to-noise-ratio (RSNR) is 7. The RSNR is defined as $\sqrt{var(f)/\sigma^2}$, where $var(f)$ is the variance of the signal $f(t)$. The number of sample points for each signal is $n = 2048$. Unless otherwise specified, we use the minimal repeated signal prefilter for the TI multiwavelet denoising experiments. The prefilter and forward multiwavelet transform are performed, and then the coefficients are thresholded using neighbouring coefficients. We experiment with both the TI and non-TI cases. Also the

thresholding is done for both term-by-term and neighbouring coefficients. The inverse multiwavelet transform and post-filter are applied to obtain the smoothed estimate of the noise-free signal. All detail scales except the five coarsest scales are thresholded. The mean square error (MSE) is used as the distance measure between the noise-free signal and the denoised signal. We use NeighCoeff to represent wavelet thresholding using neighbouring coefficients. We give our experimental results in Table 8. In this table, GHM is the well-known GHM multiwavelet transform [36]. Also, D4 is the Daubechies-4 single wavelet. It is shown that the neighbour multiwavelet thresholding method is better than the term-by-term multiwavelet thresholding method for both TI and non-TI cases. This is true for both $r = 1$ and $r = 2$. Note that TI NeighCoeff multiwavelet denoising is better than non-TI NeighCoeff multiwavelet denoising. We can also see that $r = 2$ is better than $r = 1$ for all four signals. Also, when $r = 2$ neighbour multiwavelet denoising outperforms neighbour single wavelet denoising for signals *Blocks* and *Doppler*. However, neighbour single wavelet denoising is better for *Bumps* and *HeaviSine*. We illustrate some figures from our experiments. Figure 15 shows the four noise-free signals and the noisy signals. Figure 16 illustrates the denoised signals with TI D4 term-by-term thresholding. Figure 17 shows the denoised signals with TI D4 neighbour coefficient thresholding. Figure 18 gives the denoised signals by using term-by-term multiwavelet thresholding, whereas Figure 19 illustrates the TI version denoised signals. Figure 20 and Figure 21 show the denoised signals obtained by NeighCoeff GHM ($r = 2$) and TI NeighCoeff GHM ($r = 2$), respectively.

We also tested the well-known real life image *Lena* with 256×256 pixels. Since we are considering signal denoising in this Chapter, we randomly extract a scan-line of the image, say the 128th line, and add noise to it. Again the RSNR is set to 7. All detailed scales except the six coarsest scales are thresholded. For this scan-line, the experimental results are shown in the last column of Table 8. We can see that neighbour multiwavelet denoising outperforms the neighbour single wavelet denoising. Figure 22 shows the image *Lena*, the extracted scan-line, the same scan-line with noise added, the denoised scan-line with TI NeighCoeff D4, the denoised scan-line

with term-by-term TI GHM, and the denoised scan-line with TI NeighCoeff GHM. By studying the denoised signals in Figure 22, we see that multiwavelet can capture more local features than single wavelet in signal denoising. Single wavelet denoising tends to obtain an over smoothed version of the signal. Since a natural image has a lot of details in it, we suggest to use neighbour multiwavelet for denoising natural images. As mentioned before, multiwavelets also always outperform single wavelets in the term-by-term denoising [67], [34] and [3].

	Blocks	*Bumps*	*Heavisine*	*Doppler*	*Lena*
TI D4	34.175	35.173	11.838	25.383	22.058
TI NeighCoeff D4	19.019	**15.121**	**9.588**	11.640	15.451
GHM	28.246	30.582	13.001	21.297	18.451
TI GHM	23.869	20.428	11.387	13.586	16.429
NeighCoeff GHM ($r = 1$)	22.821	21.054	12.769	15.364	16.308
TI NeighCoeff GHM ($r = 1$)	21.052	18.107	11.267	10.560	13.985
NeighCoeff GHM ($r = 2$)	19.993	19.436	12.023	14.949	15.639
TI NeighCoeff GHM ($r = 2$)	**18.077**	16.804	11.064	**10.098**	**13.802**

Table 8: MSE for TI and non-TI wavelet signal denoising.

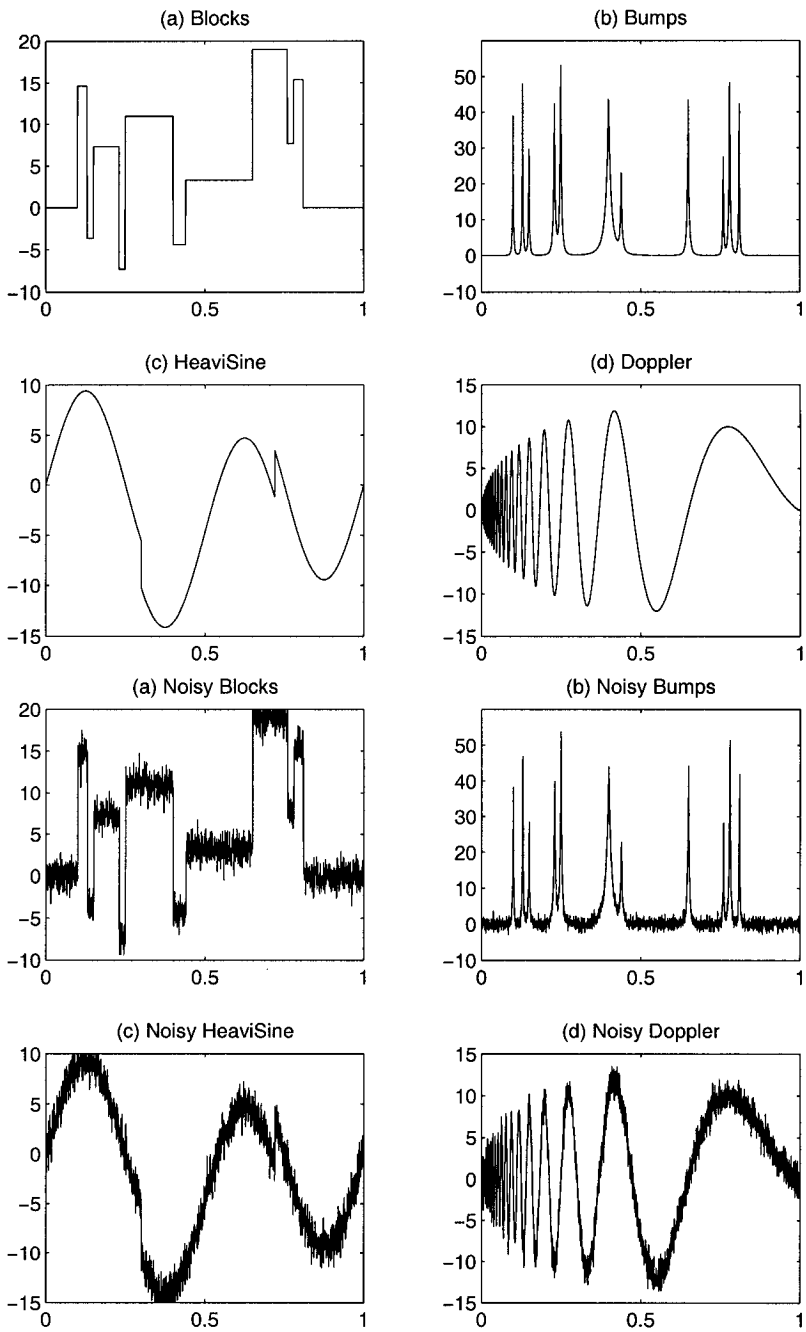

Figure 15: Four noise-free signals and four noisy signals

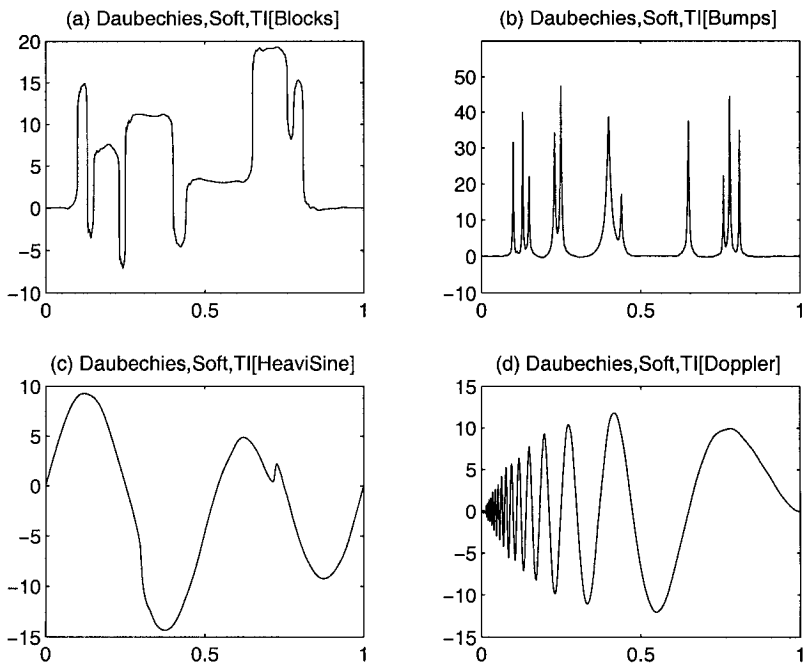

Figure 16: TI D4 denoising by term-by-term thresholding

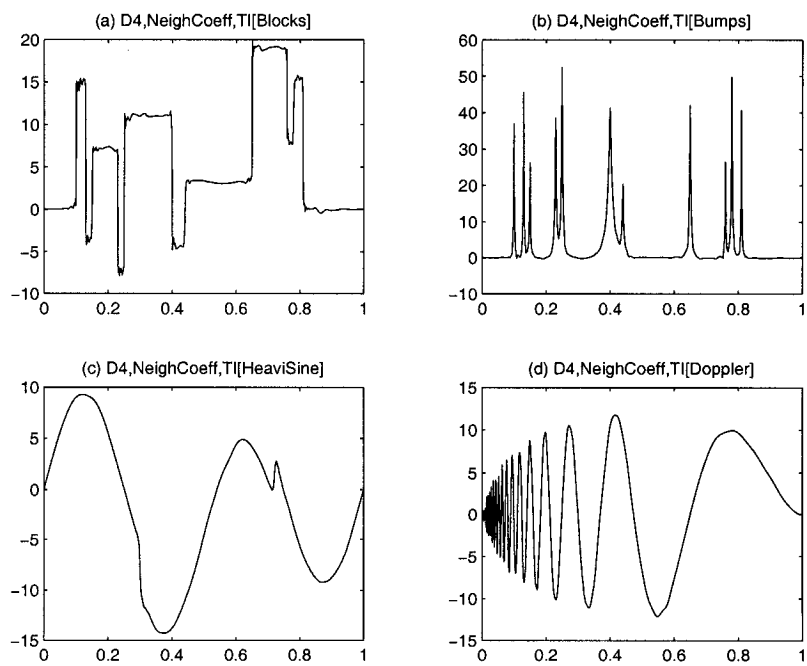

Figure 17: TI D4 denoising using neighbouring coefficients

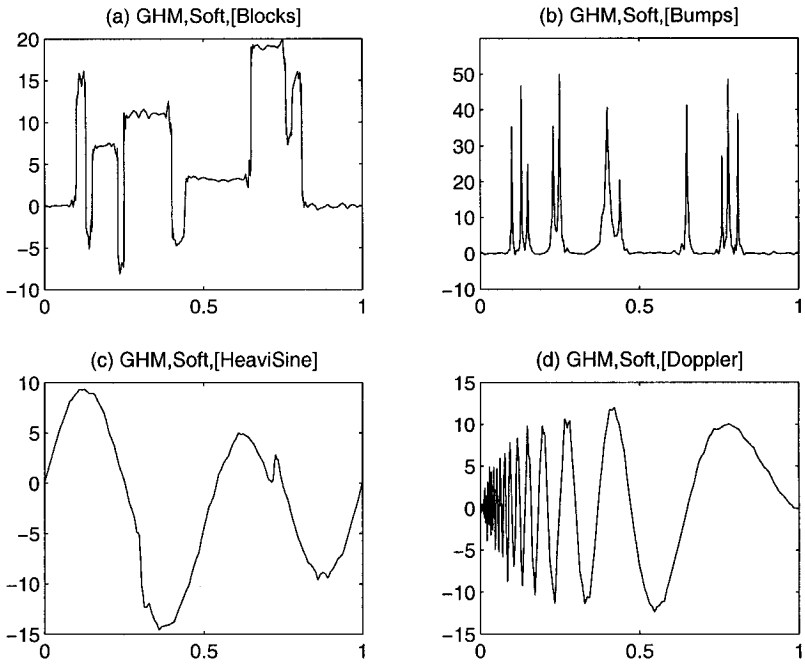

Figure 18: Multiwavelet denoising by term-by-term thresholding

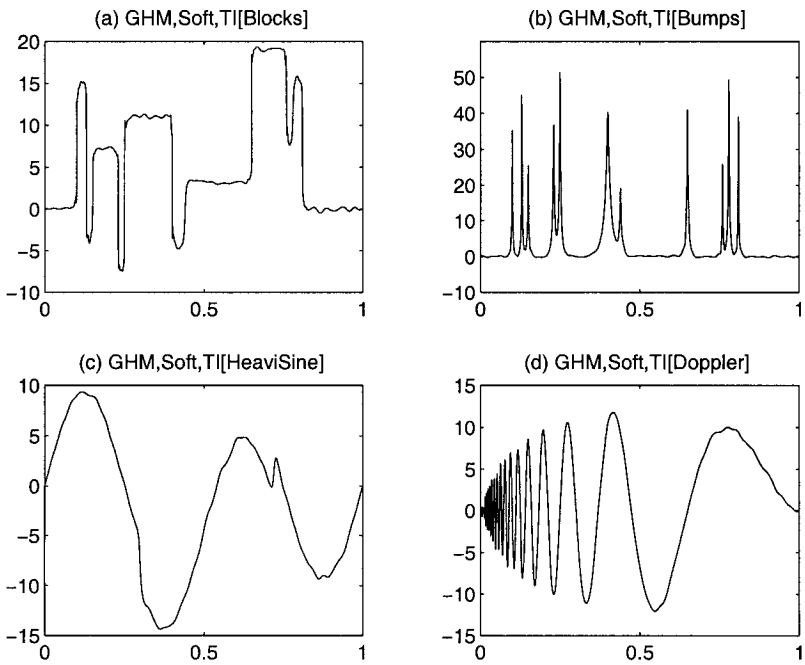

Figure 19: TI Multiwavelet denoising by term-by-term thresholding

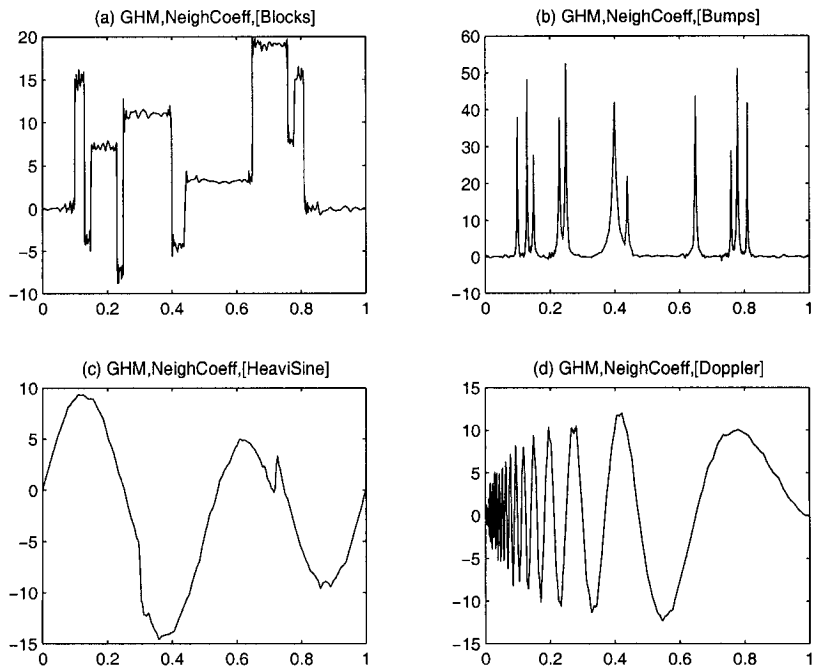

Figure 20: Multiwavelet denoising using neighbouring coefficients

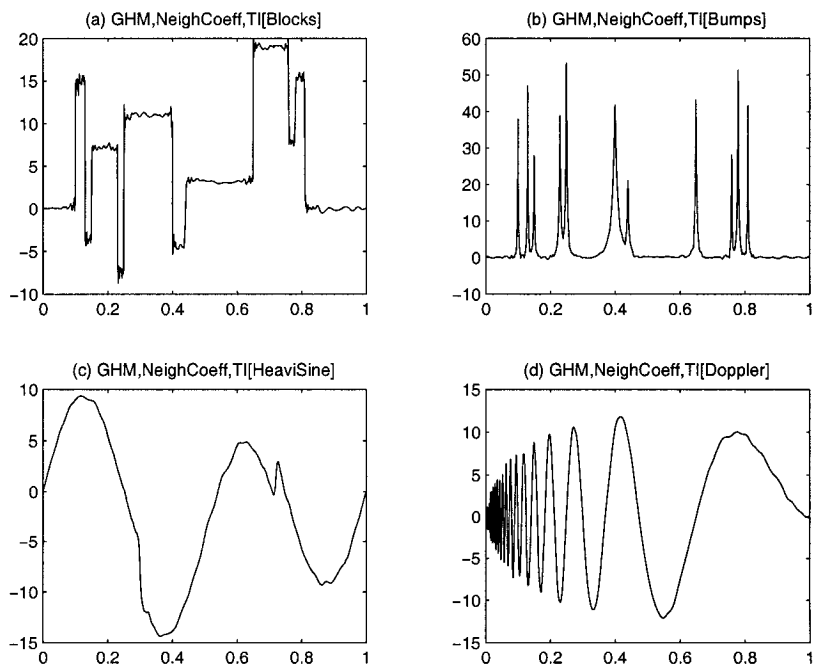

Figure 21: TI Multiwavelet denoising using neighbouring coefficients

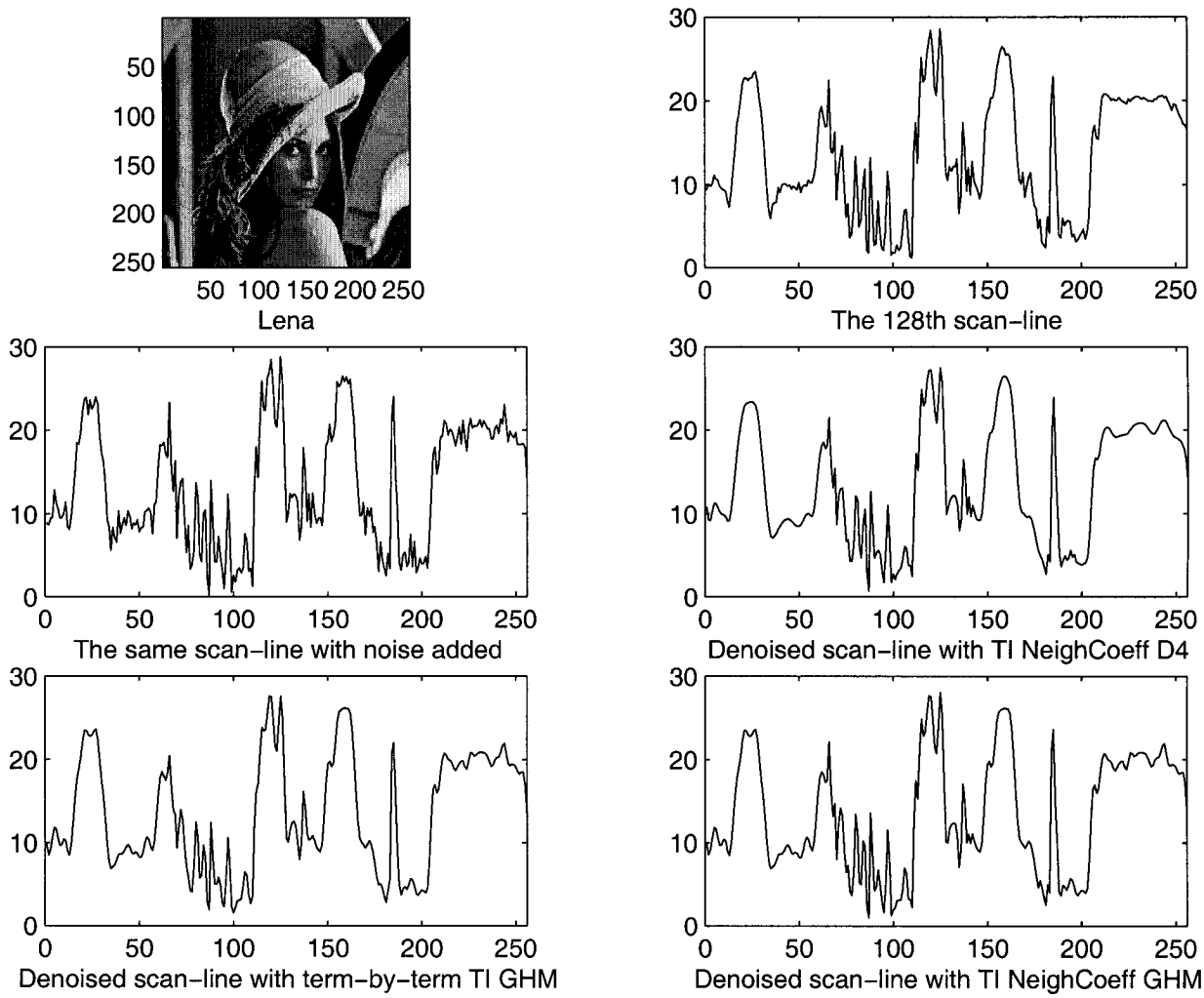

Figure 22: Denoising by using term-by-term and neighbour coefficients.

Chapter 5

Image Denoising Using Neighbouring Wavelet Coefficients

5.1 Introduction

Donoho and his coworkers pioneered a wavelet denoising scheme by using soft thresholding and hard thresholding. This can be summarized as follows. Let $A(i,j)$ be the noise-free image and $B(i,j)$ the image corrupted with white noise $Z(i,j)$, i.e., $B(i,j) = A(i,j) + \sigma Z(i,j)$, where $Z(i,j)$ has normal distribution $N(0,1)$. The Donoho's wavelet denoising scheme can be summarized as follows:

1. Transform the noisy image $B(i,j)$ into an orthogonal domain by 2D discrete wavelet transform.

2. Apply soft or hard thresholding to the resulting wavelet coefficients by using the threshold $\lambda = \sigma\sqrt{2\log N}$ where $N = n \times n$ is the number of pixels in the image.

3. Perform inverse 2D discrete wavelet transform to obtain the denoised image.

The following theorem proves that for an appropriate choice of λ, the risk of a thresholding is close to the risk of an oracle projector $r_p(f) = \sum min(|f_w[m]|^2, \sigma^2)$, where

$f_w[m]$ is the wavelet coefficients.

Theorem (Donoho & Johnstone): Let $\lambda = \sigma\sqrt{2\log N}$. The risk $r_t(f)$ of a hard or soft thresholding estimator for all $N \geq 4$ satisfies

$$r_t(f) \leq (2\log N + 1)(\sigma^2 + r_p(f)).$$

This method performs well under a number of applications because wavelet transform has the compaction property of having only a small number of large coefficients. All the rest of the wavelet coefficients are very small. The denoising is done only on the detail coefficients of the wavelet transform. It has been shown that this algorithm offers the advantages of smoothness and adaptation. However, as Coifman and Donoho [22] pointed out, this algorithm exhibits visual artifacts: Gibbs phenomena in the neighbourhood of discontinuities. Therefore, they propose in [22] a translation invariant (TI) denoising scheme to suppress such artifacts by averaging over the denoised signals of all circular shifts. The experimental results in [22] confirm that single TI wavelet denoising performs better than the traditional single wavelet denoising. Bui and Chen [3] also proposed a TI multiwavelet denoising scheme that gave better results than the TI single wavelet denoising. Recently, several important approaches are proposed by considering the influence of other wavelet coefficients on the current wavelet coefficient to be thresholded. The motivation of this idea is that a large wavelet coefficient will probably have large wavelet coefficients at its neighbours. This is because wavelet transform produces correlated wavelet coefficients. Cai and Silverman [5] proposed a thresholding scheme by taking the immediate neighbour coefficients into account. Their experimental results showed apparent advantages over the traditional term-by-term wavelet denoising. Chen and Bui [12] extended this neighbouring wavelet thresholding idea to the multiwavelet case. We found that neighbour multiwavelet denoising outperforms neighbour single wavelet denoising for some standard testing signals and real-life images. Shengqian et al. [61] proposed an adaptive shrinkage denoising scheme by using neighbourhood characteristics. They claimed

that their new scheme produced better results than Donoho's methods. Sendur and Selesnick [58] [59] proposed bivariate shrinkage functions for denoising. It is indicated that the estimated wavelet coefficients depend on the parent coefficients. The smaller the parent coefficients, the greater the shrinkage. Mihcak et al. [54] performed an approximate maximum *a posteriori* (MAP) estimation of the variance for each coefficient, using the observed noisy data in a local neighbourhood. Crouse et al. [23] developed a framework for statistical signal processing based on wavelet-domain hidden markov models (HMM). The framework enables us to concisely model the non-Gaussian statistics of individual wavelet coefficients and capture statistical dependencies between coefficients. Simoncelli and Adelson [62] gave a Bayesian wavelet coring method that incorporates the higher-order statistical regularity present in the point statistics of subband representation. The statistical model accounts for joint statistics of wavelet coefficients, both within and between bands.

In this Chapter we extend Cai and Silverman's idea to the image case. For images, we need to consider a neighbourhood window around the wavelet coefficients to be thresholded. We propose one way to thresholding the wavelet coefficients for both translation invariant (TI) and non-TI cases. Our algorithm thresholds the wavelet coefficients according to the magnitude of the square sum of all the wavelet coefficients within the neighbourhood window. Experimental results show that by using neighbouring coefficients our algorithms obtain higher Peak Signal to Noise Ratio (PSNR) for all the denoised images. Also, we find that neighbour wavelet image denoising algorithms for both TI and non-TI cases outperform the same algorithms without neighbour dependency.

The organization of this Chapter is as follows. We explain how to incorporate neighbouring wavelet coefficients into image denoising in Section 5.2. Experimental results are shown in Section 5.3.

5.2 Incorporating Neighbouring Wavelet Coefficients in Image Denoising

The wavelet transform can be accomplished by applying the low-pass and high-pass filters on the same set of low frequency coefficients recursively. That means wavelet coefficients are correlated in a small neighbourhood. A large wavelet coefficient will probably have large coefficients at its neighbours. Therefore, Cai et al. [5] proposed the following wavelet denoising scheme for 1D signal by incorporating neighbouring coefficients in the thresholding process. Suppose $d_{j,k}$ is the set of wavelet coefficients of the noisy 1D signal. If $S^2_{j,k} = d^2_{j,k-1} + d^2_{j,k} + d^2_{j,k+1}$ is less than or equal to λ^2, then we set the wavelet coefficient $d_{j,k}$ to zero. Otherwise, we shrink it according to

$$d_{j,k} = d_{j,k}(1 - \lambda^2/S^2_{j,k})$$

where $\lambda = \sqrt{2\sigma^2 \log n}$ and n is the length of the signal. Note that we should omit the first (last) term in $S^2_{j,k}$ if $d_{j,k}$ is at the left (right) boundary of level j wavelet coefficients.

For image denoising, we have to do a 2D wavelet transform. At every decomposition level, we get four frequency subbands, namely, LL, LH, HL, and HH. The next level should be applied to the low frequency subband LL only. This process is continued until a prespecified level is reached. Since the Gaussian noise will be nearly averaged out in the low frequency wavelet coefficients and we want to keep small coefficients in these frequencies, only wavelet coefficients in the high frequency levels need to be thresholded. That means we need to threshold all LH, HL, and HH within these high frequency subbands. For every wavelet coefficient $d_{j,k}$ of our interest, we need to consider a neighbourhood window $B_{j,k}$ around it. We choose the window by having the same number of pixels above, below, on the left or right of the pixel to be thresholded. That means the neighbourhood window size should be 3×3, 5×5, 7×7, 9×9, etc. Figure 23 illustrates a 3×3 neighbourhood window centered at

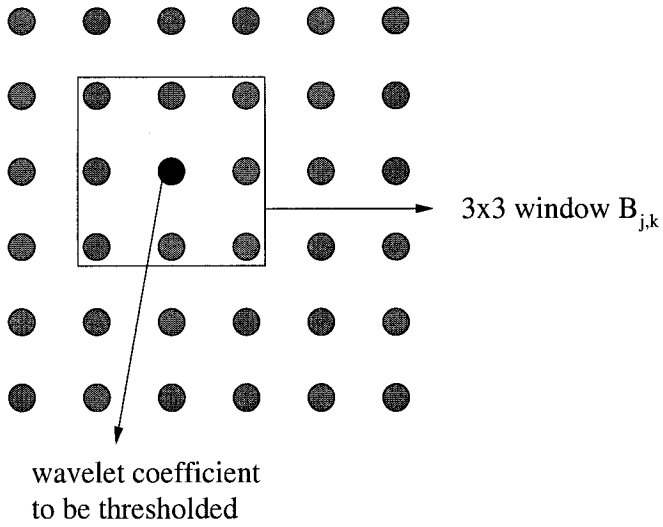

Figure 23: An illustration of the neighbourhood window centered at the wavelet coefficient to be thresholded.

the wavelet coefficient to be thresholded. We threshold different wavelet coefficient subbands independently.

Let

$$S_{j,k}^2 = \sum_{(i,l)\in B_{j,k}} d_{i,l}^2$$

when the above summation has pixel indexes out of the wavelet subband range, we omit the corresponding terms in the summation. For the wavelet coefficient to be thresholded, we shrink it according to the following formula:

$$d_{j,k} = d_{j,k}\beta_{j,k}$$

where the shrinkage factor can be defined as:

$$\beta_{j,k} = (1 - \lambda^2/S_{j,k}^2)_+$$

here, the + sign at end of the formula means to keep the positive value while set it to zero when it is negative, and $\lambda = \sqrt{2\sigma^2 \log n^2}$. Note that this thresholding formula

is a modification to the classical soft thresholding scheme developed by Donoho and his coworkers. The neighbourhood window size around the wavelet coefficient to be thresholded has influence on the denoising ability of our proposed algorithm. The larger the window, the relatively smaller the threshold is. If the size of the window around the pixel is too large, a lot of noise will be kept, so an intermediate window size of 3×3 or 5×5 should be used.

The neighbour wavelet image denoising algorithm can be described as follows:

1. Perform forward 2D wavelet decomposition (TI or non-TI) on the noisy image.
2. Apply the proposed shrinkage scheme to threshold the wavelet coefficients using a neighbourhood window $B_{j,k}$ and the universal threshold $\sqrt{2\sigma^2 \log n^2}$.
3. Perform inverse 2D wavelet transform (TI or non-TI) on the thresholded wavelet coefficients.

We call this algorithm *NeighShrink*. As we know *VisuShrink* kills too many small wavelet coefficients, our shrinkage schemes should perform better. From our experiments we find that *NeighShrink* outperforms *VisuShrink* and the TI image denoising developed by Yu et al. [81] for the TI case.

We also consider the shrinkage scheme that calculates the average value in the neighbourhood window and then threshold the current wavelet coefficient according to this average value. This thesholding formula can be given as:

$$\beta_{j,k} = (1 - N^2\lambda^2/S_{j,k}^2)_+$$

where N is the neighbourhood window size in one dimension. However, by using the average we get worse denoising results than *VisuShrink*. We give this thresholding formula because taking the average is a natural choice.

This algorithm has higher computational demands. We give the computational complexity of the algorithm for the non-TI case. The forward 2D wavelet transform

needs $2Ln^2$ flops of computation, where L is the wavelet filter length and n is the image size in one dimension. The thresholding process using neighbour information requires N^2n^2 flops of calculation, where N is the neighbourhood window size in one dimension. The inverse 2D wavelet transform also needs $2Ln^2$ flops of computation, just like the forward 2D wavelet transform. In total, the algorithm *NeighShrink* for the non-TI case takes $(4L + N^2)n^2$ flops of computation. On the other hand, *VisuShrink* only needs $4Ln^2$ flops of computation. We get better quality denoised images by sacrificing some amount of computation time.

5.3 Experimental Results

We perform our experiments on the well-known images *Lena*, *MRIScan*, *Fingerprint*, *Phone*, *Daubechies*, and *Canaletto*. We get these images from the free software package *WaveLab* developed by Donoho et al. at Stanford University. For comparison, we implement *VisuShrink*, *NeighShrink*, *TI* and *TI NeighShrink*. *VisuShrink* is the universal soft-thresholding denoising technique [31]. *TI* is the TI wavelet denoising algorithm developed by Yu et al. [81]. The Daubechies wavelet with 8 vanishing moments is used for the wavelet decomposition. Five detailed wavelet decomposition scales are thresholded using the universal threshold $\sqrt{2\sigma^2 log n^2}$. Note that this threshold is the same as Donoho's threshold for 1D signal except we replace n in 1D signal with n^2 in 2D image. For different Gaussian white noise levels, the experimental results in Peak Signal to Noise Ratio ($PSNR$) are shown in Table 9 - Table 14 for denoising images *Lena*, *MRIScan*, *Fingerprint*, *Phone*, *Daubechies*, and *Canaletto*, respectively. The $PSNR$ is defined as

$$PSNR = -10 \log_{10} \frac{\sum_{i,j}(B(i,j) - A(i,j))^2}{n^2 256^2}.$$

where B is the denoised image and A is the noise-free image. The first column in these tables is the $PSNR$ of the original noisy images, while other columns are the

PSNR of the denoised images by using different denoising methods. From Table 9 - Table 14 we can see that *NeighShrink* outperforms *VisuShrink* and the TI image denoising method developed by Yu et al. for all cases. *VisuShrink* does not have any denoising power when the noise level is low. Under such a condition, *VisuShrink* produces even worse results than the original noisy images. However, *NeighShrink* performs very well in this case. When the noise level is low, the improvement of $NeighShrink$ (TI and non-TI) over $VisuShrink$ is large. When the noise level is high, the improvement is low even though $NeighShrink$ for non-TI and TI is still better than $VisuShrink$ and the TI image denoising method developed by Yu et al. [81], respectively. Figure 24 - Figure 29 show the noise free image, the image with noise added, the denoised image with *VisuShrink*, the denoised image with *NeighShrink*, the denoised image with *TI* image denoising, and the denoised image with *TI NeighShrink* for images *Lena*, $MRIScan$, and $Fingerprint$,*Phone*, *Daubechies*, and *Canaletto*, respectively. By studying the denoised images in Figure 24 - Figure 29, we see that *NeighShrink* produces smoother and clearer denoised images. We also threshold the wavelet coefficients by looking at the average value in the neighbourhood window, and we find that it does not perform as well as $VisuShrink$ for all denoising experiments. We conduct this experiment in this Chapter because taking the average of the wavelet coefficients in the neighbourhood window is a natural choice. Unfortunately, it does not provide better performance.

In order to investigate the influence of neighbourhood window size to the denoising ability, we list the experimental results for different window sizes in Table 15. These experiments are done using non-TI *NeighShrink*. We can see that the window sizes of 3×3 and 5×5 are the best. When the window size is getting larger, the denoising ability is getting worse. However, when the window size is extremely small, just like the term-by-term thresholding, the denoising ability is not very high. We have found that the intermediate neighbourhood window sizes of 3×3 and 5×5 are good choices for our proposed algorithm $NeighShrink$.

Noisy Image	*VisuShrink*	*NeighShrink*	*TI*	*TI NeighShrink*
28.14	25.87	32.34	27.30	**33.61**
22.12	23.13	28.26	24.69	**29.38**
18.59	21.85	26.06	23.59	**27.12**
16.09	21.07	24.64	22.97	**25.67**
14.16	20.48	23.68	22.55	**24.65**
12.57	20.03	22.94	22.23	**23.87**
11.23	19.68	22.36	21.97	**23.25**

Table 9: The PSNR (dB) of the noisy images of Lena and the denoised images with different denoising methods.

Noisy Image	*VisuShrink*	*NeighShrink*	*TI*	*TI NeighShrink*
28.14	26.30	33.05	28.09	**34.49**
22.12	23.28	28.97	25.22	**30.25**
18.59	21.75	26.77	23.79	**28.01**
16.09	20.79	25.33	22.86	**26.57**
14.16	20.13	24.26	22.18	**25.49**
12.57	19.62	23.40	21.67	**24.62**
11.23	19.20	22.68	21.27	**23.87**

Table 10: The PSNR (dB) of the noisy images of MRIScan and the denoised images with different denoising methods.

Noisy Image	*VisuShrink*	*NeighShrink*	*TI*	*TI NeighShrink*
28.15	25.63	30.94	26.27	**31.89**
22.13	23.50	27.58	24.00	**28.38**
18.61	22.67	25.75	23.14	**26.46**
16.11	22.24	24.65	22.73	**25.27**
14.17	21.99	23.92	22.49	**24.46**
12.59	21.83	23.42	22.35	**23.88**
11.25	21.71	23.04	22.23	**23.45**

Table 11: The PSNR (dB) of the noisy images of Fingerprint and the denoised images with different denoising methods.

Noisy Image	*VisuShrink*	*NeighShrink*	*TI*	*TI NeighShrink*
28.13	25.07	30.08	27.42	**30.95**
22.11	22.49	26.88	25.41	**27.94**
18.59	21.29	24.95	24.39	**26.13**
16.09	20.61	23.52	23.68	**24.76**
14.15	20.20	22.45	23.05	**23.66**
12.57	19.91	21.64	22.41	**22.71**
11.23	19.65	21.06	21.76	**21.88**

Table 12: The PSNR (dB) of the noisy images of Phone and the denoised images with different denoising methods.

Noisy Image	*VisuShrink*	*NeighShrink*	*TI*	*TI NeighShrink*
28.13	31.21	35.92	33.28	**36.87**
22.11	29.16	32.75	31.53	**33.74**
18.59	28.26	30.84	30.63	**31.91**
16.09	27.73	29.68	29.89	**30.68**
14.15	27.36	28.85	29.19	**29.69**
12.57	27.15	28.18	28.51	**28.82**
11.23	26.94	27.63	27.84	**28.02**

Table 13: The PSNR (dB) of the noisy images of Daubechies and the denoised images with different denoising methods.

Noisy Image	*VisuShrink*	*NeighShrink*	*TI*	*TI NeighShrink*
28.15	25.44	31.43	26.37	**32.34**
22.13	22.61	27.97	23.49	**28.83**
18.60	21.15	26.00	22.06	**26.78**
16.11	20.25	24.59	21.21	**25.33**
14.17	19.66	23.50	20.67	**24.25**
12.58	19.24	22.61	20.30	**23.39**
11.25	18.92	21.92	20.02	**22.72**

Table 14: The PSNR (dB) of the noisy images of Canaletto and the denoised images with different denoising methods.

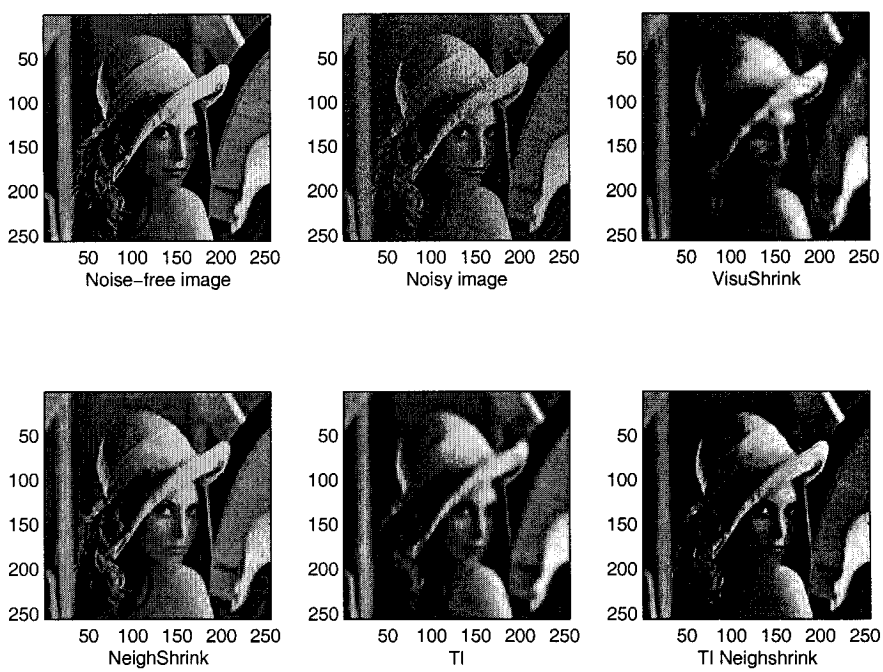

Figure 24: Image denoising by using different methods on a noisy image with PSNR = 22dB.

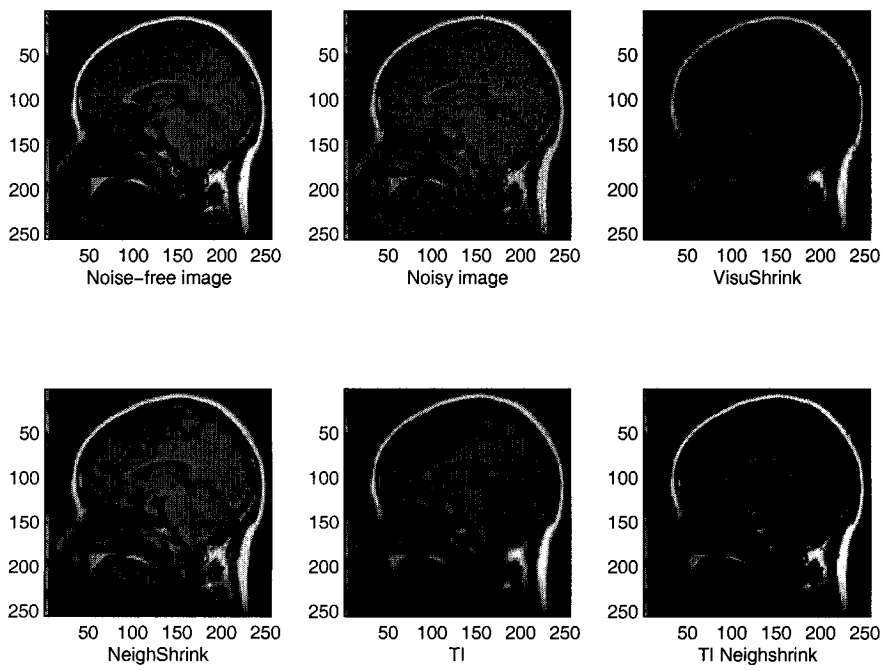

Figure 25: Image denoising by using different methods on a noisy image with PSNR = 22dB.

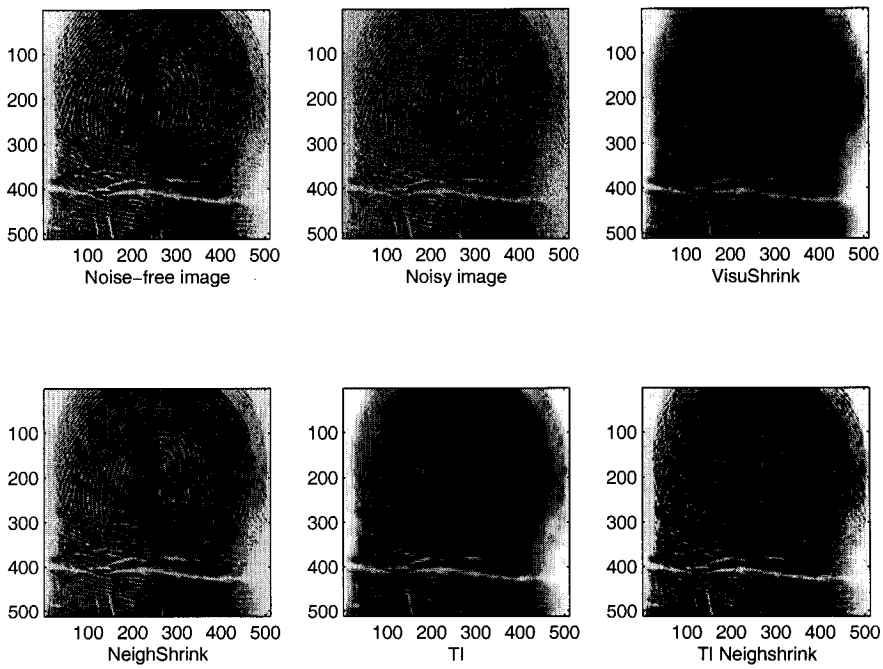

Figure 26: Image denoising by using different methods on a noisy image with PSNR = 22dB.

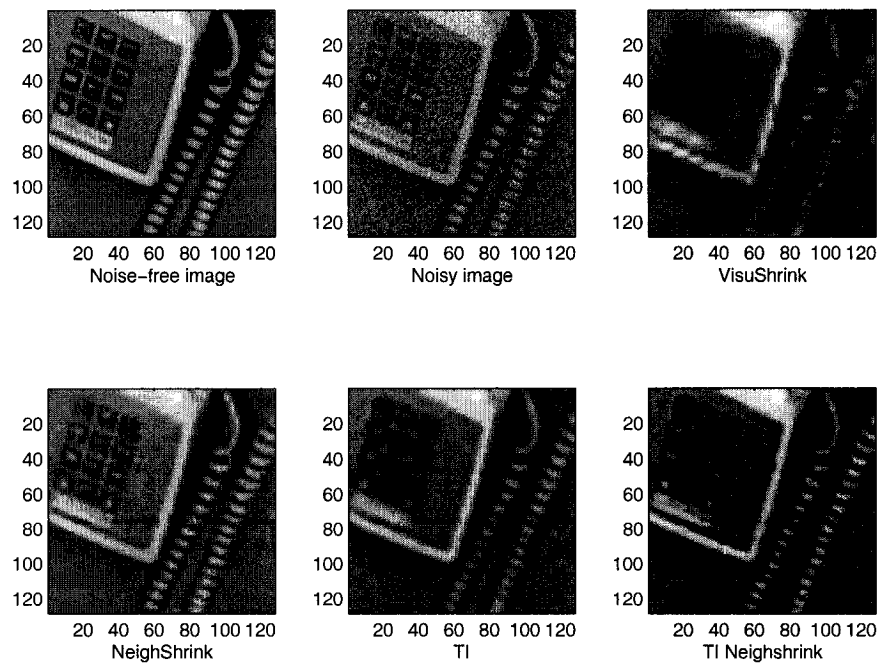

Figure 27: Image denoising by using different methods on a noisy image with PSNR = 22dB.

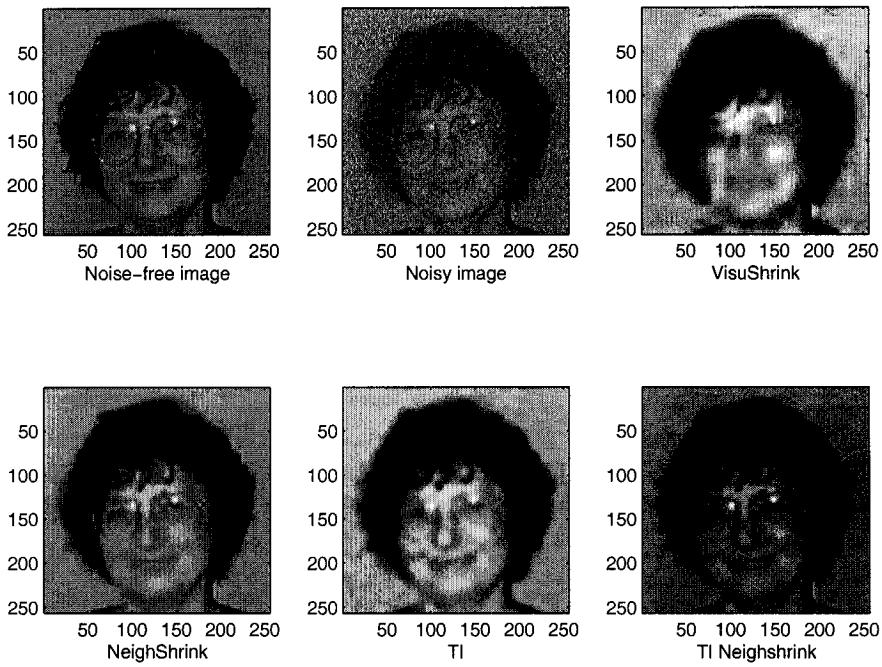

Figure 28: Image denoising by using different methods on a noisy image with PSNR = 22dB.

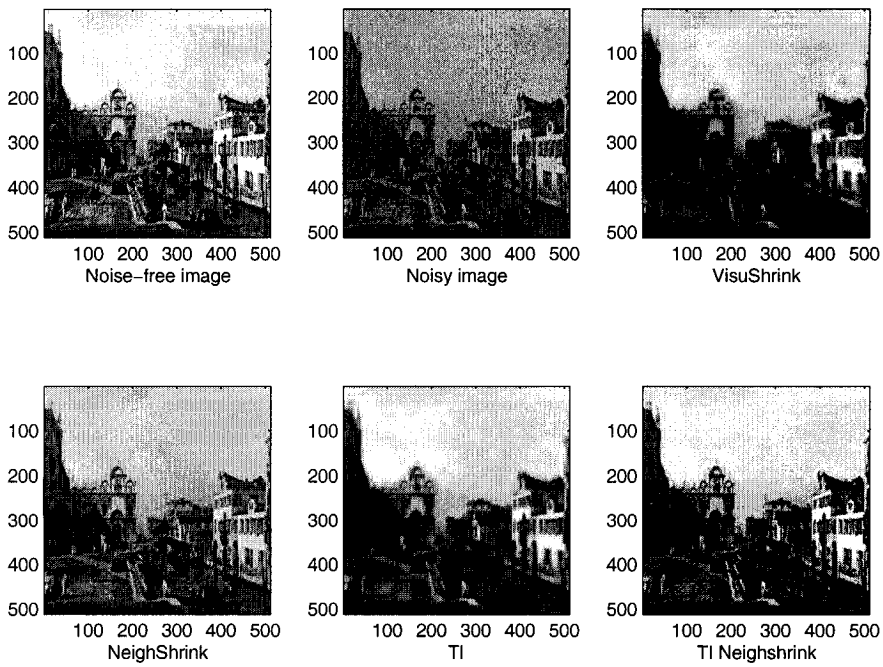

Figure 29: Image denoising by using different methods on a noisy image with PSNR = 22dB.

	Noisy	*Window Size*			
	Image	VisuShrink (1x1)	3x3	5x5	7x7
Lena	22.11	23.13	28.26	**28.39**	25.69
MRIScan	22.11	23.28	**28.97**	28.93	25.91
Fingerprint	22.13	23.50	27.58	**28.72**	26.21
Phone	22.12	22.49	**26.88**	26.85	24.89
Daubechies	22.12	29.16	**32.75**	31.33	26.71
Canaletto	22.13	22.61	27.98	**28.77**	26.17

Table 15: The PSNR (dB) of the denoised images with different neighbourhood window sizes for the non-TI case.

Chapter 6

Image Denoising with Customized Wavelet and Threshold

6.1 Introduction

Wavelet transforms have been proven to be very successful in many applications. For a given application, which wavelet should we choose to use? It is desirable to choose a wavelet that is best suitable for the problem at hand. We call this a *customized wavelet.* Customized wavelets have been used in several areas such as image compression, signal representation, signal denoising, feature extraction, face recognition, image pattern recognition, etc. Here we briefly review some of the customized wavelet papers. Mallet et al. [53] proposed a new and innovative technique based on adaptive wavelets, which aims to reduce the dimensionality and optimize the discriminatory information. Instead of using standard wavelet bases, they generate the wavelet which optimizes specified discriminant criteria. Wang et al. [76] used Genetic Algorithm (GA) to find an optimal basis derived from a combination of frequencies and orientation angles in the 2-D Gabor wavelet transform. This approach provides a more accurate and efficient projection scheme and therefore a better face classification result. Chapa et al. [9] developed a technique for deriving a bandlimited wavelet directly from the desired signal spectrum in such a way that the mean square error

between their spectra is a minimum. The technique includes an algorithm for finding the scaling function from an orthonormal wavelet, and algorithms for finding the customized wavelet magnitude and phase from a given input signal. A revised version of their paper appeared in *IEEE Transactions on Signal Processing* [10]. Zhuang et al. [84] studied the problem of choosing an image-based customized wavelet basis with compact support for image data compression and provided a general algorithm for computing the optimal wavelet basis. Tewfik et al. [70] studied the problem of choosing a discrete orthogonal wavelet with a support size that is equal to or smaller than a prespecified limit to represent a given finite support signal up to a fixed resolution scale. The techniques are based on optimizing certain cost functions. The optimization with a set of constraints can be converted to an optimization problem without constraints by means of parametrization. Das et al. [27] proposed an algorithm for construction of optimal compactly supported N-tap orthonormal wavelet for signal denoising. Simulated Annealing is used for the optimization of the parametrization of the wavelet FIR filter bank coefficients. Golden [37] considered the problem of optimizing at each resolution level the parameters of a two-band quadrature mirror filter analysis bank to achieve maximal decorrelation of the two decimated output sequences.

In this Chapter we use Simulated Annealing to find the customized wavelet filters and the customized threshold for denoising the given noisy image. We consider a neighbourhood window around the wavelet coefficients to be thresholded in the same way as in the previous Chapter. We call this method as *Customized NeighShrink*. *Customized NeighShrink* thresholds the wavelet coefficients according to the magnitude of the square sum of all the wavelet coefficients within the neighbourhood window. Experimental results show that *Customized NeighShrink* gets higher Peak Signal to Noise Ratio (PSNR) for all the denoised images. It outperforms *VisuShrink*, *NeighShrink* and *wiener2* filter for different noise levels and testing images.

The organization of this Chapter is as follows. Section 6.2 reviews how to parametrize the compactly supported wavelets. Section 6.3 explains how to denoise a noisy image

by means of customized wavelet and threshold. Section 6.4 conducts some experiments.

6.2 Parametrization of Compactly Supported Wavelets

The construction of a wavelet depends on a scaling function $\phi(x)$ that obeys a 2-scale dilation equation:

$$\phi(x) = \sqrt{2}\sum c_k \phi(2x - k)$$

where c_k is a set of real filter. The sequence $\{c_k\}$ must satisfy the following conditions [29], [65]:

$$\sum c_k = \sqrt{2}$$

$$\sum c_k c_{k+2m} = \delta(m)$$

$$\sum (-1)^k k^m c_k = 0, m = 0, 1, \cdots, M-1$$

where $M \geq 1$ and $\delta(m)$ denotes a discrete Kronecker delta function. M controls the compact support of the wavelet, and it is equal to the number of vanishing moments of the wavelet $\psi(x)$ corresponding to $\phi(x)$.

The wavelet $\psi(x)$ is defined as

$$\psi(x) = \sqrt{2}\sum d_k \phi(2x - k)$$

where $d_k = (-1)^k c_{M-1-k}$.

It can be shown that discrete orthonormal wavelets of support less than $2N - 1$,

where N is an even number, can be parametrized by $N-1$ real parameters each taking values in $[0, 2\pi)$ [85]. Let

$$H(z) = \sum c_k z^k$$

$$G(z) = \sum d_k z^k$$

where $z = e^{-jw}$. Then we can have

$$\begin{pmatrix} H(z) \\ z^{2(N-1)}G(z) \end{pmatrix} = E(z^2) \begin{pmatrix} 1 \\ z \end{pmatrix}$$

where $E(z)$ is a polyphase matrix defined by

$$E(z) = V_{N-1}(z)V_{N-2}(z)\cdots V_1(z)V_0$$

and

$$V_0 = \begin{pmatrix} cos\theta_0 & -sin\theta_0 \\ sin\theta_0 & cos\theta_0 \end{pmatrix}$$

$$V_k(z) = I + (z-1)v_k v_k^T, 1 \leq k \leq N-1.$$

$$v_k = \begin{pmatrix} cos\theta_k \\ sin\theta_k \end{pmatrix}$$

By looking at the definition of $E(z)$ we know that a sequence c_k of length $2N$ is actually parametrized by N free parameters. Furthermore, the wavelet $\psi(x)$ has at least one vanishing moment if and only if $\theta_0 = 3\pi/4$. We give some of the parametrized sequences for $N = 2$, $N = 3$ and $N = 4$, respectively. The sequences $\{c_k\}_{k=0}^{3}$ can be listed as follows:

$$
\begin{aligned}
c_0 &= sin(\theta_1) * sin(\theta_1 - \pi/4) \\
c_1 &= sin(\theta_1) * sin(\theta_1 + \pi/4) \\
c_2 &= cos(\theta_1) * sin(\theta_1 + \pi/4) \\
c_3 &= cos(\theta_1) * sin(\pi/4 - \theta_1)
\end{aligned}
$$

where $\theta_1 \in [0, 2\pi)$.

The sequences $\{c_k\}_{k=0}^{5}$ can be listed as follows:

$$
\begin{aligned}
c_0 &= (sin(\theta_2)^2 * (sin(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + \\
&\quad (sin(\theta_1) * cos(\theta_1) - cos(\theta_1)^2) * sin(\theta_2) * cos(\theta_2))/\sqrt{2} \\
c_1 &= (sin(\theta_2)^2 * (sin(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + 2 * sin(\theta_2)^2 * sin(\theta_1) * cos(\theta_1) + \\
&\quad (cos(\theta_1)^2 + sin(\theta_1) * cos(\theta_1)) * sin(\theta_2) * cos(\theta_2))/\sqrt{2} \\
c_2 &= (cos(\theta_2)^2 * (-sin(\theta_1) * cos(\theta_1) + sin(\theta_1)^2) + 2 * sin(\theta_2)^2 * sin(\theta_1) * cos(\theta_1) + \\
&\quad sin(\theta_2)^2 * (cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + \\
&\quad (cos(\theta_1)^2 - 2 * sin(\theta_1) * cos(\theta_1) - sin(\theta_1)^2) * sin(\theta_2) * cos(\theta_2))/\sqrt{2} \\
c_3 &= (cos(\theta_2)^2 * (sin(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + 2 * cos(\theta_2)^2 * sin(\theta_1) * cos(\theta_1) + \\
&\quad sin(\theta_2)^2 * (cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + \\
&\quad (-cos(\theta_1)^2 - 2 * sin(\theta_1) * cos(\theta_1) + sin(\theta_1)^2) * sin(\theta_2) * cos(\theta_2))/\sqrt{2} \\
c_4 &= (2 * cos(\theta_2)^2 * sin(\theta_1) * cos(\theta_1) + cos(\theta_2)^2 * (cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + \\
&\quad (sin(\theta_1) * cos(\theta_1) + sin(\theta_1)^2) * sin(\theta_2) * cos(\theta_2))/\sqrt{2} \\
c_5 &= (cos(\theta_2)^2 * (cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + \\
&\quad (sin(\theta_1) * cos(\theta_1) - sin(\theta_1)^2) * sin(\theta_2) * cos(\theta_2))/\sqrt{2}
\end{aligned}
$$

where $\theta_1 \in [0, 2\pi)$ and $\theta_2 \in [0, 2\pi)$.

The sequences $\{c_k\}_{k=0}^{7}$ can be listed as follows:

$$
\begin{aligned}
a_0 &= sin(\theta_2)^2 * (sin(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + \\
&\quad (sin(\theta_1) * cos(\theta_1) - cos(\theta_1)^2) * sin(\theta_2) * cos(\theta_2) \\
a_1 &= sin(\theta_2)^2 * (sin(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + 2 * sin(\theta_2)^2 * sin(\theta_1) * cos(\theta_1) + \\
&\quad (cos(\theta_1)^2 + sin(\theta_1) * cos(\theta_1)) * sin(\theta_2) * cos(\theta_2)
\end{aligned}
$$

$$
\begin{aligned}
a_2 &= cos(\theta_2)^2 * (-sin(\theta_1) * cos(\theta_1) + sin(\theta_1)^2) + 2 * sin(\theta_2)^2 * sin(\theta_1) * cos(\theta_1) + \\
&\quad sin(\theta_2)^2 * (cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + \\
&\quad (cos(\theta_1)^2 - 2 * sin(\theta_1) * cos(\theta_1) - sin(\theta_1)^2) * sin(\theta_2) * cos(\theta_2) \\
a_3 &= cos(\theta_2)^2 * (sin(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + 2 * cos(\theta_2)^2 * sin(\theta_1) * cos(\theta_1) + \\
&\quad sin(\theta_2)^2 * (cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + \\
&\quad (-cos(\theta_1)^2 - 2 * sin(\theta_1) * cos(\theta_1) + sin(\theta_1)^2) * sin(\theta_2) * cos(\theta_2) \\
a_4 &= 2 * cos(\theta_2)^2 * sin(\theta_1) * cos(\theta_1) + cos(\theta_2)^2 * (cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + \\
&\quad (sin(\theta_1) * cos(\theta_1) + sin(\theta_1)^2) * sin(\theta_2) * cos(\theta_2) \\
a_5 &= cos(\theta_2)^2 * (cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + \\
&\quad (sin(\theta_1) * cos(\theta_1) - sin(\theta_1)^2) * sin(\theta_2) * cos(\theta_2) \\
b0 &= -sin(\theta_1) * (sin(\theta_1) - cos(\theta_1)) * sin(\theta_2) * cos(\theta_2) + \\
&\quad cos(\theta_2)^2 * (cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) \\
b1 &= (-sin(\theta_1)^2 - 3 * sin(\theta_1) * cos(\theta_1)) * sin(\theta_2) * cos(\theta_2) + \\
&\quad cos(\theta_2)^2 * (-cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) \\
b2 &= (sin(\theta_1)^2 - 2 * sin(\theta_1) * cos(\theta_1) - cos(\theta_1)^2) * sin(\theta_2) * cos(\theta_2) + \\
&\quad sin(\theta_2)^2 * (cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) + \\
&\quad sin(\theta_1) * cos(\theta_1) * cos(\theta_2)^2 + cos(\theta_2)^2 * sin(\theta_1)^2 \\
b3 &= (sin(\theta_1)^2 + 2 * sin(\theta_1) * cos(\theta_1) - cos(\theta_1)^2) * sin(\theta_2) * cos(\theta_2) + \\
&\quad sin(\theta_2)^2 * (-sin(\theta_1) * cos(\theta_1) - cos(\theta_1)^2) + cos(\theta_2)^2 * (sin(\theta_1) * cos(\theta_1) - sin(\theta_1)^2) \\
b4 &= (cos(\theta_1)^2 + sin(\theta_1) * cos(\theta_1)) * sin(\theta_2) * cos(\theta_2) + \\
&\quad sin(\theta_2)^2 * (sin(\theta_1) * cos(\theta 1) + sin(\theta_1)^2) \\
b5 &= (cos(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) * sin(\theta_2) * cos(\theta_2) - \\
&\quad sin(\theta_2)^2 * (sin(\theta_1)^2 - sin(\theta_1) * cos(\theta_1)) \\
c_0 &= (a_0 * sin(\theta_3)^2 - b_0 * sin(\theta_3) * cos(\theta_3))/\sqrt{2} \\
c_1 &= (a_1 * sin(\theta_3)^2 - b_1 * sin(\theta_3) * cos(\theta_3))/\sqrt{2} \\
c_2 &= (a_0 * cos(\theta_3)^2 + a_2 * sin(\theta_3)^2 + (b_0 - b_2) * sin(\theta_3) * cos(\theta_3))/\sqrt{2}
\end{aligned}
$$

$$
\begin{aligned}
c_3 &= (a_1 * cos(\theta_3)^2 + a_3 * sin(\theta_3)^2 + (b_1 - b_3) * sin(\theta_3) * cos(\theta_3))/\sqrt{2} \\
c_4 &= (a_2 * cos(\theta_3)^2 + a_4 * sin(\theta_3)^2 + (b_2 - b_4) * sin(\theta_3) * cos(\theta_3))/\sqrt{2} \\
c_5 &= (a_3 * cos(\theta_3)^2 + a_5 * sin(\theta_3)^2 + (b_3 - b_5) * sin(\theta_3) * cos(\theta_3))/\sqrt{2} \\
c_6 &= (a_4 * cos(\theta_3)^2 + b_4 * sin(\theta_3) * cos(\theta_3))/\sqrt{2} \\
c_7 &= (a_5 * cos(\theta_3)^2 + b_5 * sin(\theta_3) * cos(\theta_3))/\sqrt{2}
\end{aligned}
$$

where $\theta_1 \in [0, 2\pi)$, $\theta_2 \in [0, 2\pi)$ and $\theta_3 \in [0, 2\pi)$.

After the parametrization, the optimization problem to find the customized wavelet for a specific cost function becomes an optimization problem without constraints. This is much easier to solve in practice. In this Chapter, we use the filter with length equal to 8. We choose 8 because the Daubechies wavelet with 8 vanishing moments is a very popular and effective choice in image denoising.

6.3 Image Denoising using Customized Wavelet and Threshold

In this section, we still consider the image denoisng technique with neighbour dependency as described in the previous Chapter. We will study the non-TI case and try to customize the wavelet filter and universal threshold used in the denoising process. As we know, the choices of the wavelet and the threshold should make significant difference in the quality of the denoised image. We propose to use Simulated Annealing to adaptively learn the customized wavelet filters and customized universal threshold at the same time. Simulated Annealing [43] is a Monte Carlo approach for minimizing an optimization problem. The term Simulated Annealing is from the physical process of heating and then slowly cooling down a substance to obtain a crystalline structure. The minimum of the cost function corresponds to this ground state of the substance. In an annealing process, the substance melts at a high temperature and it is disordered. As it cools down, it becomes more ordered and approaches a frozen ground state. The Metropolis step is the fundamental procedure of Simulated Annealing. If the change in energy is negative compared to the previous one, then the change is

accepted and the system is updated. If the energy is greater than the previous one, then it is accepted with a probability given by the Boltzman factor. This process is continued for many times until a frozen state is achieved. In Simulated Annealing, we need a cost function to minimize. This cost function is defined as the mean square error of the denoised image A with another noisy copy of the image B:

$$cost(filter, threshold) = \sum_{i,j}(A_{i,j} - B_{i,j})^2/n^2$$

where $filter$ represents the wavelet filter, $threshold$ is the threshold in the wavelet thresholding process, and n^2 is the number of pixels in the image. We are trying to customize $filter$ and $threshold$ by means of Simulated Annealing. We assume that we have two noisy copies of the same image. One is what we want to denoise and the other one is used as the target. This assumption is reasonable because sometimes we can have two sensors at the same time. Every time when we calculate the cost corresponding to the current wavelet filter and threshold, we should follow these steps:

1. Calculate the customized wavelet filter from the filter parameters.
2. Perform forward 2D wavelet transform on the noisy image using the customized wavelet filter.
3. Threshold the wavelet coefficients by considering the influence of neighbour wavelet coefficients.
4. Perform inverse 2D discrete wavelet transform to obtain the denoised image.
5. Calculate the cost based on the denoised image.

We call our neighbour wavelet image denoising algorithm with customized wavelet and threshold as *Customized NeighShrink*. From our experiments we find that *Customized NeighShrink* performs the best. It outperforms *VisuShrink*, *NeighShrink*, and *wiener2* filter for all noisy levels and testing images. For some case, choosing the customized wavelet filter and customized threshold gives us about 1.20dB improvement

over the same method by using the Daubechies wavelet with 8 vanishing moments and the universal threshold $\lambda = \sigma\sqrt{2\log n^2}$. This indicates that customizing the wavelet and the threshold makes significant improvement.

It should be mentioned that our *Customized NeighShrink* causes edges to be smoothed while denoising. This is also true for many other denoising methods. When the noise level is low, this is not a problem. However, when the noise level is high, the denoised image has visually visible smoothed edges. One way of improving this is to apply any of the edge enhancement techniques right after we get the denoised image using our proposed method *Customized NeighShrink*.

6.4 Experimental Results

We conducted some experiments in Matlab by calling the free software package *WaveLab* developed by Donoho et al. The testing images are *Lena*, *MRIScan*, *Fingerprint*, *Phone*, *Canaletto*, and *Lincoln*. These images are available in *WaveLab*. Noisy images with different noise levels are generated by adding white noise to the original noise-free images. For comparison, we implement *VisuShrink*, *Customized NeighShrink*, non-TI *NeighShrink*, and *wiener2* filter to denoise the noisy images. *VisuShrink* is the universal soft-thresholding denoising technique [31] and *NeighShrink* is our proposed algorithm in the previous Chapter. The *wiener2* function is available in the MATLAB Image Processing Toolbox, and we use a 5×5 neighbourhood of each pixel in the image for it. The *wiener2* function applies a Wiener filter (a type of linear filter) to an image adaptively, tailoring itself to the local image variance. When the variance is large, *wiener2* performs little smoothing. When the variance is small, *wiener2* performs more smoothing. This approach often produces better results than linear filtering. The adaptive filter is more selective than a comparable linear filter, preserving edges and other high frequency parts of an image. In addition, there are no design tasks; the *wiener2* function handles all preliminary computations, and implements the filter for an input image. *wiener2*, however, does require more computation

time than linear filtering. Simulated Annealing is used to find the customized wavelet filter and the customized threshold. In Simulated Annealing, we need to define a cost function. This is defined as the mean square error of the denoised image with another noisy copy of the image. We assume we have two noisy copies of the same image because sometimes we can have two sensors at the same time. In our experiments, using another noisy image as a target works very well. The wavelet filter length in our experiments is set to 8. Five detailed wavelet decomposition scales are thresholded. For different Gaussian white noise levels, the experimental results in Peak Signal to Noise Ratio ($PSNR$) are shown in Table 16 - Table 21 for denoising images *Lena*, *MRIScan*, *Fingerprint*, *Phone*, *Canaletto* and *Lincoln*, respectively. The first column in these tables is the $PSNR$ of the original noisy images, while other columns are the PSNR of the denoised images by using different denoising methods. We can see that *Customized NeighShrink* outperforms *VisuShrink*, *NeighShrink*, and *wiener2* filter for all experiments. Note that *VisuShrink* does not have any denoising power when the noise level is low. It produces even worse results than the original noisy images. However, *Customized NeighShrink* performs very well in this case. The improvement of *Customized NeighShrink* over $VisuShrink$ is large when the noise level is low. When the noise level is high, the improvement is low even though *Customized NeighShrink* is still better than $VisuShrink$. Figure 30 - Figure 35 show the noise-free images, the same image with noise added, the denoised image with *VisuShrink*, the denoised image with *Customized NeighShrink*, the denoised image with non-TI *NeighShrink*, and the denoised image with *wiener2* filter for images *Lena*, $MRIScan$, $Fingerprint$, $Phone$, $Canaletto$, and *Lincoln*, respectively. It is not difficult to see that *Customized NeighShrink* produces smoother and clearer denoised images than other denoising methods tested in this Chapter. For some case, choosing the customized wavelet filter and customized threshold gives us about 1.20dB improvement over the same method by using the Daubechies wavelet with 8 vanishing moments and the universal threshold $\lambda = \sigma\sqrt{2\log n^2}$. This indicates that customizing the wavelet and the threshold makes significant improvement.

Noisy Image	*VisuShrink*	*Customized NeighShrink*	*NeighShrink*	*wiener2*
28.14	26.35	**33.08**	32.34	30.66
22.12	23.88	**28.90**	28.26	25.00
18.60	22.87	**26.62**	26.06	21.61
16.10	22.31	**25.08**	24.64	19.17
14.14	21.90	**24.14**	23.68	17.25
12.58	21.53	**23.30**	22.94	15.66
11.24	21.23	**22.74**	22.36	14.30

Table 16: The PSNR (dB) of the noisy images of Lena and the denoised images with different denoising methods.

Noisy Image	*VisuShrink*	*Customized NeighShrink*	*NeighShrink*	*wiener2*
28.14	26.85	**33.46**	33.05	30.74
22.12	24.08	**29.34**	28.97	25.05
18.60	22.74	**27.20**	26.77	21.60
16.10	21.91	**25.92**	25.33	19.11
14.16	21.34	**24.79**	24.26	17.17
12.58	20.90	**23.68**	23.40	15.58
11.24	20.50	**22.90**	22.68	14.23

Table 17: The PSNR (dB) of the noisy images of MRIScan and the denoised images with different denoising methods.

It should be mentioned that we only implemented the non-TI *NeighShrink* for our algorithm *Customized NeighShrink*. We obtained better results than non-TI *NeighShrink* for all the testing images. However, these results are not as good as TI *NeighShrink*. We are sure that if we incorporate TI *NeighShrink* in our algorithm *Customized NeighShrink*, we can obtain better denoising results than TI *NeighShrink*. We leave this to our future work.

Noisy Image	*VisuShrink*	*Customized NeighShrink*	*NeighShrink*	*wiener2*
28.15	25.98	**32.12**	30.94	30.65
22.13	23.86	**28.13**	27.58	25.32
18.61	23.03	**26.67**	25.75	21.87
16.11	22.61	**25.17**	24.65	19.32
14.17	22.36	**24.35**	23.92	17.34
12.59	22.19	**23.96**	23.42	15.72
11.25	22.06	**23.26**	23.04	14.36

Table 18: The PSNR (dB) of the noisy images of Fingerprint and the denoised images with different denoising methods.

Noisy Image	*VisuShrink*	*Customized NeighShrink*	*NeighShrink*	*wiener2*
28.14	26.19	**30.84**	30.08	29.75
22.12	24.12	**27.06**	26.88	24.72
18.60	23.07	**25.02**	24.95	21.56
16.10	22.37	**23.70**	23.52	19.18
14.16	21.73	**22.59**	22.45	17.25
12.57	21.08	**21.68**	21.64	15.63
11.24	20.41	**21.12**	21.06	14.24

Table 19: The PSNR (dB) of the noisy images of Phone and the denoised images with different denoising methods.

Noisy Image	*VisuShrink*	*Customized NeighShrink*	*NeighShrink*	*wiener2*
28.15	25.58	**32.44**	31.43	30.38
22.13	22.85	**28.51**	27.97	25.00
18.61	21.48	**26.18**	26.00	21.70
16.11	20.67	**24.76**	24.59	19.25
14.17	20.17	**23.66**	23.50	17.32
12.59	19.83	**22.93**	22.61	15.72
11.25	19.58	**22.35**	21.92	14.37

Table 20: The PSNR (dB) of the noisy images of Canaletto and the denoised images with different denoising methods.

Noisy Image	*VisuShrink*	*Customized NeighShrink*	*NeighShrink*	*wiener2*
28.19	33.05	**33.76**	33.23	30.89
22.17	27.88	**31.17**	27.87	24.83
18.64	24.53	**29.69**	24.48	21.22
16.14	22.10	**28.25**	22.04	18.66
14.21	20.19	**24.05**	20.13	16.68
12.62	18.62	**22.62**	18.56	15.05
11.28	17.29	**21.35**	17.23	13.67

Table 21: The PSNR (dB) of the noisy images of Lincoln and the denoised images with different denoising methods.

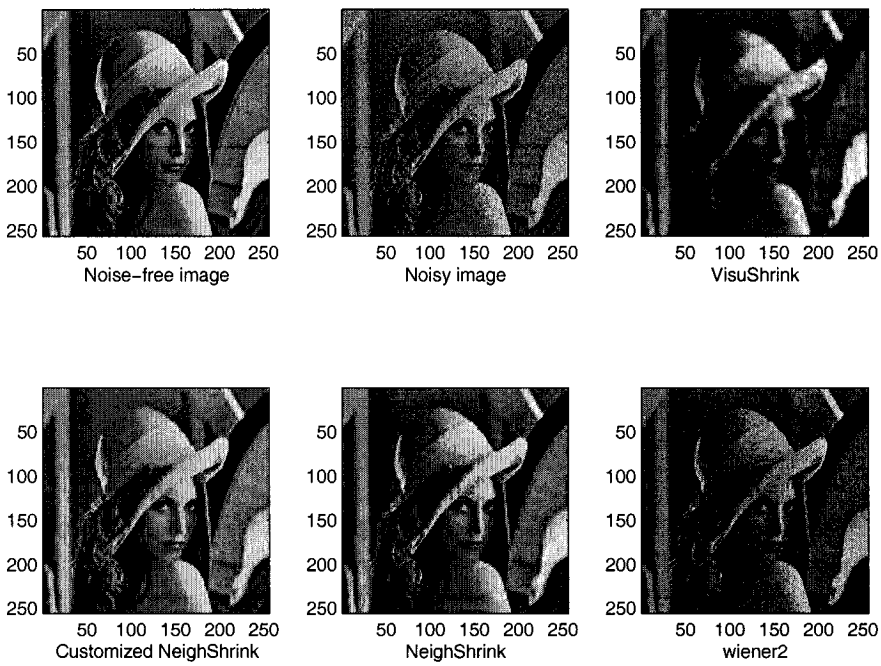

Figure 30: Image denoising by using different methods on a noisy image with PSNR = 22dB.

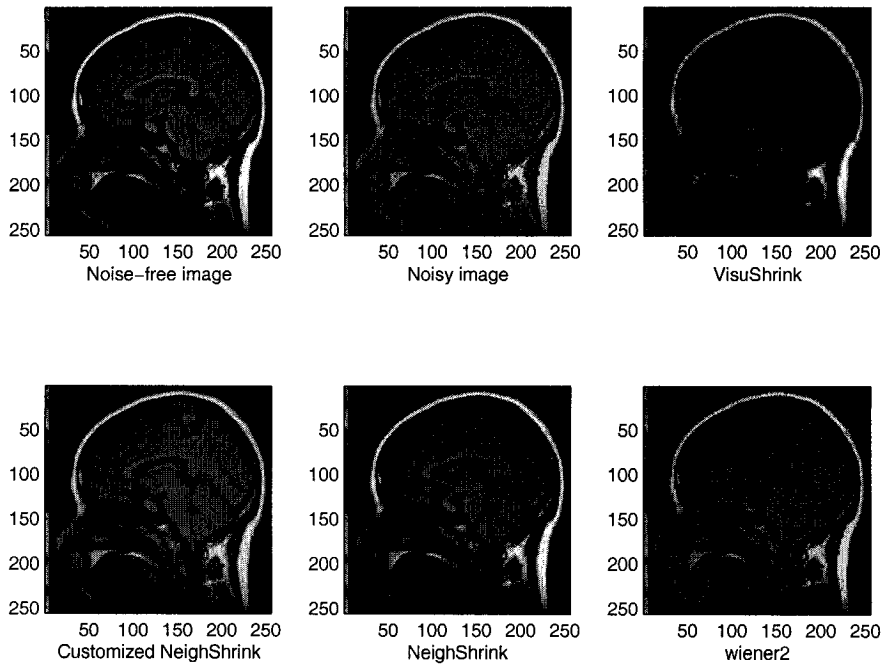

Figure 31: Image denoising by using different methods on a noisy image with PSNR = 22dB.

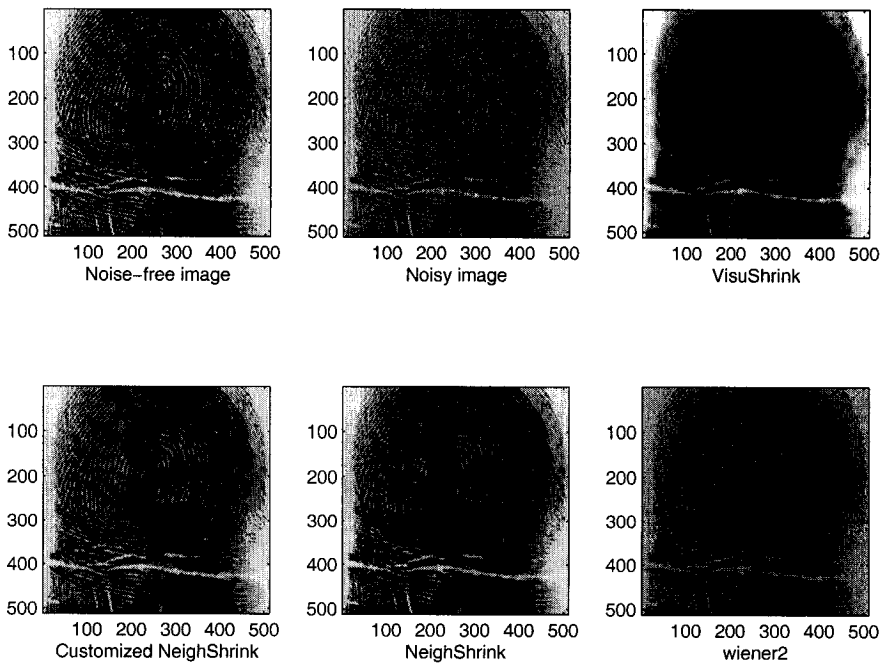

Figure 32: Image denoising by using different methods on a noisy image with PSNR = 22dB.

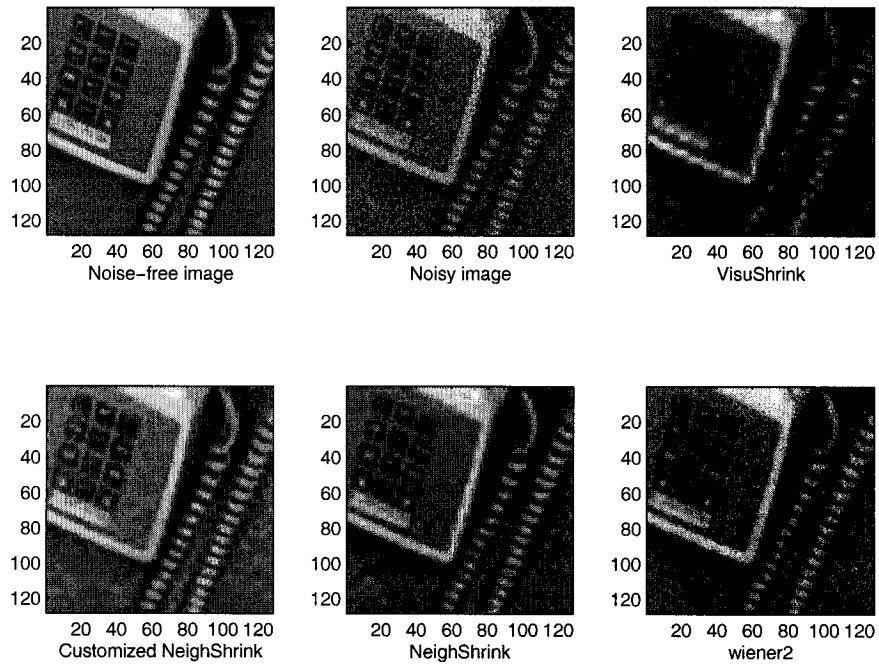

Figure 33: Image denoising by using different methods on a noisy image with PSNR = 22dB.

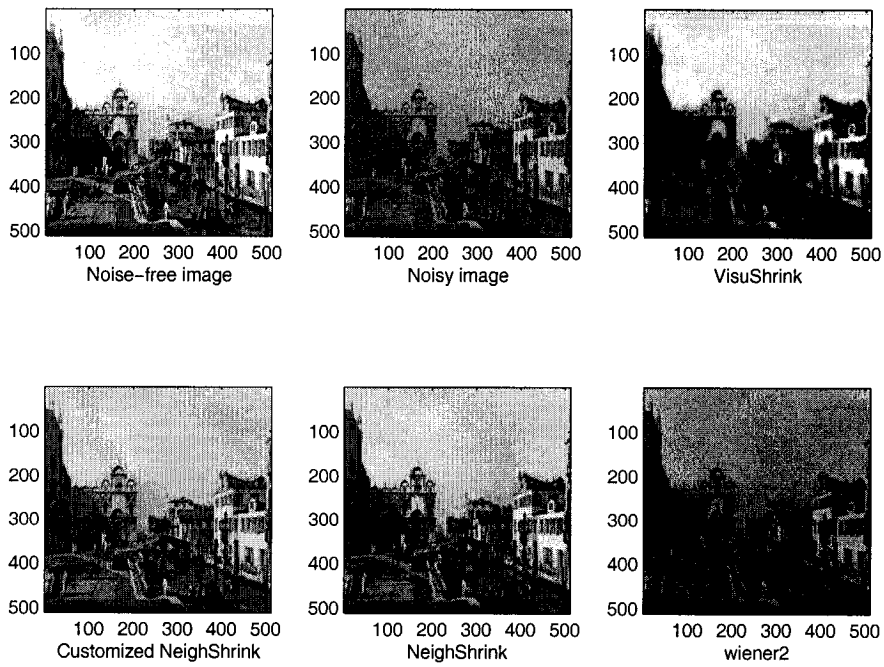

Figure 34: Image denoising by using different methods on a noisy image with PSNR = 22dB.

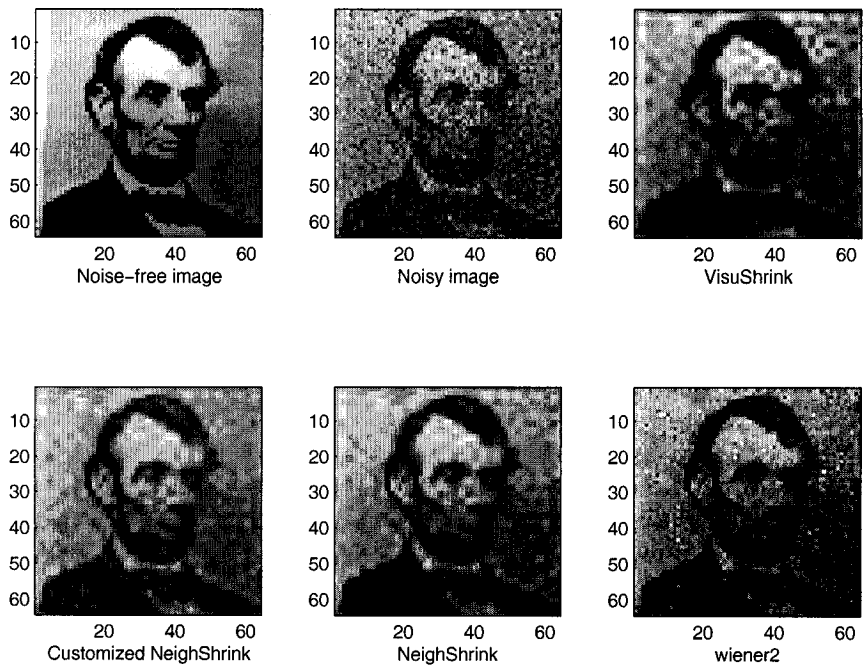

Figure 35: Image denoising by using different methods on a noisy image with PSNR = 34dB.

Chapter 7

Thesis Summary and Future Work

In this thesis, we have studied the applications of wavelets and ridgelets in two important areas: pattern recognition and denoising. We have proposed three invariant descriptors for handwritten numeral recognition and general 2D pattern recognition by using multiwavelets, ridgelets and the Fourier transform. We also proposed three methods for denoising signals and images by considering a neighbourhood of the wavelet coefficients to be thresholded and customizing the wavelet filter and threshold. Our experimental results are promising compared to the current results published in the literature.

In Chapter 2, we introduce a novel set of features that is well-suited for representing digitized handwritten numerals. The features are derived from the multiwavelet shell expansion of the numeral contour. The numeral contours are represented by fine-to-coarse approximations at different resolution levels by means of orthonormal multiwavelet expansion. It is suggested to use the low to intermediate levels of the shell coefficients since these features are relatively insensitive to the shape variation caused by the writing styles of different persons. Multiwavelet shell coefficients depend on the scale and the parameterization starting point of the original function. Therefore, we present normalization that allows us to derive a scale- and shift-invariant multiresolution representation for numerals of known orientation. Experimental results show that our proposed method is better than the scalar wavelet neural network

method in [77].

In Chapter 3, we propose two invariant descriptors for pattern recognition by using ridgelets, wavelet cycle-spinning, and Fourier transform. Our ridgelets work on the circle surrounding the pattern to be recognized. This is different from other ridgelet papers where a square is considered. We achieve rotation- invariance by taking Fourier spectra along the angle direction and by using wavelet cycle-spinning together with Fourier spectra, respectively. Note that our proposed descriptor works for not only Chinese characters, but also any other 2D pattern recognition problems. We plan to investigate texture classification by using ridgelets and wavelet packets. We are expecting a significant improvement over existing techniques for texture classification. It is evident that the two descriptors are very robust to Gaussian white noise.

In Chapter 4, we discuss and implement multiwavelet denoising by incorporating neighbouring coefficients as suggested by Cai and Silverman. Experimental results show that Neighbour multiwavelet denoising gives better results than the term-by-term multiwavelet denoising scheme for both TI and non-TI cases. In addition, neighbour multiwavelet denoising outperforms neighbour single wavelet denoising for some standard test signals and real life images. We conclude then that neighbour multiwavelet denoising can be used in place of neighbour single wavelet denoising. Future work may be done by choosing a better threshold in the neighbour multiwavelet denoising scheme. We may also consider a bigger neighbourhood instead of just the immediate neighbours.

In Chapter 5, we study image denoising by incorporating neighbouring wavelet coefficients. Experimental results show that *NeighShrink* gives better results than *VisuShrink* and the Ti image denoising method developed by Yu et al. under all experiments. It should be mentioned that in this Chapter we investigate only how the classical soft thresholding approach should be modified to take into account neighbour wavelet coefficients. We conclude that *NeighShrink*, for both TI and non-TI cases, can be used for practical image denoising applications. Future work may be done by considering the technique of incorporating neighbour multiwavelet coefficients in

the thresholding process for multiwavelet image denoising. Also, better denoising results may be obtained by combining parent-child relationship with the immediate neighbours in the same wavelet subband.

In Chapter 6, we study image denoising by using customized wavelet and customized threshold. The thresholding process incorporates neighbouring wavelet coefficients. Experimental results show that *Customized NeighShrink* gives better results than *VisuShrink*, *NeighShrink* and *wiener2* filter for all experiments. It should be mentioned that in this Chapter we only investigate how the classical soft thresholding approach should be modified to take into account neighbour wavelet coefficients. We suggest to use *Customized NeighShrink* for practical image denoising applications. Future work may be done by customizing multiwavelets in image denoising. Also, instead of using the universal customized threshold, we may also investigate different customized thresholds for different pixel locations.

Bibliography

[1] V. V. Anh, Q. Tieng, T. D. Bui and G. Y. Chen, "The Hellinger-Kakutani Metric for Pattern Recognition," in *Proceedings of IEEE International Conference on Image Processing (ICIP'97)*, vol. II, pp. 430-433, 1997.

[2] G. Beylkin, "On the Representation of Operators in Bases of Compactly Supported Wavelets," *SIAM J. Numer. Anal.*, vol. 29, pp. 1716-1740, 1992.

[3] T. D. Bui and G. Y. Chen,"Translation Invariant Denoising Using Multiwavelets," *IEEE Transactions on Signal Processing*, vol. 46, no. 12, pp. 3414-3420, 1998.

[4] T. D. Bui, G. Y. Chen and L. Feng, "An Orthonormal-Shell-Fourier Descriptor for Rapid Matching of Patterns in Image Database," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 15, no. 8, pp. 1213-1229, 2001.

[5] T. T. Cai and B. W. Silverman, "Incorporating Information on Neighbouring Coefficients into Wavelet Estimation," *Sankhya: The Indian Journal of Statistics*, vol. 63, Series B, Pt. 2, pp. 127-148, 2001.

[6] E. J. Candes, "Ridgelets and the Representation of Mutilated Sobolev Functions," *SIAM J. Math. Anal.* vol. 33, no. 2, pp. 2495-2509, 1999.

[7] E. J. Candes and D. L. Donoho, "Ridgelets: a Key to Higher-dimensional Intermittency?" *Phil. Trans. R. Soc. Lond. A.*, vol. 357, no. 1760, pp. 2495-2509, 1999.

[8] E. J. Candes, "Ridgelets: Theory and Applications," Ph.D. Thesis, Technical Report, Department of Statistics, Stanford University, 1998.

[9] J. O. Chapa and M. R. Raghuveer, "Optimal Matched Wavelet Construction and its Application to Image Pattern Recognition," *SPIE Proceedings* vol. 2491, Wavelet Applications II, 1995, Orlando, FL, USA, 1995.

[10] J. O. Chapa and M. R. Raghuveer, "Algorithms for Designing Wavelets to Match a Specified Signal," *IEEE Transactions on Signal Processing*, vol. 48, no. 12, pp. 3395-3406, 2000.

[11] G. Y. Chen, "Applications of Wavelet Transforms in Pattern Recognition and Denoising," Master Thesis, Department of Computer Science, Concordia University, Montreal, Quebec, Canada, 1999.

[12] G. Y. Chen and T. D. Bui, "Multiwavelet Denoising using Neighbouring Coefficients," *IEEE Signal Processing Letters*, vol.10, no.7, pp.211-214, 2003.

[13] G. Y. Chen and T. D. Bui, "Invariant Fourier-wavelet Descriptor for Pattern Recognition," *Pattern Recognition*, vol. 32, no. 7, pp. 1083-1088, 1999.

[14] G. Y. Chen, T. D. Bui and A. Krzyżak, "Contour-Based Handwritten Numeral Recognition using Multiwavelets and Neural Networks," *Pattern Recognition*, vol.36, no.7, pp.1597-1604, 2003.

[15] G. Y. Chen, T. D. Bui and A. Krzyżak, "Image Denoising using Neighbouring Wavelet Coefficients," Proceedings of International Conference on Acoustics, Speech, and Signal Processing, Montreal, Quebec, Canada, 2004.

[16] G. Y. Chen, T. D. Bui and A. Krzyżak, "Image Compression with Optimal Wavelet," Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, Niagara Falls, Ontario, Canada, 2004.

[17] G. Y. Chen, T. D. Bui and A. Krzyżak, "Optimal Wavelets and Neural Networks for Pattern Recognition," Proceedings of Image and Vision Computing, New Zealand, pp. 315-319, 2003.

[18] G. Y. Chen, T. D. Bui and A. Krzyżak, "Rotation Invariant Pattern Recognition using Ridgelet, Wavelet Cycle-Spinning, and Fourier Features," Submitted to *Pattern Recognition.*

[19] G. Y. Chen, T. D. Bui and A. Krzyżak, "Invariant Ridgelet-Fourier Descriptor for Pattern Recognition," Submitted to *IEEE Transactions on PAMI.*

[20] G. Y. Chen, T. D. Bui and A. Krzyżak, "Image Denoising with Neighbour Dependency and Customized Wavelet and Threshold," submitted to *Pattern Recognition.*

[21] C. K. Chui, *An Introduction to Wavelets*, Boston: Academic Press, 1992.

[22] R. R. Coifman and D. L. Donoho, "Translation Invariant Denoising," In *Wavelets and Statistics, Springer Lecture Notes in Statistics 103*, pp. 125-150, New York:Springer-Verlag.

[23] M. S. Crouse, R. D Nowak and R. G. Baraniuk, "Wavelet-based Signal Processing Using Hidden Markov Models," *IEEE Transactions on Signal Processing*, vol. 46, no. 4, pp. 886-902, 1998.

[24] Y. L. Cun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[25] Y. L. Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten Digit Recognition with a Back-propagation Network'," *Advances in Neural Information Processing Systems 2* (NIPS*89), Denver, CO, 1990. Morgan Kaufman.

[26] Y. L. Cun, L. D. Jackel, L. Bottou, A. Brunot, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Comparison of Learning Algorithms for Handwritten Digit Recognition," *International Conference on Artificial Neural Networks*, pp. 53-60, Paris, 1995.

[27] A. Das, U. B. Desai and P. P. Vaidya, "Search for Optimal Basis for Signal Denoising in the Space of n-tap Wavelets," *International Symposium on Signal Processing and its Applications (ISSPA)*, Kuala Lumpur, Malaysia, pp. 96-99, 2001.

[28] I. Daubechies, *Ten Lectures on Wavelets*, Philadelpha: SIAM, 1992.

[29] I. Daubechies, "Orthonormal Bases of Compactly Supported Wavelets," *Comm. on Pure and Applied Mathematics* **41**, pp. 909-996, 1988.

[30] M. N. Do and M. Vetterli, "The finite ridgelet transform for image representation," *IEEE Transactions on Image Processing*, vol. 12, no. 1, pp. 16-28, 2003.

[31] D. L. Donoho, "Denoising by Soft-thresholding," *IEEE Transactions on Information Theory*, vol. 41, pp. 613-627, 1995.

[32] D. L. Donoho and I. M. Johnstone, "Adaptating to Unknown Smoothness via Wavelet Shrinkage," *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1200-1224, 1995.

[33] D. L. Donoho and A. G. Flesia, "Digital Ridgelet Transform based on true ridge functions," *Beyond Wavelets*, J. Stoecker and G. V. Welland eds., Academic Press, 2001.

[34] T. R. Downie and B. W. Silverman, "The Discrete Multiple Wavelet Transform and Thresholding Methods," *IEEE Transactions on Signal Processing*, vol. 46, no. 9, pp. 2558-2561, 1998.

[35] A. G. Flesia, H. Hel-Or, A. Averbuch, E. J. Candes, R. R. Coifman and D. L. Donoho, "Digital Implementation of Ridgelet Packets," *Beyond Wavelets*, J. Stoeckler and G. V. Welland eds., Academic Press, 2001.

[36] J. S. Geronimo, D. P. Hardin and P. R. Massopust, "Fractal Functions and Wavelet Expansions Based on Several Scaling Functions," *Journal of Approximation Theory*, vol. 78, pp. 373-401, 1994.

[37] S. A. Golden, "Identifying Multiscale Statistical Models using the Wavelet Transform," M.S. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, USA, 1991.

[38] G. H. Granlund, "Fourier Processing for Hand Print Character Recognition", *IEEE Transactions on Computers*, vol. 21, no. 3, pp. 195-201, 1972.

[39] J. Hertz, A. Krogh and R. G. Palmer, "Introduction to the Theory of Neural Computation," Addision-Wesley, Redwood City, California, 1991.

[40] P. J. Huber, *Robust Statistics*, Wiley & Sons, New York, USA, 1981.

[41] M. I. Khalil and M. M. Bayoumi, "Invariant 2D object recognition using the wavelet modulus maxima," *Pattern Recognition Letters*, vol. 21, no. 9, pp. 863-872, 2000.

[42] A. Khotanzad and Y. H. Hong, "Invariant Image Recognition by Zernike Moments," *IEEE Transactions on PAMI*, vol. 12, no.5, pp. 489-497, 1990.

[43] S. Kirkpatrick, Jr. C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," Science, vol. 220, no. 4598, pp. 671-680, 1983.

[44] F. P. Kuhl and C. R. Giardina,"Elliptic Fourier Features of a Closed Contour," *Comput. Vis. Graphics Image Process*, vol. 18, no. 3, pp. 236-258, 1982.

[45] S. W. Lee, C. H. Kim, H. Ma and Y. Y. Tang, "Multiresolution Recognition of Unconstrained Handwritten Numerals with Wavelet Transform and Multilayer

Cluster Neural Network," *Pattern Recognition*, vol. 29, no. 12, pp. 1953-1961, 1996.

[46] C. C. Lin and R. Chellappa, "Classification of partial 2-D Shapes using Fourier Descriptors," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 5, pp. 686-690, 1987.

[47] C. S. Lin and C.-L. Hwang, "New Forms of Shape Invariants from Elliptic Fourier Descriptor," *Pattern Recognition*, vol, 20, no. 5, pp.535-545, 1987.

[48] S. Mallat, "Zero-crossing of a Wavelet Transform," *IEEE Transactions on Information Theory*, vol.37, no.4, pp. 1019-1033, 1991.

[49] S. Mallat, "Multifrequency Channel Decomposition of Images and Wavelet Models," *IEEE Trans. Acoust. Speech Signal Processing*, vol.37, no.12, pp. 2091-2110, 1989.

[50] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol.11, no.7, pp. 674-693, 1989.

[51] S. Mallat, "Multiresolution Approximations and Wavelet Orthonormal Bases of $L_2(R)$," *Trans. Amer. Math. Soc.* vol.315, pp. 69-87, 1989.

[52] S. Mallat and S. Zhong, "Characterisation of Signals from Multiscale Edges," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.14, no.7, pp. 710-732, 1992.

[53] Y. Mallet, D. Coomans, J. Kautsky, O. De Vel, "Classification Using Adaptive Wavelets for Feature Extraction," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 10, pp. 1058-1066, 1997

[54] M. K. Mihcak, I. Kozintsev, K. Ramchandran and P. Moulin, "Low-complexity Image Denoising Based on Statistical Modeling of Wavelet Coefficients," *IEEE Signal Processing Letters*, vol. 6, no. 12, pp. 300-303, 1999.

[55] D. W. Paglieroni and A. K. Jain, "Fast Classification of Discrete Shape Contours," *Pattern Recognition*, vol. 20, no. 6, pp. 583-598, 1987.

[56] N. Saito and G. Beylkin, "Multiresolution Representation Using the Autocorrelation Functions of Compactly Supported Wavelets," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp.3584-3590, 1993.

[57] I. Sekita, K. Toraichi, R. Mori, K. Yamamoto and H. Yamada, "Feature Extraction of Handwritten Japanese Characters by Spline Functions for Relaxation Matching," *Pattern Recognition*, vol. 21, no. 1, pp. 9-17, 1990.

[58] L. Sendur and I. W. Selesnick, "Bivariate Shrinkage Functions for Wavelet-based Denoising Exploiting Interscale Dependency," *IEEE Transactions on Signal Processing*, vol. 50, no. 11, pp. 2744-2756, 2002.

[59] L. Sendur and I. W. Selesnick, "Bivariate Shrinkage with Local Variance Estimation," *IEEE Signal Processing Letters*, vol. 9, no. 12, pp. 438-441, 2002.

[60] D. Shen and H. H. S. Ip, "Discriminative Wavelet Shape Descriptors for Recognition of 2D Patterns," *Pattern Recognition*, vol. 32, no. 2, pp. 151-165, 1999.

[61] W. Shengqian, Z. Yuanhua and Z. Daowen, "Adaptive Shrinkage Denoising Using Neighbourhood Characteristic," *Electronics Letters*, vol. 38, no. 11, pp. 502-503, 2002.

[62] E. P. Simoncelli and E. H. Adelson, "Noise Removal via Bayesian Wavelet Coring," *The 3rd International Conference on Image Processing*, Lausanne, Switzerland, 1996.

[63] J. L. Starck, E. J. Candes and D. L. Donoho, " Astronomical Image Representation by the Curvelet Transform," *Astronomy & Astrophysics*, vol. 398, no. 2, pp. 785-800, 2003.

[64] J. L. Starck, E. J. Candes and D. L. Donoho, "The Curvelet Transform for Image Denoising," *IEEE Transactions on Image Processing*, vol. 11, no. 6, pp.670–684, 2002.

[65] G. Strang, "Wavelets and Dilation Equations: A Brief Introduction." *SIAM Review*, vol. 31, no. 4, pp. 614-627, 1989.

[66] G. Strang and V. Strela, "Orthogonal Multiwavelets with Vanishing Moments," *Optical Engineering*, vol. 33, pp. 2104-2107, 1994.

[67] V. Strela, P. N. Heller, G. Strang, P. Topiwala, and C. Heil, "The Application of Multiwavelet Filter Banks to Image Processing," *IEEE Transactions on Image Processing*, vol.8, no.4, pp. 548-563, 1999.

[68] Y. Tao, E. C. M. Lam and Y. Y. Tang, "Feature extraction using wavelet and fractal," *Pattern Recognition Letters*, vol. 22, no. 3-4, pp. 271-287, 2001.

[69] T. Taxt, J. B. Olafsdottir and M. Daehlen, "Recognition of Handwritten Symbols," *Pattern Recognition*, vol. 23, no. 11, pp. 1155-1166, 1990.

[70] A. H. Tewfik, D. Sinha and P. Jorgensen, "On the Optimal Choice of a Wavelet for Signal Representation," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 747-765, 1992.

[71] Q. T. Tieng and W. W. Boles, "Recognition of 2D Object Contours using Wavelet Transform Zero-crossing Representation," *IEEE Transactions on PAMI*, vol. 19, no. 8, pp. 910-916, 1997.

[72] Q. M. Tieng and W. W. Boles, "An Application of Wavelet Based Affine-invariant Representation," *Pattern Recognition Letters*, vol. 16, no. 8, pp. 1287-1296, 1995.

[73] Q. M. Tieng and W. W. Boles, "Wavelet-based Affine Invariant Representation: A Tool for Recognizing Planar Objects in 3D Space," *IEEE Transactions on PAMI*, vol. 19, no. 8, pp. 846-857, 1997.

[74] O. D. Trier, A. K. Jain and T. Taxt, "Feature Extraction Methods for Character Recognition: A Survey," Pattern Recognition, vol. 29, pp. 641-662, 1996.

[75] S. S. Wang, P. C. Chen and W. G. Lin, "Invariant Pattern Recognition by Moment Fourier Descriptor," *Pattern Recognition*, vol. 27, no. 12, pp. 1735-1742, 1994.

[76] X. Wang and H. Qi, "Face Recognition Using Optimal Non-Orthogonal Wavelet Basis Evaluated by Information Complexity," *16 th International Conference on Pattern Recognition (ICPR'02)*, Volume 1, August 11 - 15, 2002, Quebec City, QC, Canada.

[77] P. Wunsch and A. F. Laine, "Wavelet Descriptors for Multiresolution Recognition of Handprinted Characters," *Pattern Recognition*, vol. 28, no. 8, pp. 1237-1249, 1995.

[78] X. G. Xia, J. Geronimo, D. Hardin, and B. Suter, "Design of Prefilters for Discrete Multiwavelet Transforms," *IEEE Transactions on Signal Processing*, vol. 44, pp. 25-35, 1996.

[79] L. H. Yang, C. Y. Suen, T. D. Bui and P. Zhang, "Discrimination of Similiar Handwritten Numerals Based on Invariant Curvature Features," Submitted to *Pattern Recognition.*

[80] L. H. Yang, T. D. Bui and C. Y. Suen, "Image Recognition Based on Nonlinear Wavelet Approximation," *International Journal of Wavelets, Multiresolution and Information Processing*," vol. 1, no. 2, pp. 151-161, 2003.

[81] T. P.Y. Yu, A. Stoschek and D. L. Donoho, "Translation- and Direction-Invariant Denoising of 2-D and 3-D Images," In Wavelet Applications in Signal and Image Processing IV, Proceedings SPIE, 1996.

[82] C. T. Zahn and R. C. Roskies, "Fourier Descriptors for Plane Closed Curves," *IEEE Transactions on Computers*, vol. 21, no. 3, pp. 269-281, 1972.

[83] M. Zhang, C. Y. Suen and T. D. Bui, "Feature Extraction in Character Recognition with Associative Memory Classifier," *International Journal of Pattern Recognition and Artificial Intelligence* **10**, pp. 325-348, 1996.

[84] Y. Zhuang and J. S. Baras, " Constructing Optimal Wavelet Basis for Image Compression," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 2351-2354, 1996.

[85] H. Zou and A. H. Tewfik, "Parametrization of Compactly Supported Orthonormal Wavelets," *IEEE Transactions on Signal Processing*, vol. 41, no. 3, pp. 1428-1531, 1993.