MEDIAN FILTERING AND WAVELET FILTERING: A STUDY OF NOISE REDUCTION IN PROSTATE ULTRASOUND IMAGES

THESIS

Presented to the Graduate Council of

Southwest Texas State University

in Partial Fulfillment of

the Requirements

For the Degree of

Master of Science

By

Qing Liu, B.S.

August 7, 1998

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to Dr. David E. Pitts, adjunct professor, University of Houston - Clear Lake, and former Chief, Flight Science Branch in the Solar System Exploration Division of the NASA Johnson Space Center, whose support, encouragement, and dedication were invaluable in the completion of my thesis. Dr. Pitts brought me a real world topic - noise reduction in prostate ultrasound images for my thesis. Without his suggestions, advice, providing the data and the programs, this thesis would never be completed. I am thankful for his serving as an external committee member.

I also wish to thank Dr. John Durrett who serves as my thesis committee chair. His kindness and guidance are always very helpful for going through all kinds of requirements. I am grateful to Dr. Kaikhah and Dr. Hall for serving on the committee.

Another person I want to thank is Dr. Glen A. Houston, Chair, Division of Computing and Mathematics in University of Houston - Clear Lake for his valuable suggestions.

Above all, I wish to express my appreciation to my husband, Ganyuan Xia, for his love and technical support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1. 1 Objective

The objective of this study is to find an effective way to remove the noise in prostate ultrasound images by using different image enhancement techniques. In this thesis two major techniques are carefully studied. In Chapter 2, median filtering and three other advanced median filtering techniques are used to suppress the noise. In Chapter 3, wavelet filtering technique is used to suppress the noise. Different noise reduction techniques are compared, contrasted and analyzed in this thesis. It is concluded by finding a promising approach.

1. 2 Prostate Cancer

Prostate cancer has been the most commonly diagnosed cancer among males in the United States for the past several decades. It is the second leading cause of cancer death only behind lung cancer. According to the American Cancer Society, in 1997 alone there were 184,500 new cases of prostate cancer, and 39,200 deaths related to prostate cancer. Over the past ten years, about 2,000,000 males have developed prostate cancer.

No one knows the exact cause of prostate cancer, however researchers in this field have found that certain factors are linked to an increased risk of prostate cancer. The

incidence of the disease increases with age, with more than 80 percent of all prostate cancers being diagnosed in men over age 65. African-American men and men with a family history of the disease may also be at increased risk. Studies suggest that high dietary fat intake may also play a role in prostate cancer development (Memorial Sloan-Kettering Cancer Center, 1998).

Prostate cancer arises from a group of abnormal cells within the prostate that grow out of control and may invade and destroy nearby healthy tissues or spread through the blood stream and lymphatic systems to other parts of the body forming new tumors (Jayawardena, 1997).

Most patients experience a very slow growing rate of the cancer. Generally 85% of the patients survive for an average of five years. However, it is the second leading cause of cancer death behind lung cancer, because the cancer spread very quickly. Also younger patients experience a faster growing rate of the cancer and therefore are at greater risk.

Symptoms of prostate cancer include frequent urination or an inability to urinate, trouble starting or holding back urine, and frequent pain or stiffness in the lower back, hips or upper thighs. However, these symptoms are also seen with a common, non-cancerous condition called benign prostatic hyperplasia (BPH), which is an enlargement of the prostate gland. BPH may or may not cause the cancer. It is important to seek medical attention for any of these symptoms to ensure proper diagnosis and treatment.

The prostate is the male reproductive gland, which is about the size of a walnut for a young adult. It grows slightly larger with age. Usually a normal prostate weighs about an ounce. Physically it is located just below the bladder and above the rectum. Two important functions of the prostate are bladder control and sexual functioning. The prostate produces semen which helps the sperm flow from the testicles (Houston et al, 1992).

Because the exact cause of prostate cancer is still unknown today, there is no sure way to prevent prostate cancer. However, some studies suggest that a low-fat diet can reduce prostate cancer risk. Most physicians recommend that every male of age forty or above should have a digital rectal examination and a PSA (Prostate Specific Antigen) blood test as part of an annual physical checkup to detect any problems with the prostate gland. If there is a suspicion of cancer, a prostate ultrasound examination may be used for further examination.

1. 3 Prostate Ultrasound Examination

In order to allow the urologist to closely examine the prostate gland for abnormalities, a transrectal ultrasound exam is performed. Sometimes a transrectal ultrasound image of the prostate is also used to diagnose the cause of male infertility. Usually, the urologist obtains biopsies of the prostate gland if cancer is suspected.

A prostate exam with or without a biopsy takes between fifteen and twenty-five minutes (Memorial Sloan-Kettering, 1998). The exam is performed by using an

ultrasound transducer (usually called a "camera") about the size of an index finger. The transducer is placed into the rectum, and the prostate gland is scanned.

According to the Memorial Sloan-Kettering Cancer Center, before a patient takes a transrectal prostate exam, he should take a *fleet* enema. If a biopsy is to be performed, as it usually will be, the doctor prescribes some antibiotics such as Cipro. One tablet should be taken before the biopsy, the morning of the biopsy, that evening after the exam and finally the following morning.

In an ultrasound examination room, the patient is asked to lie on a table on his side with his knees bent after changing to hospital garb. The urologist carefully inserts a transducer into the rectum. Then images and measurements of the prostate are taken.

Live ultrasound images are viewed by the urologist in order to choose the region for biopsy. During the procedure the urologist takes about six to twelve biopsies (Memorial Sloan-Kettering, 1998). A small, shallow needle is rapidly inserted into the prostate gland. Then tissue sample is collected and sent to the pathology department for biopsy. It is hoped that the improved noise reduction techniques developed here will lead to better identification of suspected lesions by the physician so that the biopsy procedure can be made more efficient and accurate.

Since the prostate has limited sensitivity to needle insertion, patients usually do not feel much pain. Some doctors say that the vast majority of men tolerate the procedure very well. Typically the anxiety they experience prior to the biopsy is considerably worse than the biopsy itself.

After the exam, a patient may experience some mucous or a small amount of bleeding in the urine and from his rectum for up to 48 hours. Blood in the semen is also common. However, these symptoms are not reasons for concern, if the amount of blood is small.

1. 4 Prostate Ultrasound Imaging

An ultrasound exam is a safe diagnostic procedure that uses very high frequency sound waves (ultrasound) to produce images of many of the internal structures of the body. Originally developed in the late 1950's, the machine has improved over the years in terms of resolution and signal to noise ratio. Even though ultrasound images are not nearly as clear as CT (Computerized Tomography) or MRI (Magnetic Resonance Imaging), they can not be replaced by CT or MRI for the following reasons. First, neither CT nor MRI can produce real time images like the ultrasound video, and therefore cannot be used to guide the biopsy needle. Second, ultrasound waves are harmless, and can be used with complete safety, even on pregnant women, whereas CT or X-rays would be inappropriate due to possible harm to the fetus or other soft tissues. Lastly, CT and MRI can not be routinely used for prostate imaging because they are very expensive.

Ultrasound sonography has become one of the most researched methods in detecting tumors and abnormalities in the prostate gland (Abeysekera, 1995). The transabdominal and transrectal sonography are two common methods of ultrasound prostate examination. The transabdominal method requires the patient to drink about thirty ounces of water to fill the bladder for better visualization. Then the urologist or a

technician moves the transducer to the area where the gland is located for clear images. Ultrasound is also used for echo cardiograms, cartoid artery, fetal exams and a variety of other applications.

Transrectal sonography was developed about thirty years ago. The transrectal method of scanning the prostate gland is different from the transabdominal method. In transrectal scanning, it is important for the transducer to be in close proximity to the gland and this is only feasible by inserting a transducer into the patient's rectum. The transrectal method was not popular initially, but it gained popularity over the years because of its better image visualization due to the advancement of ultrasound techniques. Currently, the Bruel and Kjaer (B&K) model 1846 machine is the most widely used transrectal sonography machine for prostate ultrasound images in the United States.

The B & K machine with a multi-frequency bi-planar probe can be used to obtain two kinds of images, transaxial and traverse biopsied images. The bi-planar probe is designed to change from the axial to the transverse image plane by simply turning a knob. The ultrasound data for this study was produced using a 7 MHz frequency signal. All images are first recorded on 3/4 inch video tapes in analog form. Then they are digitized by converting to 8-bit digital images. This is done at the NASA Johnson Space Center by synchronizing the real time disk with a time base corrector and all the image frames are captured on the real time disk (Jayawardena, 1997). The resulting images have 512x512 pixels with 8-bits per pixel giving gray scale levels of 0 to 255. Figure 1.1 shows such an example of prostate ultrasound image. The prostate gland is outlined in yellow oval line.

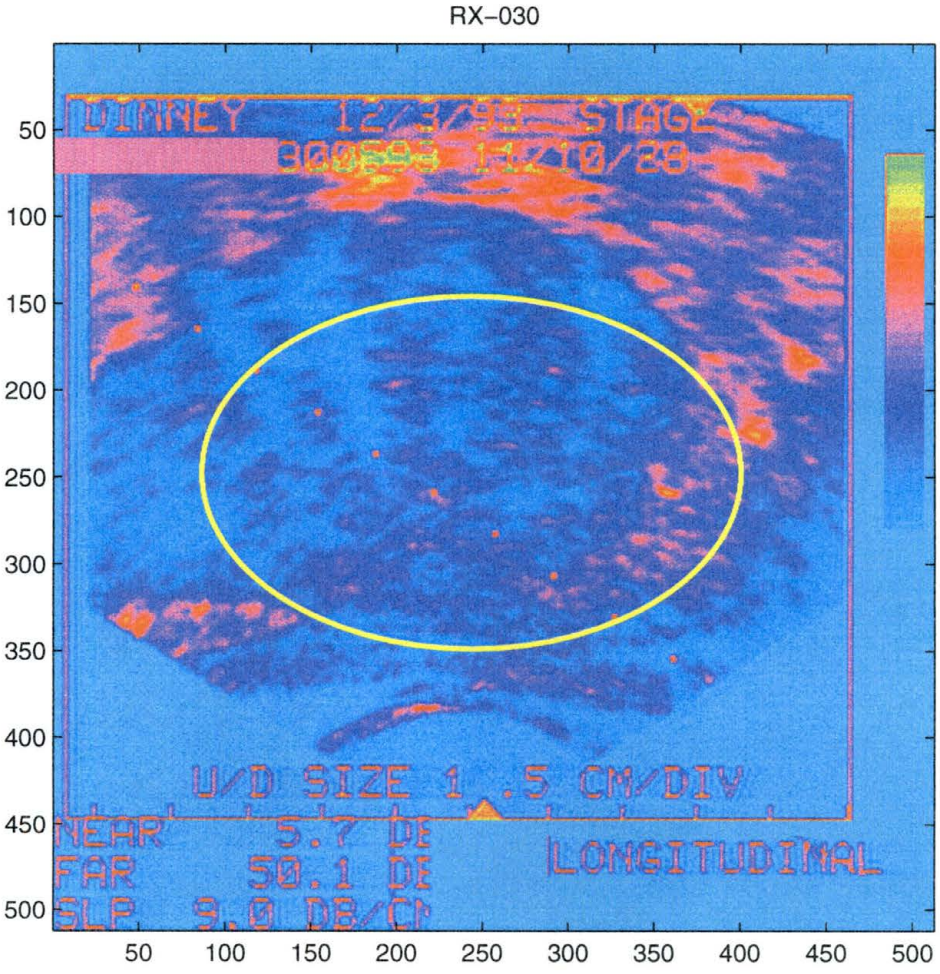

Figure 1.1 A prostate ultrasound image.

The line of red[1] dots on the image from the lower right to the upper left represents the template that shows the path of biopsy needles which are inserted from the lower right of the image toward the center of the gland.

The prostate gland can be divided into three zones: peripheral, transition, and central. The cancer cells must develop to a certain degree to cause tissue changes in order to be detected using the ultrasound imaging technique. Early investigators believed that in an ultrasound image the prostate cancer lesions appeared hyperechoic (bright), but in 1985 Lee and associates (Houston, 1995) proposed that hypoechoic (dark) regions in the prostate ultrasound image were the suspected cancerous regions. It has been shown that only 20% of the hypoechoic lesions in the prostate are cancerous (Houston, 1995). The peripheral zone is believed to be the most susceptible of the prostate zones to cancer. It is estimated that 80% of the cancers originate in the peripheral zone (Houston, 1995). Because the peripheral zone is closest to the rectum, it is most easily viewed and biopsied.

During the prostate ultrasound examination, a radiologist or an urologist will first select a peripheral region for biopsy. Usually there are three images taken for analysis: one taken in the axial view of the biopsy region, prior to the biopsy; one taken in the transverse view at the same location; and one taken in the axial view while the needle is

---

[1] The red color is the pseudo color. All ultrasound images are originally in black and white. Here all the colors are the pseudo colors which do not add any information in an image but improve visualization.

in the gland (Jayawardena, 1997). All three image views together provide more information for physicians to evaluate and diagnose.

1. 5 The Data

The data in this study are from the M.D. Anderson Cancer Center. There are two groups of data. One is the Phantom data. The other is the real data from actual patients. All the data are in "pic" format, which has a 512 byte header followed by 512x512 integers with 266 gray levels.

The phantom mimics the acoustic properties of human tissues and provides test structures within the simulated environment. Moreover, the phantom is useful to detect performance changes that occur through normal aging and deterioration of the ultrasound system components (Abeysekera, 1995). The phantom is a right circular cylinder containing water and several homogenous circular regions with known ultrasound characteristics which mimic various tissues in the prostate. The several circular regions are called targets (see Figure 1. 2). When collecting a phantom image, two targets (one

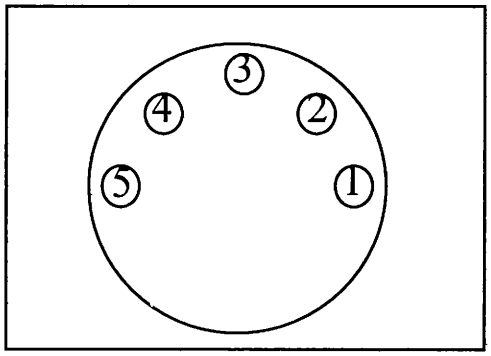

Figure 1.2 The targets of phantom data.

pair) are imaged at a time starting from the most left targets 5 and 4. Then the rotated probe is one target clockwise for the next recording. A complete set of phantom images is the series of targets as: 5 and 4, 4 and 3, 3 and 2, and 2 and 1.

Figure 1.3 shows the example of phantom data: P22157.pic. All phantom data follow the naming conventions as explained below. P2, the first two characters, indicates this is the second phantom experiment. The next sequence of two characters following P2, 2 and 1, indicate the known targets in this scene. The last two digits 5 and 7 represent the gain of 5.7 that is applied to the image. The notation P2LRDD is used where L is the left target, R is the right target, and DD indicates the gain of D.D except when DD is 12, in which case DD is 12.0. Take P25415 as another example, P25415 would indicate phantom experiment 2 on targets 5 and 4 at the gain of 1.5.

There are seven different gains used in the test. They are 1.5, 4.8, 5.4, 6.0, 6.6, 9.9, and 12.0. Each gain setting has five different targets 1, 2, 3, 4, and 5. Gain settings with a larger number like 9.9 are brighter and have greater contrast than images with smaller gains such as 1.5.

The phantom images used for this thesis are categorized into three groups according to their gain. Each group has 4 images, corresponding to a successful round of scanning with two targets in the scene each time. The total of 12 phantom images used in this study are listed as follows:

(1)	P25415.pic	P24315.pic	P23215.pic	P22115.pic
(2)	P25457.pic	P24357.pic	P23257.pic	P22157.pic
(3)	P25412.pic	P24312.pic	P23212.pic	P22112.pic

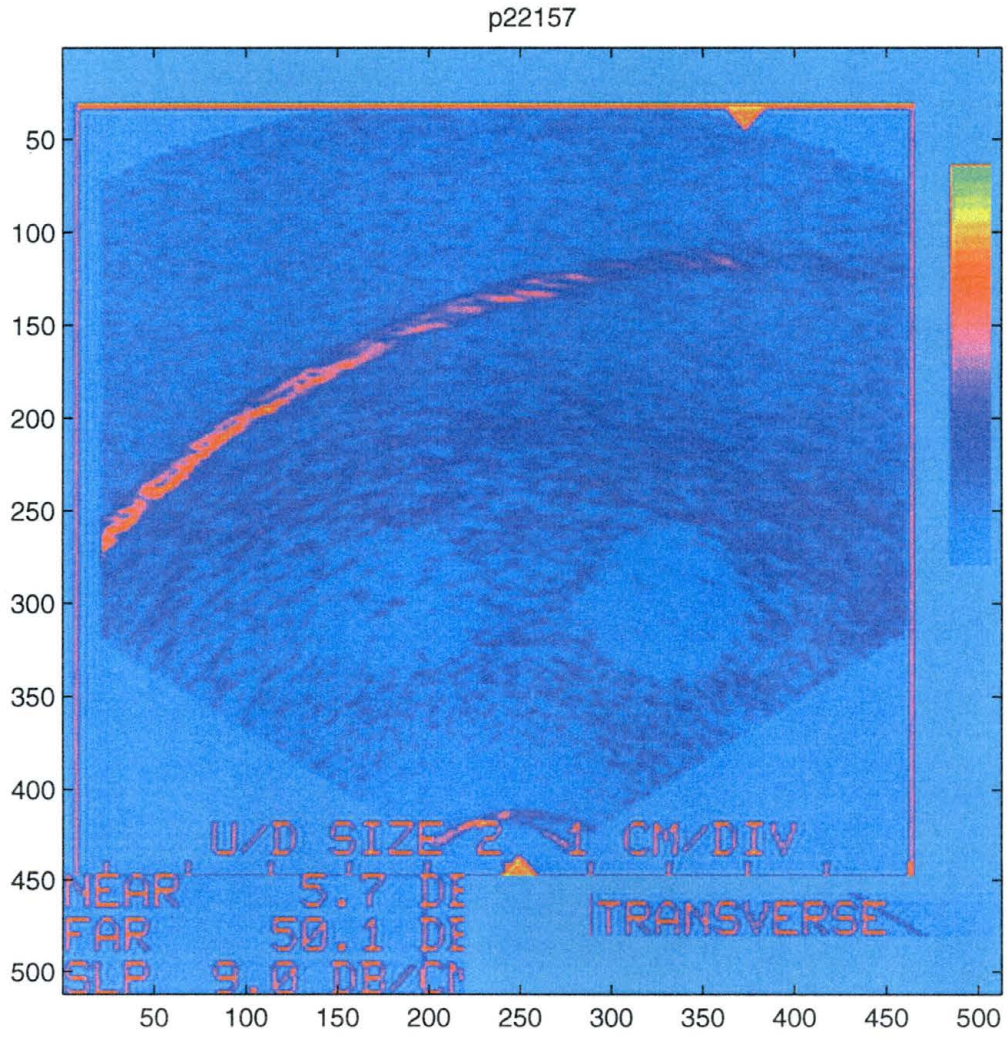

Figure 1.3. A phantom image.

The three gain settings are chosen to cover the maximum, minimum, and moderate gains for all 5 targets.

The ultrasound sensing system produces a coherent signal and through multiple reflections in the media can produce 1) destructive interference, 2) constructive interference, and 3) normal reflectance. This interference causes the noise called speckle. The phantom data are used for testing the effectiveness of different median filters and wavelet filters in reducing the noise in the ultrasound images. All the statistical analysis in this study is based on the results from phantom data because its targets give us control and chance to compare, contrast and analyze the results from different filtering techniques using known constant targets.

An example of real data (RX-031.pic) is shown in Figure 1.4. These kinds of data were collected involving a human patient. The file name involving human subjects does not include any encoding of the gain setting. The most effective denoising technique is applied to three real ultrasound images, RX-001.pic, RX-030.pic, and RX-031.pic, to show how its effectiveness in removing noise.

1. 6 Noise Reduction in Prostate Ultrasound Imagery

1. 6. 1 The Background of Image Enhancement

Digital image processing is a fascinating subject. Human beings perceive most information about our environment through the visual sense. For a long time, images could only be captured by photography, we can now capture, manipulate, and

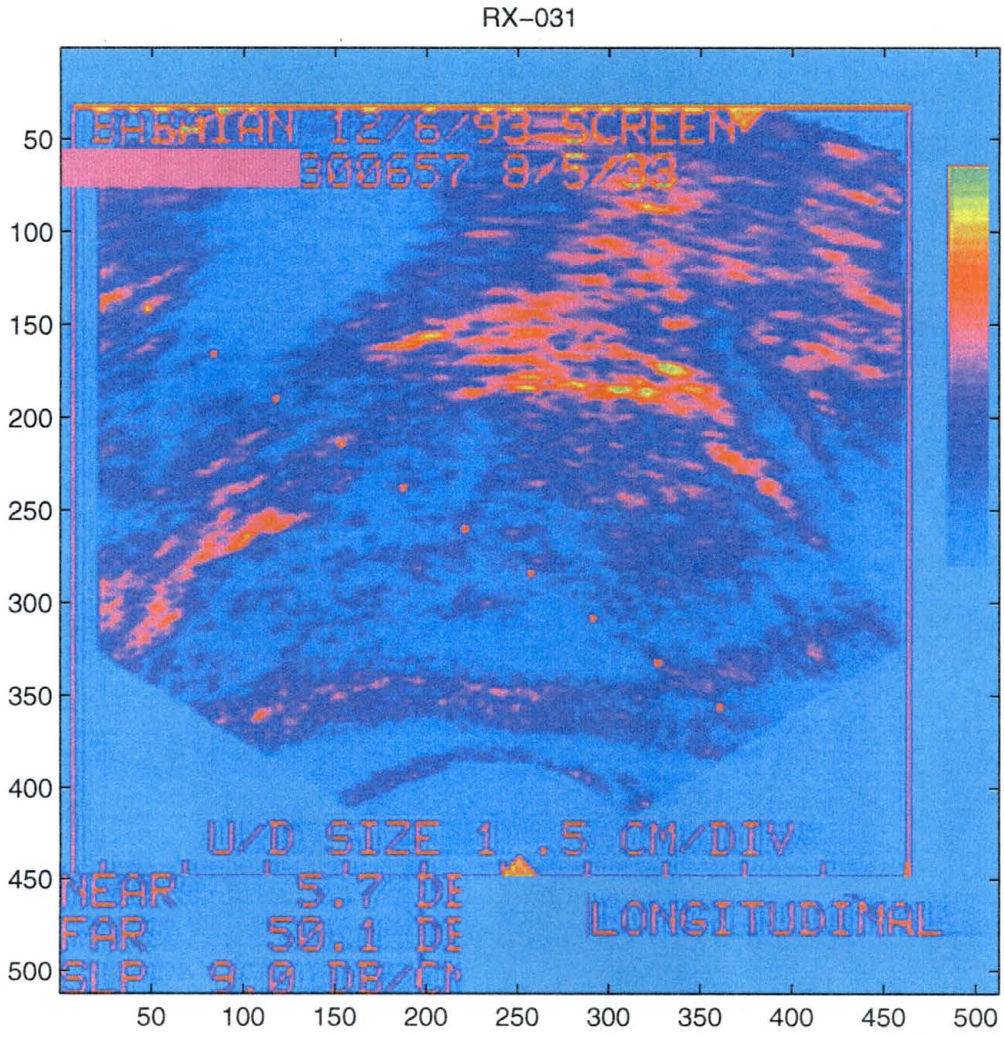

Figure 1.4. A real prostate ultrasound image.

evaluate images using computers. Image enhancement has become very important in the field of image processing because the processed result is more suitable than the original image for our specific purposes. There are two approaches to image enhancement, spatial-domain methods and frequency-domain methods.

The spatial-domain refers to the image plane itself, and those techniques are based on direct manipulation of pixels in an image (Gonzalez and Woods, 1995). According to Gonzalez and Wood (1995), image processing functions in the spatial domain may be expressed as:

$$g(x, y) = T[f(x, y)] \qquad (1.1)$$

where $f(x, y)$ is the input image, $g(x, y)$ is the processed image, and T is an operator on f, defined over some neighborhood of (x, y). In addition T can also operate on a set of input images, performing the pixel-by-pixel sum of M images for noise reduction.

The following two figures (Figure 1.5 and Figure 1.6) represent two ways to count a neighborhood, a square shaped neighborhood and a plus shaped neighborhood.

For each pixel in an image, the neighborhood is defined as a group of pixels surrounding the center pixel. Figure 1.5 illustrates the square shaped neighborhood which includes the 8 pixels in a square around pixel *a*. Figure 1.6 shows the plus shaped neighborhood which includes the 4 pixels in a plus shape around pixel *a*.

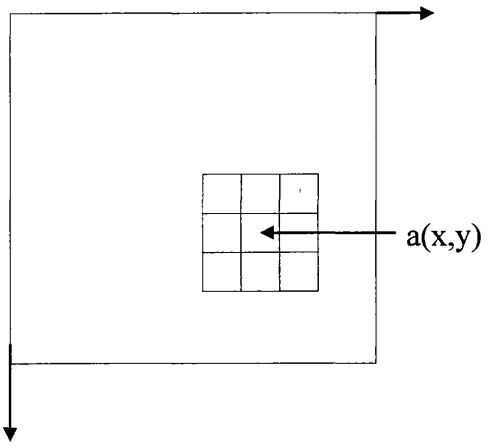

Figure 1.5. Square shaped neighborhood.

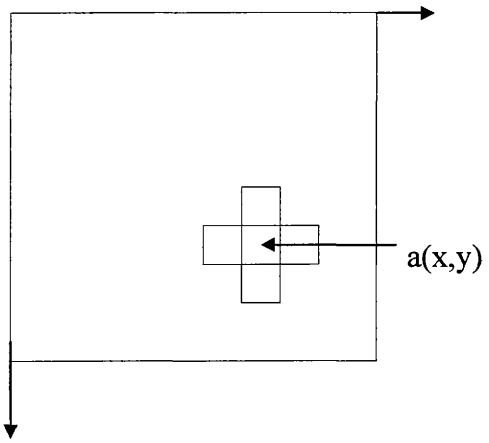

Figure 1.6. Plus shaped neighborhood.

The neighborhood is often referred to as a mask (also called a window). The mask moves through the whole image one pixel at a time so that each pixel will in turn become the center of the mask for the operation. The result of each masking operation is placed in a new resultant image as it is important not to modify the original image, otherwise unpredictable nonlinear results would be produced.

The frequency-domain processing techniques are usually based on modifying the Fourier transform of an image. According to Gonzales and Woods (1995), the Fourier transform equation can be expressed as the following:

$$\xi\{f(x)\} = F(u) = \int_{-\infty}^{\infty} F(x) \exp[-j\,\pi 2ux]dx \qquad (1.2)$$

where $f(x)$ is a continuous function of a real variable x. The Fourier transform of $f(x)$ is denoted $\xi\{f(x)\}$ (where $j = \sqrt{-1}$). So enhancement in frequency domain involves: (1) calculating Fourier transform of an image, (2) multiplying the result by a filter transfer function, (3) taking the inverse transform to produce the enhanced image.

1. 6. 2 The Noise Reduction Methods

Reducing noise and preserving details are the two main goals in the field of noise reduction. Obviously, there is a trade-off between the two. The challenge is to optimally reduce noise while preserving details.

Conventional filtering techniques do not give images that are much better than the raw images. In practice, physicians usually rely on raw images or use the images filtered by standard filters such as lowpass spatial filters (Tham, 1998).

The lowpass spatial filter, also called neighborhood averaging, is a smoothing filter. The lowpass filter is often used for blurring and noise reduction. Blurring may be an important preprocessing step for removing small details from an image prior to (large) object extraction, and bridging of small gaps in lines or curves.

The lowpass filtering method is to change an image by manipulating each moving mask which involves averaging every pixel value (gray level) in a mask and use the average value to replace the center pixel value in the mask. The coefficients for each pixel in the mask may be set by the user. For lowpass filtering, all the coefficients in a mask must be positive. The simplest arrangement would be a mask in which all coefficients have a value of 1. Then the result is the mean of the individual pixels in the mask. Therefore, the whole image is smoothed.

Figure 1.7 illustrates the effect of the lowpass filtering on image P22157.pic (see Figure 1.3). The mask is of the size of 5x5 with all the coefficients set at 1. The resultant image after applying lowpass filtering is indeed smoothed, and the noise is spread out.

By its averaging nature, the low pass filtering method generally works well for removing random noise, but it is not very effective because the replacement of an average removes substantial details. In cases where there is coherent noise or edge preservation is desired, lowpass filtering may not work. Alternative methods, such as median filters and wavelet filters are studied in this thesis. Median filtering methods are studied in Chapter 2, and noise reduction using wavelets is explored in Chapter 3. Their performances are compared with statistical measures in Chapter 4.

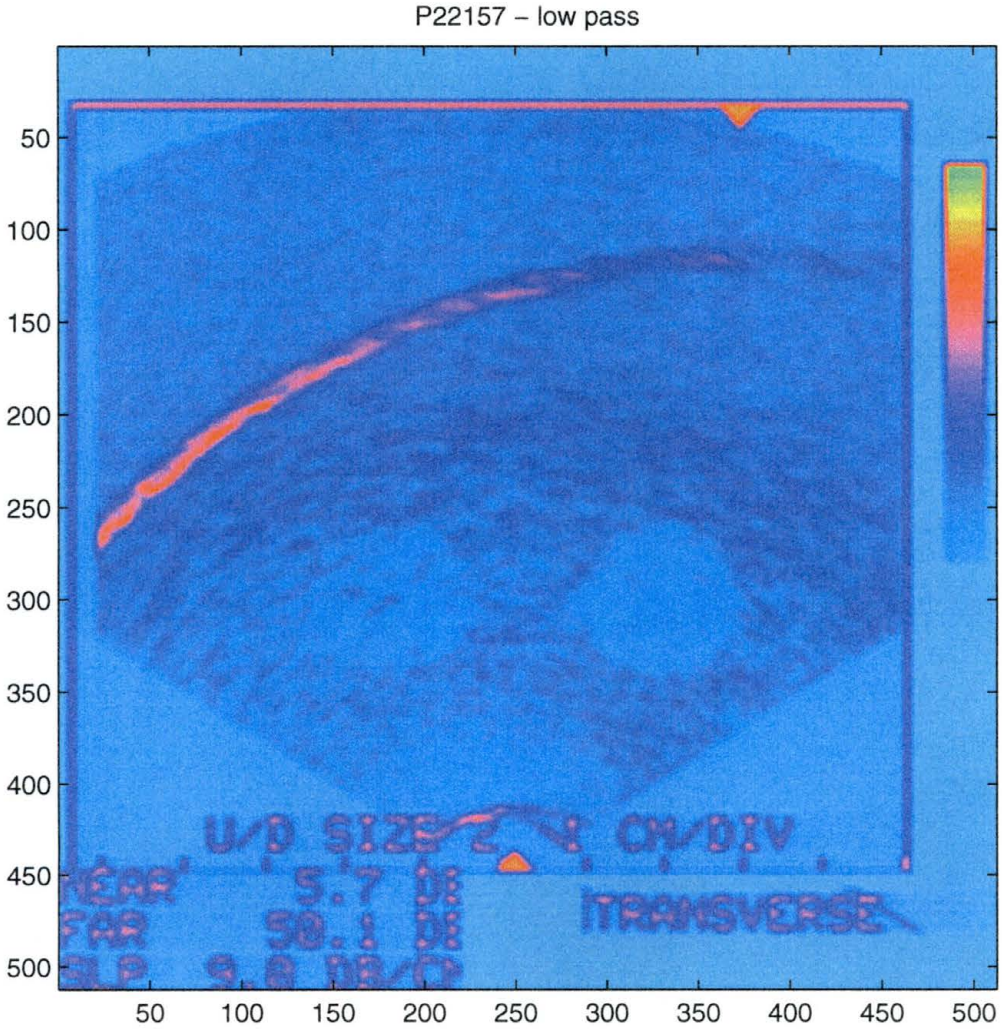

Figure 1.7. The result of lowpass filtering.

CHAPTER 2

MEDIAN FILTERING

We use prostate ultrasound because it is cheap, risk free, and relatively painless. But like all other ultrasound images, prostate ultrasound images are very noisy. The ultrasound images are noisy because of the interference. Thus the prostate ultrasound images with highly corrupted noise are not out of our expectation. As an real ultrasound image is shown in Figure 1.4, the noise percentage is extreme large. This type of noise is difficult to remove.

Traditional median filtering is very effective at removing spikelike noise, so the literature of various median filters is reviewed with the purpose to find an effective way to suppress the ultrasound noise. The results of applying various median filters on prostate ultrasound images are then presented.

Section 2.1 to Section 2.4 introduces the different median filters researched: (1) Median Filtering, (2) Threshold Control Method, (3) 2-D Quadratic Filtering, (4) Center Weighted Median (CWM) Filtering. Section 2.5 shows how effective those median filters are on prostate ultrasound images, the phantom data (see Section 1.1 for the description).

2. 1 Median Filtering

A median filter belongs to the spatial domain filter category. The method is to change an image by moving each mask throughout the image centering the mask over each pixel and then the gray level of each pixel is replaced by the median of the gray levels in a neighborhood of the pixel in each corresponding mask.

Median filtering is very effective at reducing high-frequency and impulsive noise in digital images without the extensive blurring and edge destruction associated with linear filters, such as a low-pass filter. The mask (window) size is chosen according to the noise level. Usually a small window (e.g. 3x3 or 5x5) is good enough for a small amount of noise while a large window is used for an image of higher level noise.

Figure 2.2 shows the effect of median filtering. Figure 2.1 is the well known image "***Lenna***" corrupted with salt and pepper additive noise. A 5x5 median filter effectively removes all the noise as shown Figure 2.2.

Figure 2.1. The image ***Lenna*** corrupted by salt-and-pepper noise.

Figure 2.2. The effect of median filtering of salt and pepper noise with window size of 5 by 5.

The Limitation: Though median filter is computationally efficient and suppresses impulsive noise effectively while preserving details, median filtering does have some limitations. Using larger windows (e.g.17x17) to remove larger clumps of noise causes substantial blurring of the image and this loss of detail defeats the purpose of the filtering (see Figure 2.3).

A median filter consists of a moving mask of odd length of 2N +1 (the mask with the size of L x L, where L= 2N+1) through a whole image. The center pixel in the mask is replaced by the median of the pixels within each moving mask of L x L pixels. Signal suppression using larger window median filters occurs because the replacement pixel can potentially come from data from greater distance from the center of the mask, so the replacement pixel could be from a different part of the scene. On the other hand, because it is not known which pixel is noise in an image, a pixel may be replaced by the corresponding median which includes a noise pixel in it. This is true, especially when a

large window is used, because this multiple applications of the median filter are often performed to remove noise.

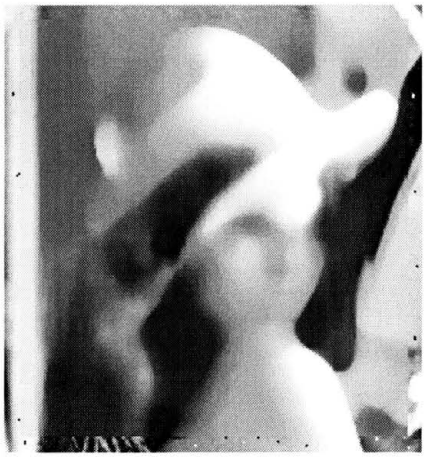

Figure 2.3. The result of median filtering of salt and pepper noise with window size of 17 by 17.

2.2 Threshold Control Method

Sometimes some distortion in the resultant images is still present after using median filtering. For example Figure 2.3 shows the results of applying a median filter with window size of 17x17 on the image "***Lenna***" with salt and pepper noise (Figure 2.1).

The threshold control method is a more effective scheme for efficient removal of impulse noise, while successfully preserving detail. It uses a detection estimation strategy in which a corrupted pixel is first detected, and then its true value is estimated based on local statistics. The threshold control is also called the "out-range method" which each pixel is sequentially examined, and if the pixel value is greater than the average value of its immediate neighbors by a pre-determined threshold value it is replaced by the average value; otherwise it is kept unchanged (Mitra et al, 1994).

There are three operations in the threshold control method (Mitra et al, 1994) for each pixel: computation of the local mean within a mask, computation of the difference between the original pixel value and its local mean, and comparison between the difference and a given threshold. The first operation can be implemented using a linear lowpass filter, and the second one can be viewed as a complementary lowpass filtering, i.e. , a linear highpass filtering. The first two operations are linear. Since the overall result depends on the comparison operation, the threshold control method is thus a nonlinear technique.

Figures 2.4, shows the effect of the threshold control method. The threshold is set to 10 here for the input image "***Lenna***" (see Figure 2.1) and yields Figure 2.4 which is as good as the resultant of median filtering (Figure 2.2) in this case.

Figure 2.4. The effect of threshold controlled median filtering with threshold of 10.

The Limitation: The threshold control method depends on the threshold used to detect a noise pixel. Since this is under the assumption of the noise being above the background by a given delta or a proportion, it is difficult to detect a noise pixel correctly

by the threshold. It is common that noise pixels could be located both on a white area (the brightest pixel value is 255) and a black area (the darkest pixel value is 0) in a corrupted image. Using the threshold may not work well for this case because a difference between a median and a center pixel (a noise pixel) in a white area is detected by the threshold while a difference between a median and a center pixel (a noise pixel) of a black area may not be detected. When the noise pattern is in such category, the threshold control method cannot outperform median filtering.

2.3 2-D Quadratic Filtering

The 2-D quadratic method was introduced by Mitra et al (1994). They found that threshold control can not effectively detect noise. They believed, for a fixed threshold, impulse noise cannot be efficiently detected simultaneously in both white and dark areas. They gave an example showing the threshold filter fails to detect a noise pixel when an impulse noise value is about 250 in an image with a dynamic range from 0 to 255. They further pointed out that based on the analysis of the threshold control method, to suppress impulse noise efficiently while successfully preserving the edges and details, a technique has to be developed to produce relatively similar results between a noisy value and a local mean with different kinds of image backgrounds. Since these characteristics can not be achieved using linear operations, Mitra et al (1994) suggested the development of some nonlinear operators as described in the following. In some cases, a fixed threshold cannot detect noise pixels. It is obvious that for an image area with a large local mean, the difference between a positive noise impulse value and the local mean is small, and for image area with a small local mean, the different is large. Mitra et al (1994) said it was

this fact that stimulated them to use the product of the difference and local mean to detect a positive noisy pixel. This kind of operation is called 2-D nonlinear operator.

The detailed steps of 2-D quadratic filtering are: (1) Check each pixel whether it is a noisy pixel by comparing the output of the proposed 2-D nonlinear operator with a given threshold, (2) Use the selective local mean, i. e. , replacing each noisy pixel value by the average value of the uncorrupted pixels within a given mask centered around that pixel. The 2-D quadratic method uses the local mean, not the median.

Teager-Kaiser's energy operator (Kaiser, 1990), a simple nonlinear operator has found many applications in signal and image processing. Mitra et al (1994) found it is sufficient to be the foundation of their 2-D quadratic method.

2.3.1 Teager-Kaiser's Algorithm

Mitra et al (1994) introduced the Teager-Kaiser's Algorithm as the following:

Let x(n) be the samples of a discrete sinusoidal signal,

$$x(n) = A \cos(\omega n + \phi), \qquad (2.1)$$

where A, ω, ϕ and n are the amplitude, frequency, the initial phase of the signal and sample index. The following Eqn. (2.2) is obtained after some algebraic manipulations:

$$E_n = x^2(n) - x(n+1)x(n-1) = A^2(n)\omega^2(n), \qquad (2.2)$$

where En denotes the energy of the oscillation system. Since this energy operator involves only three adjacent samples, the 2-D implementation of Teager's algorithm can

be limited to processing over a 3x3 mask. There are two possible versions of such extensions (Yu et al, 1991). One is to process a 2-D array into horizontal and vertical directions, in which the algorithm can be written as

$$y(m,n) = 2x^2(n) - x(m+1, n)x(m-1,n) - x(m, n+1)x(m, n-1), \quad (2.3)$$

where m and n are index numbers, y(m,n) it the output of the proposed 2-D nonlinear operator and x(m,n) is the input. It should be noted that y(m,n) has no relation with energy. However, it has been found that

$$y(m,n) \cong s(m,n)d(m,n), \quad (2.4)$$

where

$$s(m,n) = 0.2[x(m,n) + x(m+1,n) + x(m-1,n) + x(m,n-1) + x(m,n+1)], \quad (2.5)$$

and

$$d(m,n) = x(m,n) - s(m,n). \quad (2.6)$$

Therefore, Mitra et al (1994) concluded that their proposed 2-D nonlinear operator can be used for the detection of the noisy pixel corrupted by a positive impulse. They also claimed that the proposed operator can also be used to detect negative noise impulses by complementing the input image.

Figure 2.5 and Figure 2.6 show the effectiveness of 2-D quadratic filtering. Figure 2.5 is the image Lenna with the noise percentage between 10% and 20%. Figure

2.6 shows the filtered image after applying 2-D quadratic filter. The 2-D quadratic filter works well for the highly corrupted image.

Figure 2.5. A highly corrupted image.

Figure 2.6. The effect of 2-D quadratic filtering.

The Limitation: Even though there are claims that the 2-D quadratic can reduce noise and preserve details at the same time for a highly corrupted image, it cannot reduce noise in a highly corrupted image with noise level greater than 20%. Since the 2-D quadratic filter did not work well in Figure 2.6, this indicates that the filter can only work with a image with noise level between 10% to 20%. For a highly corrupted image, using

a product of the difference and local mean to detect a noisy pixel also depends on the window size which influences the local mean. That leads us back to the old problem: preserving details demands the use of a relatively small window, but a small window cannot reduce more noise. So the 2-D quadratic filter should only be applied to images with a noise level range between 10% - 20%. Therefore, the nature of a highly corrupted image suggests that a larger window median filter will work but at the expense of loss of signal information.

2.4 Center Weighted Median (CWM) Filtering

The Center Weighted Median (CWM) filter is a new kind of median filter developed by Ko and Lee (1991). It differs from the conventional median filter in that it gives more weight to the central value of a moving mask in median filtering operation.

CWM allows us to set the weights for each moving mask so that its smoothing behavior can be controlled. According to Ko and Lee (1991), this filter can preserve image details while suppressing additive white and /or impulsive-type noise. Their testing results show that the CWM filter can outperform the median filter.

The key to CWM is to properly set the center weights. Let the center weights be $2K + 1$, and the size of a moving mask be $2L + 1$. The size of a moving mask refers to total pixels in a mask. When $K = 0$, the CWM filter reduces to a regular median filter. When $2K + 1$ is greater than or equal to the size of a mask $2L + 1$, it becomes the identity filter. So K is in the range between 0 and L, and the ratio R of K and L is between 0 and 1. The relationship between K and L can be expressed as the following:

$$K(i, j) / L = R(i, j), \qquad (2.7)$$

where i, and j are the index in a moving mask. Therefore Eq. (2.7) can be expressed as:

$$K(i, j) = LR(i, j). \qquad (2.8)$$

Ko and Lee studied R(i, j), and found that there is great relationship between the variance of noise (σ_n^2) and the variance of the data (σ_x^2) within a mask related to R(i, j):

$$R(i, j) = \begin{cases} (\sigma^2_x(i,j) - \sigma^2_n) / \sigma^2_x(i,j), & \text{if } \sigma^2_x(i,j) \geq \sigma^2_n \\ 0, & \text{otherwise} \end{cases}$$

Therefore if the variance of the noise is greater than the variance of the data within a mask, R is 0 and consequently K is 0. The median filter is then used for the operation in order to make sure that the noise is removed. Otherwise, when the variance of the data is much greater than the variance of noise within a mask, R approaches 1 and this causes K to be set to L, and an identity filter is performed. This makes sense to remove noise by a median filter and leave the image alone where it is not corrupted.

The following are the figures to show the effect of CWM. Figure 2.7 is the input image with a high noise level. Figure 2.8 shows the result after applying the CWM filter.

The Limitation: CWM permits the center pixel weight to be set so that it can control when to use the median filter for noise removal and when to use the identity filter for preserving the signal information. There are two factors that affect the weight applied to the center pixel: the ratio between the variance of signal and noise, and the window size. The first factor depends on the relative noise level, while the second factor

Figure 2.7. A very noisy image.

Figure 2.8. The effect of CWM (window size 5 by 5).

affects signal preserving ability. In practice the noise level is unknown which makes it difficult to set accurate weights for the center pixel. In most cases, CWM can only reach the highest performance of a median filter.

2.5 Apply the Median Filters to Prostate Ultrasound Images (Phantom Data)

Section 2.1 to Section 2.4 discussed the major noise filtering techniques and their effectiveness including the median filter, the threshold controlled filter, the 2-D

quadratic filter, and the CWM filter. The median filter generally works for an image with spikelike noise. The threshold control filter works for better detecting noise. The 2-D quadratic filter reduces impulse noise while preserving more details for a highly corrupted image. CWM works especially well for additive or impulse noise while preserving more details. In order to compare the effectiveness of those filters, they are applied to a prostate ultrasound image, P22157.pic (see Figure 1.2). The testing results are displayed in the order of (1) the median filter, (2) the threshold controlled filter, (3) the 2-D quadratic filter, and (4) the CWM filter.

Figure 2.9 shows the result after applying the median filter with a window size of 5x5. The median filter with a larger window size of 17x17 is applied, and the result is shown in Figure 2.10. Obviously, it reduces more noise, but also introduces substantial distortion and causes the loss of more signal information.

Figure 2.11 shows resultant image after applying the threshold controlled filter. The threshold used here is 0, and window size is 5x5. It is not much improved as compared with Figure 2.9. Figure 2.12 is the result image after applying the 2-D quadratic filter. It is strange that 2-D quadratic filtering shows less reduction of noise than either the median filtering or threshold controlled filtering. Lastly Figure 2.13 shows result of applying CWM, it looks very similar to the median filtered image Figure 2.9. The noise variance chosen here is 200, and window size is again 5x5.

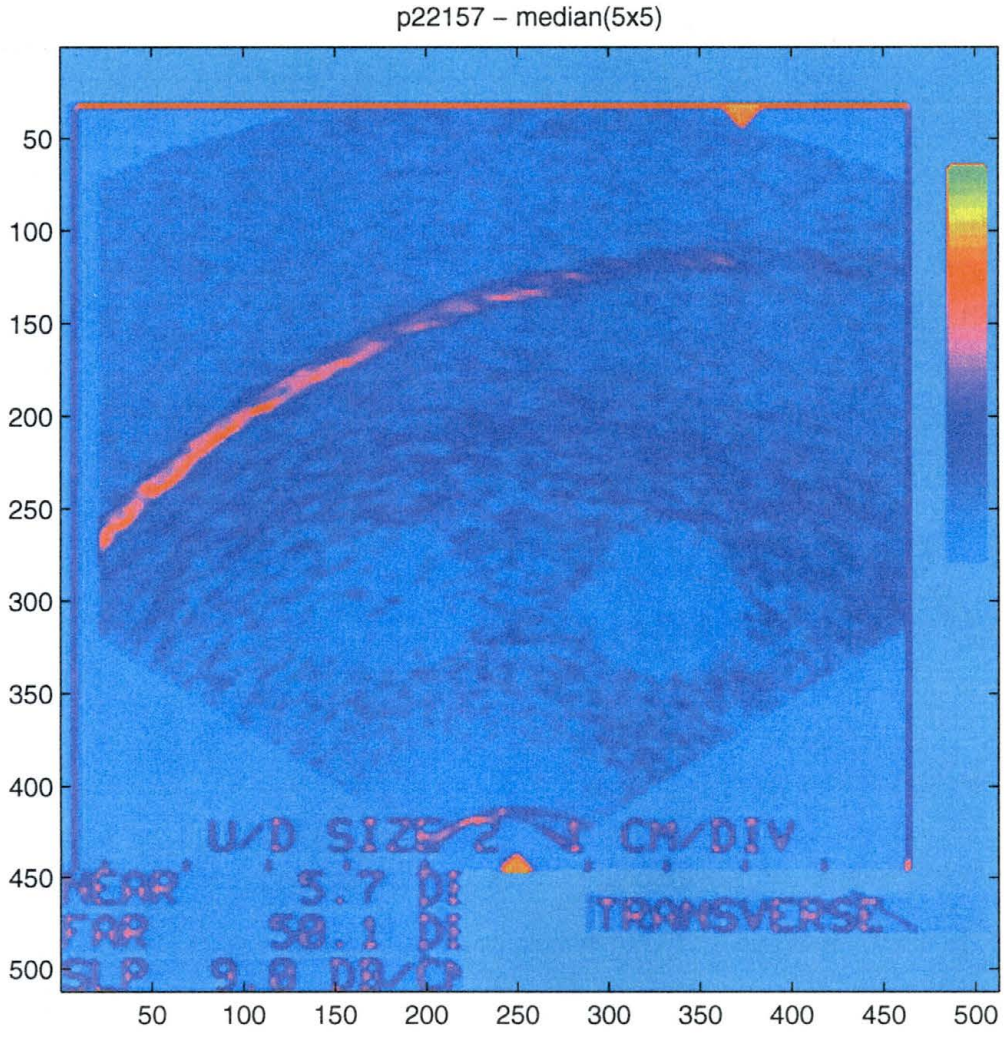

Figure 2.9. The result of median filtering (window size =5x5).

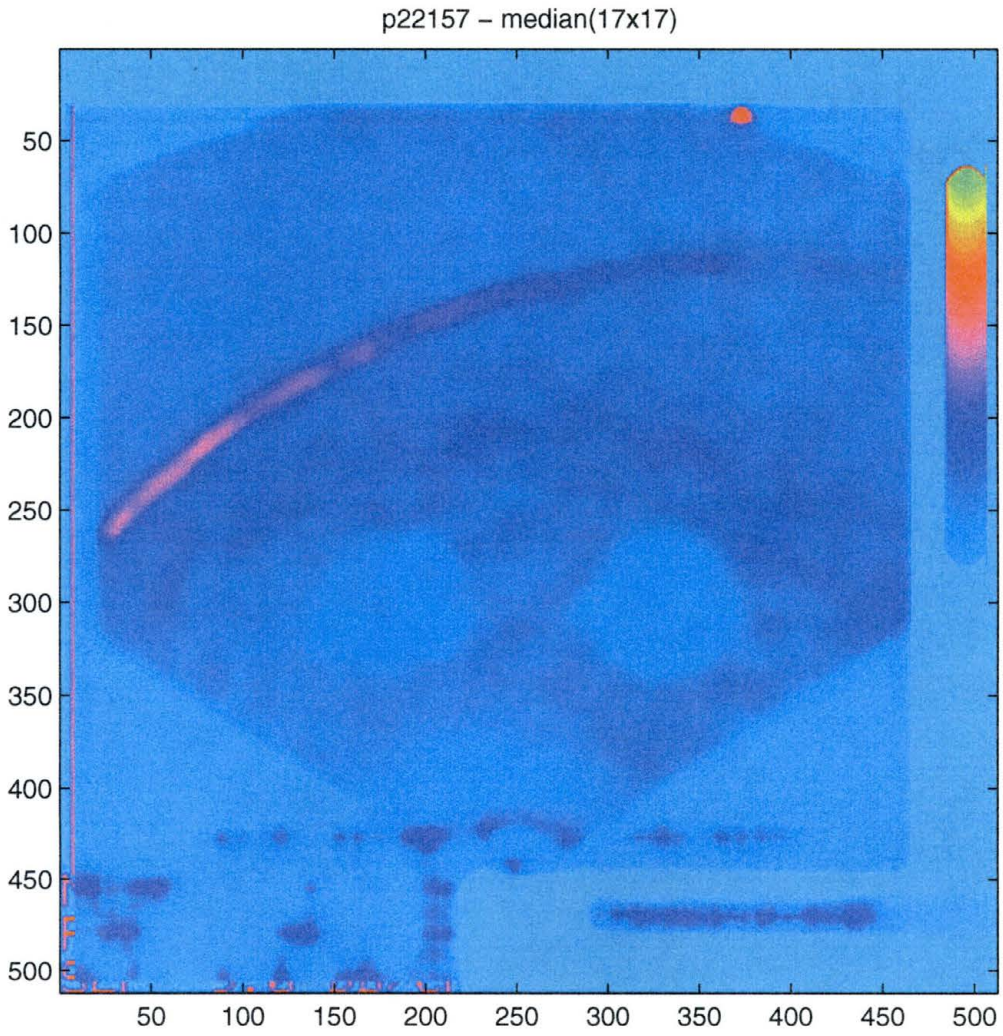

Figure 2.10. The result of median filtering (window size =17x17).

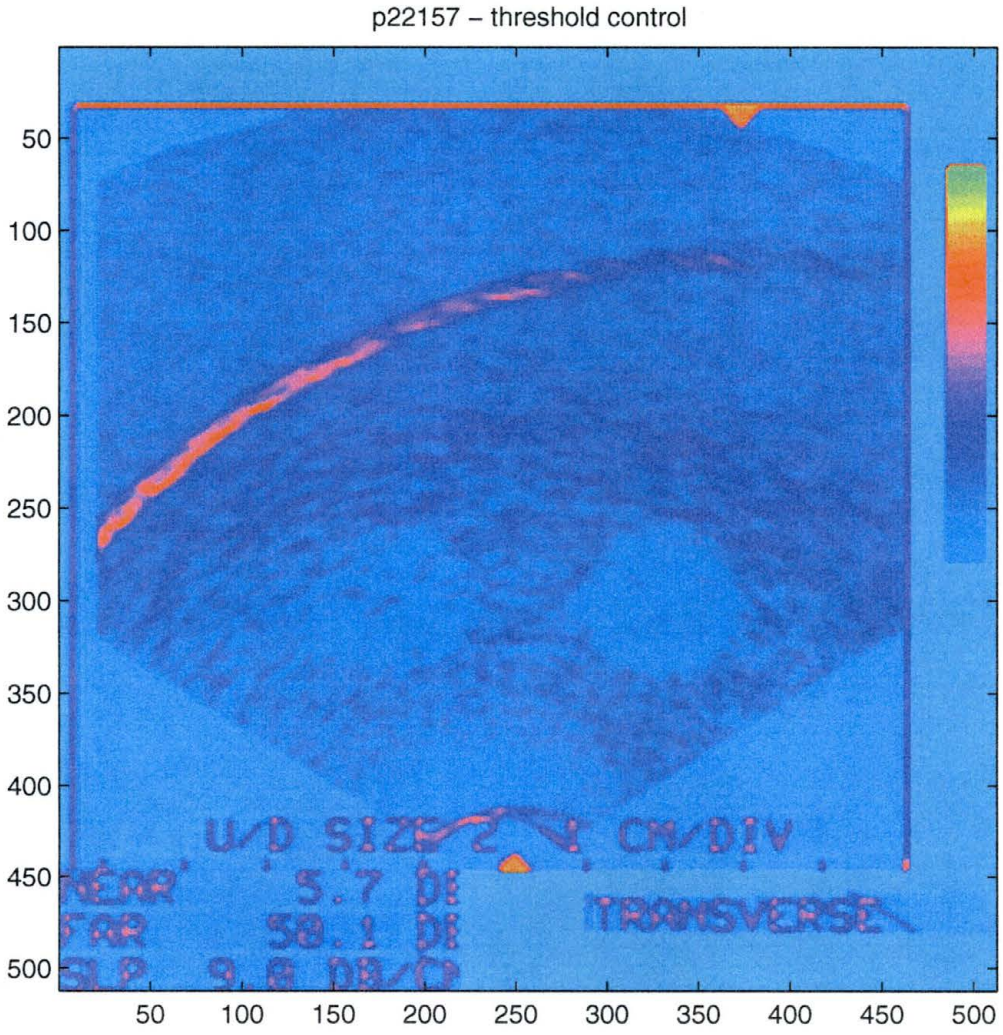

Figure 2.11. The result of threshold control method.

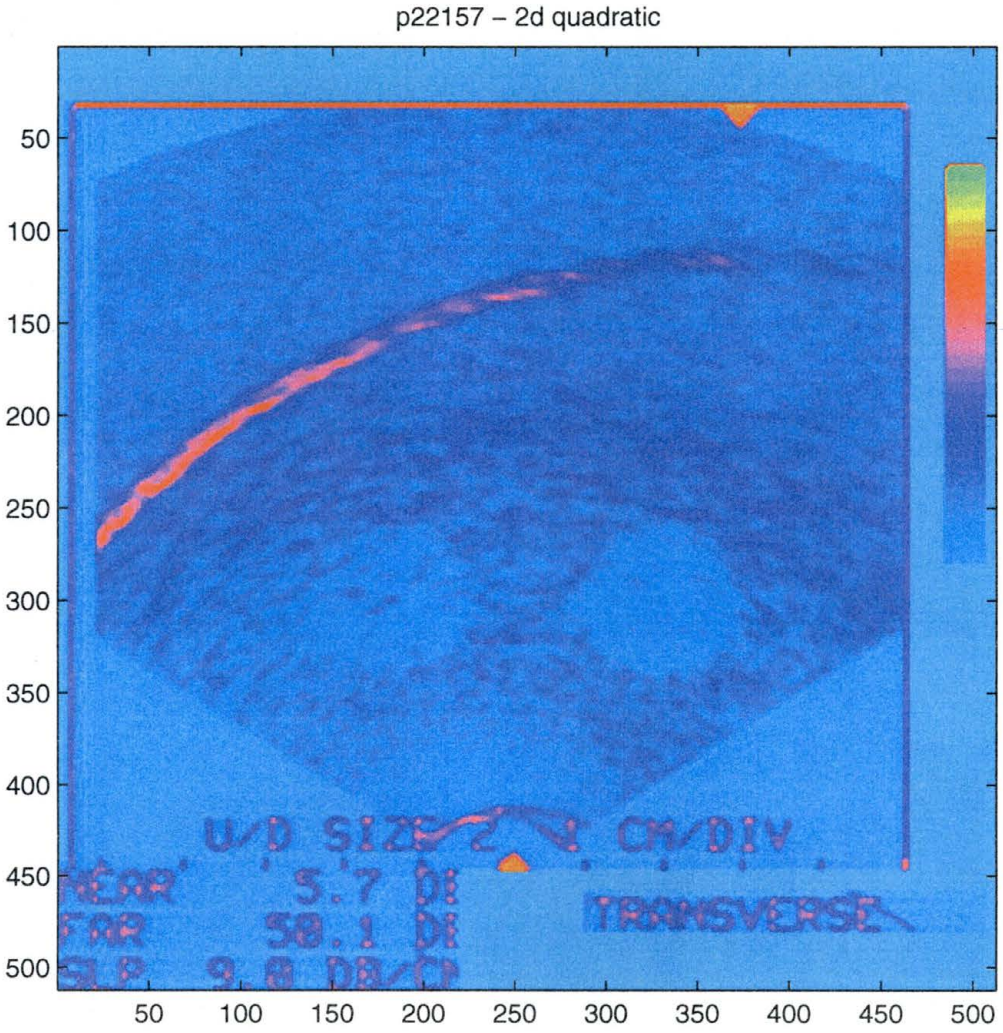

Figure 2.12. The result of 2-d quadratic filtering.

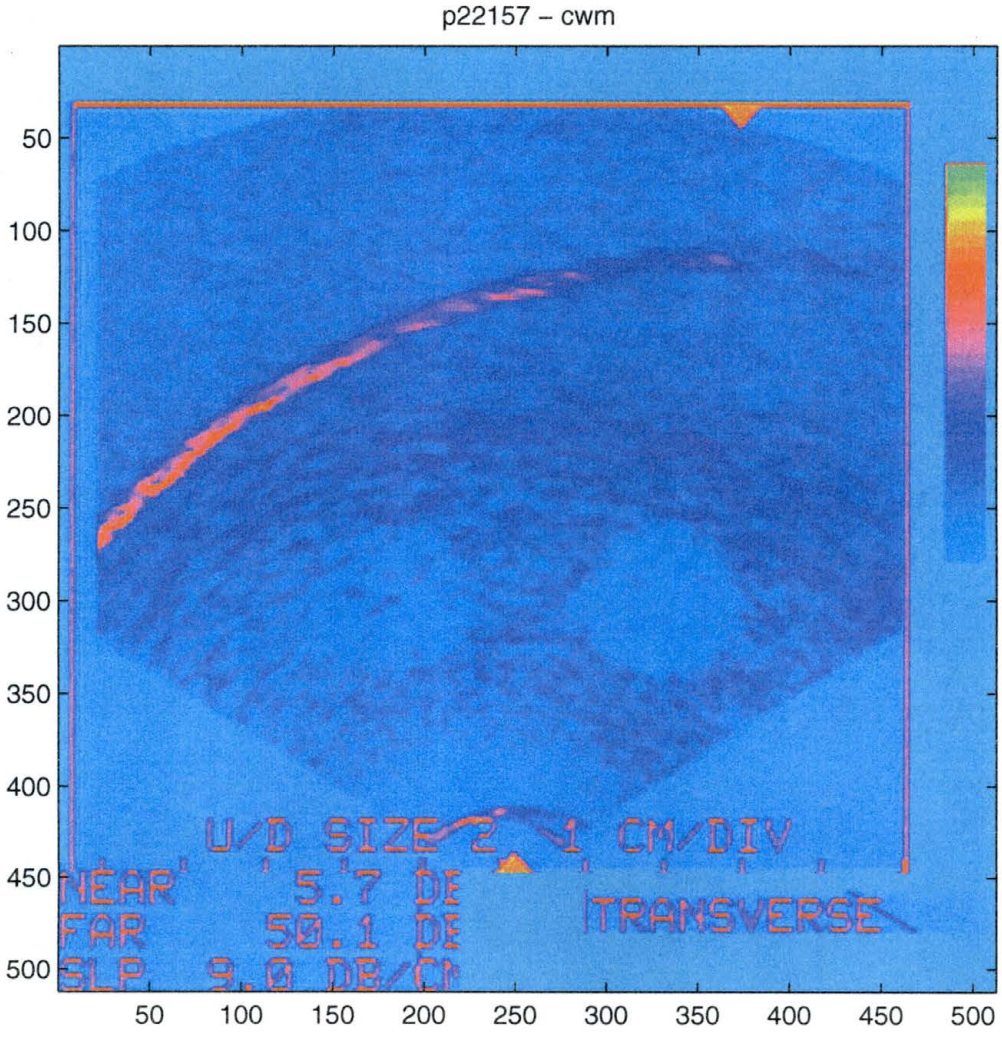

Figure 2.13. The result of CWM.

CHAPTER 3

WAVELETS AND PROSTATE ULTRASOUND IMAGES

Median filtering is very effective for removing impulsive and spikelike noise. The other filters introduced in Chapter 2 also have some advantages in reducing certain types of noise. However, none of them appear particularly effective for noise reduction in prostate ultrasound images. Wavelets filtering is applied in this chapter and it appears to do a better job in noisy prostate ultrasound images than any of the noise removal techniques studied in Chapter 2.

3.1 Wavelets

3.1.1 What Are Wavelets

Wavelets are relatively recent development in applied mathematics (Daubechis, 1992). Since early 1980s, wavelets have developed rapidly in the fields of mathematics, quantum physics, electrical engineering, and seismic geosciences (Graps, 1995). Generally, wavelets are mathematical functions that divide data into different frequency components, and then analyze each component with a resolution matched to its scale (Graps, 1995). Different from the various spatial domain median filters, wavelets filters utilize the frequency domain. Wavelets have many interesting applications such as image compression, music, and noise removal for all kinds of data. Today the most popular application of wavelets for data compression is in the geosciences.

The most important concept behind wavelets is the analysis according to scale. Wavelet algorithms process data at different scales or resolutions. In a large scale, we would notice gross features in a signal. Similarly, in a small scale, we would notice small features. What is really interesting is that wavelets let us see both the "trees and forest" as Graps (1995) described it.

The idea of wavelets comes from Fourier frequency transformation. But the real motivation is that the Fourier functions do a poor job in approximation sharp spikes. With wavelet analysis, we can use approximating functions that are contained neatly in finite domains. Wavelets are well suited for approximating data containing discontinuities and sharp spikes (Graps, 1995).

The Fourier transform analyzes time sequence data in the frequency domain. First a function is translated in the time domain into a function in the frequency domain. Then data are processed for the particular needs in the frequency domain. Last the inverse Fourier transform is used to convert data from the frequency domain back into the time domain. For images the independent variable “time” is replaced with “space” or spatial domain.

> For Fourier transform, the basis functions are sines and cosines. But for wavelet transform, the basis functions are wavelets, or mother wavelets. Individual wavelet functions are localized in space. This localization feature, along with wavelets' localization of frequency, makes many functions and operators using wavelets "sparse" when transformed into the wavelet domain. This sparseness, in turn, results in a number of useful applications such as data compression, detecting features in images, and removing noise from time series. Wavelet transforms have an infinite set

of possible basis functions. Thus wavelet analysis provides immediate access to information that can be obscured by other time-frequency methods such as Fourier analysis (Graps, 1995).

According to Graps (1995), dilations and translations of the "Mother function", $\Psi(x)$, define an orthogonal basis, the wavelet basis is (Daubechis, 1992):

$$\Psi_{(s,l)}(x) = 2^{-s/2}\Psi(2^{-s}x - l) \qquad (3.1)$$

Where variables x and l are integers that scale and dilate the mother function Ψ to generate different wavelets, such as a Harr wavelet family or a Daubechies family. The scale index s indicates the wavelet's width, and the location index l gives it position. The following are the scaling equation and the conditions that must satisfy the wavelet coefficients:

$$W(x) = \sum_{k=-1}^{N-2} (-1)^k c_{k+1}\psi(2x+k); \qquad (3.2)$$

and

$$\sum_{k=0}^{N-1} c_k = 2, \quad \sum_{k=0}^{N-1} c_k c_k + 2l = 2\delta_{l,0} \qquad (3.3)$$

where $W(x)$ is the scaling function for Ψ, the mother function, c_k are the wavelet coefficients, and δ is the delta function and l is the location index.

The wavelet transform makes an infinite set. Harr, Daubechies, Coiflet and Symmlet are four well known wavelet types. Each of them can make infinite basis wavelet functions.

3.1.3 Using Wavelets to Denoise

The beauty of wavelets is that wavelet coefficients allow the adaption of the technique for specific problems. The coefficients are placed in a transformation matrix. The transformation matrix is first applied to the raw, full-length data vector. Then the vector is smoothed and decimated by half and the matrix is applied again. Then the smoothed, halved vector is smoothed, and halved again, and the matrix applied once more. This process continues until a trivial number of "smooth-smooth-smooth..." data remain(Graps, 1995). Technically, the smoothness is achieved by arranging the coefficients into two patterns, one is for working as a smoothing filter, and the other is to bring out the data's details. In the matrix, that is to arrange the odd rows of the matrix as a smoothing filter and the even rows as the bringing out details. Therefore it is possible to remove small scale noises without substantially affecting the main features of the data (Daubechis, 1992). Usually "wavelet shrinkage and thresholding methods" are used for denoising.

Denoising is achieved by setting the coefficients to zero when they are less than a pre-determined threshold. Another controlling factor is the denoise level, a number which is the power of 2 that determines the level of details to be kept in the transformation matrix.

The following is an example used by Graps (1995) in his paper *An Introduction to Wavelets* to denoise the image *NMR* (Nuclear Magnetic Resonance) spectrum (see Figure 3.1). The denoising involves three steps:

1. transform the input image to the wavelet domain (by the family of Coiflet),
2. set the threshold to twice the standard deviation of the data,
3. inverse-transform the image back to the signal domain.

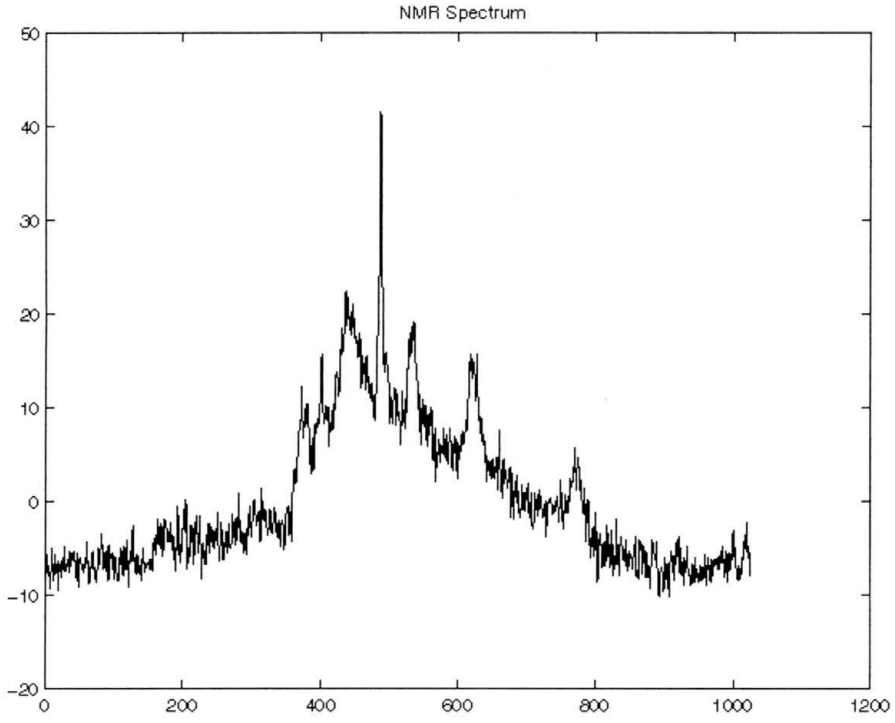

Figure 3.1. A noisy Nuclear Magnetic Resonance (NMR) spectrum.

The original input is a noisy image. However, the result (Figure 3.2) retains the main features of the original sequence, especially the huge spike in the middle, and removes ambient noise at the same time after the wavelet shrinkage and thresholding methods are applied.

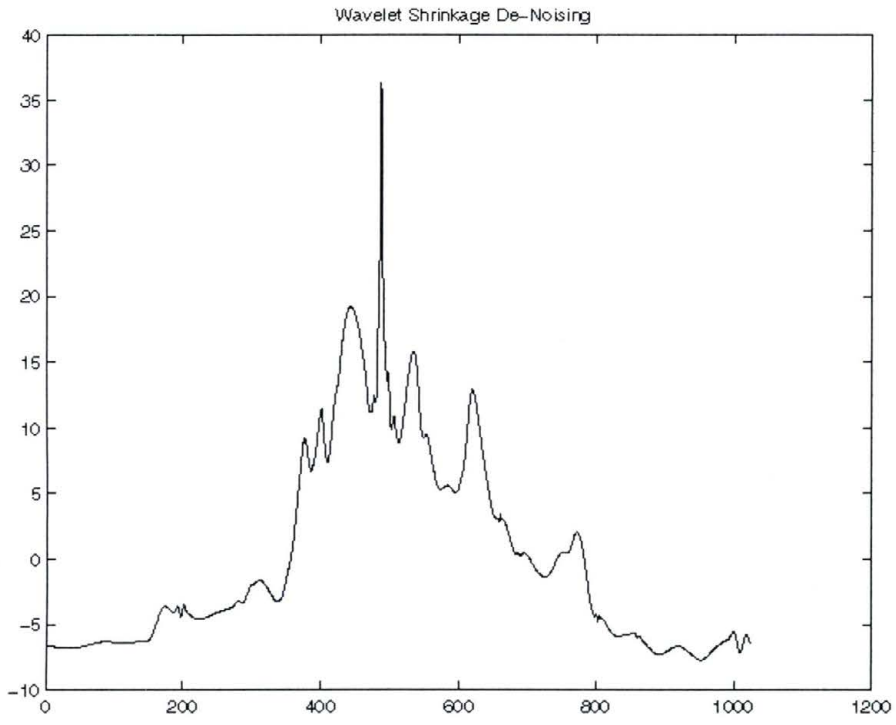

Figure 3.2. The effect of wavelet filtering.

3.2 Using Wavelets in Prostate Ultrasound Images

The computer program WaveLab is used to apply wavelets to prostate ultrasound images. WaveLab is a library of MATLAB routines for wavelet transformation functions. WaveLab is developed for research purpose by Johnathan Buckheit and his research co-workers from Stanford University. It is a free software and can be installed in UNIX, MS-Windows 3.X, and Macintosh systems. The detailed explanation of the installation of this software is documented in Appendix A.

The image used for the test is P22157.pic (Figure 1.2). The method used is "wavelet shrinkage and thresholding". The best result (Figure 3.3) for image P22157.pic is the wavelet type Coiflet, with threshold is set to 15 and denoise level is at 2. (The code

used for removing the noise in prostate ultrasound images is included in Appendix B). Usually the threshold is set to twice the standard deviation, like 15 for image P22157.pic. Another test is performed when the threshold is set to the extremely large value, 30, and denoise level is at 2. The resultant image (Figure 3.4) shows that it reduces more noise but the image is very blurred. This suggests wavelet filtering with a large threshold is not effective.

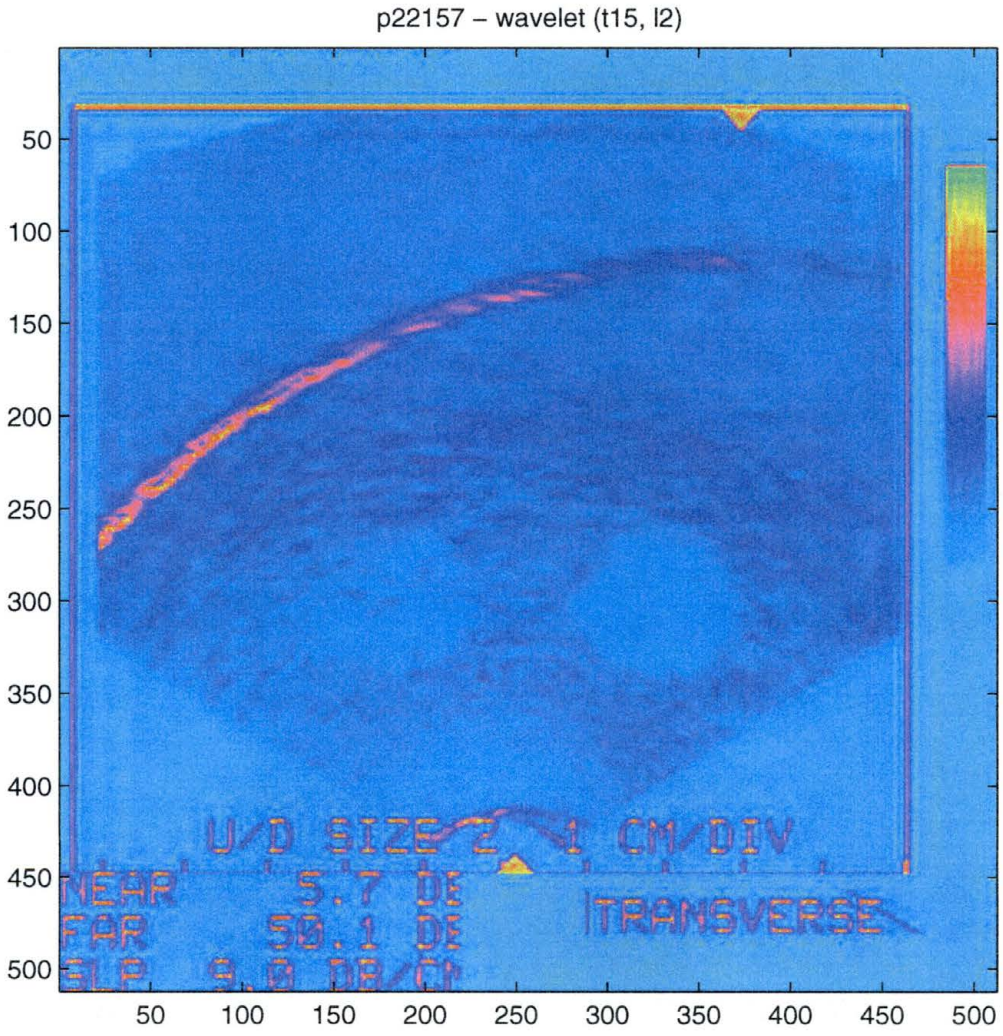

Figure 3.3. The result of wavelet filtering (threshold =15, denoise level =2).

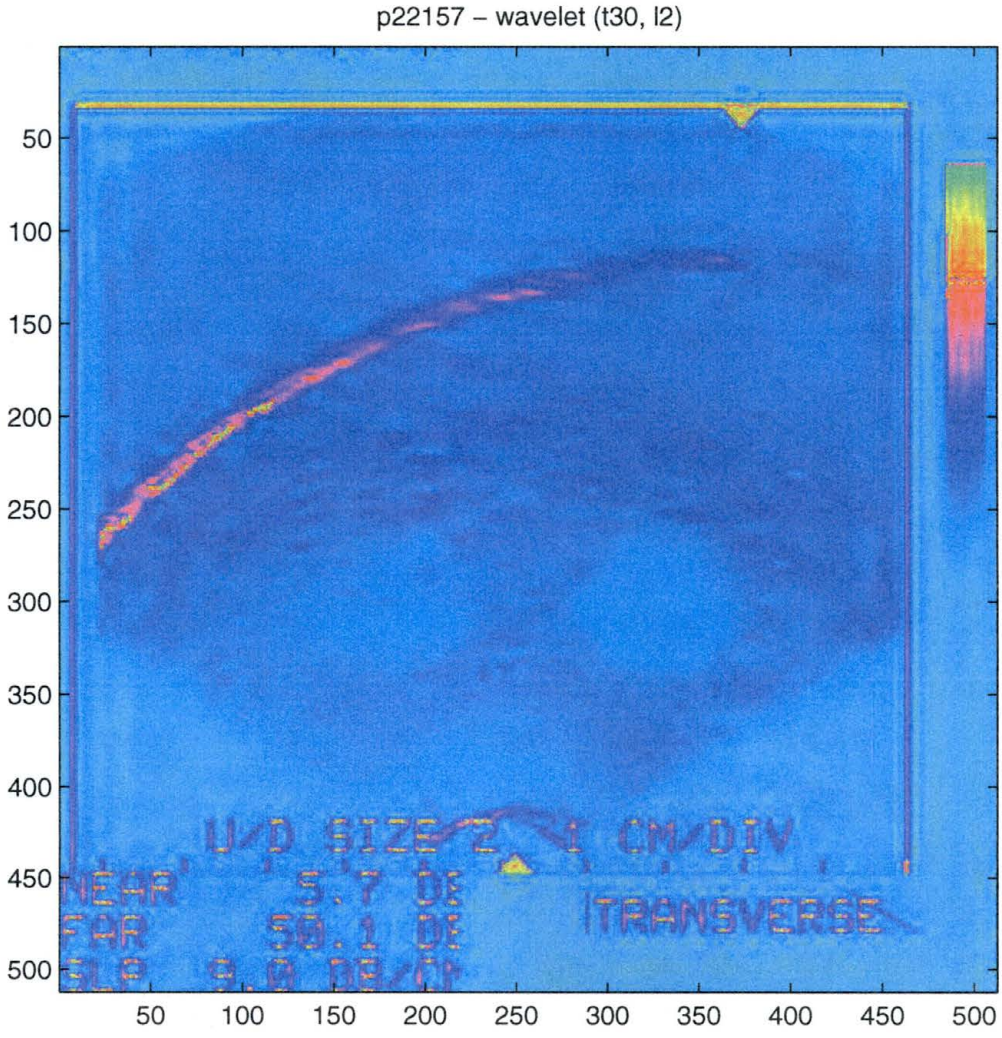

Figure 3.4. The result of wavelet filtering (threshold =30, denoise level =2).

CHAPTER 4

STATISTICAL ANALYSIS AND THE CONCLUSIONS

In Chapters 2 and 3, the results from different median and wavelet filters were compared. It appears that wavelets have the best performance in terms of reducing noise while preserving details. In this chapter, the statistical method is used to quantitatively compare, contrast, and analyze the different noise removal techniques.

4.1 The Statistical Method

The mean, standard deviation (variance), skewness, and kurtosis are often measured to give us indications of the distribution of data. The mean of a particular target indicates the average pixel value of all the pixels in that target and the standard deviation of a particular target indicates the variance of the target. The skewness and kurtosis are not discussed in this thesis because they are not relevant to our purpose. The mean and standard deviation are measured for the raw data and processed data for comparison. A circular region of interest (ROI) is placed over each target and 2285 pixels in the ROI are used to calculate the mean and standard deviation.

4.2 The Measurement of a Phantom Data P22157

The phantom image P22157 is processed with the median filter, the threshold controlled filter, the 2-D quadratic filter, and the CWM filter. When the goal of the

filtering is to remove noise while preserving details, the standard deviation should decrease while the mean remains at about the same level as that of the original data. As the standard deviation decreases, the image becomes smoother in appearance. At the same time, if the mean changes a lot, it might indicate that the image is somewhat distorted. The statistical results for the original and processed data are summarized in Table 4.1.

There are two targets in each phantom image (see Section 1.5). Those two targets are distinguished as the left target and the right target. Here in the table, "left" refers to the left target which is 2 and "right" refers to the right target which is 1 in this case (P22157).

A comparison of all the standard deviations in Table 4.1 shows that wavelets filtering gives the smallest standard deviation. The results for the 2-D quadratic filtering, CWM filtering and the threshold control method appear to be no better than median filtering. This agrees with the findings in Chapters 2 and 3. Since the performance of threshold controlled filtering, CWM filtering, and 2-D quadratic filtering with the phantom data is close to that of median filtering, only the original, median filtering, and wavelet filtering will be compared in further statistical analysis.

Table 4.1. The statistical results of P22157.

	Position	Mean	Standard Deviation	Sample Size
Original				
	left	13. 382932	5. 255149	2285
	right	11. 013567	3. 150810	2285
Median (mask = 5x5)				
	left	13. 073960	3. 945148	2285
	right	10. 879650	2. 298809	2285
Threshold Control (threshold =10, mask =5x5)				
	left	13. 073960	3. 945148	2285
	right	10. 879650	2. 298809	2285
2-D Quadratic				
	left	13. 026258	4. 346056	2285
	right	10. 898030	2. 833159	2285
CWM				
	left	13. 081401	3. 967945	2285
	right	10. 879650	2. 298809	2285
WaveLets (threshold = 10, denoise level = 2)				
	left	13. 219693	2. 765625	2285
	right	10. 858644	1. 578840	2285

4. 3 The Measurement of the Phantom Data At All Three Gain Settings

In order to complete the statistical analysis, phantom data at all three gain settings are used with the total of 12 images. Figure 4.1, Figure 4.2 and Figure 4.3 show the pixel value distribution in terms of the mean value and the associated standard deviation for five different targets in each gain setting (1.5, 5.7, and 12.0). For each gain setting, targets 2, 3 and 4 have two values because they were scanned twice during sampling data, once on the left side and once on the right side of the images. The left one is from the left position of an image (e.g. P22157.pic) and the right one is from the right position in another image (e.g. P23257.pic).

All three figures show that target 1 always has the smallest mean (darkest target) while target 5 has the largest mean (brightest target) regardless of the gain setting. Target 1 always has the smallest standard deviation while target 5 has the largest standard deviation regardless of gain setting. It is apparent that when gain increases, the noise on the targets increases too.

The goal of this research is to determine which technique has the better performance and how the performance changes with target brightness and gain setting. Figures 4.4 to 4.6 provide the answer, and show the standard deviation within the calibration targets using the original, median filtering, and wavelet filtering.

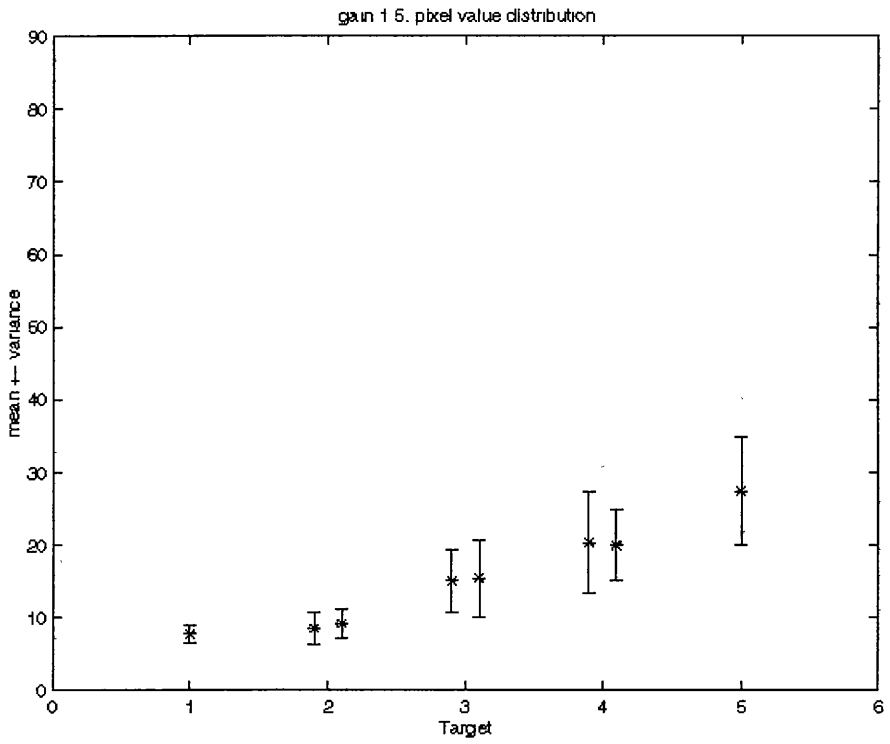

Figure 4.1. The mean of each target at gain 1.5.

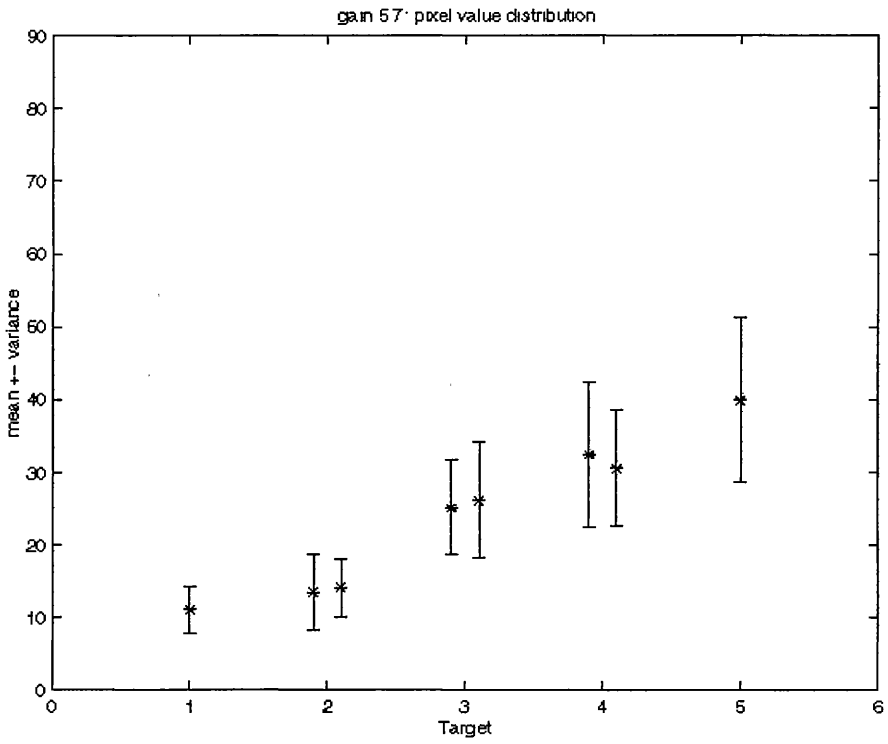

Figure 4.2. The mean of each target at gain 5.7.

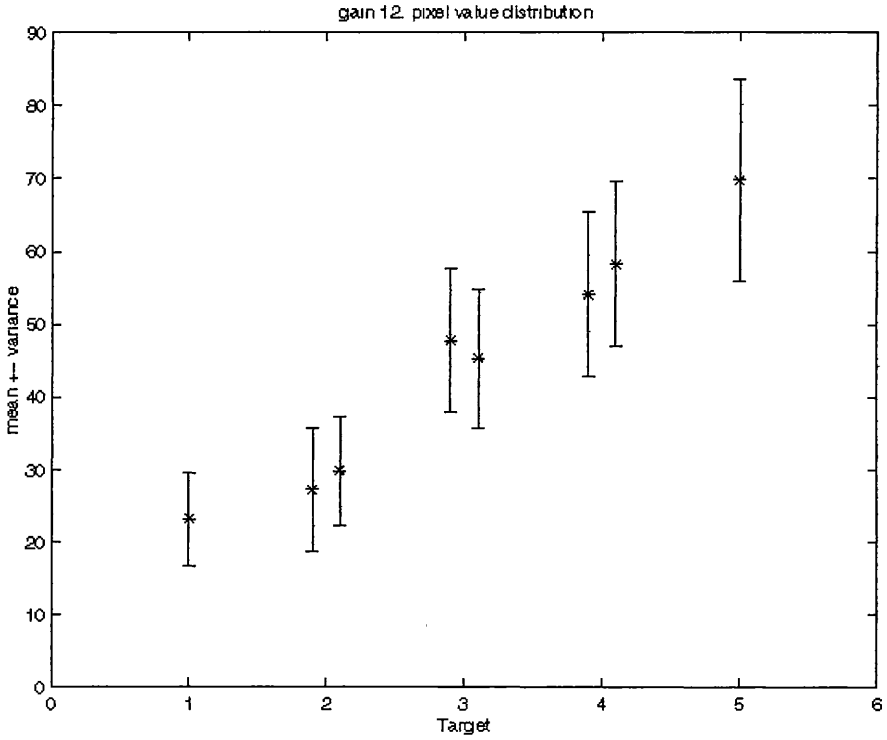

Figure 4.3. The mean of each target at gain 12.0.

It appears that wavelets filtering has the better performance for the noise removal. This is also true for the other two gain settings but these figures are not included here. So wavelet filtering consistently has the best performance independent of gain settings. In those three figures, the dotted line was used to connect the left side targets and the right side targets for better visual comparison between the results.

4.4 The Influence Of Threshold And Denoise Level For Wavelet Denoising

The function of the threshold in wavelets is to direct a wavelet filter to remove the noise component whose wavelet coefficients are below the given threshold. Usually the threshold is set to a value of twice the standard deviation. For example, if the threshold is set to 10, the wavelet filter omits details (removes noise) when the coefficients are less than 10 during the process of dividing data. A small threshold value tends to preserve more details while a larger one tends to remove more noise.

The denoise level controls which scale in the image should be kept unchanged. For example, if the denoise level is set to 2, wavelets keep data in the first $2^2 \times 2^2$ (16) times of dividing data unchanged and let the rest to be filtered in dividing data. If the denoise level is set to 3, wavelets keep data in the first $2^3 \times 2^3$ (64) times unchanged and let the rest to be filtered in dividing data. Theoretically, a large denoise level keeps an image changing less while a small denoise level changes an image a lot.

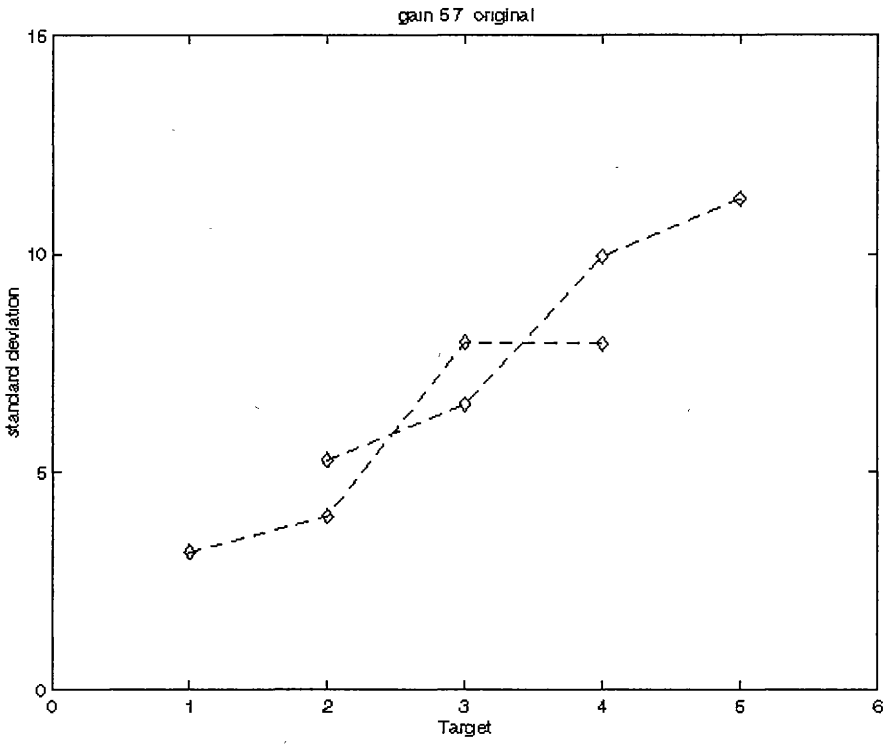

Figure 4.4. The standard deviation of each target for the original data (gain 5.7).

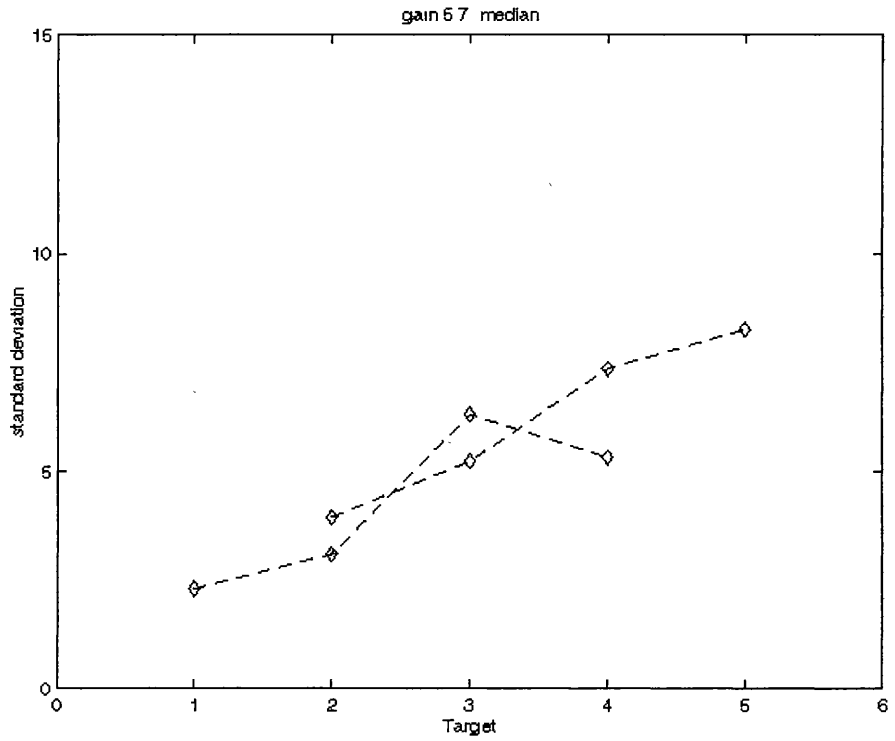

Figure 4.5. The standard deviation of each target for the median filtering processed data (gain 5.7).

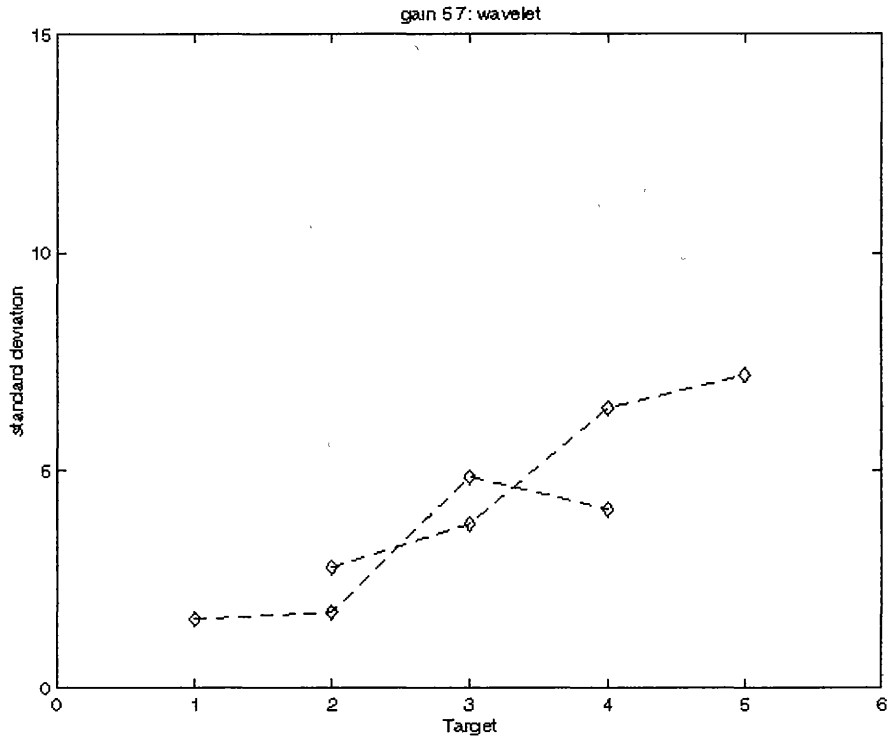

Figure 4.6. The standard deviation of each target for the wavelet filtering processed data (gain 5.7).

The performance of wavelets not only depends on the threshold but also on the denoise level. When the threshold increases, the noise level decreases. This is true for the all five targets (see Figure 4.7). And the "denoise level" does not have much impact to the image in terms of standard deviation (see Figure 4.8). The denoise level of 2 has a slightly lower standard deviation.

If we increase the threshold to the extreme, for instance, 30, the image will become very blurred. The statistical results are shown in Figure 4.9. Comparing the result of the threshold of 30 with the threshold of 15, the noise level is much decreased. However, the details and edges are not preserved. It is very much like the results of median filtering with extreme window size (see Figure 2.10 for the image, and Figure 4.10 for the statistical results), though median filtering with window size of 17x17 preserves edges better. This suggests that neither wavelets or median filtering in the extreme is effective for the purpose of removing noise.

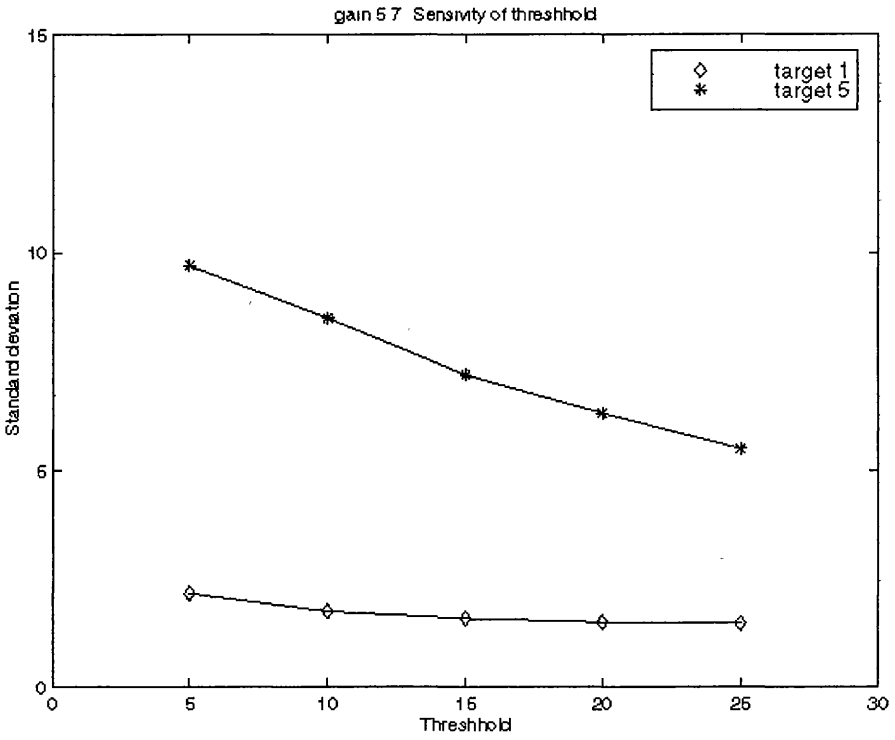

Figure 4.7. The influence of threshold in wavelets.

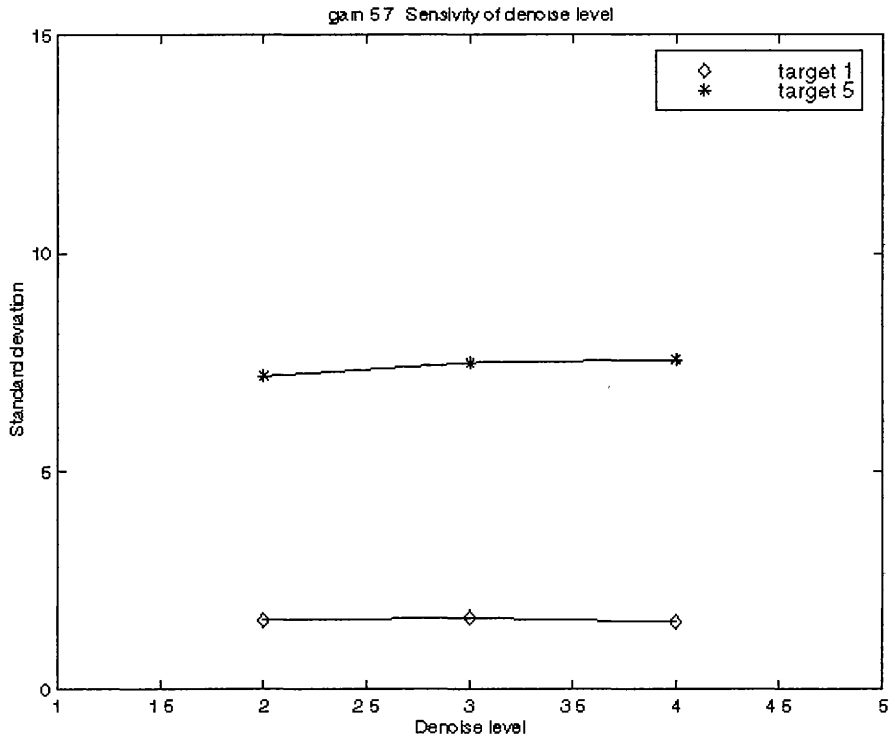

Figure 4.8. The influence of denoise level in wavelets.

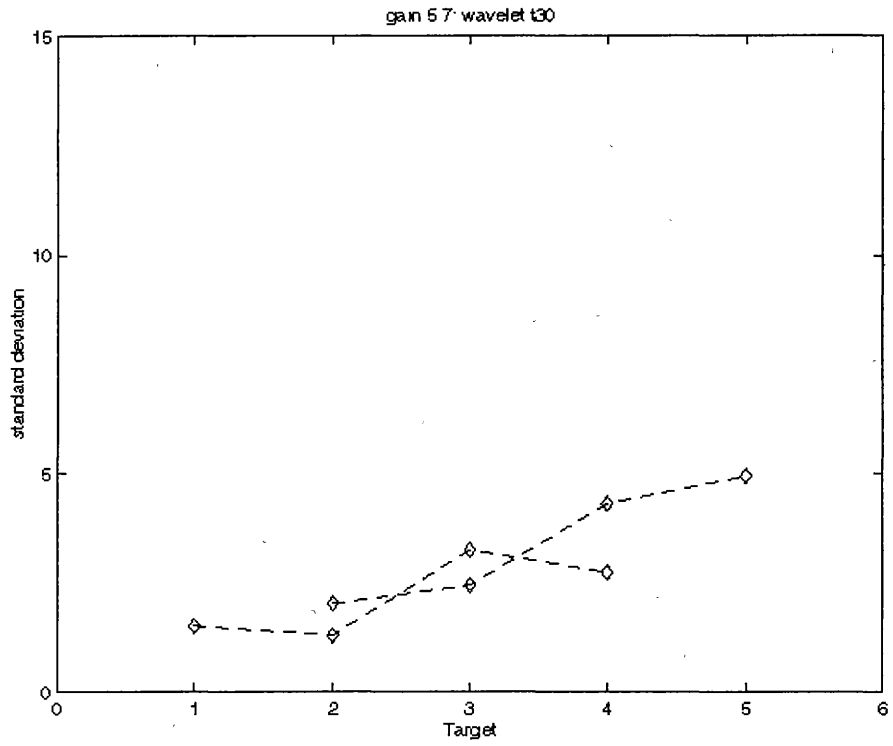

Figure 4.9. The standard deviation of each target for wavelet filtering processed data with the threshold (30).

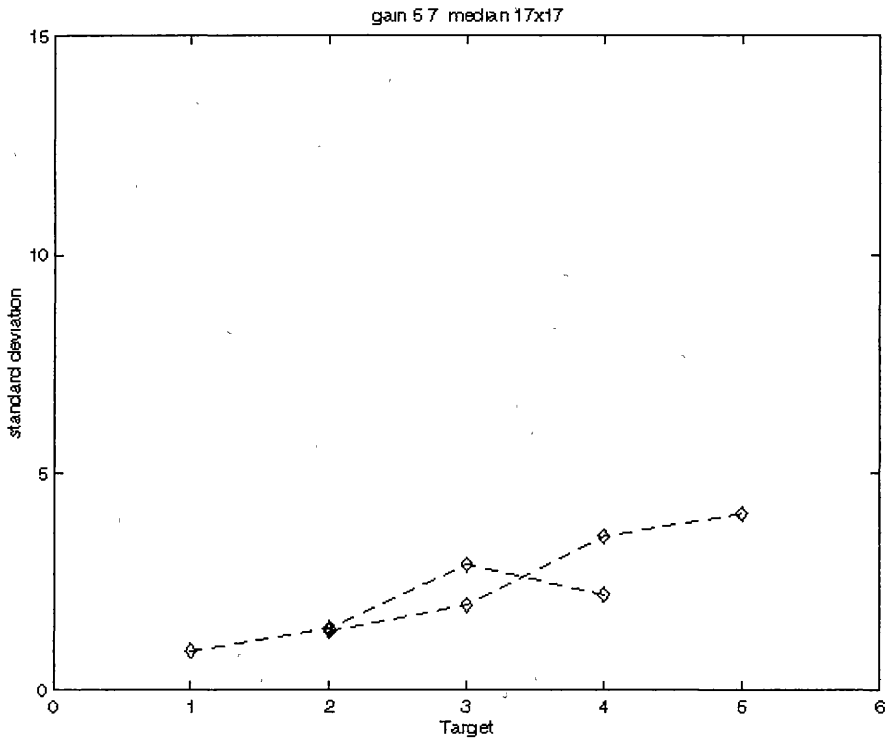

Figure 4.10. The standard deviation of each target for median filtering processed data with the extreme window size (17x17).

4.5 The Application To Real Data

From both visual inspection and statistical analysis, it was found that wavelet filtering is the most effective technique among those studied in this thesis for reducing noise in prostate ultrasound images. Therefore, wavelet filtering is applied to two real ultrasound images RX-001 (Figure 4.11) and RX-031 (Figure 4.13). The wavelet denoised results are shown in Figure 4.12 and Figure 4.14 respectively. These two resultant images provide much clearer visual image without significantly increasing blurring. Further advanced texture analysis and detection may be applied to these images, and is expected to yield better results. This certainly is beyond the scope of this thesis, and is left for ambitious readers.

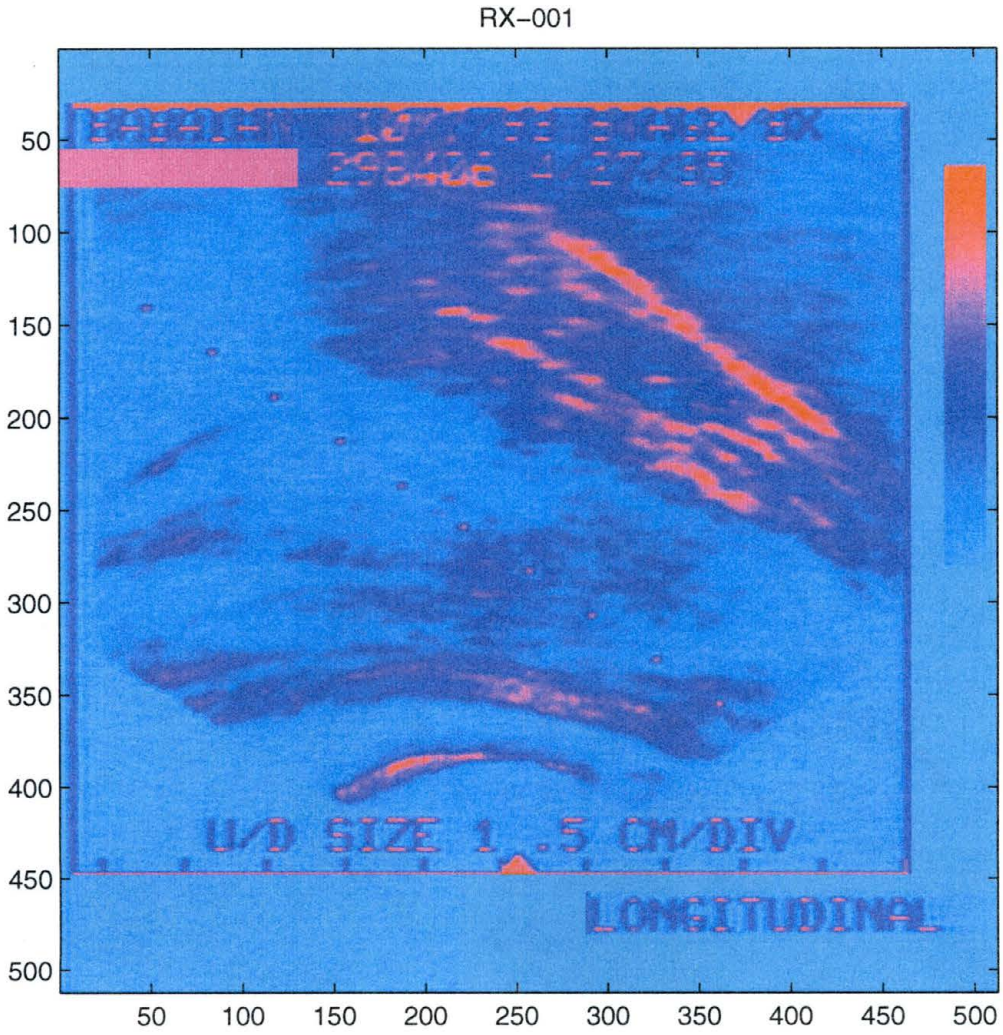

Figure 4.11. The original data (RX-001).

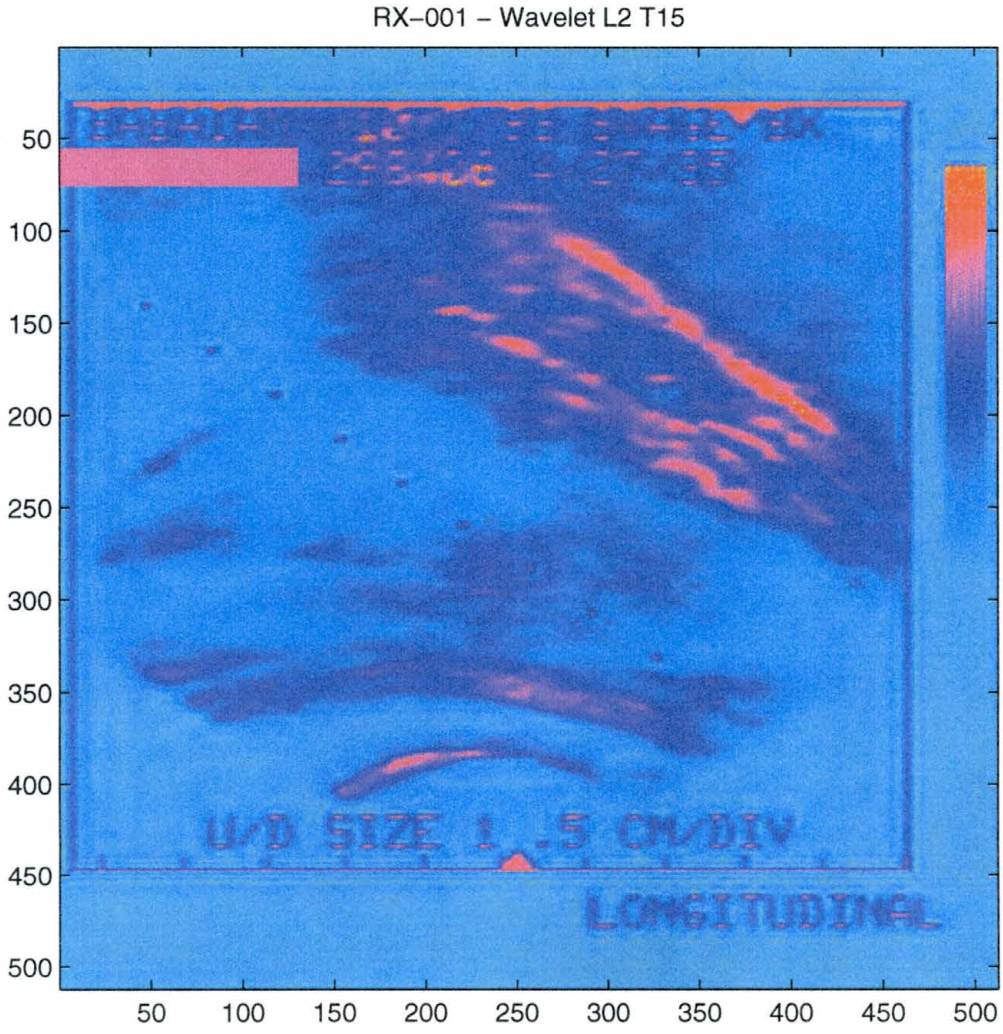

Figure 4.12. The result of wavelet filtering for the real data (RX-001).

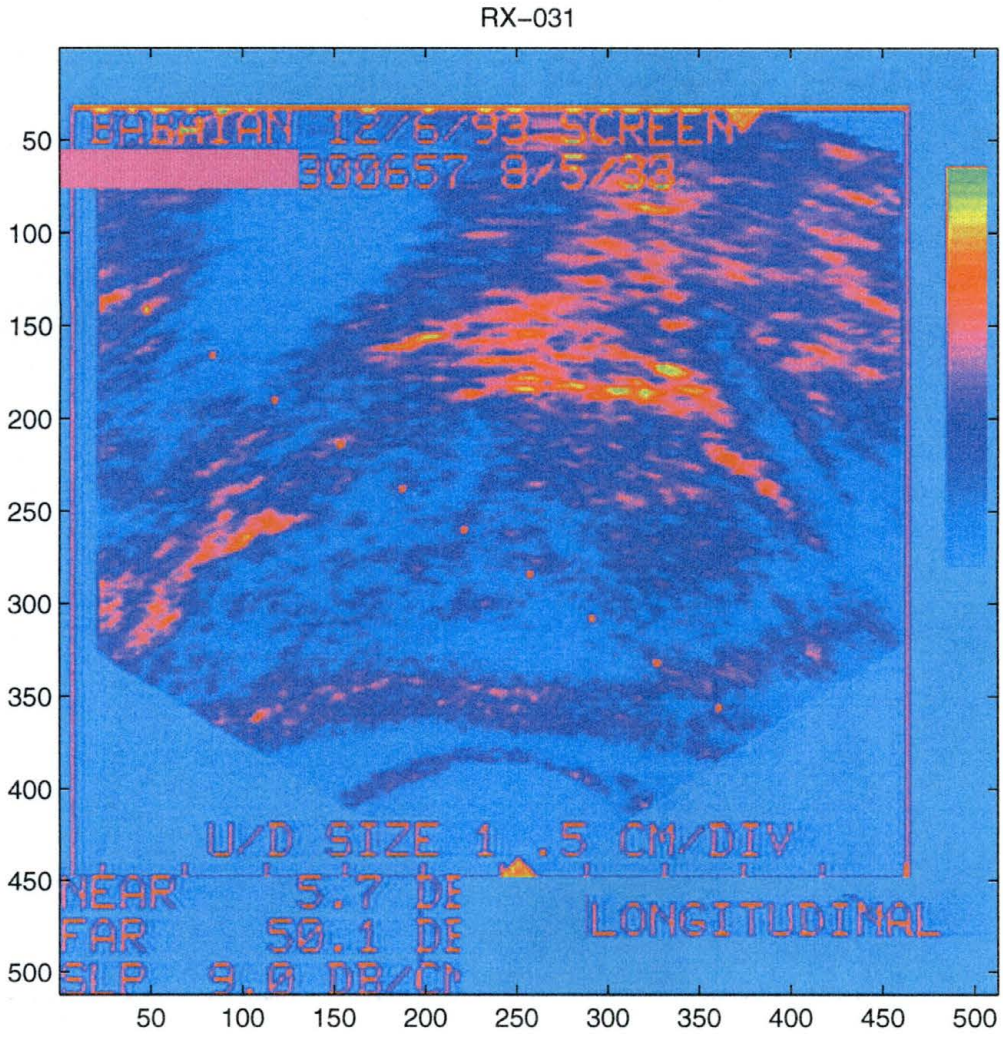

Figure 4.13. The original data (RX-031).

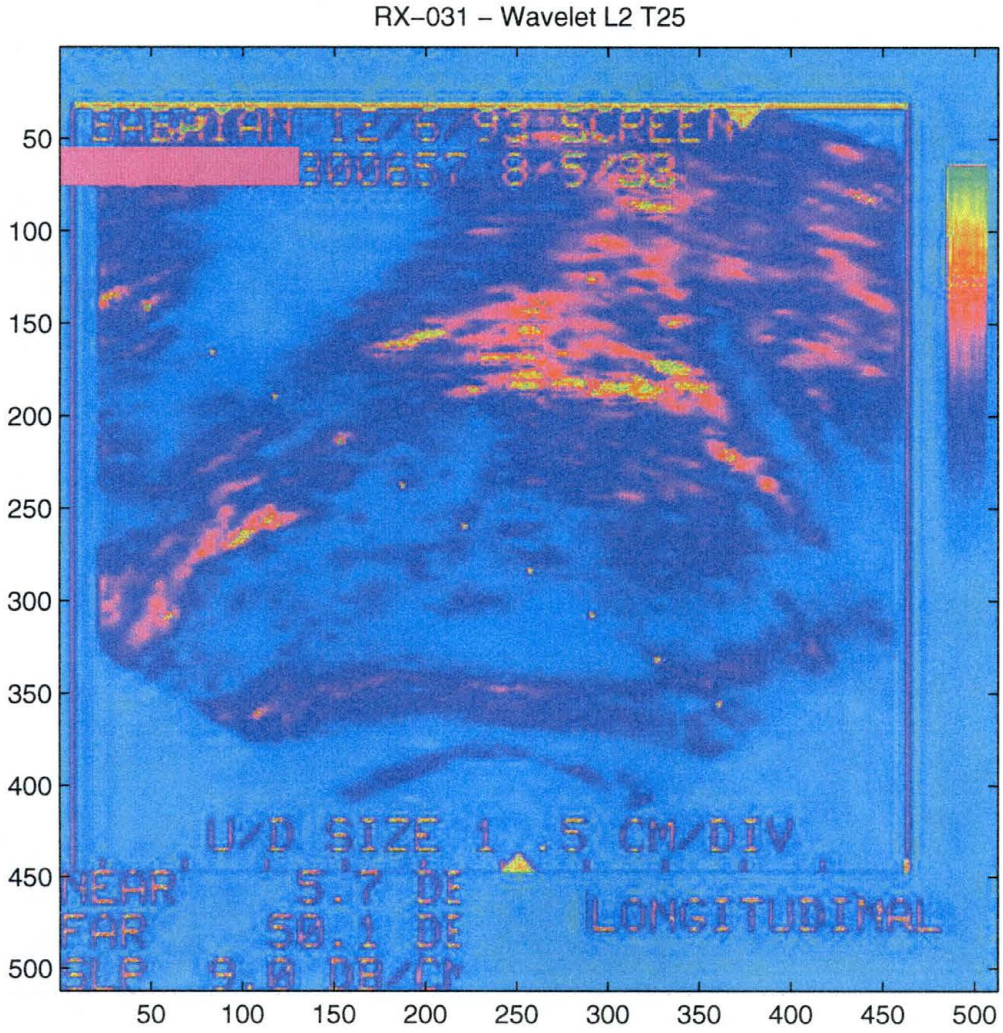

Figure 4.14. The result of wavelet filtering for the real data (RX-031).

CHAPTER 5

SUMMARY

Prostate ultrasound imaging is the most commonly used methodology for aiding physicians with needle biopsies to diagnose prostate cancer and other prostate diseases. Prostate ultrasound is used because it is cheap, risk free, and relatively painless. But like other ultrasound images, prostate ultrasound images are very noisy. With the objective of finding an effective way to remove the noise in prostate ultrasound images, the different image enhancement techniques were studied and a promising technique was found.

There are always two goals in image processing: reducing noise and preserving details. However, these two goals are always in conflict. The challenge is to reduce noise and preserve detail optimally at the same time.

Median filtering is a spatial domain method which is effective at removing spikelike noise. The window size used for the median filter has a great influence on the result: a large window removes more noise but at the expense of loss of signal information; a small window preserves more details but removes less noise.

Two more filtering techniques based on median filtering were also studied. They are 2-D quadratic filtering and center weighted median (CWM) filtering. 2-D quadratic filtering is based on a detection estimation strategy, in which the impulse noise is detected by using the product of the difference between local mean and a given threshold,

and local mean itself. If a noisy pixel is detected then the median filter is used. CWM is effective at removing impulsive and or white additive type noise. Both 2-D quadratic and CWM have some advantages in reducing certain types of noises, but the 2-D quadratic filtering is not very effective with images with noise level over 20%. CWM works for a given noise variance level. For unknown noise variance levels, a guess is necessary. In most practical situations, CWM can only reach the highest performance of a regular median filter.

The median filter, the 2-D quadratic filter and the CWM filter were all applied to phantom data, and none of them were found particularly effective. Another technique was studied: wavelets, which are mathematical functions that divide data into different frequency components, and then analyze each component with a resolution matched to its scale (Graps, 1995). Usually wavelet shrinkage and thresholding methods are used for noise removal. Denoising is achieved by setting the coefficients to zero when they are less than a pre-determined threshold during wavelet transformations, then the details with coefficients of zero are removed. By applying wavelet filtering to the phantom images, it appears that wavelets reduces noise and preserves details at the same time.

Finally, the statistical method is used to quantitatively compare, contrast, and analyze the different noise removal techniques. By studying the mean and standard deviation (the square root of the variance) for the original images and the resultant images from the different techniques, wavelets were found to decrease standard deviation while preserving the mean near the original level. Thus wavelet filtering appears best at reducing noise and preserving details at the same time for these ultrasound data.

APPENDIX A
THE INSTALLATION OF WAVELAB

WaveLab is a library of MATLAB routines specially for the various wavelet analysis. It was developed at Stanford University by Jonathan Buckheit and his research collegians in 1995. WaveLab contains about 800 .m files which are in MATLAB code format. It can only be used in MATLAB (4.X) or higher.

WaveLab is developed for use in three common operating systems: UNIX, MS-Windows and Macintosh. Since it is a free software, users can ftp to access the routines by using anonymous for login name. The ftp site is ftp://**playfair.stanford.edu/pub/wavelab**.

The name of the archives are different for the three different systems. They are **WaveLab0700.tar.Z, WaveLab0700.sea.hqx** and **wlab0700.zip** for UNIX, Macintosh and MS-Windows respectively. Because the archives are compressed, users need to decompress them after an achieves transferred to their own machines. It should be kept in mind that the software occupies about 2Mb MB disk space. For a UNIX user, use **uncompress** and **tar** to decompress a file. For a Mac user, *Stuffit, BinHex* can be used to de-binhex a file. And for a PC user, use **pkunzip** to unzip a file.

The following is an example of the checklist for PC users which was developed by the software authors and it is good for a user to make sure the installation steps:

PC Checklist

1. Binary download the file **wlab0700.zip** to your PC.
2. cd to your toolbax folder in the matlab folder.
3. **pkunzip** -d <PREFIX>**wlab0700.zip**, where <PREFIX> is the prefix making <PREFIX>**wlab0700.zip** an absolute path name reference.

After you unzip you should have the following subdirectory structure:

```
wavelab
wavelab\browsers\one-d
wavelab\cont
wavelab\datasets
wavelab\denoise
wavelab\doc
```

```
wavelab\fastalg
wavelab\interp
wavelab\meyer
wavelab\ortho
wavelab\packets
wavelab\packets2
wavelab\papers
wavelab\papers\adapt
wavelab\papers\asymp
wavelab\papers\blocky
wavelab\papers\ideal
wavelab\papers\mentseg
wavelab\papers\shortcrs
wavelab\papers\spincycl
wavelab\papers\tour
wavelab\papers\vdl
wavelab\pursuit
wavelab\station
wavelab\symm
wavelab\util
wavelab\wigner
wavelab\workouts
wavelab\workouts\bob
wavelab\workouts\mp
wavelab\workouts\multfrac
wavelab\workouts\toons
```

4. copy **c:\matlab\toolbox\wavelab\wlpath.m c:\matlab\bin**

5. Create a startup file in c:\matlab\bin which runs **wavepath** at startup time.

Users can get detailed information about WaveLab by check the Web site: **http://playfair.stanford.edu/~wavelab.**

APPENDIX B
THE CODE OF WAVELET FILTERING (DENOISING)

```
% Program: w_denoise.m
% Programmer:  Qing Liu
% This program was developed by Qing Liu in April, 1998.  It is to implement a wavelet
% filtering for removing noise of speckle type by using WaveLab routines.  It uses
Coiflet % wavelets.  The program  reads a pic file and allows a user to choose the
threshold and % the denoise level for filtering.  Then the results are written to an output
file (in pic      % format also).  It is specially desired that a user can  veiw  the image at
the same time of % runnng the program.
%
% denoising a 2-d object
%
%
% The de-noising was accomplished by
%
%    1. Transforming to the wavelet domain (Coiflet with 3 vanishing moments)
%    2. Applying a threshold at 2 standard deviations
%    3. Returning to the signal domain.
%
% read .pic file
% header:(512 bytes)
% data: 512 X 512

clear

file_in = input('input pic file = ')
f1 = fopen(file_in,'r');
hdr= fread(f1,512,'int8');
prostat = fread(f1,[512,512],'int8');
fclose(f1);clear f1

disp('Thresh hold = 2 * standard deviation');
disp(' ');
thr = input('thresh hold = ');

disp(' ');
disp('Denoise level are integers 2,3,4, etc');
lev = input('Denoise level = ');
disp(' ');

        Noisy = prostat;

        CoifQMF = MakeONFilter('Coiflet',3);
```

```
        wc_Noisy = FWT2_PO(Noisy,lev,CoifQMF);
        Coarse  = wc_Noisy(1:2^lev,1:2^lev);
        Thr_wc_Noisy = SoftThresh(wc_Noisy,thr);
        Thr_wc_Noisy(1:2^lev,1:2^lev) = Coarse;
        Clean = IWT2_PO(Thr_wc_Noisy,lev,CoifQMF);

disp(' ');
disp('To plot, enter 1; otherwise enter 0 ');
iplot = input('iplot = ');
if (iplot==1)
  figure(1)
        AutoImage(Clean');
        title([file_in(1:6) ' Level ' num2str(lev) ...
       ' thresh ' num2str(thr,2) ' De-noised']);
  figure(2)
      AutoImage(prostat');
      title([file_in(1:6)]);
end

y=Clean;
%clip to [-128, 127] range
y = fix(Clean);
for k=1:512,
 for j = 1:512,
  if (y(k,j) >= 127)
     y(k,j) = 127;
  elseif y(k,j) <= -128
     y(k,j) = -127;
  end
 end
end

% write out pic formatted data

f2 = fopen([file_in(1:6) '_t' num2str(thr,2) '_L' int2str(lev) '_wl'],'wb');
fwrite(f2,hdr,'int8');
fwrite(f2,y,'int8');
fclose(f2);clear f2
```

APPENDIX C
THE C PROGRAMS OF MEDIAN FILTERS

```
/************************        Program:  median.c        ************************
 ************************        Programmer:  Qing Liu     ************************

*****************************************************************************
* This program was developed by Qing Liu in October, 1997 based upon a program written
* by Dr.David Pitts (which is a template program - targa.c and allows a user to modify the
* image according to his own needs. The output can be displayed on Sun Solairs system.).
* Function of this program:  median filtering
* This program reads a targa format, does medaian filtering for removing spikelike noise.
* The output is also in targa format which can be displayed in a lot of softwares.
*****************************************************************************
*/

#include <stdio.h>

/************************************/
/*      Function Prototypes     */
/************************************/
void Open_io(char *, char *);           /* open a input and an output file          */
void Read_write_hdr(void);              /* read and write the header files          */
void Pause(void);                       /* pause                                    */
void Read_pixels(void);                 /* read in the pixels of an image           */
void Modify_image(void);                /* median filtering                         */
void Write_pixels(void);                /* write the pixels to the output image     */
void piksrt(int, int arr[]);            /* sorting pixels in an array               */


/************************************/
/*      Globals             */
/************************************/

FILE *fpin,*fpout;
int gpicval[512][512],gnew[512][512],gmaxx,gmaxy;
unsigned char gdata_char;

int main(int argc, char *argv[])
        /* read in a B & W targa file, allows processing, and */
        /* writes out a targa file of the same size as input  */
        /* max size of targa file is 512 lines by 512 pixels */
         /* The user should change the "Modify_image" function in */
         /* order to perform image processing functions      */

{
        Open_io (argv[1],argv[2]);      /* open an input and an output file */
        Read_write_hdr();               /* read and write the header files  */
        Pause();                        /* pause                            */
        Read_pixels();                  /* read in the pixels of an image   */

        Modify_image();                 /* median filtering                 */
```

```
        Write_pixels();                 /* write the pixels to the output image */

}               /* end of main program */

/* ____________________________________________________________ */


void Open_io(char *file_in,char *file_out)      /* open an input file and an output file */
{
        fpin=fopen(file_in,"r");    /* name of input targa file */

        if (fpin == NULL)
                {
                printf("cannot open input file\n");
                exit(0);
                }
        fpout=fopen(file_out,"w");  /* name of output targa file */
        if (fpout == NULL)
                {
                printf("cannot open output file\n");
                exit(1);
                }
}

/* ____________________________________________________________ */

void Read_write_hdr(void)
{
int j,hdr[18];
    for (j=0;j<512;j++)
        {
         fscanf(fpin,"%c",&gdata_char);      /* read input hdr */

        }
printf("\ncompleted reading %d bytes header\n",j);
    for (j=0;j<18;j++)
    {    hdr[j]=0;
    }
/* fixed header for taga format */
    hdr[2] = 3;
    hdr[13] = 2;
    hdr[15] = 2;
    hdr[16] = 8;

    for (j=0;j<18;j++)
    {

        fprintf(fpout,"%c",hdr[j]);     /* write output hdr */

    }

gmaxy=256*hdr[15]+hdr[14]; /* number of lines in image */
printf("\nNumber of lines in the image = %d\n",gmaxy);  /*  k & y  */
```

```
gmaxx=256*hdr[13]+hdr[12]; /* number of pixels per line */
printf("\nNumber of pixels per line =%d\n",gmaxx);      /* j & x */


if (hdr[16] != 8)
      {
      printf("This image file does not have 8 bit, exeuction halted\n");
      exit(2);
      }
if (gmaxx > 512 || gmaxy > 512)
      {
      printf("Image size greater than 512, execution halted\n");
      exit(3);
      }

}

/* ____________________________________________________________ */

void Pause(void)              /* wait for carriage return to continue */
{
int k,j;
          printf("\n");
          printf("Press return to continue\n");
          scanf("%c",&gdata_char);
}
/* ____________________________________________________________ */

void Read_pixels(void)
{
int k,j;
          for (k=0;k<gmaxy;k++)
          {
          for (j=0;j<gmaxx;j++)

                    {
                    fscanf(fpin,"%c",&gdata_char);
                    gpicval[j][k]=gdata_char;
                    gnew[j][k]=gdata_char;
                    }
          /*printf("Line = %d\n",k);*/
          }
}

/* ____________________________________________________________ */


void Write_pixels(void)
{
int k,j;
          for (k=0;k<gmaxy;k++)
          for (j=0;j<gmaxx;j++)
                    {
```

```
                gdata_char=gnew[j][k];
                fprintf(fpout,"%c",gdata_char);
                }
        fclose(fpin);
        fclose(fpout);
}

/* ____________________________________________________________ */

void Modify_image(void)
{
/* This procedure is to implement median filtering with a nxn box shaped      */
/* window or a plus shaped window which is at a user's choice. A user can     */
/* also chose the window size for the operation.                              */

int n;                  /* window (mask) size                          */
int choice;             /* choice for a box or a plus shaped filter    */
int k,j;                /* two index variables.                        */
int srtarr[n*n];        /* a sorting array with size of nxn.           */
int nx,ny;              /* column and row index for n square           */
int i;                  /* i is the index for sorting array            */
int X,F;                /* values of original, and the new             */


    printf("Please enter the value n = ");
    scanf("%d", &n);

    printf("\nEnter 1 for box filter, 2 for plus filter\n");
    printf("Enter your choice now:\n");
    scanf("%d", &choice);
    printf("choice=%d\n",choice);

    switch (choice) {

    case 1: {                                   /* choose to do a box shape filter */
        for (k=0;k<gmaxy;k++)
          for (j=0;j<gmaxx;j++)
              {
        if ((k<=n/2-1) || (k>=gmaxy-n/2+1) ||
           (j<=n/2-1) || (j>=gmaxx-n/2+1))
            gnew[j][k] = gpicval[j][k];
        else
           {  i = 0;
              for(nx = (k-n/2); nx <= (k+n/2); nx++)          /* manipulating each mask */
                 for(ny = (j-n/2); ny <= (j+n/2); ny++)
                  {
                    i++;
                    srtarr[i] = gpicval[ny][nx];
                  }
              piksrt(n*n, srtarr);
              gnew[j][k] = srtarr[n*n/2+1];

           }
      }
```

```
        break;
        }/*end of box choice*/

       case 2: {                                  /* choose to do a plus shape filter */
             for (k=0;k<gmaxy;k++)
               for (j=0;j<gmaxx;j++)
                    {
             if ((k<=n/2-1) || (k>=gmaxy-n/2+1) ||
               (j<=n/2-1) || (j>=gmaxx-n/2+1))
                gnew[j][k] = gpicval[j][k];
             else
               {  i = 0;
                  for(nx = (k-n/2); nx <= (k+n/2); nx++)          /* manipulating each mask */
                     for(ny = (j-n/2); ny <= (j+n/2); ny++)
                      {
                        i++;
                        if ((nx==k) ||( ny==j) )
                          srtarr[i] = gpicval[ny][nx];
                      }
                  piksrt(i, srtarr);

                  gnew[j][k] = srtarr[i/2+1];

               }
          }
        break;
       }
       default: {
         printf("Wrong choice \n");
         exit(1);
       }

}

void piksrt(int n,int arr[])   /* sorts the array arr[] of size 1..n  */
{
 int i,m;
 int a;


 for (m=2; m<=n; m++)
  {
     a=arr[m];
     i=m-1;
     while (i>0 && arr[i]>a)
           {
           arr[i+1]=arr[i];
           i--;
           }
     arr[i+1]=a;
 }
}
```

```
/************************        Program:  threshold.c        ************************
 ************************        Programmer:  Qing Liu        ************************

*****************************************************************************
* This program was developed by Qing Liu in October, 1997 based upon a program written
* by Dr.David Pitts (which is a template program - targa.c and allows a user to modify the
* image according to his own needs. The output can be displayed on Sun Solairs system.).
* Function of this program:  threshold controled median filtering
* This program reads a targa format, does threshold control filtering for better detecing noise.
* The output is also in targa format which can be displayed in a lot of softwares.
*****************************************************************************
*/
#include <stdio.h>

/*************************************/
/*      Function Prototypes      */
/*************************************/
void Open_io(char *, char *);           /* open a input and an output file        */
void Read_write_hdr(void);              /* read and write the header files        */
void Pause(void);                       /* pause                                  */
void Read_pixels(void);                 /* read in the pixels of an image         */
void Modify_image(void);                /* threshold controled median filtering   */
void Write_pixels(void);                /* write the pixels to the output image   */
void piksrt(int, int arr[]);            /* sorting pixels in an array             */


/*************************************/
/*      Globals          */
/*************************************/


FILE *fpin,*fpout;
int gpicval[512][512],gnew[512][512],gmaxx,gmaxy;
unsigned char gdata_char;

int main(int argc, char *argv[])
        /* read in a B & W targa file, allows processing, and */
        /* writes out a targa file of the same size as input  */
        /* max size of targa file is 512 lines by 512 pixels */
         /* The user should change the "Modify_image" function in */
         /* order to perform image processing functions     */

{

        Open_io (argv[1],argv[2]);      /* open an input and an output file       */
        Read_write_hdr();               /* read and write the header files        */
        Pause();                        /* pause                                  */
        Read_pixels();                  /* read in the pixels of an image         */

        Modify_image();                 /* threshold controled median filtering   */

        Write_pixels();                 /* write the pixels to the output image   */
}               /* end of main program */
```

```
/* ____________________________________________________________ */

void Open_io(char *file_in,char *file_out)       /* open an input file and an output file */
{
        fpin=fopen(file_in,"r");    /* name of input targa file */

        if (fpin == NULL)
         {
                printf("cannot open input file\n");
                exit(0);
         }
        fpout=fopen(file_out,"w");  /* name of output targa file */
        if (fpout == NULL)
        {
                printf("cannot open output file\n");
                exit(1);
        }
}

/* ____________________________________________________________ */

void Read_write_hdr(void)
{
int j,hdr[18];
     for (j=0;j<512;j++)
     {
          fscanf(fpin,"%c",&gdata_char);      /* read input hdr */

      }
printf("\ncompleted reading %d bytes header\n",j);
     for (j=0;j<18;j++)
     {     hdr[j]=0;
     }
/* fixed header for taga format */
     hdr[2] = 3;
     hdr[13] = 2;
     hdr[15] = 2;
     hdr[16] = 8;

     for (j=0;j<18;j++)
     {

          fprintf(fpout,"%c",hdr[j]);     /* write output hdr */

     }

gmaxy=256*hdr[15]+hdr[14]; /* number of lines in image */
printf("\nNumber of lines in the image = %d\n",gmaxy);  /*  k & y  */

gmaxx=256*hdr[13]+hdr[12]; /* number of pixels per line */
printf("\nNumber of pixels per line =%d\n",gmaxx);      /*  j & x  */
```

```
if (hdr[16] != 8)
 {
     printf("This image file does not have 8 bit, exeuction halted\n");
     exit(2);
 }
if (gmaxx > 512 || gmaxy > 512)
 {
     printf("Image size greater than 512, execution halted\n");
     exit(3);
  }

}

/* ____________________________________________________________ */

void Pause(void)              /* wait for carriage return to continue */
{
int k,j;
          printf("\n");
          printf("Press return to continue\n");
          scanf("%c",&gdata_char);
}
/* ____________________________________________________________ */

void Read_pixels(void)
{
int k,j;
          for (k=0;k<gmaxy;k++)
          {
                for (j=0;j<gmaxx;j++)

                {
                    fscanf(fpin,"%c",&gdata_char);
                    gpicval[j][k]=gdata_char;
                    gnew[j][k]=gdata_char;
                 }
          /*printf("Line = %d\n",k);*/
          }
}

/* ____________________________________________________________ */


void Write_pixels(void)
{
int k,j;
          for (k=0;k<gmaxy;k++)
          for (j=0;j<gmaxx;j++)
          {
                    gdata_char=gnew[j][k];
                    fprintf(fpout,"%c",gdata_char);
          }
          fclose(fpin);
          fclose(fpout);
```

```
}

/* ____________________________________________________________ */

void Modify_image(void)
{
/* This procedure is to implement threshold controled median filtering with   */
/* a nxn box shaped window or a plus shaped window which is at a user's       */
/* choice. A user will input threshold and also chose the window size for     */
/* the operation.                                                             */

int n;                  /* window (mask) size                                 */
int choice;             /* choice for a box or a plus shaped filter           */
int k,j;                /* two index variables.                               */
int srtarr[n*n];        /* a sorting array with size of nxn.                  */
int nx,ny;              /* column and row index for nxn box                   */
int i;                  /* i is the index for sorting array                   */
int X,F,t;              /* values of original, the new, and threshold         */

    printf("Plwase enter the threshold t= ");
    scanf("%d", &t);

    printf("Please enter the value n = ");
    scanf("%d", &n);

    printf("\nEnter 1 for box filter, 2 for cross filter\n");
    printf("Enter your choice now:\n");
    scanf("%d", &choice);
    printf("choice=%d\n",choice);

    switch (choice) {

    case 1: {                                          /* choose to do a box shaped filter */
        for (k=0;k<gmaxy;k++)
          for (j=0;j<gmaxx;j++)
           {
              if ((k<=n/2-1) || (k>=gmaxy-n/2+1) ||(j<=n/2-1) || (j>=gmaxx-n/2+1))
                    gnew[j][k] = gpicval[j][k];
              else
               {  i = 0;
                  for(nx = (k-n/2); nx <= (k+n/2); nx++)          /* manipulating each mask*/
                    for(ny = (j-n/2); ny <= (j+n/2); ny++)
                        {
                          i++;
                          srtarr[i] = gpicval[ny][nx];
                        }
                  piksrt(n*n, srtarr);
                  gnew[j][k] = srtarr[n*n/2+1];

                  X = gpicval[j][k];              /*original value*/
                  F = gnew[j][k];                 /* the new value*/
                  if(abs(X-F)<t)                  /*thresh control*/
                  gnew[j][k] = gpicval[j][k];
          }
```

```
        }
      break;
      }/*end of box choice*/

     case 2: {                                          /* choose to do a plus shaped filter */
           for (k=0;k<gmaxy;k++)
             for (j=0;j<gmaxx;j++)
             {
               if ((k<=n/2-1) || (k>=gmaxy-n/2+1) || (j<=n/2-1) || (j>=gmaxx-n/2+1))
                   gnew[j][k] = gpicval[j][k];
               else
                   {  i = 0;
                     for(nx = (k-n/2); nx <= (k+n/2); nx++)          /* manipulating each mask */
                        for(ny = (j-n/2); ny <= (j+n/2); ny++)
                        {
                           i++;
                           if ((nx==k) ||( ny==j) )
                                srtarr[i] = gpicval[ny][nx];
                        }
                     piksrt(i, srtarr);
                     gnew[j][k] = srtarr[i/2+1];

                     X = gpicval[j][k];                 /*original value*/
                     F = gnew[j][k];            /* the new value*/
                     if(abs(X-F)<t)             /*thresh control*/
                      gnew[j][k] = gpicval[j][k];
             }
        }
      break;
     }
     default: {
       printf("Wrong choice \n");
       exit(1);
     }
}

void piksrt(int n,int arr[])   /* sorts the array arr[] of size 1..n  */
{
 int i,m;
 int a;


 for (m=2; m<=n; m++)
  {
     a=arr[m];
     i=m-1;
     while (i>0 && arr[i]>a)
           {
           arr[i+1]=arr[i];
           i--;
           }
     arr[i+1]=a;
}
}
```

```
/************************        Program: 2-Dquadratic.c      ************************
 ************************        Programmer:  Qing Liu        ************************

*******************************************************************************
* This program was developed by Qing Liu in Febuary, 1998 based upon a program written
* by Dr.David Pitts (which is a template program - targa.c and allows a user to modify the
* image according to his own needs. The output can be displayed on Sun Solairs system.).
*  The algorithm of this program is from Mitra et al (see charpter 2 for the details).
* Function of this program:  2-D quadratic filtering
* This program reads a targa format, does 2-D quadratic filtering for removing implusive noise.
* The output is also in targa format which can be displayed in a lot of softwares.
*******************************************************************************
*/

#include <stdio.h>

/*************************************/
/*       Function Prototypes     */
/*************************************/
void Open_io(char *, char *);           /* open a input and an output file          */
void Read_write_hdr(void);              /* read and write the header files          */
void Pause(void);                       /* pause                                    */
void Read_pixels(void);                 /* read in the pixels of an image           */
void Modify_image(void);                /* threshold controled median filtering     */
void Write_pixels(void);                /* write the pixels to the output image     */
void piksrt(int, int arr[]);            /* sorting pixels in an array               */

/*************************************/
/*       Globals           */
/*************************************/


FILE *fpin,*fpout;
int gpicval[512][512],gnew[512][512],gmaxx,gmaxy;
unsigned char gdata_char;

int main(int argc, char *argv[])
        /* read in a B & W targa file, allows processing, and       */
        /* writes out a targa file of the same size as input        */
        /* max size of targa file is 512 lines by 512 pixels        */
         /* The user should change the "Modify_image" function in   */
         /* order to perform image processing functions             */

{

        Open_io (argv[1],argv[2]);      /* open an input and an output file     */
        Read_write_hdr();               /* read and write the header files      */
        Pause();                        /* pause                                */
        Read_pixels();                  /* read in the pixels of an image       */

        Modify_image();                 /* threshold controled median filtering */

        Write_pixels();                 /* write the pixels to the output image */
}               /* end of main program */
```

```
/* ____________________________________________________________ */

void Open_io(char *file_in,char *file_out)       /* open an input file and an output file */
{
        fpin=fopen(file_in,"r");    /* name of input targa file */

        if (fpin == NULL)
                {
                printf("cannot open input file\n");
                exit(0);
                }
        fpout=fopen(file_out,"w");  /* name of output targa file */
        if (fpout == NULL)
                {
                printf("cannot open output file\n");
                exit(1);
                }
}

/* ____________________________________________________________ */

void Read_write_hdr(void)
{
int j,hdr[18];
     for (j=0;j<512;j++)
          {
          fscanf(fpin,"%c",&gdata_char);      /* read input hdr */

          }
printf("\ncompleted reading %d bytes header\n",j);
     for (j=0;j<18;j++)
     {     hdr[j]=0;
     }
/* fixed header for taga format */
     hdr[2] = 3;
     hdr[13] = 2;
     hdr[15] = 2;
     hdr[16] = 8;

     for (j=0;j<18;j++)
     {

          fprintf(fpout,"%c",hdr[j]);     /* write output hdr */

     }

gmaxy=256*hdr[15]+hdr[14]; /* number of lines in image */
printf("\nNumber of lines in the image = %d\n",gmaxy);  /*  k & y  */

gmaxx=256*hdr[13]+hdr[12]; /* number of pixels per line */
printf("\nNumber of pixels per line =%d\n",gmaxx);      /*  j & x  */
```

```
if (hdr[16] != 8)
        {
        printf("This image file does not have 8 bit, exeuction halted\n");
        exit(2);
        }
if (gmaxx > 512 || gmaxy > 512)
        {
        printf("Image size greater than 512, execution halted\n");
        exit(3);
        }

}

/* ____________________________________________________________ */

void Pause(void)                /* wait for carriage return to continue */
{
int k,j;
        printf("\n");
        printf("Press return to continue\n");
        scanf("%c",&gdata_char);
}
/* ____________________________________________________________ */

void Read_pixels(void)
{
int k,j;
        for (k=0;k<gmaxy;k++)
        {
        for (j=0;j<gmaxx;j++)

                {
                fscanf(fpin,"%c",&gdata_char);
                gpicval[j][k]=gdata_char;
                gnew[j][k]=gdata_char;
                }
        /*printf("Line = %d\n",k);*/
        }
}

/* ____________________________________________________________ */


void Write_pixels(void)
{
int k,j;
        for (k=0;k<gmaxy;k++)
        for (j=0;j<gmaxx;j++)
                {
                gdata_char=gnew[j][k];
                fprintf(fpout,"%c",gdata_char);
                }
        fclose(fpin);
```

```
        fclose(fpout);
}

/* ____________________________________________________________ */

void Modify_image(void)
{
/* This procedure is to implement 2-D quadratic filtering with          */
/* a nxn box shaped window or a plus shaped window which is at a user's */
/* choice. A user will input threshold and also chose the window size for */
/* the operation.                                                       */

int n;                  /* window (mask) size                        */
int choice;             /* choice for a box or a plus shaped filter  */
int k,j;                /* two index variables.                      */
int srtarr[n*n];        /* a sorting array with size of nxn.         */
int nx,ny;              /* column and row index for nxn box          */
int i;                  /* i is the index for sorting array          */
int X,F,t;              /* values of original, the new, and threshold */
int local_sum, local_mean; /* local sum and local mean               */

    printf("Plwase enter the threshold t= ");
    scanf("%d", &t);

    printf("Please enter the value n = ");
    scanf("%d", &n);

    printf("\nEnter 1 for box filter, 2 for plus filter\n");
    printf("Enter your choice now:\n");
    scanf("%d", &choice);
    printf("choice=%d\n",choice);

    switch (choice) {

    case 1: {                                   /* choose to do a box shaped filter */
        for (k=0;k<gmaxy;k++)
          for (j=0;j<gmaxx;j++)
            {
              if ((k<=n/2-1) || (k>=gmaxy-n/2+1) || (j<=n/2-1) || (j>=gmaxx-n/2+1))
                 gnew[j][k] = gpicval[j][k];
              else
              {  i = 0;
                 local_sum = 0;
                 for(nx = (k-n/2); nx <= (k+n/2); nx++)
                   for(ny = (j-n/2); ny <= (j+n/2); ny++)
                    {
                       i++;
                       srtarr[i] = gpicval[ny][nx];
                       local_sum += gpicval[ny][nx];
                    }
                 local_mean = local_sum/i;          /*local mean calculated*/
                 piksrt(n*n, srtarr);
                 gnew[j][k] = srtarr[n*n/2+1];
```

```
                    X = gpicval[j][k];                          /*original value*/
                    F = gnew[j][k];                             /* the new value*/
                    if((abs(X-local_mean)*local_mean)<t)        /*thresh control*/
                    gnew[j][k] = gpicval[j][k];
                }
            }
        break;
    }/*end of box choice*/

    case 2: {                                                   /* choose to do a plus shaped filter */
        for (k=0;k<gmaxy;k++)
          for (j=0;j<gmaxx;j++)
           {
             if ((k<=n/2-1) || (k>=gmaxy-n/2+1) || (j<=n/2-1) || (j>=gmaxx-n/2+1))
                  gnew[j][k] = gpicval[j][k];
             else
                { i = 0;
                  local_sum = 0;
                  for(nx = (k-n/2); nx <= (k+n/2); nx++)
                     for(ny = (j-n/2); ny <= (j+n/2); ny++)
                     {
                        i++;
                        if ((nx==k) ||( ny==j) )
                        {
                            srtarr[i] = gpicval[ny][nx];
                            local_sum += gpicval[ny][nx];
                        }
                     }
                  local_mean = local_sum/i;          /*local_mean calculated*/
                  piksrt(i, srtarr);

                  gnew[j][k] = srtarr[i/2+1];

                  X = gpicval[j][k];                            /*the original value*/
                  F = gnew[j][k];                               /*the new value*/
                  if((abs(X-local_mean)*local_mean)<t)          /*threshold control*/
                       gnew[j][k] = gpicval[j][k];
                }
           }
        break;
       }
    default: {
      printf("Wrong choice \n");
      exit(1);
       }
 }

/* ____________________________________________________________ */

void piksrt(int n,int arr[])  /* sorts the array arr[] of size 1..n  */
{
 int i,m;
 int a;
```

```
 for (m=2; m<=n; m++)
  {
      a=arr[m];
      i=m-1;
      while (i>0 && arr[i]>a)
            {
            arr[i+1]=arr[i];
            i--;
            }
      arr[i+1]=a;
}
}
```

```
/************************        Program: cwm.c          ************************
 ************************        Programmer: Qing Liu    ************************

*******************************************************************************
* This program was developed by Qing Liu in March, 1998 based upon a program written
* by Dr.David Pitts (which is a template program - targa.c and allows a user to modify the
* image according to his own needs. The output can be displayed on Sun Solairs system.).
*  The algorithm of this program is from Mitra et al (see charpter 2 for the details).
* Function of this program:  CWM filtering
* This program reads a targa format, does CWM filtering for removing implusive noise.
* The output is also in targa format which can be displayed in a lot of softwares.
*******************************************************************************
*/
#include <stdio.h>

/*************************************/
/*      Function Prototypes     */
/*************************************/
void Open_io(char *, char *);           /* open a input and an output file      */
void Read_write_hdr(void);              /* read and write the header files      */
void Pause(void);                       /* pause                                */
void Read_pixels(void);                 /* read in the pixels of an image       */
void Modify_image(void);                /* threshold controled median filtering */
void Write_pixels(void);                /* write the pixels to the output image */
void piksrt(int, int arr[]);            /* sorting pixels in an array           */


/*************************************/
/*      Globals             */
/*************************************/

FILE *fpin,*fpout;
int gpicval[512][512],gnew[512][512],gmaxx,gmaxy;
unsigned char gdata_char;

int main(int argc, char *argv[])
        /* read in a B & W targa file, allows processing, and */
                /* writes out a targa file of the same size as input  */
                /* max size of targa file is 512 lines by 512 pixels */
         /* The user should change the "Modify_image" function in */
         /* order to perform image processing functions     */

{

        Open_io (argv[1],argv[2]);      /* open an input and an output file     */
        Read_write_hdr();               /* read and write the header files      */
        Pause();                        /* pause                                */
        Read_pixels();                  /* read in the pixels of an image       */

        Modify_image();                 /* threshold controled median filtering */

        Write_pixels();                 /* write the pixels to the output image */
}               /* end of main program */
```

```
/* ____________________________________________________________ */

void Open_io(char *file_in,char *file_out)      /* open an input file and an output file */
{
        fpin=fopen(file_in,"r");    /* name of input targa file */

        if (fpin == NULL)
                {
                printf("cannot open input file\n");
                exit(0);
                }
        fpout=fopen(file_out,"w");  /* name of output targa file */
        if (fpout == NULL)
                {
                printf("cannot open output file\n");
                exit(1);
                }
}

/* ____________________________________________________________ */

void Read_write_hdr(void)
{
int j,hdr[18];
     for (j=0;j<18;j++)
          {
          fscanf(fpin,"%c",&gdata_char);      /* read input hdr */
          fprintf(fpout, "%c",gdata_char);  /*write output hdr*/

          hdr[j]=gdata_char;
          printf("location=%d, value=%d\n",j, hdr[j]);
     }


gmaxy=256*hdr[15]+hdr[14]; /* number of lines in image */
printf("\nNumber of lines in the image = %d\n",gmaxy);  /*  k & y  */

gmaxx=256*hdr[13]+hdr[12]; /* number of pixels per line */
printf("\nNumber of pixels per line =%d\n",gmaxx);      /*  j & x  */


if (hdr[16] != 8)
     {
     printf("This image file does not have 8 bit, exeuction halted\n");
     exit(2);
     }
if (gmaxx > 512 || gmaxy > 512)
     {
     printf("Image size greater than 512, execution halted\n");
     exit(3);
     }

}
```

```
/* ____________________________________________________________ */

void Pause(void)              /* wait for carriage return to continue */
{
int k,j;
        printf("\n");
        printf("Press return to continue\n");
        scanf("%c",&gdata_char);
}
/* ____________________________________________________________ */

void Read_pixels(void)
{
int k,j;
        for (k=0;k<gmaxy;k++)
        {
        for (j=0;j<gmaxx;j++)

                {
                fscanf(fpin,"%c",&gdata_char);
                gpicval[j][k]=gdata_char;
                gnew[j][k]=gdata_char;
                }
        /*printf("Line = %d\n",k);*/
        }
}

/* ____________________________________________________________ */


void Write_pixels(void)
{
int k,j;
        for (k=0;k<gmaxy;k++)
        for (j=0;j<gmaxx;j++)
                {
                gdata_char=gnew[j][k];
                fprintf(fpout,"%c",gdata_char);
                }
        fclose(fpin);
        fclose(fpout);
}

/* ____________________________________________________________ */


void Modify_image(void)
{
/* This procedure is to implement cwm filtering with            */
/* a nxn box shaped window or a plus shaped window which is at a user's  */
/* choice. A user will input the varince of noise and also chose the window  */
/* size for the operation.                                      */
```

```
int n;                      /* window (mask) size                          */
int choice;                 /* choice for a box or a plus shaped filter    */
int k,j;                    /* two index variables.                        */
int srtarr[n*n];            /* a sorting array with size of nxn.           */
int nx,ny;                  /* column and row index for nxn box            */
int i;                      /* i is the index for sorting array            */
int X,F,t;                  /* values of original, the new, and threshold */
int local_sum, local_mean; /* local sum and local mean                    */
int m, kx;                  /* m is index, kx is the times of center values */
int variance_noise;         /* the given variance of noise                 */
float variance_data;        /* the given variance of data                  */


    printf("Please enter variance of noise = ");
    scanf("%d", &variance_noise);

    printf("Please enter the value n = ");
    scanf("%d", &n);

    printf("\nEnter 1 for box filter, 2 for cross filter\n");
    printf("Enter your choice now:\n");
    scanf("%d", &choice);
    printf("choice=%d\n",choice);

    switch (choice) {

    case 1: {
        for (k=0;k<gmaxy;k++)
          for (j=0;j<gmaxx;j++)
          {
              if ((k<=n/2-1) || (k>=gmaxy-n/2+1) || (j<=n/2-1) || (j>=gmaxx-n/2+1))
                  gnew[j][k] = gpicval[j][k];
              else
                { i = 0;
                  local_sum = 0;
                  for(nx = (k-n/2); nx <= (k+n/2); nx++)
                     for(ny = (j-n/2); ny <= (j+n/2); ny++)
                      {
                        i++;
                        srtarr[i] = gpicval[ny][nx];
                        local_sum += gpicval[ny][nx];
                      }
                  local_mean = local_sum/i;              /*local mean calculated*/

                  variance_data = 0;                     /* to calculate the variance of data */
                  for(nx = (k-n/2); nx <=(k+n/2); nx++)
                    for(ny= (j-n/2); ny <= (j+n/2); ny++)
                        variance_data += (gpicval[ny][nx]-local_mean)*(gpicval[ny][nx]-local_mean);

                  variance_data = variance_data/i;       /* the variance of data calculated */
/*                      printf("variance_data = %f, local_mean=%d\n", variance_data, local_mean);
*/
```

```
                    if((variance_data - variance_noise) < 0)   /*k is 0, use median filter*/
                    {
                        piksrt(n*n, srtarr);
                        gnew[j][k] = srtarr[n*n/2+1];
                    }
                    else                                    /* the following is to get kx here */
                    {                                       /*the formula: K(i,j) = LR(i,j) for kx*/
                        kx = i/2 *(variance_data -variance_noise)/variance_data;

                        for (m=1; m<=2*kx; m++)         /* copy kx times of center values*/
                            srtarr[i+m] = gpicval[j][k];
                        piksrt(n*n+2*kx, srtarr);
                        gnew[j][k] = srtarr[(n*n+2*kx)/2+1];
                    }

                }
            }
            break;
        }/*end of box choice*/

    case 2: {
            for (k=0;k<gmaxy;k++)
              for (j=0;j<gmaxx;j++)
              {
                if ((k<=n/2-1) || (k>=gmaxy-n/2+1) || (j<=n/2-1) || (j>=gmaxx-n/2+1))
                    gnew[j][k] = gpicval[j][k];
                else
                 {   i = 0;
                    local_sum = 0;
                    for(nx = (k-n/2); nx <= (k+n/2); nx++)
                       for(ny = (j-n/2); ny <= (j+n/2); ny++)
                       {
                            i++;
                            if ((nx==k) ||( ny==j) )
                            {
                               srtarr[i] = gpicval[ny][nx];
                              local_sum += gpicval[ny][nx];
                            }
                       }
                   local_mean = local_sum/i;  /*local_mean calculated*/

                   variance_data = 0;                       /* to calculate the variance of data */
                      for(nx = (k-n/2); nx <=(k+n/2); nx++)
                        for(ny= (j-n/2); ny <= (j+n/2); ny++)
                            variance_data += (gpicval[ny][nx]-local_mean)*(gpicval[ny][nx]-local_mean);

                      variance_data = variance_data/i;          /* the variance of data calculated */
/*                          printf("variance_data = %f, local_mean=%d\n", variance_data, local_mean);
*/

                      if((variance_data - variance_noise) < 0)   /*k is 0, so using median filter*/
                      {
                            piksrt(n*n, srtarr);
                            gnew[j][k] = srtarr[n*n/2+1];
```

```
                    }
                    else                                    /* the following is to get kx here*/
                    {                                       /*the formula: K(i,j) = LR(i,j)*/
                            kx = i/2 *(variance_data -variance_noise)/variance_data;

                            for (m=1; m<=2*kx; m++)         /* copy kx times of center values*/
                                srtarr[i+m] = gpicval[j][k];
                            piksrt(n*n+2*kx, srtarr);
                            gnew[j][k] = srtarr[(n*n+2*kx)/2+1];
                    }

                }
            }
            break;
           }/*end of case 2*/

    default: {
      printf("Wrong choice \n");
      exit(1);
       }

}

/* ____________________________________________________________ */

void piksrt(int n,int arr[])  /* sorts the array arr[] of size 1..n  */
{
 int i,m;
 int a;


 for (m=2; m<=n; m++)
  {
     a=arr[m];
     i=m-1;
     while (i>0 && arr[i]>a)
           {
           arr[i+1]=arr[i];
           i--;
           }
     arr[i+1]=a;
}
}
```

BIBLIOGRAPHY

Abeysekera, Ranil. (1995). Volume Visualization of the Prostate Gland Using 2-D Transrectal ulrasound Images. Master's Thesis. University of Houston - Clear Lake.

Daubechis, Ingrid. (1992). Ten Lectures on Wavelets. Society for Industrial and Applied mathematics, Philadelphia, Pennsylvania.

Donoho, D. (1993). Nonlinear Wavelet Methods for Recovery of Signals, Densities, and Spectra from Indirect and Noisy Data. Different Perspectives on Wavelets, Proceeding of Symposia in Applied Mathematics, Vol 47, I. Pp. 173-205.

Gonzalez, R. C., Woods, R.E. (1992). Digital Image Processing. Addison-Wesley, Reading, Mass.

Graps, Amara. (1995). An Introdution to Wavelets. IEEE Computation Science and Engineering. Vol 2. No. 2.

Houston, A.G., Premkumar, S.B., Pitts, D.E. and Babaian, R.J. (1992). Statistical Interpretation of Texture for Medical Applications. Biomedical Image Processing and Three-Dimensional Microscopy. SPIE (The International Society for Optical Engineering). Vol. 1660, 10-13 February, San Jose, CA, pp. 576-584.

Houston, A.G., Premkumar, S.B., Pitts, D.E. and Babaian, R.J. (1995). Prostate Ultrasound Image Analysis: Localization of Cancer Lesions to Assist Biopsy. Proceedings of the 8th IEEE Symposium Computer-Based Medical Systems, 9-11 June, Lubbock, TX.

Jayawardena, I. N. (1997). Ultrasound Image Calibration For Prostate Cancer Detection. Internship. University of Houston - Clear Lake.

Kaiser, J.F. (1990). On a Simple Algorithm to Calculate the Energy of a Signal. Pros. IEEE Int. Conf.Acoust., Speech, Sig.Processing. Albuquerque, NM, pp.381-384.

Ko, A.J., Lee,Y.H. (1991). Center Weighted Median Filters and Their Applications to Image Enhancement. IEEE Transactions on Circuits and Systems. Vol. 38. No.9. September.

Memorial Sloan-Kettering Cancer Center. (1998). http:// www.mskcc.org/document/wicproqa.htm

Mitra, S.K., Yu, T., Ali, R. (1994). Efficient Detail-Preserving Method of Impulse Noise Removal From Highly Corrupted Images. Image and Video Processing II. SPIE (The International Society for Optical Engineering). Vol.2182.

Tham, Sonny. (1998). http://ciips.ee.uwa.edu.au/~tham_s/us/index.htm.

Yu, T.H., Mitra, S.K., Kaiser, J.F. (1991). A Novel Nonliear Filter For Image Enhancement. Proc.SPIE/SPSE Symp. Elect. Imaging Sci. Tech., San Jose, CA, pp.303-309.