NOISE REDUCTION USING WAVELET METHODS

Thanakorn Piyapai

มหาวิทยาลัยเทคโนโลยีสุรนารี

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Master of Science in Applied Mathematics

Suranaree University of Technology

Academic Year 2018

การลดสัญญาณรบกวนบนภาพด้วยวิธีเวฟเล็ต

มหาวิทยาลัยเทคโนโลยีสุรนารี

นายฐนกร ปิยะไพร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2561

NOISE REDUCTION USING WAVELET METHODS

Suranaree University of Technology has approved this thesis submitted in partial fulfillment of the requirements for the Degree of Master of Science.

Thesis Examining Committee

J. Tanthanuch

(Asst. Prof. Dr. Jessada Tanthanuch)

Chairperson

(Assoc. Prof. Dr. Eckart Schulz)

Member (Thesis Advisor)

(Asst. Prof. Dr. Arjuna Chaiyasena)

Member

Benjawan Rodjanadid

(Asst. Prof. Dr. Benjawan Rodjanadid)

Member

(Assoc. Prof. Flt. Lt. Dr. Kontorn Chamniprasart)

Vice Rector for Academic Affairs

and Internationalization

(Assoc. Prof. Dr. Worawat Meevasana)

Dean of Institute of Science

มหาวิทยาลัยเทคโนโลยีสุรนารี

ฐนกร ปิยะไพร : การลดสัญญาณรบกวนบนภาพด้วยวิธีเวฟเล็ต (NOISE REDUCTION USING WAVELET METHODS) อาจารย์ที่ปรึกษา : รองศาสตราจารย์ ดร.เอ็กคาร์ท ซูลส์, 81 หน้า.

วิทยานิพนธ์นี้ได้ทำการศึกษาการประยุกต์ใช้เวฟเล็ตเพื่อลดสัญญาณรบกวนแบบสเปคเคิลที่เกิดขึ้นในภาพ ภายใต้สมมุติฐานที่ว่าสัญญาณรบกวนที่เกิดขึ้นมีรูปแบบการแจกแจงเรย์ลี การปรับปรุงภาพได้ใช้ฮาร์เวฟเล็ตและเดาเบคีส์เวฟเล็ตที่ระดับการแบ่งละเอียดที่แตกต่างกัน ประยุกต์ใช้ในวิธีขีดแบ่งแบบแข็งและแบบอ่อนในระดับขีดแบ่งที่แตกต่างกัน เพื่อปรับปรุงภาพที่สังเคราะห์จากการนำภาพปกติมาใส่สัญญาณรบกวนเรย์ลี ผลที่ได้จากการใช้วิธีเวฟเล็ตจะถูกนำไปเปรียบเทียบกับวิธีการลดสัญญาณรบกวนในภาพแบบอื่น ที่นิยมใช้กันอย่างแพร่หลาย ซึ่งประกอบไปด้วยวิธีการลดสัญญาณรบกวนด้วยวิธีการแปรค่าและการแปลงลอการิทึม การเปรียบเทียบดังกล่าวจะใช้หลายดัชนีในการเปรียบเทียบคุณภาพ

มหาวิทยาลัยเทคโนโลยีสุรนารี

สาขาวิชาคณิตศาสตร์ ลายมือชื่อนักศึกษา ฐนกร ปิยะไพร

ปีการศึกษา 2561 ลายมือชื่ออาจารย์ที่ปรึกษา

THANAKORN PIYAPAI : NOISE REDUCTION USING WAVELET METHODS. THESIS ADVISOR : ASSOC. PROF. ECKART SCHULZ, Ph.D. 81 PP.

Wavelets / Variational Model / Speckle Noise / Ultrasound Imaging

The application of wavelet methods to the reduction of speckle noise in images is studied, under the assumption that speckle noise is Rayleigh distributed. Hard and soft-thresholding methods at various threshold levels are applied to a synthetic image imposed with Rayleigh noise, using Haar wavelets and Daubechies wavelets at various refinement levels. The results are compared with other popular denoising methods, including variational models and logarithmic transformation, by applying several image quality indices.

มหาวิทยาลัยเทคโนโลยีสุรนารี

School of Mathematics

Academic Year 2018

Student's Signature ธนากร ปิยะไพร

Advisor's Signature

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my thesis advisor, Assoc. Prof. Dr. Eckart Schulz for the continuous support of my research and financial assistance, for his patience, providing me with extensive literature background and offering many useful suggestions.

In addition, I would like to acknowledge my thesis committee composed of members of the School of Mathematics, Suranaree University of Technology. These are Asst. Prof. Dr. Arjuna Chaiyasena, Asst. Prof. Dr. Benjawan Rodjanadid and Asst. Prof. Dr. Jessada Tanthanuch.

My sincere thanks also goes to Asst. Prof. Dr. Jessada Tanthanuch, for invaluable guidance in ultrasound imaging technique, for his encouragement and motivation. I am extending my thanks to the Development and Promotion of Science and Technology Talents Project (DPST) for offering a scholarship and educational opportunities. Finally, I am very grateful to my parents and my friends for all their support throughout the period of this research.

Thanakorn Piyapai

หาวิทยาลัยเทคโนโลยีสุร

CONTENTS

Page

ABSTRACT IN THAI I

ABSTRACT IN ENGLISH II

ACKNOWLEDGEMENTS III

CONTENTS IV

LIST OF TABLES VII

LIST OF FIGURES VIII

CHAPTER

I INTRODUCTION 1

- 1.1 Background 1
- 1.2 Research Objectives 3
- 1.3 Organization 3

II BASIC BACKGROUND 4

- 2.1 Image Noise 4
- 2.2 Local Filtering Techniques 5
  - 2.2.1 Mean Filter 6
  - 2.2.2 Median Filter 6
  - 2.2.3 Lee Filter 8
  - 2.2.4 Kuan Filter 9
  - 2.2.5 Lee-Sigma Filter 9
  - 2.2.6 Frost Filter 10

มหาวิทยาลัยเทคโนโลยีสุรนารี

CONTENTS (Continued)

Page

2.3 Global Filtering Techniques 11

2.3.1 Variational Methods 11

2.3.2 Thresholding Methods 12

III WAVELETS AND WAVELET THRESHOLDING 13

3.1 Discrete Wavelets 13

3.2 Wavelets from Multiresolution Analysis 16

3.2.1 Haar Wavelets 20

3.2.2 The Daubechies Wavelets 23

3.3 Two-dimensional Wavelets 24

3.4 Thresholding Methods 26

IV IMAGE QUALITY MEASURES 28

4.1 Noisy Image Generation 28

4.2 Image Quality Indices 29

V RESULTS 34

5.1 Thresholding of Wavelet Coefficients 34

5.2 Results for Wavelet Methods 37

5.3 Comparison With Other Methods 46

VI CONCLUSION 51

REFERENCES 55

APPENDICES

APPENDIX A OCTAVE CODE 59

A.1 Code to Impose Rayleigh Noise on an Image 59

A.2 Code for the Mean Filter 59

A.3 Code for the Mean Median Filter 61

มหาวิทยาลัยเทคโนโลยีสุรนารี

CONTENTS (Continued)

Page

A.4 Code for the Multiplicative Lee Filter 62

A.5 Code for the Seekot's Variational Method 64

A.6 Code for the ROF Variational Method 66

A.7 Code for the Haar Wavelet Method with Hard Thresholding 68

A.8 Code for the Haar Wavelet Method with Soft Thresholding 72

A.9 Code for the Daubechies Wavelet Method with Hard Thresholding 76

A.10 Code for the Daubechies Wavelet Method with Soft Thresholding . 78

CURRICULUM VITAE 81

มหาวิทยาลัยเทคโนโลยีสุรนารี

LIST OF TABLES

Table **Page**

3.1 The nonzero scaling coefficients for the *db*2 - *db*10 wavelets (Source: Daubechies (1992)). 23

5.1 Comparison of wavelet noise reduction methods without log transform applied 39

5.2 Comparison of wavelet noise reduction methods with log transform applied 40

5.3 Comparison of various noise reduction methods without log transform applied 47

5.4 Comparison of various noise reduction methods with log transform applied 48

มหาวิทยาลัยเทคโนโลยีสุรนารี

LIST OF FIGURES

Figure **Page**

3.1 Bottom to top : the Haar scaling function ϕ, Haar mother wavelet ψ and wavelets $\psi_{j,k}$ at scales $j = 1, 2$ on the interval $[0, T] = [0, 1]$. (Source: https://www.researchgate.net) 22

3.2 The Daubechies scaling functions $D4$, $D8$, $D12$ and $D16$. 24

3.3 The Daubechies scaling function db20. 25

3.4 The Daubechies mother wavelet db20. 25

4.1 The noise-free Lena image 30

4.2 The Lena image with Rayleigh noise 30

5.1 1/4 thresholding 35

5.2 1/2 thresholding 35

5.3 3/4 thresholding 36

5.4 15/16 thresholding 37

5.5 63/64 thresholding 37

5.6 Haar, 1/4 hard thresholding 41

5.7 Haar, 1/4 soft thresholding 41

5.8 Haar, 1/2 hard thresholding 41

5.9 Haar, 1/2 soft thresholding 41

5.10 Haar, 3/4 hard thresholding 42

5.11 Haar, 3/4 soft thresholding 42

5.12 Haar, 15/16 hard thresholding 42

5.13 Haar, 15/16 soft thresholding 42

มหาวิทยาลัยเทคโนโลยีสุรนารี

LIST OF FIGURES (Continued)

Page

5.14 Haar, 63/64 hard thresholding 43
5.15 Haar, 63/64 soft thresholding 43
5.16 Haar, hard thresholding applied to all wavelet coefficients 43
5.17 Haar, soft thresholding applied to all wavelet coefficients 43
5.18 db20, 63/64 hard thresholding 44
5.19 db20, 63/64 soft thresholding 44
5.20 db20, hard thresholding applied to all wavelet coefficients 44
5.21 db20 soft thresholding applied to all wavelet coefficients 44
5.22 Haar, 63/64 soft thresholding, log applied 45
5.23 Haar, 63/64 soft thresholding, log not applied 45
5.24 db20, 63/64 soft thresholding, log applied 45
5.25 db20, 63/64 soft thresholding, log not applied 45
5.26 Mean filter, 5 × 5 window 49
5.27 Mean filter, 7 × 7 window 49
5.28 Median filter, 5 × 5 window 49
5.29 Median filter, 7 × 7 window 49
5.30 Multiplicative Lee filter, 5 × 5 window 50
5.31 Multiplicative Lee filter, 7 × 7 window 50
5.32 ROF variational filter 50
5.33 Seekot variational filter 50

มหาวิทยาลัยเทคโนโลยีสุรนารี

CHAPTER I

INTRODUCTION

1.1 Background

Image noise is present in practically all kinds of imaging, whether analog or digital. The type of noise most noticeable in ultrasound images and small aperture radar (SAR) images is called *speckle noise*, because it appears in an image as very dark or very bright spots. Speckle noise results in reduced quality of an image by blurring fine details such as edges, shapes, pixel intensity values, etc. Naturally, image noise reduction or removal is a very active and intense field of research, and there is a vast collection of literature available. Many popular denoising techniques involve local averaging. These include the Lee filter (Lee 1980), the Frost filter (Frost et al., 1982) and the Kuan filter (Kuan et al., 1985). Over time there have been numerous refinements to these types of filters in order to improve their performance for specific applications. Another set of noise reduction techniques is global in nature. Among the most common are variational and thresholding methods. The variational method was popularized by the paper of Rudin, Osher and Fatemi (1992). Here, the denoised image is obtained as the minimizer of a certain functional which measures the combination of the average gradient (i.e. total variation) throughout the desired image together with its distance to the noisy image. Thus, the image which minimizes this functional can be considered the denoised image. Experiments have shown that this method preserves edges relatively well. Its drawback is that low-amplitude variations such as image texture disappear. Donoho and Johnstone (1994) and Donoho (1995) were the first to show that wavelet transform thresholding methods can be used not only for image compression, but also for filtering / denoising purposes. The idea is that the wavelet coefficients which fall below a certain threshold value are attributable to noise; thus setting them to zero will remove this noise. The drawback is that Gibbs-like effects can appear, which are very detrimental to image quality. Furthermore, some experimental results indicate that wavelet transform thresholding is inferior to Fourier transform thresholding (Mateo and Fernández-Caballero, 2009). To overcome this problem, there have been some studies which combine the threshold with the variational method. Chan and Zhou (2000) first apply wavelet thresholding, followed by the variational method. Krommweh and Ma (2010) do not set the wavelet coefficients falling below the threshold to zero, but set them to a value which minimizes the total variation.

Most studies and experiments on noise reduction by wavelets assume that noise is additive and Gaussian distributed. On the other hand, speckle noise in ultrasound images is usually modeled to be of multiplicative nature having the Rayleigh distribution. In addition, the denoising quality naturally depends on the choice of wavelet used, particularly its smoothness. It is of interest to investigate what noise reduction techniques are appropriate for speckle noise and what wavelets to use for this purpose.

มหาวิทยาลัยเทคโนโลยีสุรนารี

In practical applications, ultrasound images are frequently modified by a logarithmic transformation to improve readability. This transformation changes the multiplicative Rayleigh distributed noise to additive-type noise, and one would expect that linear methods, of which the wavelet method is one, should perform better in this case. It is thus interesting to compare the performance of noise reduction with and without the logarithmic transformation.

1.2 Research Objectives

The main objective of this thesis is to study the effectiveness of the wavelet transform combined with thresholding to reduce noise in a digital image whose pixel intensities are Rayleigh distributed. In particular, it seeks to find out what wavelets are best suited for this purpose and to which wavelet coefficients thresholding should be applied.

The second objective is to investigate whether applying the logarithmic transform will improve the performance of the noise-reduction methods, in particular the wavelet transform method.

Lastly, the performance of the wavelet noise reduction method is to be compared with other methods.

1.3 Organization

This thesis is organized as follows. In Chapter II, the types of noise and common techniques for noise reduction are reviewed. The theory of wavelets is presented in Chapter III, and the wavelets used in this thesis are introduced. Chapter IV discusses several metrics that can employed to measure image quality. Chapter V presents the experimental results of the investigations, as well as a discussion of them. The final Chapter VI then summarizes the results of this study. For the sake of completeness, the Octave computer code used for computations is listed in the Appendix.

มหาวิทยาลัยเทคโนโลยีสุรนารี

CHAPTER II

BASIC BACKGROUND

In this chapter, we introduce the representation of digital images and review the various common image noise reduction methods.

2.1 Image Noise

In digital image processing, an image is commonly divided into a finite number of dots, called *pixels*. Thus, the image can be represented by an array of some size $m \times n$, and each entry of this array is providing information on the corresponding pixel of the image. For example, if the image is black and white, as in an ultrasound image, then a single numerical value is assigned to each pixel, representing the gray-level or brightness or intensity of the image at this pixel.

In order to obtain a workable model, image noise is usually considered as consisting of two components, an additive and a multiplicative one. Mathematically, one uses an equation

$$g(x,y) = f(x,y) \cdot u(x,y) + \eta(x,y) \tag{2.1}$$

where $f(x,y)$ denotes the brightness of the original noiseless image $g(x,y)$ the brightness of the corrupted image, $u(x,y)$ is the multiplicative component and $\eta(x,y)$ is the additive component of noise, all at pixel (x,y). This means that multiplicative noise is amplified by pixel brightness, while additive noise is independent of pixel brightness.

In ultrasound imaging, the additive component is insignificant when com-

มหาวิทยาลัยเทคโนโลยีสุรนารี

pared to the multiplicative component, and may thus be ignored. Thus, Equation (2.1) simplifies to

$$g(x, y) = f(x, y) \cdot u(x, y). \tag{2.2}$$

The goal of noise reduction is to find an algorithm which replaces the noisy pixel values $g(x, y)$ with some “denoised” values $\tilde{f}(x, y)$ which are close to the noiseless values $f(x, y)$.

There is vast literature on image denoising techniques. Broadly speaking, they fall into two categories: local methods and global methods. In addition, statistical tools are often part of the denoising process. In the following, we will review the most common of these denoising methods.

2.2 Local Filtering Techniques

The most simple and common noise reduction techniques in digital image processing are of local nature. The idea is that the pixels in vicinity of a given pixel should have similar values. Hence, given a pixel (x, y), a small area of surrounding pixels is designated as a window, and the numerical value of the given pixel is replaced by some type of average of the values of all the pixels within this window. In practice, one usually chooses a window of square size where the pixel (x, y) is located at the center of the square. In the following, S_{xy} will denote the set of coordinates of all the pixels inside the window around pixel (x, y). Thus, its cardinality $|S_{xy}|$ will denote the number of pixels in this window. Furthermore, $\tilde{f}(x, y)$ will denote the value of the denoised image at pixel (x, y).

2.2.1 Mean Filter

The *mean filter* is the simplest of the local filters. It simply replaces the pixel value $g(x, y)$ of the noisy image at pixel (x, y) with the arithmetic mean of the values of all pixels inside the window:

$$\tilde{f}(x, y) = \frac{1}{|S_{xy}|} \sum_{(s,t) \in S_{xy}} g(s, t). \tag{2.3}$$

As a drawback, this filter tends to blur the image and results in loss of detail, unless the window is chosen extremely small.

The mean filter works best when noise is additive with zero mean. We note that multiplicative noise can always be converted to additive noise: Applying the natural logarithm to (2.2) we obtain

$$\ln g(x, y) = \ln f(x, y) + \ln u(x, y),$$

assuming that all quantities are positive.

2.2.2 Median Filter

Alternatively, in the *median filter* one replaces the given pixel value with the median of all the pixel values in the window. Its mathematical model is given by the equation

$$\tilde{f}(x, y) = \operatorname*{median}_{(s,t) \in S_{xy}} g(s, t).$$

This filter is good at reducing impulsive noise and theoretically preserves edges better than the mean filter.

Many local filters are based on the statistical properties of the noise. One assumes that *all quantities are random variables* and that noise whether of additive or multiplicative type at different pixels is identically distributed and uncorrelated, and independent of the pixel intensity. Thus, one considers models

$$g(x,y) = f(x,y) + u(x,y),$$

resp.

$$g(x,y) = f(x,y) \cdot u(x,y).$$

where at each pixel (x,y), the random variables $f(x,y)$ and $u(x,y)$ are independent, $u(x,y)$ and $u(\tilde{x},\tilde{y})$ are uncorrelated, yet identically distributed for $(x,y) \neq (\tilde{x},\tilde{y})$.

We will use the following notations:

$\nu = \nu(x,y) = E(f(x,y))$: mean of the original, noise-free image intensity at pixel (x,y)

$\sigma_f^2 = \sigma_f^2(x,y)$: variance of the original, noise-free image at pixel (x,y)

$\mu = \mu(x,y) = E(g(x,y))$: mean of the noisy image intensity at pixel (x,y)

$\sigma_g^2 = \sigma_g^2(x,y)$: variance of the noisy image at pixel (x,y)

$\overline{n} = E(u)$: mean of noise u

σ_n^2 : variance of noise.

It is natural to assume that noise is pixel-independent, and that the noise-free pixel values $f(x,y)$ and noise $u(x,y)$ are independent random variables. The values of ν and σ_f^2 are in general unknown. The values of μ and σ_g^2 can be estimated from the noisy image by local averaging: Placing a small window S_{xy} around pixel (x,y), then $\mu(x,y)$ is estimated by equation (2.3) while $\sigma_g^2(x,y)$ can be estimated by

$$\sigma_g^2(x,y) = \frac{1}{|S_{xy}|} \sum_{(s,t)\in S_{xy}} \left(g(s,t) - \mu(x,y) \right)^2. \tag{2.4}$$

Since noise is pixel-independent, the values of $\overline{n}$ and σ_n^2 can be estimated from the image in a similar way, by considering a window which lies in a uniform region of the image, provided that the type of noise (additive vs. multiplicative) is known.

หาวิทยาลัยเทคโนโลยีสุร

The first two basic filters below perform some weighting between no filtering at all and mean filtering by setting

$$\tilde{f}(x,y) = \beta(x,y)g(x,y) + (1-\beta(x,y))\mu(x,y) \qquad (0 \leq \beta \leq 1). \tag{2.5}$$

Each filter computes the appropriate value of the weight $\beta = \beta(x,y)$ from the above local statistics differently. A value of β closed to 1 means that no filtering takes place, while a value closed to 0 means that mean filtering is taking place.

2.2.3 Lee Filter

The basic idea (Lee, 1980) is as follows: if the variance among the pixel values within the window is small, then this variance stems mainly from noise, and one can use mean filtering. On the other hand, a large variance is the result of edges within the window which must not be blurred out, and hence no filtering should take place. To be precise, the weight for additive, uncorrelated noise with zero mean ($\overline{n} = 0$) is

$$\beta = \beta(x,y) = \frac{\sigma_g^2}{\sigma_g^2 + \sigma_n^2}.$$

Note that since noise is of additive type in this model, then $\nu = \mu$ and $\sigma_f^2 = \sigma_g^2 - \sigma_n^2$.

On the other hand, when noise is multiplicative with mean $\overline{n} = 1$, one obtains that $\nu = \mu$ and

$$\sigma_f^2 = \frac{\sigma_g^2 - \sigma_n^2\mu^2}{(1+\sigma_n^2)}. \tag{2.6}$$

In this case, after linear approximation of Equation (2.2), as derived in (Lee, 1983),

$$g(x,y) \approx A(x,y)f(x,y) + B(x,y)u(x,y) + C(x,y),$$

noise can be considered of additive kind (no longer of zero mean or pixel independent), and the weight is now approximated slightly differently (Lee, 1980) by

$$\beta = \beta(x,y) = \frac{\sigma_f^2}{\sigma_f^2 + \mu^2\sigma_n^2}.$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

Since in practice, all the variables must be estimated from the noisy image, it may happen that the above fraction becomes negative; hence one must choose

$$\sigma_f^2 = \max\left(\frac{\sigma_g^2 - \sigma_n^2\mu^2}{1+\sigma_n^2}, 0\right).$$

2.2.4 Kuan Filter

Assuming multiplicative noise of mean $\overline{n} = 1$, noise is first converted to additive noise by writing

$$g(x,y) = f(x,y) + (u(x,y) - 1)f(x,y).$$

The second term on the right is now considered as signal dependent, additive noise (which differs from the usual assumption that noise be pixel-independent). From here Kuan et al. (1985) derive formula (2.5), where now

$$\beta = \beta(x,y) = \frac{\sigma_f^2}{\sigma_f^2 + (\mu^2 + \sigma_f^2)\sigma_n^2},$$

and σ_f^2 is still as in (2.6).

2.2.5 Lee-Sigma Filter

The Lee-Sigma filter (Lee, 1983) was designed specifically for speckle noise reduction in SAR images. It is essentially the mean filter, with one major modification: It is assumed that those pixels inside the window whose values lie within the range of two-standard deviations from the value of the center pixel do not represent speckle noise, and are therefore included in the averaging process. Pixels whose values lie outside of this range represent speckle noise and are excluded from averaging. Assuming multiplicative noise of mean $\overline{n} = 1$, one sets

$$Q_{xy} = \{\ (s,t) \in S_{xy} : |g(s,t) - g(x,y)| \le 2\sigma_n g(x,y)\ \}$$

and averages by

$$\tilde{f}(x,y) = \frac{1}{|Q_{xy}|} \sum_{(s,t)\in Q_{xy}} g(s,t).$$

However, some particular situations require separate treatment: when the center pixel (x,y) has intensity which is very different from all of the surrounding pixels – meaning it represents spot noise – then the cardinality of the set Q_{xy} will be very small, possibly one, so that no or little averaging takes place. In this case, in order to remove the noisy center pixel, one replaces the above set Q_{xy} with a very small window around (x,y) which only contains the direct neighbors of (x,y).

2.2.6 Frost Filter

This is another type of mean filter developed for image enhancement in SAR images, assuming multiplicative noise. The contribution to the averaging process of each pixel within the window is now position dependent; pixels closer to the center carry a larger weight. The model is

$$\tilde{f}(x,y) = K_1 \sum_{(s,t)\in S_{xy}} \gamma(s,t) g(s,t),$$

where the weight γ is given by

$$\gamma(s,t) = exp\left[-K \left(\frac{\sigma_g(x,y)}{\mu(x,y)} \right)^2 \big\| (s,t) - (x,y) \big\| \right].$$

Here, $\|\cdot\|$ denotes the Euclidean norm, so that $\big\|(s,t) - (x,y)\big\|$ represents the distance between pixel (s,t) and the center pixel (x,y). The constant K is a parameter influencing the filter characteristics, and K_1 is chosen so that mean image intensity remains unchanged,

$$K_1 = \left[\sum_{(s,t)\in S_{xy}} \gamma(s,t) \right]^{-1}.$$

In regions where the ratio σ_g/μ is large signifying edges in the image, $\gamma(s,t)$ will be close to zero at all except the central pixel, so that no filtering takes place. In

มหาวิทยาลัยเทคโนโลยีสุรนารี

regions where this ratio is small, $\gamma(s,t)$ will be close to 1, and averaging will take place.

2.3 Global Filtering Techniques

As for global filtering techniques, here one considers the image as a bounded function defined on the rectangle $I = [0,m]\times[0,n]$, the value of the function $f(x,y)$ at each point (x,y) denoting the image brightness at that point. Two methods have become popular : variational methods and thresholding methods.

2.3.1 Variational Methods

In the variational model popularized by the paper of Rudin, Osher and Fatemi (1992), the image denoising problem is formulated as an optimization problem: The denoised image $\tilde{f}$ is obtained as the minimizer of a functional

$$E(\tilde{f}) = \iint_I \left[|\nabla \tilde{f}| + \lambda J(\tilde{f},g) \right] d(x,y),$$

where $\nabla \tilde{f}$ denotes the gradient and $J(\tilde{f},g)$ is a data-fidelity term. The purpose of the first term involving the gradient is to obtain a smooth denoised image, while the data fidelity term measures how well the denoised image $\tilde{f}$ matches the unknown noise-free image f; it is derived from assumptions on the probability distribution of the noisy image by using Bayesian statistics. The parameter λ specifies how the two components of this functional are weighed against another. By the method of calculus of variations, the minimizer $\tilde{f}$ can be found numerically as the solution u of an Euler-Lagrange partial differential equation,

$$\frac{\partial}{\partial x}\left(\frac{u_x}{\sqrt{u_x^2+u_y^2}}\right) + \frac{\partial}{\partial y}\left(\frac{u_y}{\sqrt{u_x^2+u_y^2}}\right) - \lambda \frac{\partial}{\partial u} J(u,g) = 0,$$

where the outward normal vanishes on the boundary of I. The data fidelity term $J(\tilde{f},g)$ is chosen appropriate for the characteristics of the noise. In ultrasound

มหาวิทยาลัยเทคโนโลยีสุรนารี

imaging, the noisy image is commonly modeled to have the Rayleigh distribution (Burckhardt, 1978). Seekot (2008) has derived a data-fidelity term for the variational model corresponding to this distribution, giving the Euler-Lagrange equation

$$\frac{\partial}{\partial x}\left(\frac{u_x}{\sqrt{u_x^2+u_y^2}}\right)+\frac{\partial}{\partial y}\left(\frac{u_y}{\sqrt{u_x^2+u_y^2}}\right)+2\lambda\frac{g^2-u^2}{u^3}=0.$$

2.3.2 Thresholding Methods

Another popular way to reduce noise in an image is by thresholding. This concept will be explained at the end of the next chapter.

มหาวิทยาลัยเทคโนโลยีสุรนารี

CHAPTER III

WAVELETS AND WAVELET THRESHOLDING

In this chapter, we review the mathematical background from wavelets required. In particular, we introduce the Haar and the Daubechies wavelets db20. We also explain how wavelet techniques can be used in denoising or compression using thresholding.

3.1 Discrete Wavelets

The basic idea of one-dimensional wavelet theory is to decompose the Hilbert space $L^2(\mathbb{R})$ into an infinite direct sum $\{W_j\}_{j\in\mathbb{Z}}$ of subspaces

$$L^2(\mathbb{R}^n) = \bigoplus_{j=-\infty}^{\infty} W_j$$

of special form:

1. All the spaces W_j are isomorphic. In fact, for each j, the dilation operator $D : f(x) \mapsto \sqrt{2}f(2x)$ is a Hilbert space isomorphism of W_j onto W_{j+1}.

   Therefore, the j-fold composition D^j of this dilation operator, $D^j : f(x) \mapsto 2^{j/2}f(2^jx)$ is a Hilbert space isomorphism of W_0 onto W_j, for each j.

2. The space W_0 has an orthonormal basis consisting of integer translates of a single function ψ, that is,

$$\mathcal{B}_0 := \{\psi_{0,k} : \psi_{0,k}(x) = \psi(x-k), \quad k \in \mathbb{Z}\}$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

is an orthonormal basis of W_0. By this, the space W_0 is invariant under integer translations:

$$f(x) \in W_0 \Leftrightarrow f(x-k) \in W_0 \quad \text{for all } k \in \mathbb{Z}.$$

This function ψ is called the *mother wavelet.*

Then by 1., the j-fold composition D^j maps the basis $\mathcal{B}_0$ to a basis

$$\mathcal{B}_j := \left\{\psi_{j,k} : \psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k), \quad k \in \mathbb{Z}\right\}$$

of W_j, for each j. By this, the space W_j is invariant under integer dyadic translations:

$$f(x) \in W_j \Leftrightarrow f\left(x - \frac{k}{2^j}\right) \in W_j \quad \text{for all } k \in \mathbb{Z}.$$

The spaces W_j are called *detail spaces.* Thus,

$$\mathcal{B} = \bigcup_{j=-\infty}^{\infty} \mathcal{B}_j = \{\psi_{j,k} : j,k \in \mathbb{Z}\}$$

is an orthonormal bases of $L^2(\mathbb{R}^n)$, called a *wavelet basis.*

3. Thus, one can express an element $f \in L^2(\mathbb{R})$ in terms of these subspaces,

$$f = \sum_{j=-\infty}^{\infty} Q_j f$$

where Q_j denotes the projection of f onto the space W_j,

$$Q_j f = \sum_{k \in \mathbb{Z}} d_{j,k}\psi_{j,k}$$

and where

$$d_{j,k} = \langle f, \psi_{j,k}\rangle = 2^{j/2}\int_{\mathbb{R}} f(x)\overline{\psi\left(2^j x - k\right)}\, dx.$$

For fixed scale parameter j, the collection $\{d_{j,k} : k \in \mathbb{Z}\}$ is called the set of *wavelet coefficients of f at scale 2^j*, and the total collection $\{d_{j,k} : j,k \in \mathbb{Z}\}$ is called the *wavelet transform of of f* for the mother wavelet ψ.

By Parseval's identity, the doubly-indexed sequence $\left\{d_{j,k}\right\}_{j,k\in\mathbb{Z}}$ is an element of $\ell^2(\mathbb{Z}\times\mathbb{Z})$, so that the map

$$WT : f \in L^2(\mathbb{R}) \mapsto \left\{d_{j,k} :, k \in \mathbb{Z}\right\} \in \ell^2(\mathbb{Z}\times\mathbb{Z})$$

is a linear isomorphism, called the *discrete wavelet transform.* Then $f \in L^2(\mathbb{R})$ can be expressed in this bases as

$$f = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} d_{j,k}\psi_{j,k} \tag{3.1}$$

with convergence in the L^2-norm. This reconstruction is called the *inverse wavelet transform.*

4. In practice, one cannot work with infinitely many subspaces W_j. One therefore designates a smallest and largest scale level j_{min} and j_{max}, respectively, and works in the subspace

   $$\bigoplus_{j=j_{min}}^{j_{max}} W_j \quad \text{of} \quad L^2(\mathbb{R}). \tag{3.2}$$

   Now if in addition

   (a) the function f has compact support,

   (b) the mother wavelet ψ has compact support,

   then for each j, only finitely many coefficients $d_{j,k}$ are nonzero, and an element f in the space (3.2) can be expressed as a finite sum

   $$f = \sum_{j=j_{min}}^{j_{max}} \sum_{k=k_{min}(j)}^{k_{max}(j)} d_{j,k}\psi_{j,k},$$

   that is,

   $$f = \sum_{j=j_{min}}^{j_{max}} 2^{j/2} \sum_{k=k_{min}(j)}^{k_{max}(j)} d_{j,k}\psi\left(2^j x - k\right),$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

3.2 Wavelets from Multiresolution Analysis

Almost all wavelets, and in particular, all wavelets of compact support (Hernandez, 1996) can be obtained by a process called *multiresolution analysis* as introduced by Mallat (1989). When wavelets are constructed by such a multiresolution analysis, then (3.2) can be improved to working in the space

$$V_{j_{min}} \oplus \bigoplus_{j=j_{min}}^{j_{max}} W_j,$$

where $V_{j_{min}}$ is again a shift-invariant subspace of $L^2(\mathbb{R})$, and

$$V_{j_{min}} = \bigoplus_{j=-\infty}^{j_{min}-1} W_j.$$

Definition 3.1. (Multiresolution analysis)

A *multiresolution analysis* (MRA) on $L^2(\mathbb{R})$ is a sequence of closed subspaces $\{V_j\}_{j\in\mathbb{Z}}$ of $L^2(\mathbb{R})$ satisfying the following properties:

$(M1)$: $V_j \subseteq V_{j+1}$ for all $j \in \mathbb{Z}$ ("nested sequence"),

$(M2)$: $\bigcup_{j\in\mathbb{Z}} V_j$ is dense in $L^2(\mathbb{R})$,

$(M3)$: $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$,

$(M4)$: $f(x) \in V_0$ if and only if $f(2^j x) \in V_j$, for all j,

$(M5)$: there exists a function $\varphi(x) \in L^2(\mathbb{R})$, called the *scaling function*, such that the collection of integer translates $\{\varphi(x-k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis of V_0.

Let us briefly outline how wavelets can be obtained from a multiresolution analysis. By (M4), the dilation operator D is a linear isometry of V_j onto V_{j+1} for all j. Applying the dilation operator j-times, it follows that the collection

$$\{D^j \varphi_{0,k}\}_{k\in\mathbb{Z}} = \{\varphi_{j,k}\}_{k\in\mathbb{Z}} = \{2^{j/2}\varphi(2^j x - k)\}_{k\in\mathbb{Z}}$$

is an orthonormal basis of V_j, for all j. In particular, when $j = 1$ then

$$\{\varphi_{1,k}\}_{k\in\mathbb{Z}} = \{\sqrt{2}\varphi(2x-k)\}_{k\in\mathbb{Z}}$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

is an orthonormal basis of V_1. Now by (M1), $\varphi \in V_1$ as well, and thus it can be expressed in terms of this basis,

$$\varphi = \sum_{k\in\mathbb{Z}} g_k \varphi_{1,k} \tag{3.3}$$

where

$$g_k =< \varphi, \varphi_{1,k} >= \sqrt{2}\int_{-\infty}^{\infty} \varphi(x)\overline{\varphi(2x-k)}\,dx. \tag{3.4}$$

The sequence $\{g_k\}_{k\in\mathbb{Z}}$ is called the *scaling filter* and is an element of $\ell^2(\mathbb{Z})$, in fact by Parseval's identity,

$$\|\{g_k\}\|_{l^2(\mathbb{Z})} = \|\varphi\|_{L^2(\mathbb{R})} = 1. \tag{3.5}$$

It follows that the sequence $\{h_k\}_{k\in\mathbb{Z}}$ defined by

$$h_k = (-1)^k\overline{g_{1-k}}$$

is also an element of $\ell^2(\mathbb{Z})$ of norm one, so that the function ψ defined by

$$\psi = \sum_{k\in\mathbb{Z}} h_k \varphi_{(1,k)} \tag{3.6}$$

is an element of V_1. The sequence $\{h_k\}$ is called the *wavelet filter*.

Now express the integer translates of φ and ψ in terms of the basis vectors $\varphi_{1,k}$ of V_1. By (3.3),

$$\begin{aligned}\varphi(x-l) &= \sum_{k\in\mathbb{Z}} g_k\varphi_{1,k}(x-l) = \sum_{k\in\mathbb{Z}} g_k\sqrt{2}\varphi(2(x-l)-k)\\ &= \sum_{k\in\mathbb{Z}} g_k\sqrt{2}\varphi\left(2x-(k+2l)\right) = \sum_{k\in\mathbb{Z}} g_{k-2l}\varphi_{1,k}(x)\end{aligned}$$

and similarly by (3.6),

$$\psi(x-l) = \sum_{k\in\mathbb{Z}} h_{k-2l}\varphi_{1,k}(x).$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

It follows by Parseval's equality that

$$
\begin{aligned}
&<\varphi(x-m),\psi(x-l)>=\sum_{k\in\mathbb{Z}} g_{k-2m}\overline{h_{k-2l}}=\sum_{k\in\mathbb{Z}} g_{k-2m}(-1)^{k-2l}g_{1-k+2l}\\
&=\sum_{k\in\mathbb{Z}}(-1)^{k+m-l}g_{k+(l-m)}g_{1-k+(l-m)}\\
&=\sum_{k=1}^{\infty}(-1)^{k+m-l}g_{k+(l-m)}g_{1-k+(l-m)}+\sum_{k=0}^{\infty}(-1)^{-k+m-l}g_{-k+(l-m)}g_{1+k+(l-m)}\\
&=\sum_{k=1}^{\infty}(-1)^{k+m-l}g_{k+(l-m)}g_{1-k+(l-m)}+\sum_{k=1}^{\infty}(-1)^{1-k+m-l}g_{1-k+(l-m)}g_{k+(l-m)}\\
&=\sum_{k=1}^{\infty}\left[(-1)^{k}-(-1)^{-k}\right](-1)^{m-l}g_{k+(l-m)}g_{1-k+(l-m)}=0.
\end{aligned}
$$

Similar computations give

$$
\begin{aligned}
&<\psi(x-m),\psi(x-l)>=\sum_{k\in\mathbb{Z}} h_{k-2m}\overline{h_{k-2l}}\\
&=\sum_{k\in\mathbb{Z}}(-1)^{k-2m}\overline{g_{1-k+2m}}(-1)^{k-2l}g_{1-k+2l}=\sum_{k\in\mathbb{Z}} g_{1-k+2l}\overline{g_{1-k+2m}}\\
&=\sum_{k\in\mathbb{Z}} g_{k+2l}\overline{g_{k+2m}}=<\varphi(x+l),\varphi(x+m)>=\delta_{l,m}
\end{aligned}
$$

by (M5). This shows that the collection $\{\varphi_{0,k},\psi_{0,k}\}_{k\in\mathbb{Z}}$ is orthonormal in V_1. Thus, if we let W_0 denote the orthonormal complement of V_0 in V_1,

$$V_1 = V_0 \oplus W_0$$

then $\{\psi_{0,k}\}_{k\in\mathbb{Z}}$ will be an orthonormal set in W_0. One can show that this collection is total in W_0, that is, is a basis of W_0.

In general, for each j, let W_j denote the orthogonal complement of V_j in V_{j+1}, that is,

$$V_{j+1} = V_j \oplus W_j.$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

Applying induction, we have for each $n > j_o$,

$$\begin{aligned} V_n &= V_{n-1} \oplus W_{n-1} \\ &= V_{n-2} \oplus W_{n-2} \oplus W_{n-1} \\ &= V_{n-3} \oplus W_{n-3} \oplus W_{n-2} \oplus W_{n-1} \\ &\;\;\vdots \\ &= V_{j_o} \oplus W_{j_o} \oplus W_{j_o+1} \oplus \cdots \oplus W_{n-1}. \end{aligned} \tag{3.7}$$

One can show that (M1)–(M3) imply that for each $j_o \in \mathbb{Z}$,

$$L^2(\mathbb{R}) = V_{j_o} \oplus \bigoplus_{j=j_0}^{\infty} W_j$$

and also

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

Now since D^j maps W_0 isomorphically onto W_j, the collection $\{\psi_{j,k}\}_{k\in\mathbb{Z}} = \{D^j\psi_{0,k}\}_{k\in\mathbb{Z}}$ is an orthonormal basis for W_j, for each j. This shows that $\{\psi_{j,k} : j, k \in \mathbb{Z}\}$ is a basis of $L^2(\mathbb{R})$, and hence ψ is a mother wavelet.

Remark. Given smallest and largest scaling levels j_{min} and j_{max}, then (choosing $n = j_{max} + 1$, $j_o = j_{min}$ we obtain from (3.7) that

$$V_n = V_{j_{min}} \oplus \bigoplus_{j=j_{min}}^{j_{max}} W_j. \tag{3.8}$$

Since for each j, $\{\varphi_{j,k} : k \in \mathbb{Z}\}$ is an orthonormal basis of V_j, where

$$\varphi_{j,k}(x) = 2^{j/2}\varphi(2^j x - k),$$

then

$$\{\varphi_{j_{min},k} : k \in \mathbb{Z}\} \cup \{\psi_{j,k} : j_{min} \le j \le j_{max}\,,\ k \in \mathbb{Z}\}$$

is an orthonormal basis of $V_n = V_{j_{max}+1}$.

Given $f \in L^2(\mathbb{R})$, we first denote its orthogonal projection onto the space V_n by $\hat{f}$, so that

$$\hat{f} = \sum_{k=-\infty}^{\infty} c_{n,k}\varphi_{n,k}, \qquad c_{n,k} = \langle f, \varphi_{n,k}\rangle, \quad n = j_{max} + 1.$$

Next, the decomposition (3.8) gives

$$\hat{f} = \sum_{k=-\infty}^{\infty} c_{j_o,k}\varphi_{j_o,k} + \sum_{j=j_o}^{j_{max}} \sum_{k=-\infty}^{\infty} d_{j,k}\psi_{j,k},$$

where $j_o = j_{min}$ and the scaling coefficients $c_{j_o,k}$ and the wavelet coefficients $d_{j,k}$ are given by

$$c_{j_o,k} = \langle \hat{f}, \varphi_{j_o,k}\rangle = \langle f, \varphi_{j_o,k}\rangle \qquad \text{and} \qquad d_{j,k} = \langle \hat{f}, \psi_{j,k}\rangle = \langle f, \psi_{j,k}\rangle,$$

respectively. We observe that when f and the wavelets have compact support, then only finitely many of these coefficients are zero.

Remark. One can show that if the scaling function and the wavelet are also elements of $L^1(\mathbb{R})$ (Walnut, 2002), for example if φ has compact support, then

$$\left|\int_{-\infty}^{\infty} \varphi(x)\, dx\right| = 1 \qquad \text{and} \qquad \int_{-\infty}^{\infty} \psi(x)\, dx = 0. \tag{3.9}$$

Because of this, compactly supported mother wavelets have a graph that looks similar to an oscillation wave, thus the name ”wavelet”.

มหาวิทยาลัยเทคโนโลยีสุรนารี

3.2.1 Haar Wavelets

Haar wavelets are the simplest types of wavelets, and were known long before the development of wavelet theory.

Example: Let us explain this concept by a simple example. The Haar scaling function is the characteristic function of the unit interval,

$$\phi(x) = \varphi(x) = \begin{cases} 1 & \text{if } x \in [0, 1) \\ 0 & \text{else} \end{cases}$$

and the mother wavelet is

$$\psi(x) = \begin{cases} 1 & \text{if } x \in [0, 1/2) \\ -1 & \text{if } x \in [1/2, 1) \\ 0 & \text{else.} \end{cases}$$

We choose to work in the space V_4, and use the decomposition $V_4 = V_0 \oplus W_0 \oplus W_1 \oplus W_2 \oplus W_3$. Now a basis of V_4 is given by the set of functions $\{\varphi_{4,k}(x) = 2^{4/2}\varphi(2^4 x - k) : k \in \mathbb{Z}\}$, that is,

$$\varphi_{4,k}(x) = \begin{cases} 4 & \text{if } x \in \left[\frac{k}{16}, \frac{k+1}{16}\right) \\ 0 & \text{else.} \end{cases} \tag{3.10}$$

Similarly, bases of $V_0, W_0, \ldots, W_3$ are given by the functions

$$\varphi_{0,k}(x) = \begin{cases} 1 & \text{if } x \in [k, k+1) \\ 0 & \text{else} \end{cases} \quad \text{and} \quad \psi_{j,k}(x) = \begin{cases} 2^{j/2} & \text{if } x \in \left[\frac{k}{2^j}, \frac{k+1/2}{2^j}\right) \\ -2^{j/2} & \text{if } x \in \left[\frac{k+1/2}{2^j}, \frac{k+1}{2^j}\right) \\ 0 & \text{else,} \end{cases} \tag{3.11}$$

for $j = 0, 1, 2, 3$. By (3.10), elements of V_4 are piecewise constant on intervals of length $1/16$ so this is the best resolution which we can work with. The scaling function, mother wavelet and the first few wavelets $\psi_{j,k}$ are shown in Figure 3.1.

Now consider a function f supported on the interval $[0, 2^n]$. Then by (3.11), the only possibly nonzero scaling and wavelet coefficients are

$$c_{0,k} = \langle f, \varphi_{0,k} \rangle = \int_0^{2^n} f(x)\varphi_{0,k}(x)\,dx = \int_k^{k+1} f(x)\,dx, \qquad k = 0 \ldots 2^n - 1$$

and

$$\begin{aligned} d_{j,k} = \langle f, \psi_{j,k} \rangle &= \int_0^{2^n} f(x)\psi_{j,k}(x)\,dx \\ &= 2^{j/2} \int_{\frac{k}{2^j}}^{\frac{2k+1}{2^{j+1}}} f(x)\,dx - 2^{j/2} \int_{\frac{2k+1}{2^{j+1}}}^{\frac{k+1}{2^j}} f(x)\,dx, \qquad k = 0 \ldots 2^{n+j} - 1, \end{aligned}$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

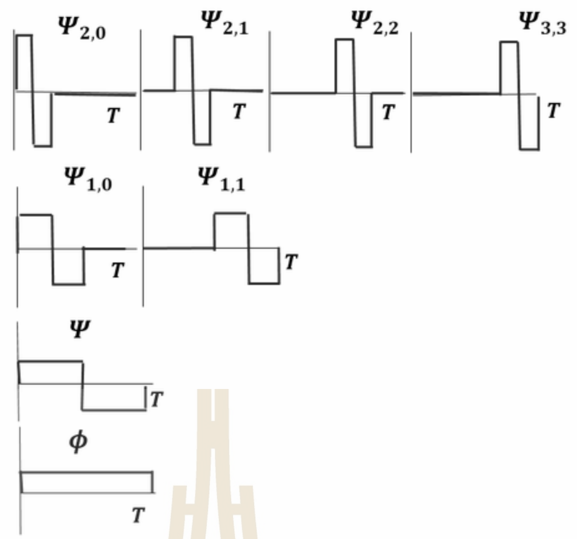

Figure 3.1 Bottom to top : the Haar scaling function ϕ, Haar mother wavelet ψ and wavelets $\psi_{j,k}$ at scales $j = 1, 2$ on the interval $[0, T] = [0, 1]$. (Source: https://www.researchgate.net)

so that

$$\begin{aligned}\hat{f} &= \sum_{k=0}^{2^n-1} c_{0,k}\varphi_{0,k} + \sum_{j=0}^{3}\sum_{k=0}^{2^{n+j}-1} d_{j,k}\psi_{j,k} \\ &= \sum_{k=0}^{2^n-1} c_{0,k}\varphi_{0,k} + \sum_{k=0}^{2^n-1} d_{0,k}\psi_{0,k} + \sum_{k=0}^{2^{n+1}-1} d_{1,k}\psi_{1,k} + \sum_{k=0}^{2^{n+2}-1} d_{2,k}\psi_{2,k} + \sum_{k=0}^{2^{n+3}-1} d_{3,k}\psi_{3,k}.\end{aligned}$$

There are altogether 2^{n+4} coefficients to be computed.

For ease of notation, let use set

$$\psi_{-1,k}(x) = \varphi_{0,k} \qquad \text{and} \qquad d_{-1,k} = c_{0,k}$$

so that the above becomes

$$\hat{f} = \sum_{j=-1}^{3}\sum_{k=0}^{N_j} d_{j,k}\psi_{j,k} \tag{3.12}$$

where $N_j = 2^n - 1$ when $j = 0, -1$ and $N_j = 2^{n+j} - 1$ else.

3.2.2 The Daubechies Wavelets

The Daubechies wavelets (Daubechies, 1988) are the most commonly used wavelets. This is a family of wavelets with compact support, labeled $db\{2N\}$, N a positive integer. The scaling function of a $db\{2N\}$-wavelet is supported on the interval $[0, N-1]$, and the scaling filter has length N. The corresponding wavelets have vanishing p-th moments for $0 \leq p \leq N-1$. Table 3.1 shows the scaling coefficients of the $db2$–$db10$ wavelets. There is no closed form known for these wavelets, although their values at dyadic rationals can be computed by an appropriate algorithm.

Table 3.1 The nonzero scaling coefficients for the $db2$ - $db10$ wavelets (Source: Daubechies (1992)).

	db2	*db4*	*db6*	*db8*	*db10*
a_0	1	0.6830127	0.4704672	0.3258034	0.2264190
a_1	1	1.1830127	1.1411169	1.0109457	0.8539435
a_2		0.3169873	0.6503650	0.8922014	1.0243269
a_3		−0.1830127	−0.1909344	−0.0395750	0.1957670
a_4			−0.1208322	−0.2645072	−0.3426567
a_5			0.0498175	0.0436163	−0.0456011
a_6				0.0465036	0.1097027
a_7				−0.0149870	−0.0088268
a_8					−0.0177919
a_9					0.0047174

มหาวิทยาลัยเทคโนโลยีสุรนารี

Figure 3.2 shows the graphs of various Daubechies scaling functions. They become smoother with increasing N, in fact, they are C^1 functions for $N \geq 3$.

In this thesis, Haar wavelets and Daubechies wavelets db20 are used. The graphs of the db20 scaling function and mother wavelet are shown in figures 3.3 and 3.4, respectively. There is a related class of wavelets called *Symlets* whose graphs show better symmetry than the Daubechies wavelets. In our experiments, the db20 wavelets performed marginally better than the Sym20 wavelets, therefore

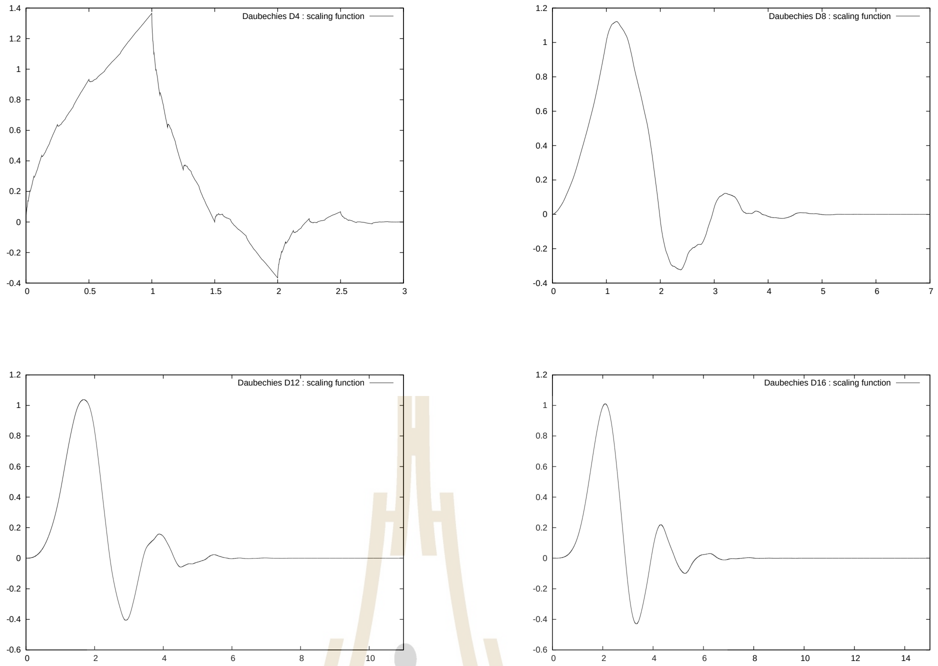

Figure 3.2 The Daubechies scaling functions $D4$, $D8$, $D12$ and $D16$.

only results for the Daubechies wavelets are presented.

3.3 Two-dimensional Wavelets

Since images are two-dimensional objects, one also needs a two-dimensional wavelet transform. One way to deal with a two-dimensional domain is to compute two one-dimensional wavelet transforms separately, first in y-direction and then in x-direction. This is the way chosen in this thesis. We explain the idea by using the above example of the Haar wavelet.

Consider a noisy image $g(x, y)$ on the square $I_2 = [0, 2^n] \times [0, 2^n]$, and assume that the best resolution possible is $\frac{1}{16}$ length units. The image pixels will be spaced accordingly in each direction by this distance, and this resolution is achieved with scaling level $j_{max} = 3$; i.e. working in the space V_4 in each direction.

มหาวิทยาลัยเทคโนโลยีสุรนารี

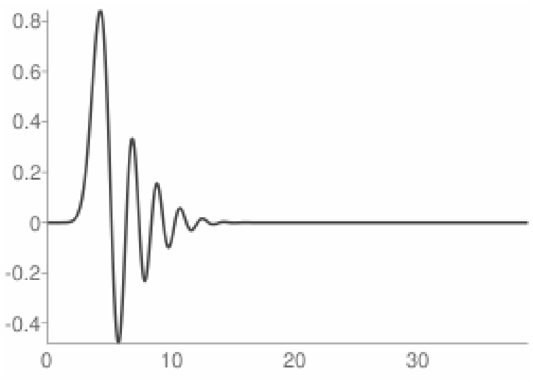

Figure 3.3 The Daubechies scaling function db20.

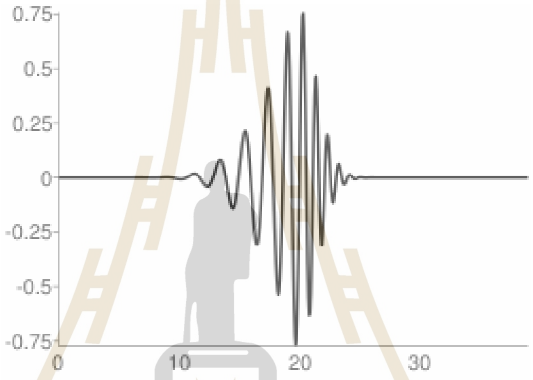

Figure 3.4 The Daubechies mother wavelet db20.

Taking the one-dimensional wavelet transform in y-direction gives by (3.12)

$$g(x,y) = \sum_{l=-1}^{3}\sum_{m=0}^{N_l} d_{l,m}(x)\psi_{l,m}(y), \qquad d_{l,m}(x) = \int_0^{2^n} g(x,y)\psi_{l,m}(y)\,dy,$$

the coefficients $d_{l,m}(x)$ still depending on x. Next taking the wavelet transform of each coefficient function $d_{l,m}(x)$ gives

$$d_{l,m}(x) = \sum_{j=-1}^{3}\sum_{k=0}^{N_j} d_{l,m}^{j,k}\psi_{j,k}(x), \qquad d_{l,m}^{j,k} = \int_0^{2^n} d_{l,m}(x)\psi_{j,k}(x)\,dx.$$

Combining both,

$$g(x,y) = \sum_{j=-1}^{3}\sum_{l=-1}^{3}\sum_{k=0}^{N_j}\sum_{m=0}^{N_l} d_{l,m}^{j,k}\psi_{j,k}(x)\psi_{l,m}(y), \tag{3.13}$$

$$d_{l,m}^{j,k} = \int_0^{2^n} d_{l,m}(x)\psi_{j,k}(x)\,dx \;=\; \int_0^{2^n}\int_0^{2^n} g(x,y)\psi_{j,k}(x)\psi_{l,m}(y)\,dx\,dy.$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

(To be precise, $g(x,y)$ in (3.13) above denotes the projection of $g(x,y)$ onto the subspace of $L^2(I_2)$ spanned by the orthonormal family $\{\psi_{j,k}(x)\psi_{l,m}(y) : -1 \leq j,l \leq 3, k = 0 \ldots N_j,\ m = 0 \ldots N_l\}$.)

3.4 Thresholding Methods

Thresholding is a common technique used in filtering or in data compression. For simplicity, we explain this idea with the Fourier transform first.

Let $f(x)$ be a continuous real-valued function on $[-\pi,\pi]$. Then it can be expressed as a Fourier series,

$$f(x) = \frac{a_o}{2} + \sum_{k=1}^{\infty} \left[a_k \cos(kx) + b_k \sin(kx)\right].$$

where

$$a_k = \langle f(x), \cos(kx) \rangle = \int_{-\pi}^{\pi} f(x) \cos(kx)\, dx \quad (k = 0, 1, 2, \ldots).$$

$$b_k = \langle f(x), \sin(kx) \rangle = \int_{-\pi}^{\pi} f(x) \sin(kx)\, dx \quad (k = 1, 2, 3, \ldots).$$

The pair of coefficients (a_k, b_k) expresses the contents of frequency $\frac{k}{2\pi}$ in the "signal" $f(x)$. In practice, there is an upper limit to the frequencies considered, so

$$f(x) \approx \frac{a_o}{2} + \sum_{k=1}^{N} \left[a_k \cos(kx) + b_k \sin(kx)\right]$$

for some integer N.

The idea in compression is that small absolute values of the coefficients a_k and b_k do not contribute much to the signal and may be discarded. In denoising, one assumes that frequencies with small absolute values of these coefficients represent (additive) noise and should be removed. This is the idea of thresholding: all coefficients whose absolute values are below some threshold ϵ should be changed to zero.

มหาวิทยาลัยเทคโนโลยีสุรนารี

One distinguishes between *hard thresholding* and *soft thresholding*. In general, suppose a function $f(x)$ is expressed in terms of a basis,

$$g = \sum_k c_k \varphi_k.$$

First fix a threshold value ϵ. In *hard thresholding*, one sets all coefficients whose values fall below this threshold to zero,

$$\tilde{c}_k = \begin{cases} 0 & \text{if } |c_k| < \epsilon \\ c_k & \text{else} \end{cases}$$

which means that the modified "signal" is

$$\tilde{f} = \sum_k \tilde{c}_k \varphi_k = \sum_{\substack{k \\ |c_k| \geq \epsilon}} c_k \varphi_k.$$

Because the map $c_k \mapsto \tilde{c}_k$ is discontinuous, there may be artifacts in the modified signal. Thus in soft thresholding one makes this function continuous, for example by setting

$$\tilde{c}_k = s\big(|c_k|\big)c_k \qquad \text{where} \qquad s(t) = \max\left(0, 1 - \frac{\epsilon}{t}\right).$$

Then $c_k \mapsto \tilde{c}_k$ is a continuous function.

In this thesis, thresholding is applied to the wavelet coefficients of an image. For example, when $g(x,y)$ represents the brightness of an image, and we apply the 2-dimensional Haar wavelet transform as in (3.13),

$$g(x,y) = \sum_{j=-1}^{3} \sum_{l=-1}^{3} \sum_{k=0}^{N_j} \sum_{m=0}^{N_l} d_{l,m}^{j,k} \psi_{j,k}(x) \psi_{l,m}(y)$$

then we may apply thresholding to some or all of the wavelet coefficients $d_{l,m}^{j,k}$. Results of this will be presented in Chapter V.

CHAPTER IV

IMAGE QUALITY MEASURES

In this chapter, we first explain how the sample image was generated, and review the image quality metrics commonly used in this thesis to measure the quality of an image.

4.1 Noisy Image Generation

Noise present in coherent imaging, such as laser optics and ultrasound, is mostly modeled to be Rayleigh distributed. The underlying model assumes that waves which are reflected from a surface, due to slight irregularities on the surface, are composed of a large number of reflected waves which possess small random amplitudes, and also uniformly distributed random phase shifts. Because of these phase shifts, the waves with different phases may amplify or cancel out another at the detector, leading to a noisy image. When the detector measures the wave amplitude, then the image pixels will be modeled to be Rayleigh distributed random variables (Burckhardt, 1978 and Goodman, 1985). On the other hand, when the detector measure the wave power, then the image pixels will be modeled to be exponentially distributed random variables (Aubert and Aujol, 2008).

In the case of Rayleigh distributed noise, the parameter σ of the distribution is proportional to the signal amplitude, or equivalently, pixel brightness. Thus, the probability density function is

$$p(r) = \frac{r}{\sigma^2} e^{-r^2/2\sigma^2} \qquad (r \geq 0)$$

มหาวิทยาลัยเทคโนโลยีสุรนารี

where

$$\sigma = \frac{u}{\sqrt{2}},$$

u denoting the noise-free amplitude of the pixel. This type of noise is usually considered as multiplicative noise.

It is well known and can easily be shown that $p(r)$ has mean $\sigma\sqrt{\pi/2}$ and second moment $2\sigma^2$. Thus, variance is proportional to the square of the mean.

It should be made clear here that “Rayleigh noise” does not mean that Rayleigh-distributed noise is added to the image. Instead, it means that each pixel of an image is a random variable that is Rayleigh distributed; the parameter σ of this distribution depending on the brightness of the pixel in the noise-free image. Furthermore, it is assumed that the brightness intensities of different pixels are independent random variables.

Because of the difficulty of obtaining good quality, raw ultrasound images, for the purpose of evaluating the noise reduction techniques, a synthetic, noisy version of the well-known Lena image was produced, creating Rayleigh-distributed noise with the Octave program according to the above formula (See the Appendix). The original, noise-free Lena image is shown in Figure 4.1, while the noisy image can be seen in Figure 4.2.

มหาวิทยาลัยเทคโนโลยีสุรนารี

4.2 Image Quality Indices

There are many ways to measure image quality. Naturally, the best way is evaluation by the human eye, because in the end it is humans who make use of and evaluate image content. However, this type of evaluation is subjective and cannot be automated, therefore a number of metrics have been developed for objective evaluation.

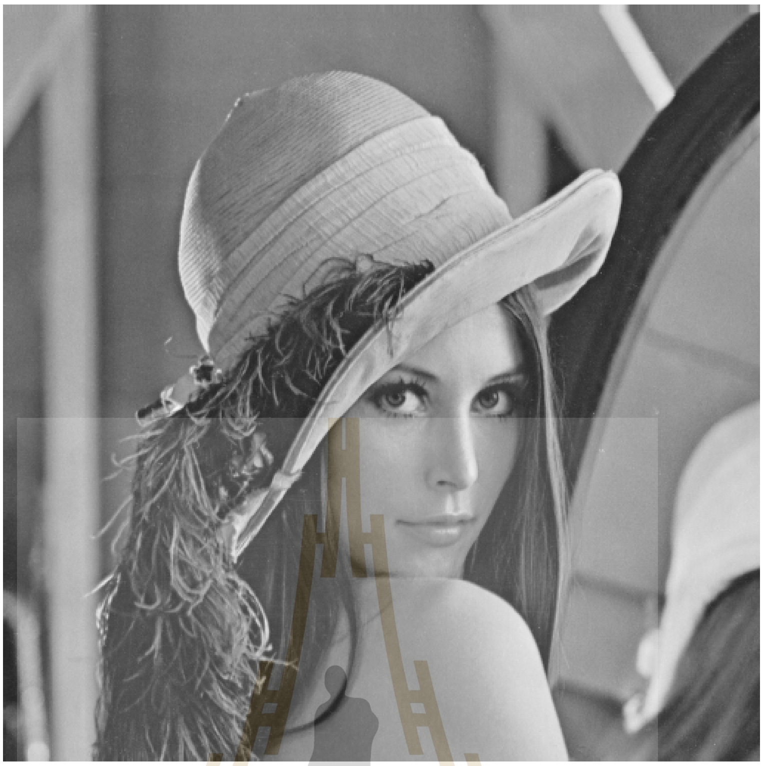

Figure 4.1 The noise-free Lena image

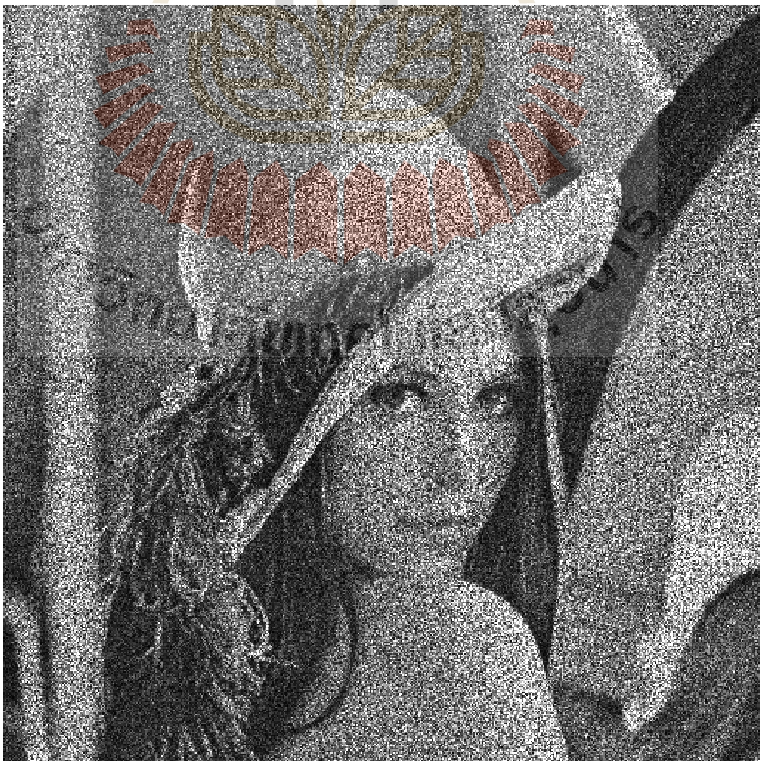

Figure 4.2 The Lena image with Rayleigh noise

We will first explain their mathematical meanings, and then the practical implementation. Throughout, we consider the image a region I_2 in the plane which will be given the Lebesgue measure normalized so that the total measure of I_2 equals one.

1. **Average Difference (AD)** or **Mean Absolute Error (MAE).** This is the the L^1-distance between the original noisy image f and the denoised image $\tilde{f}$,

   $$AD = \|\tilde{f} - f\|_1. \tag{4.1}$$

   All areas in the image region I_2 carry an identical weight.

   If the images are pixellated, of size $m \times n$, then

   $$AD = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left| \tilde{f}(i,j) - f(i,j) \right|.$$

   The division by mn corresponds to normalization of the Lebesgue measure in (4.1) above. Thus, the AD-value measures the average pixel error. Naturally, the smaller this value the better the two images match; a value of $AD = 0$ signifies a perfect match.

2. **Root Mean Square Error (RMSE).** This is the L^2-distance between the two images,

   $$RMSE = \|\tilde{f} - f\|_2.$$

   Areas in the image where the two functions differ by large amounts carry more weight than areas of small differences.

   In the pixellated case,

   $$RMSE = \sqrt{\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\tilde{f}(i,j) - f(i,j) \right)^2}.$$

   Alternatively, the **Mean Square Error (MSE)** is often used, $MSE = RMSE^2$. Again, a value of $RMSE = 0$ signifies a perfect match.

มหาวิทยาลัยเทคโนโลยีสุรนารี

3. **The Correlation Coefficient (Corr).** The correlation coefficient between two random variables is essentially an L^2-inner product of normalized vectors, measuring how similar both are; therefore

$$Corr = \frac{\langle \tilde{f} - \mu_{\tilde{f}}, f - \mu_f \rangle}{\left\| \tilde{f} - \mu_{\tilde{f}} \right\|_2 \left\| f - \mu_f \right\|_2},$$

where $\mu_f$and $\mu_{\tilde{f}}$ denote the means of f and $\tilde{f}$, respectively, $\mu_f = \int_{I_2} f(x, y)\, dA$.

In the pixellated case,

$$Corr = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \left(\tilde{f}(i,j) - \mu_{\tilde{f}} \right) \left(f(i,j) - \mu_f \right)}{\sqrt{\left(\sum_{i=1}^{m} \sum_{j=1}^{n} (\tilde{f}(i,j) - \mu_{\tilde{f}})^2 \right) \left(\sum_{i=1}^{m} \sum_{j=1}^{n} (f(i,j) - \mu_f)^2 \right)}}$$

where

$$\mu_f = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} f(i,j).$$

The value of this index ranges betwwen 0 and 1, with a value of $Corr = 1$ signifying a perfect match of the two images.

4. **The Signal to Noise Ratio (SNR).** There are different definitions of the SNR. The common concept used in image processing is the ratio of mean over standard deviation of a uniform section of the image,

$$SNR = \frac{\mu_{\tilde{f}}}{\sigma_{\tilde{f}}}.$$

This index is special in that it does not compare two images, but tries to extract the image quality information from the image to be evaluated only. This is the reason that one needs to consider a relatively uniform section of the image for its evaluation. Obviously, the larger this index the less noise is present in the image; there is a common criterion (Rose, 1973) which states that a SNR of at least 5 is needed to be able to distinguish image features at 100% certainty.

มหาวิทยาลัยเทคโนโลยีสุรนารี

In the pixellated case,

$$SNR = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\tilde{f}(i,j)}{\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\big(\tilde{f}(i,j)-f(i,j)\big)^2}}\,.$$

For additive, pixel-independent noise this is a good measure, but less so for multiplicative noise such as Rayleigh noise.

5. **Image Quality Index (IQI).** This is a relatively new index: it was first introduced by Wang and Bowik (2002). It is an index comparing two images and is composed of three equally weighted component indices: correlation, luminance and contrast, and it is therefore a more broadly based measure of image quality,

$$IQI = \underbrace{\frac{\sigma_{f\tilde{f}}}{\sigma_f\sigma_{\tilde{f}}}}_{\text{correlation}} \cdot \underbrace{\frac{2\mu_f\mu_{\tilde{f}}}{\mu_f^2+\mu_{\tilde{f}}^2}}_{\text{luminance}} \cdot \underbrace{\frac{2\sigma_f\sigma_{\tilde{f}}}{\sigma_f^2+\sigma_{\tilde{f}}^2}}_{\text{contrast}}$$

where as usual, μ_f and $\mu_{\tilde{f}}$ denote the means of the two images to be compared, while σ_f^2 and $\sigma_{\tilde{f}}^2$ denote their variances. For a pixellated image,

$$\mu_f = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} f(i,j), \qquad \mu_{\tilde{f}} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\tilde{f}(i,j),$$

$$\sigma_{f\tilde{f}} = \frac{1}{mn-1}\sum_{i=1}^{m}\sum_{j=1}^{n}(f(i,j)-\mu_f)(\tilde{f}(i,j)-\mu_{\tilde{f}}),$$

$$\sigma_f^2 = \frac{1}{mn-1}\sum_{i=1}^{m}\sum_{j=1}^{n}(f(i,j)-\mu_f)^2,$$

$$\sigma_{\tilde{f}}^2 = \frac{1}{mn-1}\sum_{i=1}^{m}\sum_{j=1}^{n}(\tilde{f}(i,j)-\mu_{\tilde{f}})^2.$$

The value of this index ranges between 0 and 1, with a value of $IQI = 1$ signifying a perfect match of the two images.

มหาวิทยาลัยเทคโนโลยีสุร

CHAPTER V

RESULTS

In this chapter, the results of the experiments are presented. We begin by summarizing the collection of results from the wavelet transform. We then compare them with other noise reduction methods, both with and without the logarithmic transform. But first a note on what wavelet coefficients were chosen for thresholding.

5.1 Thresholding of Wavelet Coefficients

The Lena image used in this work is the standard, grayscale Lena image freely available at many internet sites. It is a 512×512 bitmap image with grayscale values in the 0-255 integer range, and can thus be presented in a computer program as an array of size 512×512. Internally in the computer program, integer values are converted to floating point values.

มหาวิทยาลัยเทคโนโลยีสุรนารี

The largest scaling level which makes sense for an image of that size with the Haar wavelets is $j_{max} = 8$. In order to compare the Daubechies wavelets with the Haar wavelet, this largest scaling level was also chosen for the Daubechies wavelets. Experiments have shown that even in case of the Daubechies wavelets, this scaling level of $j_{max} = 8$ produces noticeably better results than lower maximun scaling levels.

Thus, an image is decomposed as a wavelet sum

$$\sum_{j=-1}^{8}\sum_{l=-1}^{8}\sum_{k=0}^{N_j}\sum_{m=0}^{N_l} d_{l,m}^{j,k}\psi_{j,k}(x)\psi_{l,m}(y)$$

as described in Chapter III. The wavelet transform of an image is thus also represented in form of a square matrix, and moving along the rows and columns increases the scale level in x and y directions, respectively. With increasing scaling level the number of wavelet coefficients doubles, so most of the entries in the matrix correspond to coefficients belonging to large scale levels j.

For example, Figure 5.1 shows the matrix of wavelet coeffcients divided into four equal parts. The coefficients in the grayed-out bottom right quarter belong to scaling levels j_{max} in both directions. In “1/4-thresholding”, thresholding is applied to these coeffcients at highest scale $j = l = j_{max} = 8$ only.

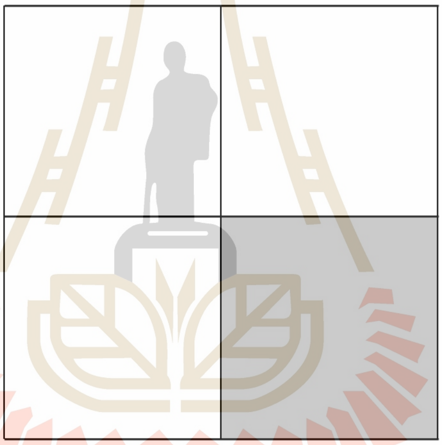

Figure 5.1 1/4 thresholding

มหาวิทยาลัยเทคโนโลยีสุรนารี

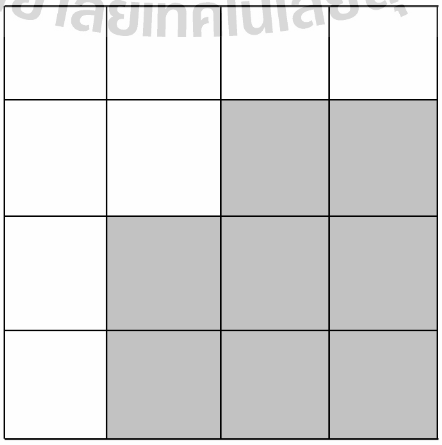

Figure 5.2 1/2 thresholding

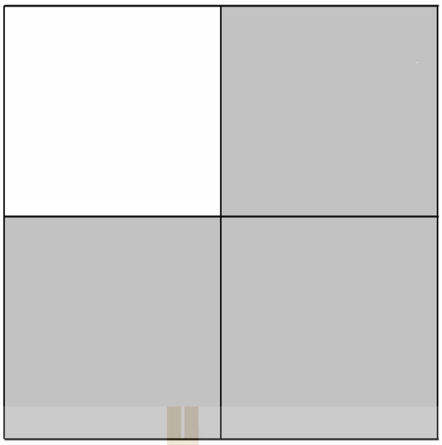

Figure 5.3 3/4 thresholding

Next is "1/2 thresholding" as shown in Figure 5.2. Here, already half of the coeffcients undergo thresholding, which are the coefficients $d_{l,m}^{j,k}$ where $(j, l) \in \{(7, 8), (8, 7), (8, 8)\}$.

In "3/4 thresholding" as shown in Figure 5.3, 75% of the coeffcients undergo thresholding, namely those where at least one of the scale levels has largest value: $j = 8$ or $l = 8$.

In "15/16 thresholding" as shown in Figure 5.4, all coefficients that have at least one scale level, j or l in the range $\{7, 8\}$ experience thresholding.

Finally, there is "63/64 thresholding" of Figure 5.5, all coefficients that have at least one scale level, j or l in the range $\{6, 7, 8\}$ experience thresholding.

"Normal thresholding" finally means that all wavelet coeffcients undergo thresholding.

The best threshold value of $\epsilon = 0.0007$ in the case of Haar wavelets, and $\epsilon = 0.15$ in the case of Daubechies wavelets db20 was obtained by trial-and-error.

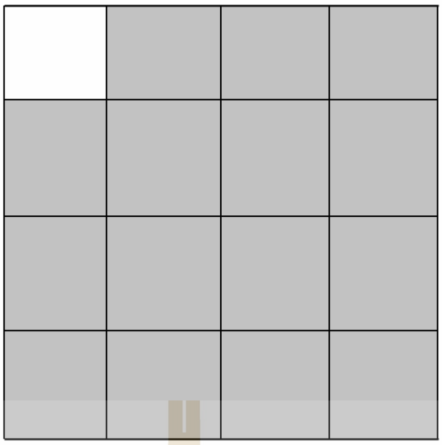

Figure 5.4 15/16 thresholding

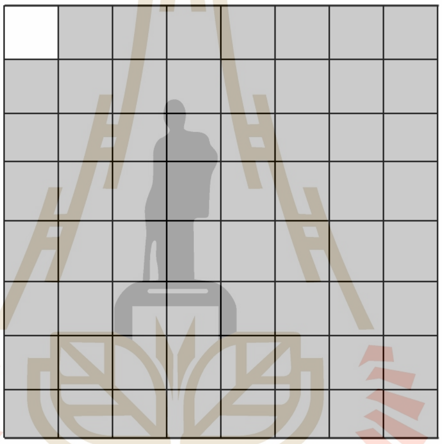

Figure 5.5 63/64 thresholding

มหาวิทยาลัยเทคโนโลยีสุรนารี

5.2 Results for Wavelet Methods

In this section, the results for the wavelet method are presented. Tables 5.1 and 5.2 show several image quality indices, for images without and with the log transform applied, respectively. Images after denoising – for the case where the logarithmic transform was not applied – are presented in Figures 5.20 through 5.15. The images are sorted in increasing order as to the number of wavelet coefficients to which thresholding is applied, beginning with Haar wavelets.

1. All quality indices are best in case of the 63/64 soft thresholding. There is, however, one exception: in case of the Haar wavelet without log transform,

15/16 thresholding gives a marginally better IQI index.

2. Soft-thresholding outperforms hard-thresholding by a wide margin in practically all indices. Visual inspection of the images shows that soft-thresholding is also much better in removing noise.

3. By all indices, except the IQI index, when the log transform is not applied in case of 63/64 soft thresholding, the Haar wavelets appear marginally better than the Daubechies wavelets. However, inspection of the images shows that there is some pixellation in case of the Haar wavelet.

4. Haar wavelets appear to have a much better SNR ratio than Daubechies wavelets. One should note, however, that the SNR ratio is derived from evaluating a uniform area of the original image, and thus neglects effects such as edges in the image.

5. Applying the log transform has different effects, depending on the thresholding method.

   For hard thresholding, applying the log transform improved all metrics. This is more pronounced for the Haar wavelets than the Daubechies wavelets.

   For soft thresholding, the indices measuring differences (MSE, RMSE, AD) show better results without application of the log transform. On the other hand, the indices that measure how two images match up (Correlation Coefficient, IQI) perform better when the log transfom is applied.

   Figures 5.22 and 5.23 give a visual comparison between log and non-log applications, for the best performing Haar wavelets (63/64 soft thresholding). Figures 5.24 and 5.25 do the same for the best performing Daubechies db20 wavelets

Table 5.1 **Table 5.1** Comparison of wavelet noise reduction methods without log transform applied

Denoising Method (no log transform) $\epsilon = 0.0007$ (Haar) $\epsilon = 0.15$ (db20)	MSE	RMSE	AD	Correlation Coeff.	IQI denoised vs.original	IQI denoised vs.noisy	IQI noisy vs.original	SNR
Haar (hard) $j_{max} = 8, m = 9$	0.031577	0.17770	0.13048	0.67479	0.66324	0.80316	0.54095	7.6326
Haar (soft) $j_{max} = 8, m = 9$	0.012987	0.11396	0.087553	0.88627	0.84310	0.57576	0.54095	35.745
Haar (1/2 hard) $j_{max} = 8, m = 9$	0.041083	0.20269	0.15484	0.62887	0.60504	0.91159	0.54095	5.542
Haar (1/2 soft) $j_{max} = 8, m = 9$	0.030464	0.17454	0.13408	0.69501	0.68582	0.84499	0.54095	7.215
Haar (1/4 hard) $j_{max} = 8, m = 9$	0.046742	0.21620	0.16802	0.60137	0.57164	0.95768	0.54095	5.0354
Haar (1/4 soft)) $j_{max} = 8, m = 9$	0.04142	0.20352	0.15806	0.62786	0.60567	0.92904	0.54095	5.6317
Haar (3/4 hard) $j_{max} = 8, m = 9$	0.035556	0.18856	0.14182	0.65859	0.64072	0.85999	0.54095	6.3944
Haar (3/4 soft) $j_{max} = 8, m = 9$	0.01977	0.14060	0.10767	0.78732	0.78496	0.74136	0.54095	11.717
Haar (15/16 hard) $j_{max} = 8, m = 9$	0.031966	0.17879	0.13182	0.67762	0.66427	0.81767	0.54095	7.2803
Haar (15/16 soft) $j_{max} = 8, m = 9$	0.012462	0.11163	0.085326	0.88115	0.86906	0.64341	0.54095	22.857
Haar (63/64 hard) $j_{max} = 8, m = 9$	0.031511	0.17751	0.13038	0.67719	0.66521	0.80619	0.54095	7.5428
Haar (63/64 soft) $j_{max} = 8, m = 9$	0.011547	0.10746	0.082532	0.89895	0.87727	0.60857	0.54095	31.212
db20 (hard) $j_{max} = 8, m = 9$	0.049182	0.22177	0.17304	0.58875	0.55651	0.97473	0.54095	4.8001
db20 (soft) $j_{max} = 8, m = 9$	0.022616	0.15039	0.11279	0.74486	0.74144	0.87093	0.54095	10.326
db20 (63/64 hard) $j_{max} = 8, m = 9$	0.029491	0.17173	0.13218	0.69174	0.68907	0.77791	0.54095	7.7205
db20 (63/64 soft) $j_{max} = 8, m = 9$	0.011828	0.10875	0.084473	0.89219	0.87812	0.60878	0.54095	26.05

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 5.2 Comparison of wavelet noise reduction methods with log transform applied

Denoising Method (with log transform) $\epsilon = 0.0007$ (Haar) $\epsilon = 0.15$ (db20)	MSE	RMSE	AD	Correlation Coeff.	IQI denoised vs.original	IQI denoised vs.noisy	IQI noisy vs.original	SNR
Haar (hard) $j_{max} = 8, m = 9$	0.019560	0.13986	0.10576	0.80094	0.7884	0.59643	0.54095	15.227
Haar (soft) $j_{max} = 8, m = 9$	0.016094	0.12686	0.10285	0.91136	0.81714	0.43350	0.54095	28.583
Haar (1/2 hard) $j_{max} = 8, m = 9$	0.034219	0.18498	0.14153	0.67031	0.65630	0.83470	0.54095	6.4703
Haar (1/2 soft) $j_{max} = 8, m = 9$	0.030828	0.17558	0.13547	0.69410	0.68487	0.80358	0.54095	6.9958
Haar (1/4 hard) $j_{max} = 8, m = 9$	0.043294	0.20807	0.16156	0.61772	0.59306	0.92278	0.54095	5.3129
Haar (1/4 soft) $j_{max} = 8, m = 9$	0.041663	0.20412	0.15872	0.62575	0.60376	0.91009	0.54095	5.5059
Haar (3/4 hard) $j_{max} = 8, m = 9$	0.025530	0.15978	0.12062	0.73858	0.73131	0.72977	0.54095	8.9230
Haar (3/4 soft) $j_{max} = 8, m = 9$	0.020403	0.14284	0.11017	0.79410	0.78775	0.67040	0.54095	10.858
Haar (15/16 hard) $j_{max} = 8, m = 9$	0.019925	0.14116	0.10579	0.79864	0.78907	0.63471	0.54095	12.823
Haar (15/16 soft) $j_{max} = 8, m = 9$	0.013576	0.11652	0.092276	0.90392	0.87474	0.54084	0.54095	19.891
Haar (63/64 hard) $j_{max} = 8, m = 9$	0.019290	0.13889	0.10491	0.80576	0.79369	0.60550	0.54095	14.475
Haar (63/64 soft) $j_{max} = 8, m = 9$	0.013156	0.11470	0.09331	0.92547	0.87800	0.49148	0.54095	26.077
db20 (hard) $j_{max} = 8, m = 9$	0.044589	0.21116	0.16269	0.60897	0.58211	0.93282	0.54095	5.1122
db20 (soft) $j_{max} = 8, m = 9$	0.018939	0.13762	0.10359	0.81195	0.79347	0.75111	0.54095	12.772
db20 (63/64 hard) $j_{max} = 8, m = 9$	0.028615	0.16916	0.12982	0.70333	0.69906	0.74540	0.54095	7.6180
db20 (63/64 soft) $j_{max} = 8, m = 9$	0.012829	0.11327	0.091726	0.92311	0.88641	0.53275	0.54095	23.112

มหาวิทยาลัยเทคโนโลยีสุรนารี

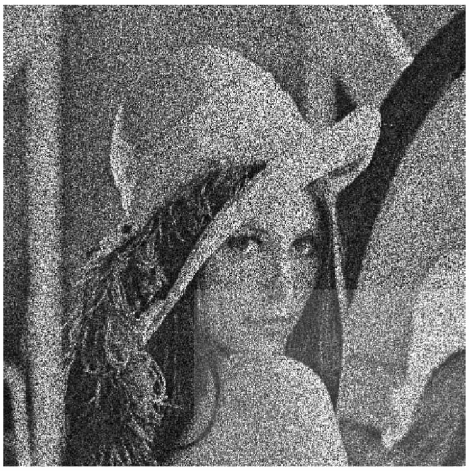

Figure 5.6 Haar, 1/4 hard thresholding

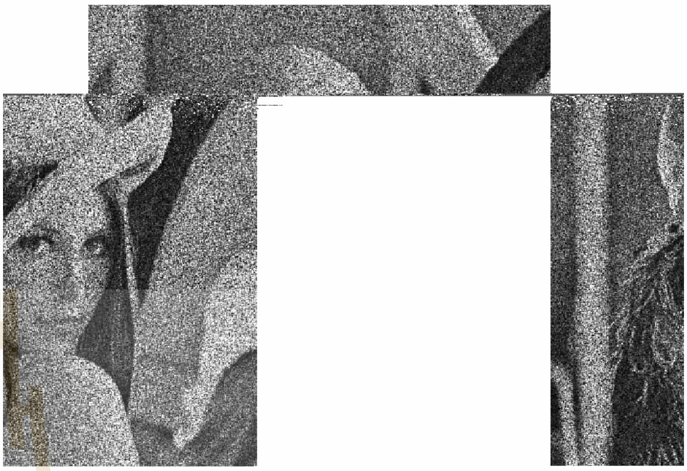

Figure 5.7 Haar, 1/4 soft thresholding

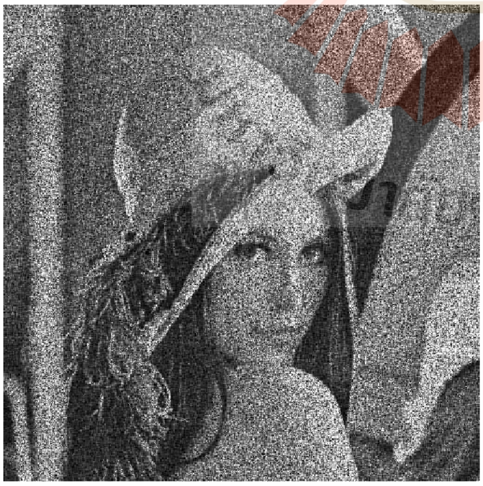

Figure 5.8 Haar, 1/2 hard thresholding

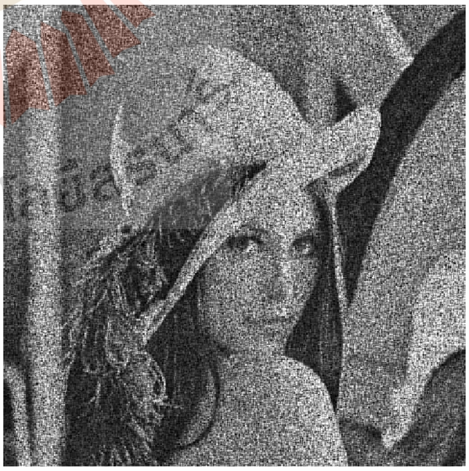

Figure 5.9 Haar, 1/2 soft thresholding

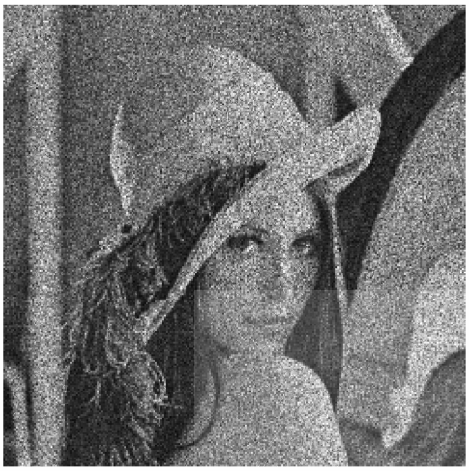

Figure 5.10 Haar, 3/4 hard thresholding

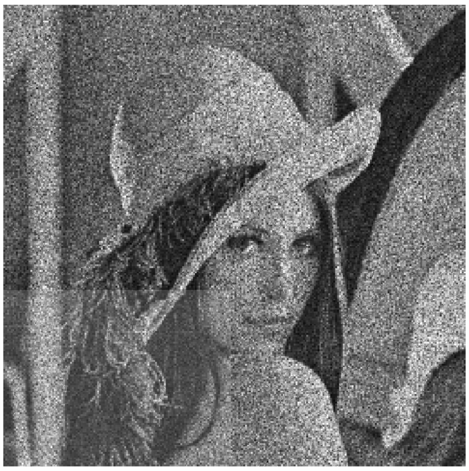

Figure 5.11 Haar, 3/4 soft thresholding

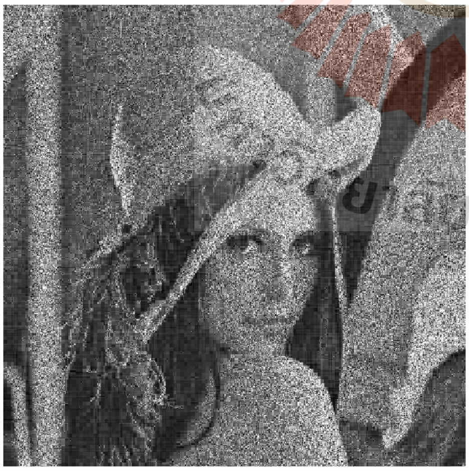

Figure 5.12 Haar, 15/16 hard thresholding

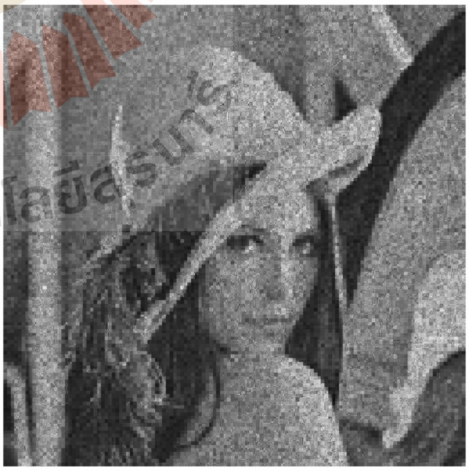

Figure 5.13 Haar, 15/16 soft thresholding

มหาวิทยาลัยเทคโนโลยีสุรนารี

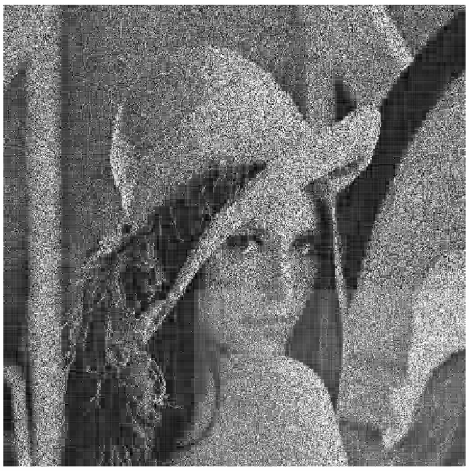

Figure 5.14 Haar, 63/64 hard thresholding

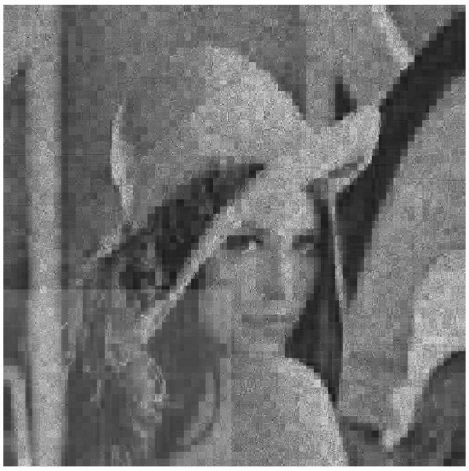

Figure 5.15 Haar, 63/64 soft thresholding

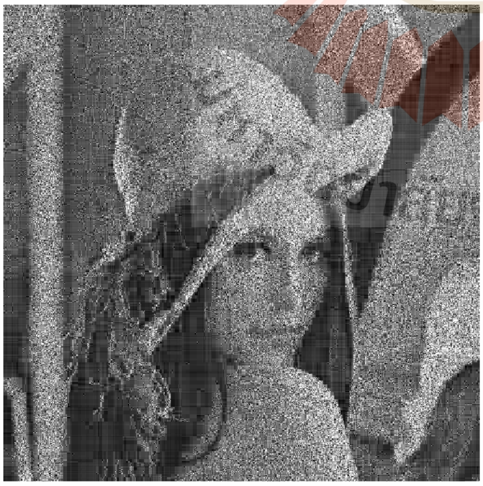

Figure 5.16 Haar, hard thresholding applied to all wavelet coefficients

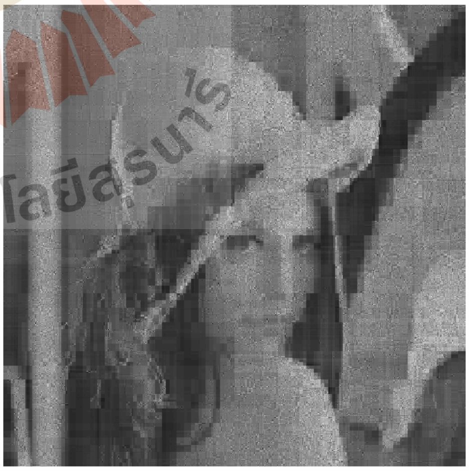

Figure 5.17 Haar, soft thresholding applied to all wavelet coefficients

มหาวิทยาลัยเทคโนโลยีสุรนารี

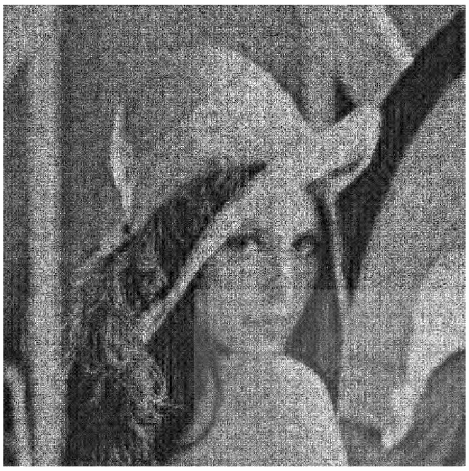

Figure 5.18 db20, 63/64 hard thresholding

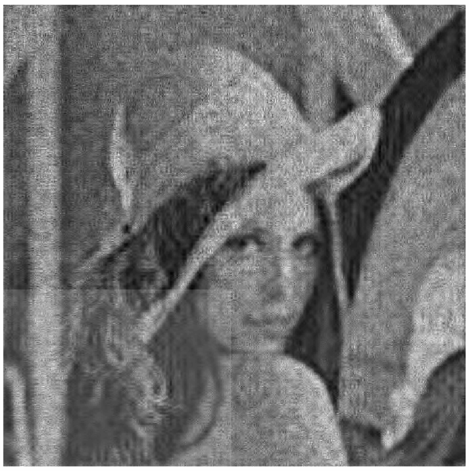

Figure 5.19 db20, 63/64 soft thresholding

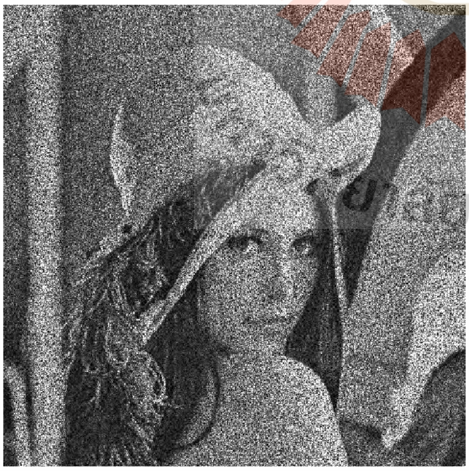

Figure 5.20 db20, hard thresholding applied to all wavelet coefficients

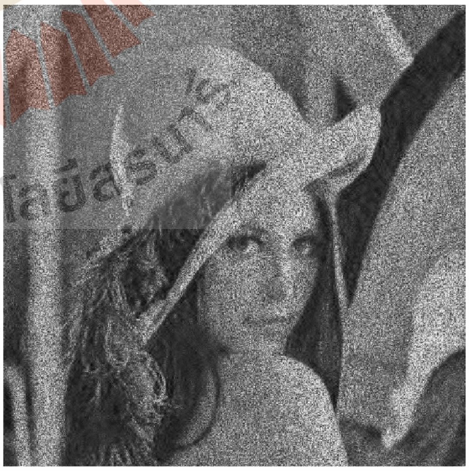

Figure 5.21 db20 soft thresholding applied to all wavelet coefficients

มหาวิทยาลัยเทคโนโลยีสุรนารี

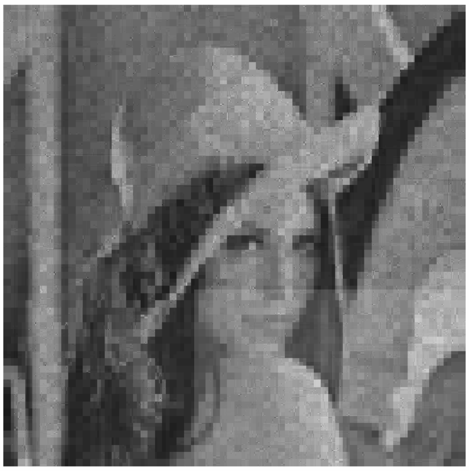

Figure 5.22 Haar, 63/64 soft thresholding, log applied

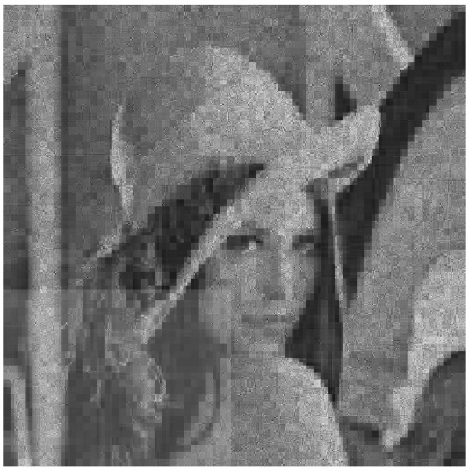

Figure 5.23 Haar, 63/64 soft thresholding, log not applied

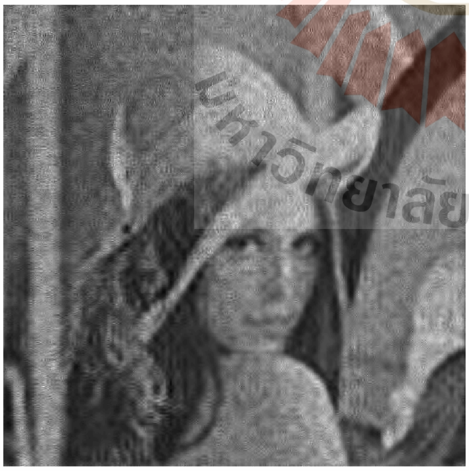

Figure 5.24 db20, 63/64 soft thresholding, log applied

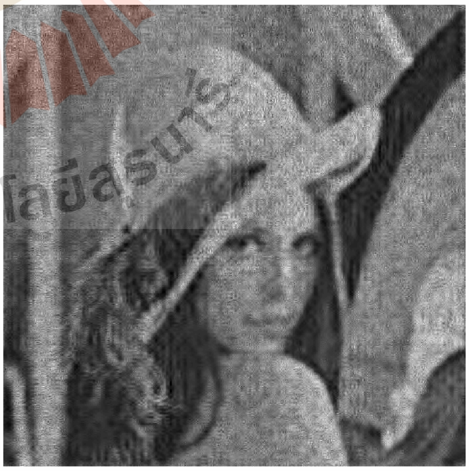

Figure 5.25 db20, 63/64 soft thresholding, log not applied

มหาวิทยาลัยเทคโนโลยีสุรนารี

5.3 Comparison With Other Methods

In this section, the performance of the best wavelet methods is compared with that of other methods. Tables 5.3 and 5.4 show several image quality indices, for images without and with the log transform applied, respectively. Images after denoising – for the case where the logarithmic transform was not applied – are presented in Figures 5.26 through 5.33, with the exception of the wavelet pictures already listed above.

1. The best results overall are for Seekot's variational method. In this method, all metrics are noticeable above those for the other methods. This method has been especially designed for Rayleigh noise.
2. The runner-up is the Mean filtering method. Although it is a simple method, it even surpasses the other variational method, the ROF method.
3. Without the log transform, the ROF method produces better results than the wavelet method. However, after the log transform has been applied, both methods show similar performance. This is surprising, because the ROF model is for additive-type noise.
4. All models which do not use wavelets perform better without log transform. The exception is the median filter, which performs exactly the same, with or without the log transformed applied. This makes sense as the median pixel in a window does not change when taking the logarithm.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Table 5.3 Comparison of various noise reduction methods without log transform applied

Denoising Method (no log transform) $\epsilon = 0.0007$ (Haar) $\epsilon = 0.15$ (db20)	MSE	RMSE	AD	Correlation Coeff.	IQI denoised vs.original	IQI denoised vs.noisy	IQI noisy vs.original	SNR
Mean (5x5 Window)	0.0091111	0.095452	0.075622	0.94240	0.91648	0.51493	0.54110	25.021
Mean (7x7 Window)	0.0089513	0.094612	0.076213	0.95081	0.91727	0.49107	0.54131	23.73
Median (5x5 Window)	0.013367	0.11562	0.09255	0.91198	0.89187	0.51913	0.54110	13.973
Median (7x7 Window)	0.012662	0.11253	0.092145	0.93371	0.90538	0.49833	0.54131	14.648
Multiplicative Lee (5x5 Window)	0.010568	0.10280	0.08001	0.91389	0.89725	0.69497	0.54110	19.5
Multiplicative Lee (7x7 Window)	0.010142	0.10071	0.079112	0.92362	0.90202	0.68324	0.54131	19.049
Seekot's method (beta = 5.75)	0.0057399	0.075762	0.056294	0.94531	0.93041	0.56392	0.54095	71.538
ROF Model (beta = 0.251)	0.01037	0.10183	0.078911	0.91759	0.89615	0.65746	0.54095	25.362
Haar (hard) $j_{max} = 8, m = 9$	0.031577	0.17770	0.13048	0.67479	0.66324	0.80316	0.54095	7.6326
Haar (soft) $j_{max} = 8, m = 9$	0.012987	0.11396	0.087553	0.88627	0.84310	0.57576	0.54095	35.745
Haar (63/64 hard) $j_{max} = 8, m = 9$	0.031511	0.17751	0.13038	0.67719	0.66521	0.80619	0.54095	7.5428
Haar (63/64 soft) $j_{max} = 8, m = 9$	0.011547	0.10746	0.082532	0.89895	0.87727	0.60857	0.54095	31.212
db20 (hard) $j_{max} = 8, m = 9$	0.049182	0.22177	0.17304	0.58875	0.55651	0.97473	0.54095	4.8001
db20 (soft) $j_{max} = 8, m = 9$	0.022616	0.15039	0.11279	0.74486	0.74144	0.87093	0.54095	10.326
db20 (63/64 hard) $j_{max} = 8, m = 9$	0.029491	0.17173	0.13218	0.69174	0.68907	0.77791	0.54095	7.7205
db20 (63/64 soft) $j_{max} = 8, m = 9$	0.011828	70.10875	0.084473	0.89219	0.87812	0.60878	0.54095	26.05

Table 5.4 Comparison of various noise reduction methods with log transform applied

Denoising Method (with log transform) $\epsilon = 0.0007$ (Haar) $\epsilon = 0.0007$ (db20)	MSE	RMSE	AD	Correlation Coeff.	IQI denoised vs.original	IQI denoised vs.noisy	IQI noisy vs.original	SNR
Mean (5x5 Window)	0.011952	0.10933	0.088541	0.93987	0.89977	0.50306	0.54110	18.167
Mean (7x7 Window)	0.011913	0.10915	0.089999	0.94946	0.89971	0.47869	0.54131	17.53
Median (5x5 Window)	0.013367	0.11562	0.09255	0.91198	0.89187	0.51913	0.54110	13.973
Median (7x7 Window)	0.012662	0.11253	0.092145	0.93371	0.90538	0.49833	0.54131	14.648
Multiplicative Lee (5x5 Window)	0.012645	0.11245	0.089234	0.92230	0.89002	0.64749	0.54110	16.312
Multiplicative Lee (7x7 Window)	0.012368	0.11121	0.089526	0.93369	0.89371	0.63242	0.54131	15.983
Seekot's method (beta = 5.75)	0.0089028	0.094355	0.073201	0.93096	0.90470	0.57810	0.54095	35.103
ROF Model (beta = 0.251)	0.012631	0.11239	0.09077	0.92678	0.88663	0.59560	0.54095	19.707
Haar (hard) $j_{max} = 8, m = 9$	0.019560	0.13986	0.10576	0.80094	0.78840	0.59643	0.54095	15.227
Haar (soft) $j_{max} = 8, m = 97$	0.016094	0.12686	0.10285	0.91136	0.81714	0.43350	0.54095	28.583
Haar (63/64 hard) $j_{max} = 8, m = 9$	0.019290	0.13889	0.10491	0.80576	0.79369	0.60550	0.54095	14.475
Haar (63/64 soft) $j_{max} = 8, m = 9$	0.013156	0.11470	0.09331	0.92547	0.87800	0.49148	0.54095	26.077
db20 (hard) db20, $j_{max} = 8, m = 9$	0.044589	0.21116	0.16269	0.60897	0.58211	0.93282	0.54095	5.1122
db20 (soft) db20, $j_{max} = 8, m = 9$	0.018939	0.13762	0.10359	0.81195	0.79347	0.75111	0.54095	12.772
db20 (63/64 hard), $j_{max} = 8, m = 9$	0.028615	0.16916	0.12982	0.70333	0.69906	0.74540	0.54095	7.6180
db20 (63/64 soft) $j_{max} = 8, m = 9$	0.012829	0.11327	0.091726	0.92311	0.88641	0.53275	0.54095	23.112

มหาวิทยาลัยเทคโนโลยีสุรนารี

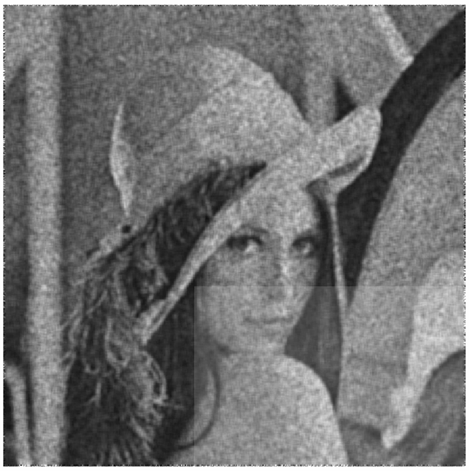

Figure 5.26 Mean filter, 5 × 5 window

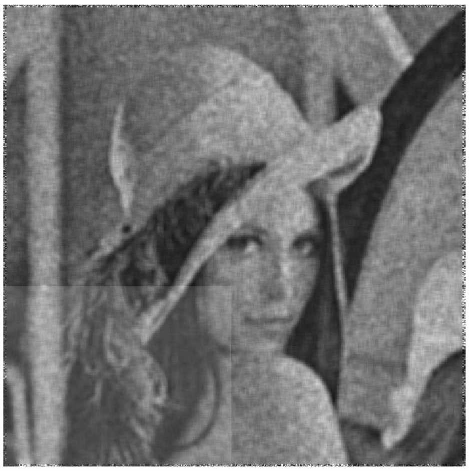

Figure 5.27 Mean filter, 7 × 7 window

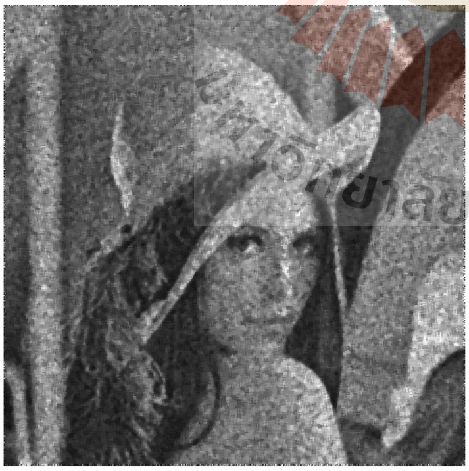

Figure 5.28 Median filter, 5×5 window

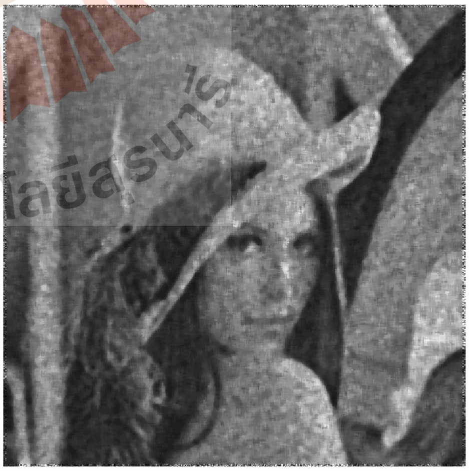

Figure 5.29 Median filter, 7×7 window

มหาวิทยาลัยเทคโนโลยีสุรนารี

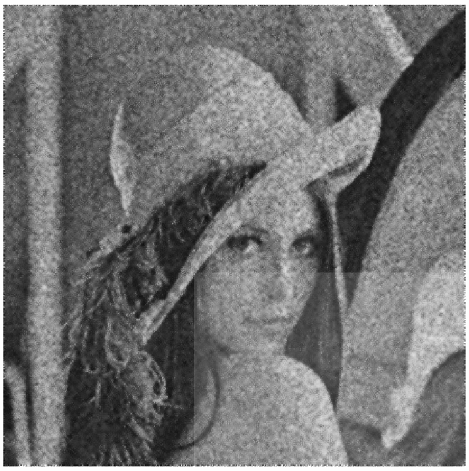

Figure 5.30 Multiplicative Lee filter, 5 × 5 window

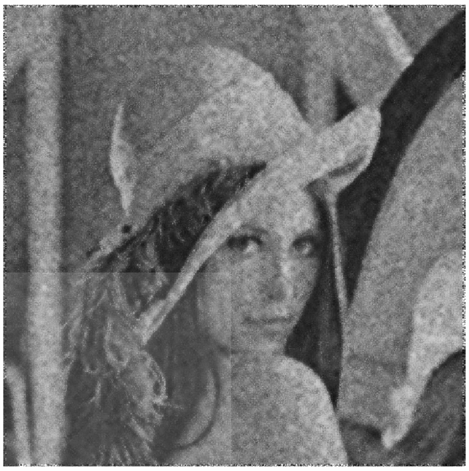

Figure 5.31 Multiplicative Lee filter, 7 × 7 window

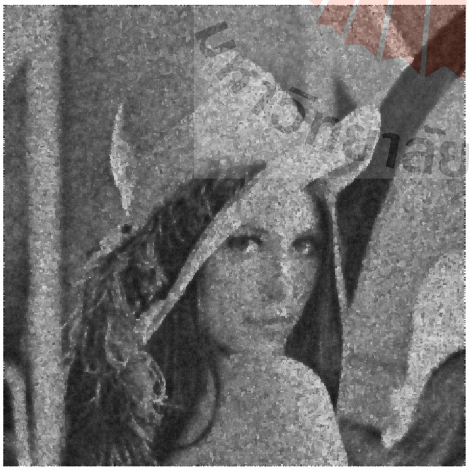

Figure 5.32 ROF variational filter

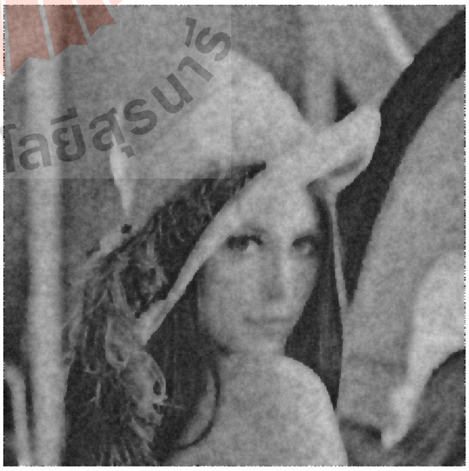

Figure 5.33 Seekot variational filter

มหาวิทยาลัยเทคโนโลยีสุรนารี

CHAPTER VI

CONCLUSION

In this thesis, several methods for reducing Rayleigh noise in a synthetic noisy image were studied, with emphasis on wavelet filters. In particular, the following questions were investigated experimentally:

1. Are wavelet filters able to remove this type of noise ?
2. What wavelets are best suited for image denoising ?
3. How do wavelet filters compare with other filters ?
4. Will applying the log transform to an image improve the performance of noise reduction filters ?

Many experiments shown in the literature were performed with lightly noised images, for example with Gaussian noise. In this thesis on the other hand, a synthetic image was used for the experiments, severely degraded by Rayleigh noise in accordance with the theoretical model.

Altogether, these experiments demonstrate that wavelets can be successfully used for noise reduction in images. The choice of wavelet has some impact on noise reduction performance, but to a lesser degree than expected. Daubechies wavelets and Symlets perform marginally better than Haar wavelets, but one should use wavelets of high order, Daubechies wavelets of type db20 worked better than lower order Daubechies wavelets.

More importantly, the experiments showed that soft-thresholding is far superior to hard-thresholding. In addition, thresholding should be applied to the

มหาวิทยาลัยเทคโนโลยีสุร

wavelet coefficients at all except the lowest scales. There is, however, a side effect in that images can become pixellated. This pixellization is only noticeable by the human eye, and is not reflected in the various performance metrics. There is room for further investigation, for example, whether one should impose different thresholds at different scales, or whether one can improve the soft-threshold function used.

In this thesis, two one-dimensional wavelet transforms – in each of the two directions – were applied to obtain a two dimensional transform. Thus, the wavelet coefficients are determined by brightness changes in x and y directions. There exist other more complicated two-dimensional transforms, which might be better provide noise reduction in other directions as well.

Interestingly and surprisingly however, some of the other methods produced even better results. The simplest of the filters, the mean filter, produced relatively good metric results. However, images tend to get blurred at larger window sizes.

Seekot's variational method stood out, both when measured with objective metrics, or by the human eye. However, the variational methods produces softer edges than other methods. It would be interesting to try a combination of the variational and wavelet method to increase image sharpness. In addition, this methods depends on a weight parameter which here was obtained by trial-and-error; it would be worthwhile to design an algorithm which can derive the optimal choice of this parameter in an image.

Another result of this work shows that applying the log transform does not produce better denoised images for many of the methods. This is surprising in that Rayleigh noise is of multiplicative type, so it is natural to expect that taking the logarithmic transform should make methods which work well with additive noise,

มหาวิทยาลัยเทคโนโลยีสุรนารี

perform better. That our experiments show the opposite is definitely worthy of further detailed investigation. However, for the wavelets with hard thresholding, taking the log improved performance across all metrics as expected. For soft thresholding, on the other hand, the log transform only improved on the "match-up" indices, faring worse with the "error" indices.

In practical implementations of an ultrasound scanner, there are big brightness differences present within a raw ultrasound image. In order to improve the visuals, manufacturers often apply the logarithmic transform to make image brightness more balanced and suitable for the eye. It remains to be investigated, in particular using the variational methods, how to best reduce noise in such a modified image, as the experiments show that all except the wavelet methods perform worse after the log transform has been applied.

มหาวิทยาลัยเทคโนโลยีสุรนารี

REFERENCES

มหาวิทยาลัยเทคโนโลยีสุรนารี

REFERENCES

Aubert, G. and Aujol, J.F. (2008). A variational approach to remove multiplicative noise. **SIAM J. Appl. Math.** (68): 925-946.

Burckhardt, C.B. (1978). Speckle in ultrasound B-mode scans. **IEEE Transactions on Sonics and Ultrasonics**. 25: 1-6.

Chambolle, A. (2004). An algorithm for total variation minimization and applications. **Interfaces and Free Boundaries**. 6: 1-24.

Chan, T.F. and Zhou, H.M. (2000). Total variation improved wavelet thresholding in image compression, **Proceedings of the Seventh International Conference on Image Processing ICIP 2000**, 391-394.

Daubechies, I. (1992). **Ten Lectures on Wavelets**. SIAM, Philadelphia.

Donoho, D.L. and Johnstone, I.M. (1994). Ideal spatial adaptation via wavelet shrinkage. **Biometrica**. 81: 425-455.

Donoho, D. L. (1995). De-noising by soft thresholding. **IEEE Transactions on Information Theory**. 41(3): 613-627.

Frost, V., Stiles, J., Shanmugan, K. and Holzman, J. (1982). A model for radar images and its application to adaptive digital filtering and multiplicative noise. **IEEE Transactions on Patterns Analysis and Machine Intelligence**. 4: 157-166.

Jansen, M. (2001). **Noise Reduction by Wavelet Thresholding**. Springer Verlag.

Goodman, J.W. (1985). **Statistical Optics**. John Wiley & Sons.

Hernandez, E., and Weiss, G. (1996). **A First Course on Wavelets**, CRC Press, Boca Raton.

Jaybhjay, J. and Shastri, R. (2015). A study of speckle noise reduction filters. **Signal and Image Processing: An International Journal (SIPJ)**. 6(3): 71-75.

Krommweh, J. and Ma, J. (2010). Tetrolet shrinkage with anisotropic total variation minimization for image approximation, **Signal Processing**. 90: 2529-2539.

Kuan, D.T., Sawchuk, A.A., Strand, T.C. and Chavel, P. (1985). Adaptive noise smoothing filter with signal-dependent noise. **IEEE Transactions on Patterns Analysis and Machine Intelligence**. 7: 165-177.

Lee, J.S. (1980). Digital image enhancement and noise filtering using local statistics. **IEEE Transactions on Patterns Analysis and Machine Intelligence**. 2: 165-168.

Lee, J.S. (1981). Speckle analysis and smoothing of synthetic aperture radar images. **Computer Graphics and Image Processing**. 17: 24-32.

Lee, J.S. (1983). A simple speckle smoothing algorithm for synthetic aperture radar images. **IEEE Transactions on Systems, Man, and Cybernetics**. 13(1): 85-87.

Mallat S. G. (1989). Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbb{R})$. **Trans. Amer. Math. Soc.** 315: 69-87.

มหาวิทยาลัยเทคโนโลยีสุรนารี

Mascarenhas, N.D.A. (1997). An overview of speckle noise filtering in SAR images. **Proceedings of the First Latino-American Seminar on Radar Remote Sensing Image Processing Techniques**, Buenos Aires, Argentina, 2-4 December 1996, 71-73.

Mateo, J.L. and Fernández-Caballero, A. (2009). Finding out general tendencies in speckle noise reduction in ultrasound images, **Expert Systems with Applications**. 36: 7786-7797.

Rose, A. (1973). **Vision - Human and Electronic**. Plenum Press.

Rudin L.I., Osher, S. and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. **Physica D**. 60: 259-268.

Santoso, A. W., Bayuaji, L., Sze, L. T., Lateh, H., and Zain, J. M. (2016). Comparison of various speckle noise reduction filters on synthetic aperture radar image. **International Journal of Applied Engineering Research**. 11(15): 8760-8761.

Seekot, W. (2008). **Ultrasound Image Enhancement by Means of a Variational Approach**. Master's Thesis, Suranaree University of Technology, Thailand.

Walnut, D. (2002). **An Introduction to Wavelet Analysis**. Birkhäuser, Boston.

Wang, Z. and Bovik, A.C. (2002). A universal image quality index. **IEEE Signal Processing Letters**. 9: 81-84.

มหาวิทยาลัยเทคโนโลยีสุรนารี

APPENDICES

มหาวิทยาลัยเทคโนโลยีสุรนารี

APPENDIX A

OCTAVE CODE

This appendix lists some of the Octave computer code used in this thesis.

A.1 Code to Impose Rayleigh Noise on an Image

```
pkg load image; // Load imaging octave package
ref = imread('Downloads/lena512.bmp');

[m,n]=size(ref);
new=im2double(ref);

for i=1:m
    for j=1:n
       x = double( new(i,j) );
                %%%  Rayleigh with sigma^2=x^2/2  is the same as Weibull with
                %%%  scale lambda=x  and shape k=2.
       new(i,j) = wblrnd(x,2);
   end;
end;

    outfilename=strcat('D:/temp/lena512␣noisy␣rayleigh␣unmod.bmp');
    imwrite(new,outfilename);
               %%%%  create another image with unchamged mean
    new1=new*(2/sqrt(pi));
    outfilename=strcat('D:/temp/lena512␣noisy␣rayleigh_mod.bmp');
    imwrite(new1,outfilename);
```

A.2 Code for the Mean Filter

```
pkg load image; // Load package image

w = input('enter␣windows␣size␣n␣␣(2n+1)x(2n+1)␣:␣');

// Read original image
ref_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512.bmp');

// Read noisy image
noisy_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512_noise_rayleigh_.bmp');

ref_dble = log1p(im2double(ref_int)); // Log transform applied
%%ref_dble = im2double(ref_int); // Convert image to double precision
noisy_dble = log1p(im2double(noisy_int));
%%noisy_dble = im2double(noisy_int);

[m,n]=size(ref_dble);
new=noisy_dble;

// Mean Filtering //
```

```
for i=(1+w):(m-w)
  for j=(1+w):(n-w)
    zz = noisy_dble( (i-w):(i+w), (j-w):(j+w) );
    z = zz( : );
    new(i,j) = mean(z);
  end;
end;

// Reverse log data
ref_dble = expm1(ref_dble);
noisy_dble = expm1(noisy_dble);
new = expm1(new);

figure, imshow(ref_dble); // Show original image
figure, imshow(noisy_dble); // Show noisy image
figure, imshow(new); // Show denoised image

// Calculate MSE, AD & Correlation Coefficient //

ref_small   = ref_dble( (1+w):(m-w),(1+w):(n-w) );
noisy_small = noisy_dble( (1+w):(m-w),(1+w):(n-w) );
new_small   =  new( (1+w):(m-w),(1+w):(n-w) );
mymeas = abs ( ref_small - new_small );
mymeas1 = mymeas.*mymeas;
mymeas2 = ref_small.*ref_small;
MSE = sum(mymeas1(:));
MSE = MSE/(m-2*w);
MSE = MSE/(n-2*w)
RMSE = sqrt(MSE)
AD = sum( mymeas(:) );
AD = AD/(m-2*w);
AD = AD/(n-2*w)
Corr2 = corr2(ref_small,new_small)

// Calculate Image Quality Index //

function ret = Image_Quality_Index(f,g)
mean_f = mean(f(:));
mean_g = mean(g(:));
contrast_fg = mean((f - mean_f).*(g - mean_g));
contrast_f_sq = mean((f - mean_f).*(f - mean_f));
contrast_g_sq = mean((g - mean_g).*(g - mean_g));
ret = (4*mean_f*mean_g/(mean_f^2 + mean_g^2))
  *(contrast_fg/(contrast_f_sq + contrast_g_sq));
endfunction;

IQI_Denoising_Original = Image_Quality_Index(new_small,ref_small)
IQI_Denoising_Noisy= Image_Quality_Index(new_small,noisy_small)
IQI_Noisy_Original = Image_Quality_Index(noisy_small,ref_small)

// Calculate SNR on uniform region of original image //

zz1 = new( 1:38,  137:300 );
bg1 = ref_dble( 1:38,  137:300 );
z1 = zz1.^2;
v1 = (zz1-bg1).^2;
snr1 = sum(z1(:))/sum(v1(:));

zz2 = new( 164:215,  480:512 );
bg2 = ref_dble( 164:215,  480:512 );
z2 = zz2.^2;
v2 = (zz2-bg2).^2;
snr2 = sum(z2(:))/sum(v2(:));

zz3 = new( 480:512,  230:258 );
bg3 = ref_dble( 480:512,  230:258 );
z3 = zz3.^2;
v3 = (zz3-bg3).^2;
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
snr3 = sum(z3(:))/sum(v3(:));

snrvec = [ snr1 snr2 snr3];
SNR = max( snrvec )
```

A.3 Code for the Mean Median Filter

```
pkg load image; // Load package image

w = input('enter␣windows␣size␣n␣␣(2n+1)x(2n+1)␣:␣');

// Read original image
ref_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512.bmp');

// Read noisy image
noisy_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512_noise_rayleigh_.bmp');

ref_dble = log1p(im2double(ref_int)); // Log transform applied
%%ref_dble = im2double(ref_int); // Convert image to double precision
noisy_dble = log1p(im2double(noisy_int));
%%noisy_dble = im2double(noisy_int);

[m,n]=size(ref_dble);
new=noisy_dble;

// Median Filtering //

for i=(1+w):(m-w)
  for j=(1+w):(n-w)
    zz = noisy_dble( (i-w):(i+w), (j-w):(j+w) );
    z = zz( : );
    new(i,j) = median(z);
  end;
end;

// Reverse log data
ref_dble = expm1(ref_dble);
noisy_dble = expm1(noisy_dble);
new = expm1(new);

figure, imshow(ref_dble); // Show original image
figure, imshow(noisy_dble); // Show noisy image
figure, imshow(new); // Show denoised image

// Calculate MSE, AD & Correlation Coefficient //

ref_small   = ref_dble( (1+w):(m-w),(1+w):(n-w) );
noisy_small = noisy_dble( (1+w):(m-w),(1+w):(n-w) );
new_small   =  new( (1+w):(m-w),(1+w):(n-w) );
mymeas = abs ( ref_small - new_small );
mymeas1 = mymeas.*mymeas;
mymeas2 = ref_small.*ref_small;
MSE = sum(mymeas1(:));
MSE = MSE/(m-2*w);
MSE = MSE/(n-2*w)
RMSE = sqrt(MSE)
AD = sum( mymeas(:) );
AD = AD/(m-2*w);
AD = AD/(n-2*w)
Corr2 = corr2(ref_small,new_small)

// Calculate Image Quality Index //
```

```
function ret = Image_Quality_Index(f,g)
mean_f = mean(f(:));
mean_g = mean(g(:));
contrast_fg = mean((f - mean_f).*(g - mean_g));
contrast_f_sq = mean((f - mean_f).*(f - mean_f));
contrast_g_sq = mean((g - mean_g).*(g - mean_g));
ret = (4*mean_f*mean_g/(mean_f^2 + mean_g^2))
     *(contrast_fg/(contrast_f_sq + contrast_g_sq));
endfunction;

IQI_Denoising_Original = Image_Quality_Index(new_small,ref_small)
IQI_Denoising_Noisy= Image_Quality_Index(new_small,noisy_small)
IQI_Noisy_Original = Image_Quality_Index(noisy_small,ref_small)

// Calculate SNR on uniform region of original image //

zz1 = new( 1:38,  137:300 );
bg1 = ref_dble( 1:38,  137:300 );
z1 = zz1.^2;
v1 = (zz1-bg1).^2;
snr1 = sum(z1(:))/sum(v1(:));

zz2 = new( 164:215,  480:512 );
bg2 = ref_dble( 164:215,  480:512 );
z2 = zz2.^2;
v2 = (zz2-bg2).^2;
snr2 = sum(z2(:))/sum(v2(:));

zz3 = new( 480:512,  230:258 );
bg3 = ref_dble( 480:512,  230:258 );
z3 = zz3.^2;
v3 = (zz3-bg3).^2;
snr3 = sum(z3(:))/sum(v3(:));

snrvec = [ snr1 snr2 snr3];
SNR = max( snrvec )
```

A.4 Code for the Multiplicative Lee Filter

```
pkg load image; // Load package image

w = input('enter␣windows␣size␣n␣␣(2n+1)x(2n+1)␣:␣');

// Read original image
ref_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512.bmp');

// Read noisy image
noisy_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512_noise_rayleigh_.bmp');

ref_dble = log1p(im2double(ref_int)); // Log transform applied
%%ref_dble = im2double(ref_int); // Convert image to double precision
noisy_dble = log1p(im2double(noisy_int));
%%noisy_dble = im2double(noisy_int);

function ret = varq(u) // Variance
mm = mean(u);
ret = var( u-mm, 1);
endfunction;

[m,n]=size(ref_dble);
new=noisy_dble;

// Multiplicative Lee Filtering //
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
var_ref = 1
new=noisy_dble;

for i=(1+w):(m-w)
  for j=(1+w):(n-w)
    zz = noisy_dble( (i-w):(i+w), (j-w):(j+w) );
    z = zz( : );
    v = varq(z);
    s_z = mean(z);
    K = v/(v + s_z*s_z*var_ref); // Weighting parameter
    new(i,j) = s_z + K*(noisy_dble(i,j) - s_z);
  end;
end;

// Reverse log data
ref_dble = expm1(ref_dble);
noisy_dble = expm1(noisy_dble);
new = expm1(new);

figure, imshow(ref_dble); // Show original image
figure, imshow(noisy_dble); // Show noisy image
figure, imshow(new); // Show denoised image

// Calculate MSE, AD & Correlation Coefficient //

ref_small   = ref_dble( (1+w):(m-w),(1+w):(n-w) );
noisy_small = noisy_dble( (1+w):(m-w),(1+w):(n-w) );
new_small   =  new( (1+w):(m-w),(1+w):(n-w) );
mymeas = abs ( ref_small - new_small );
mymeas1 = mymeas.*mymeas;
mymeas2 = ref_small.*ref_small;
MSE = sum(mymeas1(:));
MSE = MSE/(m-2*w);
MSE = MSE/(n-2*w)
RMSE = sqrt(MSE)
AD = sum( mymeas(:) );
AD = AD/(m-2*w);
AD = AD/(n-2*w)
Corr2 = corr2(ref_small,new_small)

// Calculate Image Quality Index //

function ret = Image_Quality_Index(f,g)
mean_f = mean(f(:));
mean_g = mean(g(:));
contrast_fg = mean((f - mean_f).*(g - mean_g));
contrast_f_sq = mean((f - mean_f).*(f - mean_f));
contrast_g_sq = mean((g - mean_g).*(g - mean_g));
ret = (4*mean_f*mean_g/(mean_f^2 + mean_g^2))
*(contrast_fg/(contrast_f_sq + contrast_g_sq));
endfunction;

IQI_Denoising_Original = Image_Quality_Index(new_small,ref_small)
IQI_Denoising_Noisy= Image_Quality_Index(new_small,noisy_small)
IQI_Noisy_Original = Image_Quality_Index(noisy_small,ref_small)

// Calculate SNR on uniform region of original image //

zz1 = new( 1:38,  137:300 );
bg1 = ref_dble( 1:38,  137:300 );
z1 = zz1.^2;
v1 = (zz1-bg1).^2;
snr1 = sum(z1(:))/sum(v1(:));

zz2 = new( 164:215,  480:512 );
bg2 = ref_dble( 164:215,  480:512 );
z2 = zz2.^2;
v2 = (zz2-bg2).^2;
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
snr2 = sum(z2(:))/sum(v2(:));

zz3 = new( 480:512,  230:258 );
bg3 = ref_dble( 480:512,  230:258 );
z3 = zz3.^2;
v3 = (zz3-bg3).^2;
snr3 = sum(z3(:))/sum(v3(:));

snrvec = [ snr1 snr2 snr3];
SNR = max( snrvec )
```

A.5 Code for the Seekot's Variational Method

```
pkg load image; // Load package image

//Set up constant
delta_t = 0.01;
h = 1;
beta = 5.75
%%best_beta_log1p = 5.75;
K1 = delta_t/h;
K2 = 2*delta_t/beta;

// Read original image
ref_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512.bmp');

// Read noisy image
noisy_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512_noise_rayleigh_.bmp');

[m,n]=size(noisy_int);

ref_dble = log1p(im2double(ref_int)); // Log transform applied
%%ref_dble = im2double(ref_int); // Convert image to double precision
noisy_dble = log1p(im2double(noisy_int));
%%noisy_dble = im2double(noisy_int);

// Seekot Method (Solving Euler-Lagrange Equation) //

u_old = ones(m,n)*0.5;
u_old(1,:) = noisy_dble(1,:);   u_old(m,:) = noisy_dble(m,:);
u_old(:,1) = noisy_dble(:,1);   u_old(:,n) = noisy_dble(:,n);
u_new = u_old;

function ret = M(a,b)
ret = (sign(a)+sign(b)).*min(abs(a),abs(b))/2;
endfunction;

for k=1:1000
  delta1000= u_old(3:m,2:n-1)   - u_old(2:m-1,2:n-1);
  delta00_10   = u_old(2:m-1,2:n-1) - u_old(1:m-2,2:n-1);
  delta0100= u_old(2:m-1,3:n)   - u_old(2:m-1,2:n-1);
  delta000_1   = u_old(2:m-1,2:n-1) - u_old(2:m-1,1:n-2);
  delta_11_10  = u_old(1:m-2,3:n)   - u_old(1:m-2,2:n-1);
  delta_10_1_1 = u_old(1:m-2,2:n-1) - u_old(2:m-1,1:n-2);
  delta1_10_1  = u_old(3:m,1:n-2)   - u_old(2:m-1,1:n-2);
  delta0_1_1_1 = u_old(2:m-1,1:n-2) - u_old(1:m-2,1:n-2);

  m1 = M(delta0100,delta000_1);
  m2 = M(delta1000,delta00_10);
  m3 = M(delta_11_10,delta_10_1_1);
  m4 = M(delta1_10_1,delta0_1_1_1);

  Term1 = delta1000./sqrt(delta1000.*delta1000 + m1.*m1);
  Term1(abs(delta1000)<1E-5)  = 0.0;
  Term2 = delta0100./sqrt(delta0100.*delta0100 + m2.*m2);
  Term2(abs(delta0100)<1E-5)  = 0.0;
  Term3 = -delta00_10./sqrt(delta00_10.*delta00_10 + m3.*m3);
  Term3(abs(delta00_10)<1E-5) = 0.0;
  Term4 = -delta000_1./sqrt(delta000_1.*delta000_1 + m4.*m4);
  Term4(abs(delta000_1)<1E-5) = 0.0;

  usquare = u_old.*u_old;
  ucube = usquare.*u_old;
  ucube(abs(ucube)<1E-2)=1E-2;
  noisy_dble_square = noisy_dble.*noisy_dble;
  noisy_term = (noisy_dble_square(2:m-1,2:n-1) - usquare(2:m-1,2:n-1))./ucube(2:m
      -1,2:n-1);
  u_new(2:m-1,2:n-1) =  u_old(2:m-1,2:n-1)
    + K1*(Term1+Term2+Term3+Term4)
    + K2*noisy_term;
  u_old = u_new;
end;

// Reverse log data

u_old = expm1(u_old);
ref_dble = expm1(ref_dble);
noisy_dble = expm1(noisy_dble);


figure, imshow(ref_dble); // Show original image
figure, imshow(noisy_dble); // Show noisy image
figure, imshow(u_old); // Show denoised image

// Calculate MSE, AD & Correlation Coefficient //

mymeas = abs ( ref_dble - u_old );
mymeas1 = mymeas.*mymeas;
MSE = sum(mymeas1(:));
MSE = MSE/m;
MSE = MSE/n
RMSE = sqrt(MSE)
AD = sum(mymeas(:) );
AD = AD/m;
AD = AD/n
Corr2 = corr2(ref_dble,u_old)

// Calculate Image Quality Index //

function ret = Image_Quality_Index(f,g)
  mean_f = mean(f(:));
  mean_g = mean(g(:));
  contrast_fg = mean((f - mean_f).*(g - mean_g));
  contrast_f_sq = mean((f - mean_f).*(f - mean_f));
  contrast_g_sq = mean((g - mean_g).*(g - mean_g));
  ret = (4*mean_f*mean_g/(mean_f^2 + mean_g^2)) *(contrast_fg/(contrast_f_sq +
      contrast_g_sq));
endfunction;

IQI_Denoising_Original = Image_Quality_Index(u_old,ref_dble)
IQI_Denoising_Noisy= Image_Quality_Index(u_old,noisy_dble)
IQI_Noisy_Original = Image_Quality_Index(noisy_dble,ref_dble)

// Calculate SNR on uniform region of original image //

zz1 = u_old( 1:38,  137:300 );
bg1 = ref_dble( 1:38,  137:300 );
z1 = zz1.^2;
v1 = (zz1-bg1).^2;
snr1 = sum(z1(:))/sum(v1(:));

zz2 = u_old( 164:215,  480:512 );
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
bg2 = ref_dble( 164:215,  480:512 );
z2 = zz2.^2;
v2 = (zz2-bg2).^2;
snr2 = sum(z2(:))/sum(v2(:));

zz3 = u_old( 480:512,  230:258 );
bg3 = ref_dble( 480:512,  230:258 );
z3 = zz3.^2;
v3 = (zz3-bg3).^2;
snr3 = sum(z3(:))/sum(v3(:));

snrvec = [ snr1 snr2 snr3];
SNR = max( snrvec )
```

A.6 Code for the ROF Variational Method

```
pkg load image; // Load package image

//Set up constant
delta_t = 0.01;
h = 1;
beta = 0.251
%%best_beta_log1p = 0.251;
K1 = delta_t/h;
K2 = 2*delta_t/beta;

// Read original image
ref_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512.bmp');

// Read noisy image
noisy_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512_noise_rayleigh_.bmp');

[m,n]=size(noisy_int);

ref_dble = log1p(im2double(ref_int)); // Log transform applied
%%ref_dble = im2double(ref_int); // Convert image to double precision
noisy_dble = log1p(im2double(noisy_int));
%%noisy_dble = im2double(noisy_int);

// ROF Model (Solving Euler-Lagrange Equation) //

u_old = ones(m,n)*0.5;
u_old(1,:) = noisy_dble(1,:);   u_old(m,:) = noisy_dble(m,:);
u_old(:,1) = noisy_dble(:,1);   u_old(:,n) = noisy_dble(:,n);
u_new = u_old;

function ret = M(a,b)
  ret = (sign(a)+sign(b)).*min(abs(a),abs(b))/2;
endfunction;

for k=1:1000
  delta1000= u_old(3:m,2:n-1)   - u_old(2:m-1,2:n-1);
  delta00_10   = u_old(2:m-1,2:n-1) - u_old(1:m-2,2:n-1);
  delta0100= u_old(2:m-1,3:n)   - u_old(2:m-1,2:n-1);
  delta000_1   = u_old(2:m-1,2:n-1) - u_old(2:m-1,1:n-2);
  delta_11_10  = u_old(1:m-2,3:n)   - u_old(1:m-2,2:n-1);
  delta_10_1_1 = u_old(1:m-2,2:n-1) - u_old(1:m-2,1:n-2);
  delta1_10_1  = u_old(3:m,1:n-2)   - u_old(2:m-1,1:n-2);
  delta0_1_1_1 = u_old(2:m-1,1:n-2) - u_old(1:m-2,1:n-2);

  m1 = M(delta0100,delta000_1);
  m2 = M(delta1000,delta00_10);
  m3 = M(delta_11_10,delta_10_1_1);
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
  m4 = M(delta1_10_1,delta0_1_1_1);

  Term1 = delta1000./sqrt(delta1000.*delta1000 + m1.*m1);
  Term1(abs(delta1000)<1E-5)  = 0.0;
  Term2 = delta0100./sqrt(delta0100.*delta0100 + m2.*m2);
  Term2(abs(delta0100)<1E-5)  = 0.0;
  Term3 = -delta00_10./sqrt(delta00_10.*delta00_10 + m3.*m3);
  Term3(abs(delta00_10)<1E-5) = 0.0;
  Term4 = -delta000_1./sqrt(delta000_1.*delta000_1 + m4.*m4);
  Term4(abs(delta000_1)<1E-5) = 0.0;

  noisy_term = noisy_dble(2:m-1,2:n-1) - u_old(2:m-1,2:n-1);
  u_new(2:m-1,2:n-1) =  u_old(2:m-1,2:n-1) + K1*(Term1+Term2+Term3+Term4) + K2*
      noisy_term;
  u_old = u_new;
end;

// Reverse log data

u_old = expm1(u_old);
ref_dble = expm1(ref_dble);
noisy_dble = expm1(noisy_dble);


figure, imshow(ref_dble); // Show original image
figure, imshow(noisy_dble); // Show noisy image
figure, imshow(u_old); // Show denoised image

// Calculate MSE, AD & Correlation Coefficient //

mymeas = abs ( ref_dble - u_old );
mymeas1 = mymeas.*mymeas;
MSE = sum(mymeas1(:));
MSE = MSE/m;
MSE = MSE/n
RMSE = sqrt(MSE)
AD = sum(mymeas(:) );
AD = AD/m;
AD = AD/n
Corr2 = corr2(ref_dble,u_old)

// Calculate Image Quality Index //

function ret = Image_Quality_Index(f,g)
  mean_f = mean(f(:));
  mean_g = mean(g(:));
  contrast_fg = mean((f - mean_f).*(g - mean_g));
  contrast_f_sq = mean((f - mean_f).*(f - mean_f));
  contrast_g_sq = mean((g - mean_g).*(g - mean_g));
  ret = (4*mean_f*mean_g/(mean_f^2 + mean_g^2)) *(contrast_fg/(contrast_f_sq +
      contrast_g_sq));
endfunction;

IQI_Denoising_Original = Image_Quality_Index(u_old,ref_dble)
IQI_Denoising_Noisy= Image_Quality_Index(u_old,noisy_dble)
IQI_Noisy_Original = Image_Quality_Index(noisy_dble,ref_dble)

// Calculate SNR on uniform region of original image //

zz1 = u_old( 1:38,  137:300 );
bg1 = ref_dble( 1:38,  137:300 );
z1 = zz1.^2;
v1 = (zz1-bg1).^2;
snr1 = sum(z1(:))/sum(v1(:));

zz2 = u_old( 164:215,  480:512 );
bg2 = ref_dble( 164:215,  480:512 );
z2 = zz2.^2;
v2 = (zz2-bg2).^2;
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
snr2 = sum(z2(:))/sum(v2(:));

zz3 = u_old( 480:512,  230:258 );
bg3 = ref_dble( 480:512,  230:258 );
z3 = zz3.^2;
v3 = (zz3-bg3).^2;
snr3 = sum(z3(:))/sum(v3(:));

snrvec = [ snr1 snr2 snr3];
SNR = max( snrvec )
```

A.7 Code for the Haar Wavelet Method with Hard Thresholding

```
pkg load image; // Load package image

// Read original image //
original_int = imread('C:/Users/xiangeh/Desktop/Denoising␣Speckle␣Noise␣in␣the␣
    image/lena512.bmp');

// Read noisy image //
noisy_int = imread('C:/Users/xiangeh/Desktop/Denoising␣Speckle␣Noise␣in␣the␣image
    /lena512_noise_rayleigh_.bmp');

original_dble = log1p(im2double(original_int));  // Log transform applied
%%original_dble = im2double(original_int); // Convert image to double precision
noisy_dble = log1p(im2double(noisy_int));
%%noisy_dble = im2double(noisy_int);
[m1,n1]= size(original_dble);

//Set up constant
thold = 0.0007 // Thresholding value
j_max = 8   // should be <= m;
m = 9 // 256 pixels
M = 2^m;

Delta_x = 1/M;
N = 2^(j_max);
//  vector containing wavelet coefficients in one dimension
wt1 = zeros( M,  2*N );
inv_wt1 = wt1;
//  vector containing wavelet coefficients in two dimensions
wt2 = zeros( 2*N,2*N );
inv = zeros(M,M);

// Haar Wavelet Method //

function ret = Haar_0th_scaling_function(k,M)
  if (k==0)
    ret = ones(M,1);
  else
    ret = zeros(M,1);
  end;
endfunction;

function ret = Haar_jth_mother_wavelet(j,k,M)  // allow j=0.._j_max
  ret = zeros(M,1);
  x0_lower=2*k*M;
  x0_medium=(2*k+1)*M;
  x0_upper=2*(k+1)*M;
  twopj=2^(j/2);
  for i=1:M
  xx=2*i*(2^j);
  if ( ( x0_lower < xx ) && ( xx <= x0_medium ))
    ret(i) =  twopj;
    elseif ( ( x0_medium < xx ) && ( xx<= x0_upper ))
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
      ret(i) = -twopj;
    end;
  end;
endfunction;

function ret = wavelet_transform(j_max, M, g)
  N = 2^(j_max);
  ret = zeros(2*N,1);
  Delta_x = 1/M;

  z = Haar_0th_scaling_function(0,M);
  ret( 1 ) = dot(g,z)*Delta_x;

  for j=0:j_max
    power=2^j-1;
    for k=0:power
      z = Haar_jth_mother_wavelet(j,k,M);
      ret( power + k + 2) = dot(g,z)*Delta_x;
    end;
  end;
endfunction;

function ret = inverse_wavelet_transform(j_max, M, wt)
  N = 2^(j_max);
  ret = zeros(M,1);
  ret =  wt(1)*(Haar_0th_scaling_function(0,M));
  for j=0:j_max
    power=2^j-1;
    for k=0:power
      ret = ret + wt( power + k + 2) *(Haar_jth_mother_wavelet(j,k,M));
    end;
  end;
endfunction;


// take wt of each image row;
for i=1:M
  wt1(i,:) = wavelet_transform( j_max, M, noisy_dble( i,:) );
end;
// take wt of each image column;
for i=1:2*N
  wt2(:,i) = wavelet_transform( j_max, M, wt1(:,i) );
end;

// Thresholding Method //

// 1/2 Piece Hard Thresholding //

for i=N+1:2*N
  for j=(N+2)/2:N
    if abs(wt2(i,j)) < thold
      wt2(i,j) = 0;
    end;
  end;
end;

for i=N/2+1:2*N
    for j=N+1:2*N
    if abs(wt2(i,j)) < thold
      wt2(i,j) = 0;
    end;
  end;
end;

// 1/4 Piece Hard Thresholding //

for i=N+1:2*N
  for j=N+1:2*N
    if abs(wt2(i,j)) < thold
```

```
      wt2(i,j) = 0;
    end;
  end;
end;

// 3/4 Piece Hard Thresholding //

for i=N+1:2*N
  for j=1:N
    if abs(wt2(i,j)) < thold
      wt2(i,j) = 0;
    end;
  end;
end;

for i=1:2*N
  for j=N+1:2*N
    if abs(wt2(i,j)) < thold
      wt2(i,j) = 0;
    end;
  end;
end;

// 15/16 Piece Hard Thresholding //

for i=1:N/2
  for j=N/2:2*N
    if abs(wt2(i,j)) < thold
      wt2(i,j) = 0;
    end;
  end;
end;

for i=N/2+1:2*N
  for j=1:2*N
    if abs(wt2(i,j)) < thold
      wt2(i,j) = 0;
    end;
  end;
end;

// 63/64 Piece Hard Thresholding //

for i=1:N/4
  for j=N/4:2*N
    if abs(wt2(i,j)) < thold
      wt2(i,j) = 0;
    end;
  end;
end;

for i=N/4+1:2*N
  for j=1:2*N
    if abs(wt2(i,j)) < thold
      wt2(i,j) = 0;
    end;
  end;
end;

// Normal Hard Thresholding //

   wt2( abs(wt2) < thold ) = 0.0;

// Now we have the Wavelet transform. Take the inverse transform //

for i=1:2*N
  inv_wt1(:,i) = inverse_wavelet_transform( j_max, M, wt2(:,i) );
end;
for i=1:M
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
  inv(i,:)  =  inverse_wavelet_transform( j_max, M, inv_wt1(i,:) );
end;

//Reverse log data

inv = expm1(inv);
original_dble = expm1(original_dble);
noisy_dble= expm1(noisy_dble);

figure, imshow(original_dble); // Show original image
figure, imshow(noisy_dble); // Show noisy image
figure, imshow(inv); // Show denoised image

// Calculate MSE, AD & Correlation Coeifficient //

mymeas = abs ( original_dble - inv );
mymeas1 = mymeas.*mymeas;
MSE = sum(mymeas1(:));
MSE = MSE/m1;
MSE = MSE/n1
RMSE = sqrt(MSE)
AD = sum(mymeas(:) );
AD = AD/m1;
AD = AD/n1
Corr2 = corr2(original_dble,inv)

// Calculate Image Quality Index //

function ret = Image_Quality_Index(f,g)
  mean_f = mean(f(:));
  mean_g = mean(g(:));
  contrast_fg = mean((f - mean_f).*(g - mean_g));
  contrast_f_sq = mean((f - mean_f).*(f - mean_f));
  contrast_g_sq = mean((g - mean_g).*(g - mean_g));
  ret = (4*mean_f*mean_g/(mean_f^2 + mean_g^2))  *(contrast_fg/(contrast_f_sq +
      contrast_g_sq));
endfunction;

IQI_Denoising_Original = Image_Quality_Index(inv,original_dble)
IQI_Denoising_Noisy= Image_Quality_Index(inv,noisy_dble)
IQI_Noisy_Original = Image_Quality_Index(noisy_dble,original_dble)

// Calcualte SNR on uniform region of original image //

zz1 = inv( 1:38,  137:300 );
bg1 = original_dble( 1:38,  137:300 );
z1 = zz1.^2;
v1 = (zz1-bg1).^2;
snr1 = sum(z1(:))/sum(v1(:));

zz2 = inv( 164:215,  480:512 );
bg2 = original_dble( 164:215,  480:512 );
z2 = zz2.^2;
v2 = (zz2-bg2).^2;
snr2 = sum(z2(:))/sum(v2(:));

zz3 = inv( 480:512,  230:258 );
bg3 = original_dble( 480:512,  230:258 );
z3 = zz3.^2;
v3 = (zz3-bg3).^2;
snr3 = sum(z3(:))/sum(v3(:));

snrvec = [ snr1 snr2 snr3];
SNR = max( snrvec )
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

A.8 Code for the Haar Wavelet Method with Soft Thresholding

```
pkg load image; // Load package image

// Read original image //
original_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512.bmp');

// Read noisy image //
noisy_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512_noise_rayleigh_.bmp');

original_dble = log1p(im2double(original_int));  // Log transform applied
%%original_dble = im2double(original_int); // Convert image to double precision
noisy_dble = log1p(im2double(noisy_int));
%%noisy_dble = im2double(noisy_int);
[m1,n1]= size(original_dble);

//Set up constant
thold = 0.0007 // Thresholding value
j_max = 8   // should be <= m;
m = 9 // 256 pixels
M = 2^m;

Delta_x = 1/M;
N = 2^(j_max);
//  vector containing wavelet coefficients in one dimension
wt1 = zeros( M,  2*N );
inv_wt1 = wt1;
//  vector containing wavelet coefficients in two dimensions
wt2 = zeros( 2*N,2*N );
inv = zeros(M,M);

// Haar Wavelet Method //

function ret = Haar_0th_scaling_function(k,M)
  if (k==0)
    ret = ones(M,1);
  else
    ret = zeros(M,1);
  end;
endfunction;

function ret = Haar_jth_mother_wavelet(j,k,M)  // allow j=0.._j_max
  ret = zeros(M,1);
  x0_lower=2*k*M;
  x0_medium=(2*k+1)*M;
  x0_upper=2*(k+1)*M;
  twopj=2^(j/2);
  for i=1:M
    xx=2*i*(2^j);
      if ( ( x0_lower < xx ) && ( xx <= x0_medium ))
        ret(i) =  twopj;
        elseif ( ( x0_medium < xx ) && ( xx<= x0_upper ))
          ret(i) = -twopj;
      end;
  end;
endfunction;

function ret = wavelet_transform(j_max, M, g)
  N = 2^(j_max);
  ret = zeros(2*N,1);
  Delta_x = 1/M;
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
  z = Haar_0th_scaling_function(0,M);
  ret( 1 ) = dot(g,z)*Delta_x;

  for j=0:j_max
    power=2^j-1;
    for k=0:power
      z = Haar_jth_mother_wavelet(j,k,M);
      ret( power + k + 2) = dot(g,z)*Delta_x;
    end;
  end;
endfunction;

function ret = inverse_wavelet_transform(j_max, M, wt)
N = 2^(j_max);
ret = zeros(M,1);
ret =  wt(1)*(Haar_0th_scaling_function(0,M));
for j=0:j_max
power=2^j-1;
for k=0:power
ret = ret + wt( power + k + 2)
*(Haar_jth_mother_wavelet(j,k,M));
end;
end;
endfunction;


// take wt of each image row;
for i=1:M
wt1(i,:) = wavelet_transform( j_max, M, noisy_dble( i,:) );
end;
// take wt of each image column;
for i=1:2*N
wt2(:,i) = wavelet_transform( j_max, M, wt1(:,i) );
end;

// Thresholding Method //

// 1/2 Piece Soft Thresholding //

for i=N+1:2*N
for j=(N+2)/2:N
if abs(wt2(i,j)) < thold
wt2(i,j) = 0;
end;
wt2(i,j) = wt2(i,j) - thold*sign(wt2(i,j));
end;
end;

for i=N/2+1:2*N
for j=N+1:2*N
if abs(wt2(i,j)) < thold
wt2(i,j) = 0;
end;
wt2(i,j) = wt2(i,j) - thold*sign(wt2(i,j));
end;
end;

// 1/4 Piece Soft Thresholding //

for i=N+1:2*N
for j=N+1:2*N
if abs(wt2(i,j)) < thold
wt2(i,j) = 0;
end;
wt2(i,j) = wt2(i,j) - thold*sign(wt2(i,j));
end;
end;

// 3/4 Piece Soft Thresholding //
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
for i=N+1:2*N
for j=1:N
if abs(wt2(i,j)) < thold
wt2(i,j) = 0;
end;
wt2(i,j) = wt2(i,j) - thold*sign(wt2(i,j));
end;
end;

for i=1:2*N
for j=N+1:2*N
if abs(wt2(i,j)) < thold
wt2(i,j) = 0;
end;
wt2(i,j) = wt2(i,j) - thold*sign(wt2(i,j));
end;
end;

// 15/16 Piece Soft Thresholding //

for i=1:N/2
for j=N/2:2*N
if abs(wt2(i,j)) < thold
wt2(i,j) = 0;
end;
wt2(i,j) = wt2(i,j) - thold*sign(wt2(i,j));
end;
end;

for i=N/2+1:2*N
for j=1:2*N
if abs(wt2(i,j)) < thold
wt2(i,j) = 0;
end;
wt2(i,j) = wt2(i,j) - thold*sign(wt2(i,j));
end;
end;

// 63/64 Piece Soft Thresholding //

for i=1:N/4
for j=N/4:2*N
if abs(wt2(i,j)) < thold
wt2(i,j) = 0;
end;
wt2(i,j) = wt2(i,j) - thold*sign(wt2(i,j));
end;
end;

for i=N/4+1:2*N
for j=1:2*N
if abs(wt2(i,j)) < thold
wt2(i,j) = 0;
end;
wt2(i,j) = wt2(i,j) - thold*sign(wt2(i,j));
end;
end;

// Normal Soft Thresholding //

wt2( abs(wt2) < thold ) = 0.0;
wt2 = wt2 - thold*sign(wt2);

// Now we have the Wavelet transform. Take the inverse transform //

for i=1:2*N
inv_wt1(:,i) = inverse_wavelet_transform( j_max, M, wt2(:,i) );
end;
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
for i=1:M
inv(i,:)  =  inverse_wavelet_transform( j_max, M, inv_wt1(i,:) );
end;

//Reverse log data

inv = expm1(inv);
original_dble = expm1(original_dble);
noisy_dble= expm1(noisy_dble);

figure, imshow(original_dble); // Show original image
figure, imshow(noisy_dble); // Show noisy image
figure, imshow(inv); // Show denoised image

// Calculate MSE, AD & Correlation Coeifficient //

mymeas = abs ( original_dble - inv );
mymeas1 = mymeas.*mymeas;
MSE = sum(mymeas1(:));
MSE = MSE/m1;
MSE = MSE/n1
RMSE = sqrt(MSE)
AD = sum(mymeas(:) );
AD = AD/m1;
AD = AD/n1
Corr2 = corr2(original_dble,inv)

// Calculate Image Quality Index //

function ret = Image_Quality_Index(f,g)
mean_f = mean(f(:));
mean_g = mean(g(:));
contrast_fg = mean((f - mean_f).*(g - mean_g));
contrast_f_sq = mean((f - mean_f).*(f - mean_f));
contrast_g_sq = mean((g - mean_g).*(g - mean_g));
ret = (4*mean_f*mean_g/(mean_f^2 + mean_g^2))
*(contrast_fg/(contrast_f_sq + contrast_g_sq));
endfunction;

IQI_Denoising_Original = Image_Quality_Index(inv,original_dble)
IQI_Denoising_Noisy= Image_Quality_Index(inv,noisy_dble)
IQI_Noisy_Original  = Image_Quality_Index(noisy_dble,original_dble)

// Calcualte SNR on uniform region of original image //

zz1 = inv( 1:38,  137:300 );
bg1 = original_dble( 1:38,  137:300 );
z1 = zz1.^2;
v1 = (zz1-bg1).^2;
snr1 = sum(z1(:))/sum(v1(:));

zz2 = inv( 164:215,  480:512 );
bg2 = original_dble( 164:215,  480:512 );
z2 = zz2.^2;
v2 = (zz2-bg2).^2;
snr2 = sum(z2(:))/sum(v2(:));

zz3 = inv( 480:512,  230:258 );
bg3 = original_dble( 480:512,  230:258 );
z3 = zz3.^2;
v3 = (zz3-bg3).^2;
snr3 = sum(z3(:))/sum(v3(:));

snrvec = [ snr1 snr2 snr3];
SNR = max( snrvec )
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

A.9 Code for the Daubechies Wavelet Method with Hard Thresholding

```
pkg load image; // Load package image
pkg load ltfat; // Load packe The Large Time-Frequency Analysis

//Set up constant
thold = 0.15 // Thresholding value
j_max = 8 // should be <= m;

// Read original image
ref_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512.bmp');

// Read noisy image
noisy_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512_noise_rayleigh_.bmp');

original_dble = log1p(im2double(original_int)); // Log transform applied
%%original_dble = im2double(original_int); // Convert image to double precision
noisy_dble = log1p(im2double(noisy_int));
%%noisy_dble = im2double(noisy_int);
[m1,n1]=size(noisy_int);

// Wavelet Choose Daubechies 20 tap
WaveletType = 'db20';
// Take Wavelet Transform
wt = fwt2(noisy_dble,WaveletType,j_max);

// 1/2 Piece Hard Thresholding //

for i = m1/2+1:m1
for j = n1/4+1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
end;
end;

for i = m1/4+1:m1
for j = n1/2+1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
end;
end;

// 1/4 Piece Hard Thresholding //

for i = m1/2+1:m1
for j = n1/2+1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
end;
end;

// 3/4 Piece Hard Thresholding //

for i=m1/2+1:m1
for j=1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
end;
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
end;

for i=1:m1
for j=n1/2+1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
end;
end;

// 63/64 Piece Hard Thresholding //

for i=m1/8+1:m1
for j=1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
end;
end;

for i=1:m1
for j=n1/8+1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
end;
end;

// Normal Hard Thresholding //

wt( abs(wt) < thold ) = 0.0;

// Take the inverse wavelet transform //

inv = ifwt2(wt,WaveletType,j_max);

// Reverse log data

inv = expm1(inv);
original_dble = expm1(original_dble);
noisy_dble= expm1(noisy_dble);

figure, imshow(original_dble); // Show original image
figure, imshow(noisy_dble); // Show noisy image
figure, imshow(inv); // Show denoised image

// Calculate MSE, AD & Correlation Coefficient //

mymeas = abs ( original_dble - inv );
mymeas1 = mymeas.*mymeas;
MSE = sum(mymeas1(:));
MSE = MSE/m1;
MSE = MSE/n1
RMSE = sqrt(MSE)
AD = sum(mymeas(:) );
AD = AD/m1;
AD = AD/n1
Corr2 = corr2(original_dble,inv)

// Calculate Image Quality Index //

function ret = Image_Quality_Index(f,g)
mean_f = mean(f(:));
mean_g = mean(g(:));
contrast_fg = mean((f - mean_f).*(g - mean_g));
contrast_f_sq = mean((f - mean_f).*(f - mean_f));
contrast_g_sq = mean((g - mean_g).*(g - mean_g));
ret = (4*mean_f*mean_g/(mean_f^2 + mean_g^2))
*(contrast_fg/(contrast_f_sq + contrast_g_sq));
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
endfunction;

IQI_Denoising_Original = Image_Quality_Index(inv,original_dble)
IQI_Denoising_Noisy= Image_Quality_Index(inv,noisy_dble)
IQI_Noisy_Original = Image_Quality_Index(noisy_dble,original_dble)

// Calculate SNR on uniform region of original image //

zz1 = inv( 1:38,  137:300 );
bg1 = original_dble( 1:38,  137:300 );
z1 = zz1.^2;
v1 = (zz1-bg1).^2;
snr1 = sum(z1(:))/sum(v1(:));

zz2 = inv( 164:215,  480:512 );
bg2 = original_dble( 164:215,  480:512 );
z2 = zz2.^2;
v2 = (zz2-bg2).^2;
snr2 = sum(z2(:))/sum(v2(:));


zz3 = inv( 480:512,  230:258 );
bg3 = original_dble( 480:512,  230:258 );
z3 = zz3.^2;
v3 = (zz3-bg3).^2;
snr3 = sum(z3(:))/sum(v3(:));

snrvec = [ snr1 snr2 snr3];
SNR = max( snrvec )
```

A.10 Code for the Daubechies Wavelet Method with Soft Thresholding

```
pkg load image; // Load package image
pkg load ltfat; // Load packe The Large Time-Frequency Analysis

//Set up constant
thold = 0.15 // Thresholding value
j_max = 8 // should be <= m;

// Read original image
ref_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512.bmp');

// Read noisy image
noisy_int = imread('C:/Users/xiangeh/Desktop
/Denoising␣Speckle␣Noise␣in␣the␣image
/lena512_noise_rayleigh_.bmp');

original_dble = log1p(im2double(original_int)); // Log transform applied
%%original_dble = im2double(original_int); // Convert image to double precision
noisy_dble = log1p(im2double(noisy_int));
%%noisy_dble = im2double(noisy_int);
[m1,n1]=size(noisy_int);

// Wavelet Choose Daubechies 20 tap
WaveletType = 'db20';
// Take Wavelet Transform
wt = fwt2(noisy_dble,WaveletType,j_max);

// 1/2 Piece Soft Thresholding //

for i = m1/2+1:m1
for j = n1/4+1:n1
if abs(wt(i,j)) < thold
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
wt(i,j) = 0;
end;
wt(i,j) = wt(i,j) - thold*sign(wt(i,j));
end;
end;

for i = m1/4+1:m1
for j = n1/2+1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
wt(i,j) = wt(i,j) - thold*sign(wt(i,j));
end;
end;

// 1/4 Piece Soft Thresholding //

for i = m1/2+1:m1
for j = n1/2+1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
wt(i,j) = wt(i,j) - thold*sign(wt(i,j));
end;
end;

// 3/4 Piece Soft Thresholding //

for i=m1/2+1:m1
for j=1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
wt(i,j) = wt(i,j) - thold*sign(wt(i,j));
end;
end;

for i=1:m1
for j=n1/2+1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
wt(i,j) = wt(i,j) - thold*sign(wt(i,j));
end;
%%end;

// 63/64 Piece Soft Thresholding //

for i=m1/8+1:m1
for j=1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
wt(i,j) = wt(i,j) - thold*sign(wt(i,j));
end;
end;

for i=1:m1
for j=n1/8+1:n1
if abs(wt(i,j)) < thold
wt(i,j) = 0;
end;
wt(i,j) = wt(i,j) - thold*sign(wt(i,j));
end;
end;

// Normal Soft Thresholding //

wt( abs(wt) < thold ) = 0.0;
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

```
wt = wt - thold*sign(wt);

// Take the inverse wavelet transform //

inv = ifwt2(wt,WaveletType,j_max);

// Reverse log data

inv = expm1(inv);
original_dble = expm1(original_dble);
noisy_dble= expm1(noisy_dble);

figure, imshow(original_dble); // Show original image
figure, imshow(noisy_dble); // Show noisy image
figure, imshow(inv); // Show denoised image

// Calculate MSE, AD & Correlation Coefficient //

mymeas = abs ( original_dble - inv );
mymeas1 = mymeas.*mymeas;
MSE = sum(mymeas1(:));
MSE = MSE/m1;
MSE = MSE/n1
RMSE = sqrt(MSE)
AD = sum(mymeas(:) );
AD = AD/m1;
AD = AD/n1
Corr2 = corr2(original_dble,inv)

// Calculate Image Quality Index //

function ret = Image_Quality_Index(f,g)
mean_f = mean(f(:));
mean_g = mean(g(:));
contrast_fg = mean((f - mean_f).*(g - mean_g));
contrast_f_sq = mean((f - mean_f).*(f - mean_f));
contrast_g_sq = mean((g - mean_g).*(g - mean_g));
ret = (4*mean_f*mean_g/(mean_f^2 + mean_g^2))
*(contrast_fg/(contrast_f_sq + contrast_g_sq));
endfunction;

IQI_Denoising_Original = Image_Quality_Index(inv,original_dble)
IQI_Denoising_Noisy= Image_Quality_Index(inv,noisy_dble)
IQI_Noisy_Original = Image_Quality_Index(noisy_dble,original_dble)

// Calculate SNR on uniform region of original image //

zz1 = inv( 1:38,  137:300 );
bg1 = original_dble( 1:38,  137:300 );
z1 = zz1.^2;
v1 = (zz1-bg1).^2;
snr1 = sum(z1(:))/sum(v1(:));

zz2 = inv( 164:215,  480:512 );
bg2 = original_dble( 164:215,  480:512 );
z2 = zz2.^2;
v2 = (zz2-bg2).^2;
snr2 = sum(z2(:))/sum(v2(:));


zz3 = inv( 480:512,  230:258 );
bg3 = original_dble( 480:512,  230:258 );
z3 = zz3.^2;
v3 = (zz3-bg3).^2;
snr3 = sum(z3(:))/sum(v3(:));

snrvec = [ snr1 snr2 snr3];
SNR = max( snrvec )
```

มหาวิทยาลัยเทคโนโลยีสุรนารี

CURRICULUM VITAE

NAME: Thanakorn Piyapai **GENDER:** Male

NATIONALITY: Thai **MARITAL STATUS:** Single

DATE OF BIRTH: April 13, 1992

EDUCATIONAL BACKGROUND:

- M.Sc. in Applied Mathematics (in progress), Suranaree University of Technology, Nakhon Ratchasima, Thailand, 2019.
- B.Sc. in Physics, Khon Kaen University, Khon Kaen, Thailand, 2014.

RESEARCH INTERESTS:

- Ultrasound Speckle Noise Reduction Using Wavelet Methods.
- Studying Magnetization with the Ising Model.
- Brownian Motion.

ATTENDED:

- The 9th Conference on Science and Technology for Youths, Bangkok International Trade & Exhibition Centre: BITEC, Bangkok, Thailand, 30 May - 1 June, 2014.

SKILLS & INTERESTS :

Languages: TOEIC Listening & Reading Score 515 (Lower Intermediate Level).

Computer skills: C Programming, Octave and PTC Mathcad.

Interests: Advanced Mathematics, Systematic Trading, Cryptocurrency, Value Investing in Stocks and World Economy.

มหาวิทยาลัยเทคโนโลยีสุรนารี