VOL. 11, NO. 13, JULY 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

EFFICIENT ANALYSIS OF MEDICAL IMAGE DE-NOISING FOR MRI AND ULTRASOUND IMAGES

Mohamed Saleh Abuazoumy and Abdirahman Mohamud Shire
Faculty of Electrical and Electronic Engineering, University Tun Hussein Onn Malaysia Parit Raja, Johor, Malaysia
E-Mail: mhmd_abu76@yahoo.com

ABSTRACT

Magnetic resonance imaging (MRI) and ultrasound images have been widely exploited for more truthful pathological changes as well as diagnosis. However, they suffer from a number of shortcomings and these includes: acquisition noise from the equipment, ambient noise from the environment, the presence of background tissue, other organs and anatomical influences such as body fat, and breathing motion. Therefore, noise reduction is very important, as various types of noise generated limits the effectiveness of medical image diagnosis. In this paper, an efficient analysis of MRI and ultrasound modalities was investigated. Three experiments were carried out that include various filters (Median, Gaussian and Wiener filter). To validate the outcomes of medical image de-noising, both objective (quantitative) and subjective (qualitative) tests were used. Since the noise levels are scanner-dependent, this study has underlined several significant parameters to be considered in a generic de-noising algorithm for MRI and ultrasound modalities.

Keywords: magnetic resonance imaging (MRI), ultrasound images, median filter, gaussian filter, wiener filter, peak signal-to-noise ratio (PSNR).

INTRODUCTION

The arrival of digital medical imaging technologies such as positron emission tomography (PET), magnetic resonance imaging (MRI), computerized tomography (CT) and ultrasound imaging has revolutionized modern medicine. Today, many patients no longer need to go through invasive and often dangerous procedures to diagnose a wide variety of illnesses [1-3]. With the widespread use of digital imaging in medicine today, the quality of digital medical images becomes an important issue [4-5]. To achieve the best possible diagnosis, it is important that medical images be sharp, clear, and free of noise and artefacts. While the technologies for acquiring digital medical images continue to improve, resulting in images of higher and higher resolution and quality, removing noise in these digital images remains one of the major challenges in the study of medical imaging, because they could mask and blur important subtle features in the images, many proposed de-noising techniques have their own problems. Image de-noising still remains a challenge for researchers because noise removal introduces artefacts and causes blurring of the images [5].

This paper describes different methodologies for de-noising giving an insight as to which algorithm should be used to find the most reliable estimate of the original image data given its degraded version. Noise modelling in medical images is greatly affected by capturing instruments, data transmission media, image quantization and discrete sources of radiation. Most of images are assumed to have additive random noise which is modelled as a white Gaussian noise; therefore it is required to exterminate a variety of types of de-noising algorithm for MRI and ultrasound modalities.

MATHEMATICAL FORMULATION

Median filter

The median filter is a popular nonlinear digital filtering technique, often used to remove noise. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very widely used in digital image processing because under certain conditions, it preserves edges while removing noise. Which has been shown to be good at removing salt-and-pepper noise in images [6]. Sometimes known as a rank filter, this spatial filter suppresses isolated noise by replacing each pixel's intensity by the median of the intensities of the pixels in its neighbourhood. It is widely used in de-noising and image smoothing applications. Median filters exhibit edge-preserving characteristics (linear methods such as average filtering tends to blur edges), which is very desirable for many image processing applications as edges contain important information for segmenting, labelling and preserving detail in images [7]. This filter may be represented by (1).

$$G(u,v) = median\{I(x,y),(x,y) \in wF\} \quad (1)$$

Where
$wF = w\ x\ w$Filter window with pixel (u,v) as its middle

Gaussian filter

Gaussian filter is a filter whose impulse response is a Gaussian function. Gaussian filters are designed to give no overshoot to a step function input while minimizing the rise and fall time [8]. This behaviour is closely connected to the fact that the Gaussian filter has the minimum possible group delay. Mathematically, a Gaussian filter modifies the input signal by convolution with a Gaussian function; this transformation is also

known as the Weierstrass transform. Smoothing is commonly undertaken using linear filters such as the Gaussian function (the kernel is based on the normal distribution curve), which tends to produce good results in reducing the influence of noise with respect to the image. The 1D and 2D Gaussian distributions with standard deviation for a data point (x) and pixel (x,y), are given by (2) and (3), respectively [9].

$$G(x) = \frac{1}{2\pi\sigma^2}.e^{-\frac{x^2}{2\sigma^2}} \quad (2)$$

$$G(x,y) = \frac{1}{2\pi\sigma^2}.e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3)$$

The kernel could be extended to further dimensions as well. For an image, the 2D Gaussian distribution is used to provide a point-spread; i.e. blurring over neighbouring pixels. This is implemented on every pixel in the image using the convolution operation. The degree of blurring is controlled by the sigma or blurring coefficient, as well as the size of the kernel used (squares with an odd number of pixels; e.g. 3×3, 5×5 pixels, so that the pixel being acted upon is in the middle). The processing can be speeded up by implementing the filtering in the frequency rather than spatial domain, especially for the slower convolution operation (which is implemented as the faster multiplication operation in the frequency domain).

Wiener filter

Wiener filters are a class of optimum linear filters which involve linear estimation of a desired signal sequence from another related sequence. It is not an adaptive filter. The wiener filter's main purpose is to reduce the amount of noise present in an image by comparison with an estimation of the desired noiseless image. The Wiener filter may also be used for smoothing. This filter is the mean squares error-optimal stationary linear filter for images degraded by additive noise and blurring. It is usually applied in the frequency domain (by taking the Fourier transform) [10], due to linear motion or unfocused optics Wiener filter is the most important technique for removal of blur in images. From a signal processing standpoint. Each pixel in a digital representation of the photograph should represent the intensity of a single stationary point in front of the camera. Unfortunately, if the shutter speed is too slow and the camera is in motion, a given pixel will be an amalgram of intensities from points along the line of the camera's motion.

The goal of the Wiener filter is to filter out noise that has corrupted a signal. It is based on a statistical approach. Typical filters are designed for a desired frequency response. The Wiener filter approaches filtering from a different angle. One is assumed to have knowledge of the spectral properties of the original signal and the noise, and one seeks the LTI filter whose output would come as close to the original signal as possible [11]. Wiener filters are characterized by the following:

1. Assumption: signal and (additive) noise are stationary linear random processes with known spectral characteristics.
2. Requirement: the filter must be physically realizable, i.e. Causal (this requirement can be dropped, resulting in a non-causal solution).
3. Performance criteria: minimum mean-square error.
Wiener Filter in the Fourier Domain as in (4).

$$G(u,v) = \frac{H^*(u,v)P_S(u,v)}{|H(u,v)|^2 P_S(u,v)+P_n(u,v)} \quad (4)$$

Where

$H(u,v)$ = Fourier transform of the point spread function

$Ps(u,v)$ = Power spectrum of the signal process, obtained by taking the Fourier transform of the signal autocorrelation

$Pn(u,v)$ = Power spectrum of the noise process, obtained by taking the Fourier transform of the noise autocorrelation

It should be noted that there are some known limitations to Wiener filters. They are able to suppress frequency components that have been degraded by noise, but do not reconstruct them. Wiener filters are also unable to undo blurring caused by band limiting of $H(u,v)$, which is a phenomenon in real-world imaging systems.

Peak Signal-to-Noise ratio

The peak signal-to-noise ratio (PSNR) is the ratio between a signal's maximum power and the power of the signal's noise. Engineers commonly use the PSNR to measure the quality of reconstructed images that have been compressed. Each picture element (pixel) has a colour value that can change when an image is compressed and then uncompressed. Signals can have a wide dynamic range, so PSNR is usually expressed in decibels, which is a logarithmic scale.

The PSNR is most commonly used as a measure of quality of reconstruction of lossy compression codecs(e.g., For image compression). The signal in this case is the original data, and the noise is the error introduced by compression. When comparing compression codecs it is used as an approximation to human perception of reconstruction quality, therefore in some cases, one reconstruction may appear to be closer to the original than another, even though it has a lower PSNR (a higher PSNR would normally indicate that the reconstruction is of higher quality). One has to be extremely careful with the range of validity of this metric; it is only conclusively valid when it is used to compare results from the same codec (or codec type) and same content [12]. It is most easily defined via the mean squared error (MSE), where it denotes the mean square error for two m×n images I (i, j) and I (i, j) where one of the images is considered a noisy approximation of the other and is given by (5),(6).

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad (5)$$

The PSNR is defined by (6)

$$PSNR_{dB} = 10 \cdot log_{10}\left(\frac{MAX_I^2}{MSE}\right) \quad (6)$$

Where, MAXI is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, which is equal to 255. More generally, when samples are represented using linear PCM with B bits per sample, MAXI is $2B-1$.

Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, where higher is better. Acceptable values for wireless transmission quality loss are considered to be about 20 dB to 25 dB [13].

RESEARCH METHODOLOGY

The concept of de-noising filters

The idea of every de-noising filter is different from other filters because every filter has its own function. To give simple explanation of the de-noising filters window 3×3 is used for median, Gaussian and Wiener filter.

How does Median filter work?

The median filter considers each pixel in the image in turn and looks at its nearby neighbours to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the *mean* of neighbouring pixel values, it replaces it with the median of those values [10].

Median filter controls the pepper and Gaussian noises. The median filter is reputed to be edge preserving. The transfer function used here is

$$T(x,y) = I\left(\frac{(n\,x\,n)}{2}\right) \quad (7)$$
$$I1 \leq I2 \leq I3 \leq \ldots \leq In\,x\,n$$

Where $I\,(nxn)\,/\,2$ is the intensity value in the middle position of the sorted array of the neighbouring pixels.

0	0	0
0	12	9
0	22	17

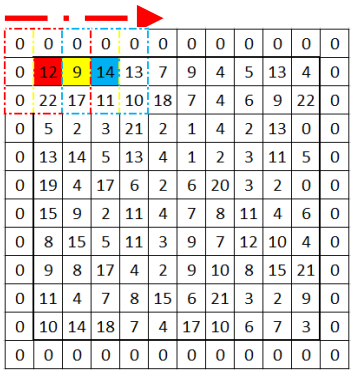
Neighbourhood values are (0, 0, 0, 0, 0, 4, 4, 12, 22)
Median value is 0

Figure-1. Sorting neighbourhood values and determines median value.

The median is calculated by first sorting all the pixel values from the surrounding neighbourhood into numerical order and then replacing the pixel being considered with the middle pixel value as was showed in Figure-1. If the neighbourhood under consideration contains an even number of pixels, the average of the two middle pixel values is used. The pattern of neighbours is called the "window", which slides, pixel by pixel over the entire image.

12	9	14	13	7	9	4	5	13	4
22	17	11	10	18	7	4	6	9	22
5	2	3	21	2	1	4	2	13	0
13	14	5	13	4	1	2	3	11	5
19	4	17	6	2	6	20	3	2	0
15	9	2	11	4	7	8	11	4	6
8	15	5	11	3	9	7	12	10	4
9	8	17	4	2	9	10	8	15	21
11	4	7	8	15	6	21	3	2	9
10	14	18	7	4	17	10	6	7	3

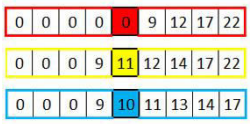
Figure-2. Assumed pixels' window represented on MRI.



0	0	0	0	0	0	0	0	0	0	0	0
0	12	9	14	13	7	9	4	5	13	4	0
0	22	17	11	10	18	7	4	6	9	22	0
0	5	2	3	21	2	1	4	2	13	0	0
0	13	14	5	13	4	1	2	3	11	5	0
0	19	4	17	6	2	6	20	3	2	0	0
0	15	9	2	11	4	7	8	11	4	6	0
0	8	15	5	11	3	9	7	12	10	4	0
0	9	8	17	4	2	9	10	8	15	21	0
0	11	4	7	8	15	6	21	3	2	9	0
0	10	14	18	7	4	17	10	6	7	3	0
0	0	0	0	0	0	0	0	0	0	0	0

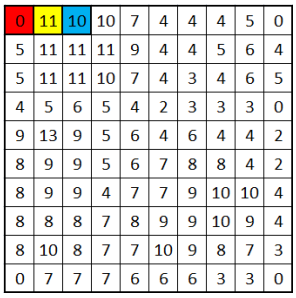
Figure-3. The movement of the window 3×3 (mask) on the pixels.

Median filtering using a 3x3 sampling window with the extending border values outside with 0s



0	0	0	0	0	9	12	17	22
0	0	0	9	11	12	14	17	22
0	0	0	9	10	11	13	14	17

Figure-4. Sorting the pixels and determining middle pixel value.



0	11	10	10	7	4	4	4	5	0
5	11	11	11	9	4	4	5	6	4
5	11	11	10	7	4	3	4	6	5
4	5	6	5	4	2	3	3	3	0
9	13	9	5	6	4	6	4	4	2
8	9	9	5	6	7	8	8	4	2
8	9	9	4	7	7	9	10	10	4
8	8	8	7	8	9	9	10	9	4
8	10	8	7	7	10	9	8	7	3
0	7	7	7	6	6	6	3	3	0

Figure-5. Pixels window after applying Median filter on all pixels.

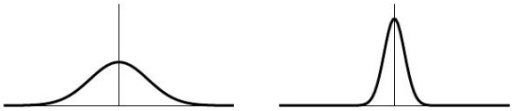
How does Gaussian filter work?

Gaussian filter are a class of low-pass filter, all based on the Gaussian probability distribution function

used to blur images and remove noise and detail. In one dimension, the Gaussian function is:

$$G(x) = \frac{1}{2\pi\sigma^2}.e^{-\frac{x^2}{2\sigma^2}} \quad (8)$$

Where σ is the standard deviation of the distribution the distribution is assumed to have a mean of 0. Shown graphically, we see the familiar bell shaped Gaussian distribution, where a large value of σ produces to a flatter curve, and a small value leads to a "pointier" curve. Figure-6 shows examples of such one dimensional Gaussians [14].

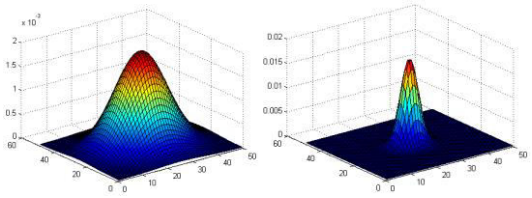


Large value of σ Small value ofσ

Figure-6. One dimensional Gaussians.

When working with images we need to use the two dimensional Gaussian function Figure-7. This is simply the product of two 1D Gaussian functions (one for each direction) and is given by:

$$G(x,y) = \frac{1}{2\pi\sigma^2}.e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (9)$$



$\sigma = 9\sigma = 3$

Figure-7. Two dimensional Gaussians.

The Gaussian filter works by using the 2D distribution as a point-spread function. This is achieved by convolving the 2D Gaussian distribution function with the image. Before we can perform the convolution a collection of discrete pixels we need to produce a discrete approximation to the Gaussian function. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point. The kernel coefficients diminish with increasing distance from the kernel's centre. Central pixels have a higher weighting than those on the periphery. Larger values of σ produce a wider peak (greater blurring). Kernel size must increase with increasing σ to maintain the Gaussian nature of the filter. Gaussian kernel coefficients depend on the value of σ. Figure-8 shows a different convolution kernel that approximates a Gaussian with σ.

0.063	0.125	0.063	0.075	0.124	0.075	0.102	0.115	0.102
0.125	0.250	0.125	0.124	0.204	0.124	0.115	0.131	0.115
0.063	0.125	0.063	0.075	0.124	0.075	0.102	0.115	0.102

Figure-8. 3×3 windows with $\sigma = 0.849$, $\sigma = 1$, $\sigma =$. The idea of these windows distributions comes from

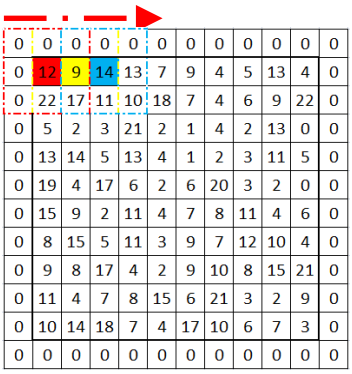
$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 0.063 & 0.125 & 0.063 \\ 0.125 & 0.250 & 0.125 \\ 0.063 & 0.125 & 0.063 \end{bmatrix}$$

Figure-9. 3×3 windows with $\sigma = 0.849$.

Where $\sigma = 0.849$& 16 is the summation of the values in 3×3 window with $\sigma = 0.849$ this window is usingto explain how Gaussian filter is working by convolute it on the pixels of MRI.

12	9	14	13	7	9	4	5	13	4
22	17	11	10	18	7	4	6	9	22
5	2	3	21	2	1	4	2	13	0
13	14	5	13	4	1	2	3	11	5
19	4	17	6	2	6	20	3	2	0
15	9	2	11	4	7	8	11	4	6
8	15	5	11	3	9	7	12	10	4
9	8	17	4	2	9	10	8	15	21
11	4	7	8	15	6	21	3	2	9
10	14	18	7	4	17	10	6	7	3

Figure-10. Assumed pixels window represented on MRI.



0	0	0	0	0	0	0	0	0	0	0	0
0	12	9	14	13	7	9	4	5	13	4	0
0	22	17	11	10	18	7	4	6	9	22	0
0	5	2	3	21	2	1	4	2	13	0	0
0	13	14	5	13	4	1	2	3	11	5	0
0	19	4	17	6	2	6	20	3	2	0	0
0	15	9	2	11	4	7	8	11	4	6	0
0	8	15	5	11	3	9	7	12	10	4	0
0	9	8	17	4	2	9	10	8	15	21	0
0	11	4	7	8	15	6	21	3	2	9	0
0	10	14	18	7	4	17	10	6	7	3	0
0	0	0	0	0	0	0	0	0	0	0	0

Figure-11. The movement of the window 3 x 3 (mask) on the pixels.

To achieve the function of Gaussian filter, the 2D Gaussian distribution of 3×3 window with $\sigma =$ 0.849 must be convolved on the first window of MRI pixels to get first value of filtered pixel.

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 \\ 0 & 12 & 9 \\ 0 & 22 & 17 \end{bmatrix}$$

Figure-12. 3×3 windows with σ = 0.849, first window of MRI pixels.

$$1st\ P = \frac{1\times 0 + 2\times 0 + 1\times 0 + 2\times 0 + 4\times 12 + 2\times 9 + 1\times 0 + 2\times 22 + 1\times 17}{16}$$

$$1st\ filtered\ pixel = \frac{1127}{16} = 7.9375 = 8$$

$$2nd\ P = \frac{1\times 0 + 2\times 0 + 1\times 0 + 2\times 12 + 4\times 9 + 2\times 14 + 1\times 22 + 2\times 17 + 1\times 11}{16}$$

$$2nd\ filtered\ pixel = \frac{155}{16} = 9.6875 = 10$$

$$3rd\ P = \frac{1\times 0 + 2\times 0 + 1\times 0 + 2\times 9 + 4\times 14 + 2\times 13 + 1\times 17 + 2\times 11 + 1\times 10}{16}$$

$$3rd\ filtered\ pixel = \frac{149}{16} = 9.3125 = 9$$

8	10	9	9	8	6	4	5	7	6
10	12	11	12	11	7	5	6	10	9
8	9	9	11	8	4	3	5	8	6
8	9	9	9	5	4	5	5	6	4
10	11	9	8	5	6	9	7	4	3
9	10	8	7	6	7	10	9	6	3
8	10	9	7	6	7	9	10	10	7
7	10	10	8	6	8	10	9	11	10
7	9	10	9	9	11	11	8	7	7
6	9	9	7	7	9	9	5	4	3

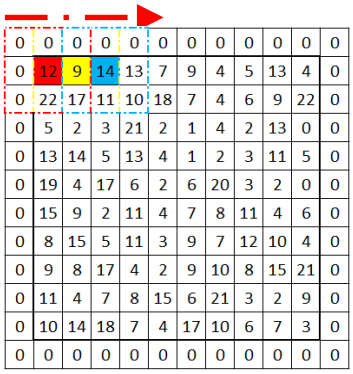
Figure-13. Pixels window after applying Gaussian filter on all pixels.

How does Wiener filter work?

Wiener filters are a class of optimum linear filters which involve linear estimation of a desired signal sequence from another related sequence. In this filter the same window of MRI pixels will be used to illustrate howthe Wiener filter is working [15].

12	9	14	13	7	9	4	5	13	4
22	17	11	10	18	7	4	6	9	22
5	2	3	21	2	1	4	2	13	0
13	14	5	13	4	1	2	3	11	5
19	4	17	6	2	6	20	3	2	0
15	9	2	11	4	7	8	11	4	6
8	15	5	11	3	9	7	12	10	4
9	8	17	4	2	9	10	8	15	21
11	4	7	8	15	6	21	3	2	9
10	14	18	7	4	17	10	6	7	3

Figure-14. Assumed pixels window represented on MRI.



0	0	0	0	0	0	0	0	0	0	0	0
0	12	9	14	13	7	9	4	5	13	4	0
0	22	17	11	10	18	7	4	6	9	22	0
0	5	2	3	21	2	1	4	2	13	0	0
0	13	14	5	13	4	1	2	3	11	5	0
0	19	4	17	6	2	6	20	3	2	0	0
0	15	9	2	11	4	7	8	11	4	6	0
0	8	15	5	11	3	9	7	12	10	4	0
0	9	8	17	4	2	9	10	8	15	21	0
0	11	4	7	8	15	6	21	3	2	9	0
0	10	14	18	7	4	17	10	6	7	3	0
0	0	0	0	0	0	0	0	0	0	0	0

Figure-15. The movement of the window 3 x 3 (mask) on the pixels.

By applying Wiener filter, the first window of all the pixels Figure-16 will be calculated by estimating the local mean and variance around each pixel to find the value of the central window.

0	0	0
0	12	9
0	22	17

Figure-16. First window of the pixels.

To find the value of the central window, $\mu$by calculating the equation (9)

$$\mu = \frac{1}{NM}\sum_{n_1,n_2\in\eta} a(n_1,n_2) \quad (9)$$

$$\mu = \frac{1}{3*3}[0+0+0+0+12+9+0+22+17]$$

$$\mu = \frac{1}{9}[60] = \frac{60}{9} = 6.6667$$

$$\sigma^2 = \frac{1}{NM}\sum_{n_1,n_2\in\eta} a^2(n_1,n_2) - \mu^2 \quad (10)$$

$$\sigma^2 = \frac{1}{9}[(0^2-(6.6667)^2)+(0^2-(6.6667)^2)+(0^2-(6.6667)^2) + (0^2-(6.6667)^2)+(12^2-(6.6667)^2)+(9^2 - (6.6667)^2)+(0^2-(6.6667)^2)+(22^2 - (6.6667)^2)+(17^2-(6.6667)^2)]$$

$$\sigma^2 = 66.4440$$

Where is the *N*-by-*M* local neighborhood of each pixel in the image then creates a pixelwise Wiener filter using these estimates,

$$b(n_1,n_2) = \mu + \frac{\sigma^2 - v^2}{\sigma^2}(a(n_1,n_2) - \mu^2) \quad (11)$$

Where v^2 is the noise variance. In this 3×3 window the noise variance is 30.9790 which is calculated by Matlab according to the pixels input.

$$b = 6.6667 + \frac{66.4440 - 30.9790}{66.4440} \times (12 - 6.6667) = 9.5134 = 10$$

This is the value of the first pixel after Wiener filter calculation, and then we move to the right one step to find the second pixel.

$$\mu = \frac{1}{3*3}[0+0+0+12+9+14+22+17+11] = 9.4444$$

$$\sigma^2 = \frac{1}{9}[(0^2-(9.4444)^2)+(0^2-(9.4444)^2)+(0^2-(9.4444)^2)+(12^2-(9.4444)^2)+(9^2-(9.4444)^2)+(14^2-(9.4444)^2)+(22^2-(9.4444)^2)+(17^2-(9.4444)^2)+(11^2-(9.4444)^2)] = 56.9144$$

$$b = 9.4444 + \frac{56.9144-30.9790}{56.9144} \times (9-9.4444) = 9.2419 = (9)$$

We move to the right one step to find the Third pixel.

$$\mu = \frac{1}{3*3}[0+0+0+9+14+13+17+11+10] = 8.2222$$

$$\sigma^2 = \frac{1}{9}[(0^2-(8.2222)^2)+(0^2-(8.2222)^2)+(0^2-(8.2222)^2)+(9^2-(8.2222)^2)+(14^2-(8.2222)^2)+(13^2-(8.2222)^2)+(17^2-(8.2222)^2)+(11^2-(8.2222)^2)+(10^2-(8.2222)^2)] = 38.6176$$

$$b = 8.2222 + \frac{38.6176-30.9790}{38.6176} \times (14-8.2222) = 9.3651 = (9)$$

10	9	9	9	7	5	4	5	9	5
14	12	11	11	12	6	5	7	8	14
7	8	9	12	6	5	3	6	9	4
8	10	8	10	6	5	5	6	4	3
12	10	9	7	6	6	7	7	5	3
10	10	9	7	7	7	9	9	6	3
7	10	9	7	7	7	9	9	10	6
6	9	9	8	7	9	9	10	10	12
6	11	10	9	8	10	10	9	8	7
4	9	8	7	6	12	8	6	3	2

Figure-17. Pixels window after applying Wiener filter on all pixels.

PSNR Calculation

Define the mean squared error (MSE) between two monochromatic images, where one image is considered to be an approximation of the other. The MSE can be described as the mean of the square of the differences in the pixel values between the corresponding pixels of the two images [16].

The MSE can be calculated using equation (12).

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j)-K(i,j)]^2 \quad (12)$$

Where I and K are matrices that represent the images being compared. The two summations are performed for the dimensions "i" and "j." Therefore I(i,j) represents the value of pixel (i,j) of image I. Determine the maximum possible value of the pixels in image I. Typically, this may be given as (2^n) - 1 where n is the number of bits that represent the pixel. Thus, an 8-bit pixel would have a maximum value of (2^8) - 1 = 255. Let the maximum value for pixels in image I be MAX_I.

Express the PSNR in decibels.

$$PSNR_{dB} = 10 \cdot log_{10}\left(\frac{MAX_I^2}{MSE}\right) \quad (13)$$

$$PSNR_{dB} = 20 \cdot log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \quad (14)$$

Description of methodology

The description of working in order to reduce the noise medical images (ultrasound image and MRI), is illustrated by referring to image de-noising chart which represented in Figure-18.

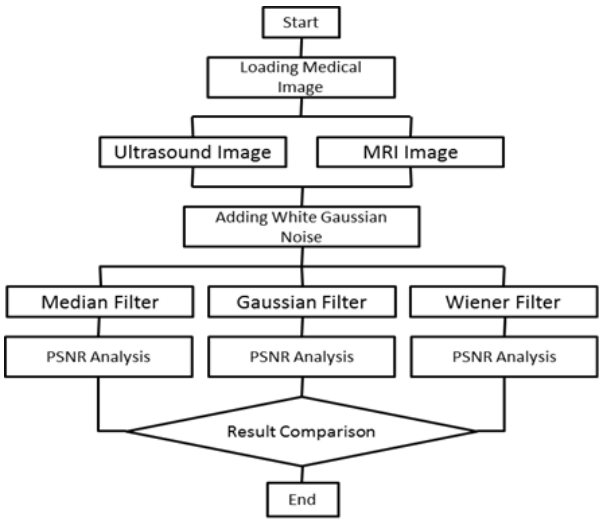
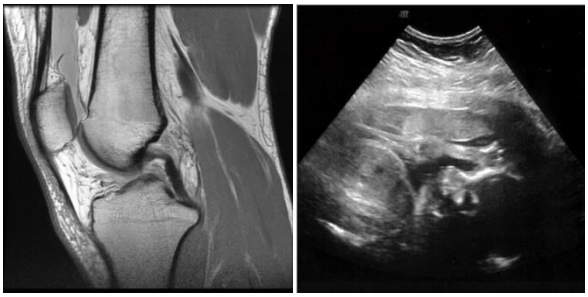


Figure-18. Image De-noising chart.

The methodology of this study requires collecting medical image data which include two types of modalities MRI and ultrasound image which are respectively in Figure-19 (a) and (b).



(a) (b)

Figure-19. MRI Image (a) and Ultrasound image (b).

Median filter implementation

Applying median filter on MRI

1. Loading MRI image in the command window of the matlab software to perform the needed operations as shown in Figure-20.

Figure-20. MRI image.

2. Generate White Gaussian Noise on the MRI image to become noisy image as depicted in Figure-21.

Figure-21. Noisy MRI image.

3. Applying Median filter on the MRI image with different parameters to determine determine the reliable size of the window (kernel) which including the sizes from (1×1) to (6×6).

4. Calculating PSNR for noisy MRI Image and de-noised MRI with every windows size from (1×1) to (6×6).

5. Comparing PSNR result of noisy MRI with every PSNR result and determine whether we got better PSNR or no, which is known by getting PSNR result greater than PSNR result of noisy MRI.

Applying median filter on Ultrasound image

1. Loading ultrasound image in the command window of the matlab software to perform the needed operations as shown in Figure-22.

Figure-22. Ultrasound image.

2. Generate White Gaussian Noise on the Ultrasound Image to become noisy image as depicted in Figure-23.

Figure-23. Noisy ultrasound image.

3. Applying Median filter on the Ultrasound Image with different parameters to determine the reliable size of the window (kernel) which including the sizes from (1×1) to (6×6).

4. Calculating PSNR for noisy Ultrasound Image and de-noised Ultrasound Image with every windows size from (1×1) to (6×6).

5. Comparing PSNR result of noisy Ultrasound Image with every PSNR result and determine whether we got better PSNR or no, which is known by getting PSNR result greater than PSNR result of noisy Ultrasound.

Gaussian filter implementation

Applying Gaussian filter on MRI

1. Loading MRI image in the command window of the matlab software to perform the needed operations as shown in Figure-24.

Figure-24. MRI Image.

2. Generate White Gaussian Noise on the MRI image to become noisy image as depicted in Figure-25.

Figure-25. Noisy MRI Image.

3. Applying Gaussian filter on the MRI image with different parameters to determine determine the

reliable size of the window (kernel) which including the sizes from (1×1) to (6×6) and standard deviation σ.

4. Calculating PSNR for noisy MRI Image and de-noised MRI with every windows size from (1×1) to (6×6) with different standard deviation σ.

5. Comparing PSNR result of noisy MRI with every PSNR result and determine whether we got better PSNR or no, which is known by getting PSNR result greater than PSNR result of noisy MRI.

Applying Gaussian filter on Ultrasound image

1. Loading ultrasound image in the command window of the matlab software to perform the needed operations as shown in Figure-26.

Figure-26. Ultrasound Image.

2. Generate White Gaussian Noise on the MRI image to become noisy image as depicted in Figure-27.

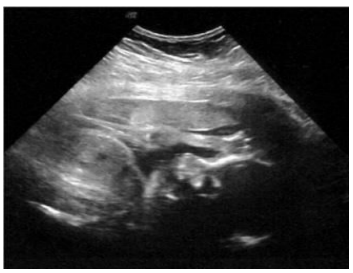


Figure-27. Noisy Ultrasound Image.

3. Applying Gaussian filter on the Ultrasound image with different parameters to determine determine the reliable size of the window (kernel) which including the sizes from (1×1) to (6×6) and standard deviation σ.

4. Calculating PSNR for noisy Ultrasound Image and de-noised Ultrasound Image with every windows size from (1×1) to (6×6) with different standard deviation σ.

5. Comparing PSNR result of noisy Ultrasound Image with every PSNR result and determine whether we got better PSNR or no, which is known by getting PSNR result greater than PSNR result of noisy MRI.

Wiener filter implementation

Applying Wiener filter on MRI

1. Loading MRI image in the command window of the matlab software to perform the needed operations as shown in Figure-28.

Figure-28. MRI Image.

2. Generate noise like White Gaussian Noise on the MRI image to become noisy image as depicted in Figure-29.

Figure-29. Noisy MRI Image.

3. Applying Wiener filter on the MRI image with different parameters to determine determine the reliable size of the window (kernel) which including the sizes from (1×1) to (6×6).

4. Calculating PSNR for noisy MRI Image and de-noised MRI with every windows size from (1×1) to (6×6).

5. Comparing PSNR result of noisy MRI with every PSNR result and determine whether we got better PSNR or no, which is known by getting PSNR result greater than PSNR result of noisy MRI.

Applying Wiener filter on Ultrasound image

1. Loading ultrasound image in the command window of the matlab software to perform the needed operations as shown in Figure-30.

Figure-30. Ultrasound Image.

2. Generate White Gaussian Noise on the Ultrasound Image to become noisy image as depicted in Figure-31.

Figure-31. Noisy Ultrasound Image.

3. Applying Wiener filter on the Ultrasound Image with different parameters to determine the reliable size of the window (kernel) which including the sizes from (1×1) to (6×6).

4. Calculating PSNR for noisy Ultrasound Image and de-noised Ultrasound Image with every windows size from (1×1) to (6×6).

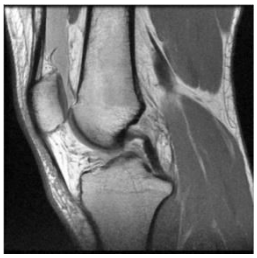
5. Comparing PSNR result of noisy Ultrasound Image with every PSNR result and determine whether we got better PSNR or no, which is known by getting PSNR result greater than PSNR result of noisy Ultrasound Image.

RESULTS AND ANALYSIS

The result of Median filter on MRI and ultrasound

1. PSNR results of de-noised MRI by Median filter from window size (1×1) to (6×6) which show this window size (3×3) is the best with PSNR=14.01 Figure-32a, because it is greater than other PSNR result and noisy MRI which is (9.19).

[1 1]	[1 2]	[1 3]	[1 4]	[1 5]	[1 6]
9.19	10.72	11.73	11.45	11.83	10.91
[2 1]	[2 2]	[2 3]	[2 4]	[2 5]	[2 6]
11.45	11.25	13.29	11.72	12.76	11.03
[3 1]	[3 2]	[3 3]	[3 4]	[3 5]	[3 6]
11.75	12.06	14.01	12.30	13.18	11.39
[4 1]	[4 2]	[4 3]	[4 4]	[4 5]	[4 6]
12.37	11.73	13.65	11.81	12.78	10.98
[5 1]	[5 2]	[5 3]	[5 4]	[5 5]	[5 6]
11.94	11.68	13.30	11.66	12.40	10.84
[6 1]	[6 2]	[6 3]	[6 4]	[6 5]	[6 6]
12.16	11.36	12.95	11.30	12.11	10.54



(a) **(b)**

Figure-32. (a) PSNR results of de-noised MRI Image by Median filter with windows size from (1×1) to (6×6), (b) De-noised MRI by Median filter PSNR= 14.01.

2. PSNR results of de-noised Ultrasound Image by Median filter from window size (1×1) to (6×6) which show this window size (3×3) is the best with PSNR=14.98 Figure-33a, because it is greater than other PSNR result and noisy Ultrasound Image which is (11.77).

[1 1]	[1 2]	[1 3]	[1 4]	[1 5]	[1 6]
11.78	13.96	13.81	14.47	14.01	14.23
[2 1]	[2 2]	[2 3]	[2 4]	[2 5]	[2 6]
13.17	13.29	13.94	13.60	13.72	13.36
[3 1]	[3 2]	[3 3]	[3 4]	[3 5]	[3 6]
13.49	14.66	14.98	14.79	14.48	14.25
[4 1]	[4 2]	[4 3]	[4 4]	[4 5]	[4 6]
13.20	13.24	13.70	13.30	13.28	12.90
[5 1]	[5 2]	[5 3]	[5 4]	[5 5]	[5 6]
12.75	13.44	13.53	13.39	13.07	12.91
[6 1]	[6 2]	[6 3]	[6 4]	[6 5]	[6 6]
12.07	12.09	12.35	12.05	12.00	11.71

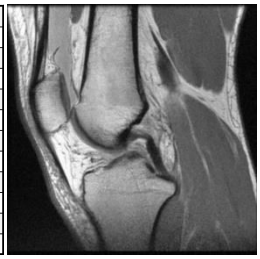
(a) **(b)**

Figure-33. (a) PSNR results of de-noised ultrasound Image by Median filter with windows size from (1×1) to (6×6), (b) De-noised ultrasound Image by Median filter PSNR= 14.98.

The result of Gaussian filter on MRI and ultrasound

1. PSNR results of de-noised MRI by Gaussian filter from window size (1×1) to (6×6) which show this window size (3×3) is the best with PSNR=15.53 with standard deviation $\sigma = 0.849$ Figure-34a, because it is greater than other PSNR result and noisy MRI which is (9.19).

[1 1]	[1 2]	[1 3]	[1 4]	[1 5]	[1 6]
9.24	10.43	13.01	11.39	13.24	11.45
[2 1]	[2 2]	[2 3]	[2 4]	[2 5]	[2 6]
11.16	11.56	13.63	12.12	13.67	12.13
[3 1]	[3 2]	[3 3]	[3 4]	[3 5]	[3 6]
13.00	12.51	15.53	12.97	15.40	12.97
[4 1]	[4 2]	[4 3]	[4 4]	[4 5]	[4 6]
12.29	12.15	14.21	12.51	14.14	12.50
[5 1]	[5 2]	[5 3]	[5 4]	[5 5]	[5 6]
13.30	12.60	15.52	12.98	15.36	12.97
[6 1]	[6 2]	[6 3]	[6 4]	[6 5]	[6 6]
12.36	12.17	14.22	12.51	14.14	12.50



(a) **(b)**

Figure-34. (a) PSNR results of de-noised MRI Image by Gaussian filter with windows size from (1×1) to (6×6),$\sigma = 0.849$ (b) De-noised MRI by Median filter PSNR= 15.53

2. PSNR results of de-noised Ultrasound Image by Gaussian filter from window size (1×1) to (6×6) which show this window size (3×3) is the best with PSNR=17.37 Figure-35a, because it is greater than other PSNR result and noisy Ultrasound Image which is (11.77).

[1 1]	[1 2]	[1 3]	[1 4]	[1 5]	[1 6]
11.77	13.61	15.55	14.75	15.86	14.83
[2 1]	[2 2]	[2 3]	[2 4]	[2 5]	[2 6]
12.85	13.86	14.89	14.44	15.00	14.47
[3 1]	[3 2]	[3 3]	[3 4]	[3 5]	[3 6]
15.24	15.71	17.37	16.21	17.39	16.22
[4 1]	[4 2]	[4 3]	[4 4]	[4 5]	[4 6]
13.76	14.33	15.22	14.69	15.25	14.70
[5 1]	[5 2]	[5 3]	[5 4]	[5 5]	[5 6]
15.35	15.66	17.15	16.06	17.15	16.07
[6 1]	[6 2]	[6 3]	[6 4]	[6 5]	[6 6]
13.81	14.33	15.21	14.68	15.23	14.69

(a) **(b)**

Figure-35. (a) PSNR results of de-noised Ultrasound Image by Gaussian filter with windows size from (1×1) to (6×6),$\sigma = 0.849$, (b) De-noised Ultrasound Image by Gaussian filter PSNR= 17.37.

The result of Wiener filter on MRI and ultrasound

1. PSNR results of de-noised MRI by Wiener filter from window size (1×1) to (6×6) which show this window size (3×3) is the best with PSNR=15.35 Figure 36a, because it is greater than other PSNR result and noisy MRI which is (9.19).

[1 1]	[1 2]	[1 3]	[1 4]	[1 5]	[1 6]
-19.70	11.03	12.83	13.16	13.32	12.86
[2 1]	[2 2]	[2 3]	[2 4]	[2 5]	[2 6]
11.04	12.61	14.03	13.84	13.70	13.10
[3 1]	[3 2]	[3 3]	[3 4]	[3 5]	[3 6]
12.58	13.67	15.35	14.52	14.38	13.51
[4 1]	[4 2]	[4 3]	[4 4]	[4 5]	[4 6]
13.05	13.66	14.73	14.00	13.71	13.03
[5 1]	[5 2]	[5 3]	[5 4]	[5 5]	[5 6]
13.34	13.59	14.58	13.82	13.59	12.86
[6 1]	[6 2]	[6 3]	[6 4]	[6 5]	[6 6]
13.25	13.40	14.07	13.40	13.16	12.52

(a)

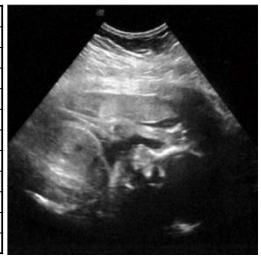
(b)

Figure-36. (a) PSNR results of de-noised MRI Image by Wiener filter with windows size from (1×1) to (6×6), (b) De-noised MRI by Wiener filter PSNR= 15.35.

2. PSNR results of de-noised Ultrasound Image by Wiener filter from window size (1×1) to (6×6) which show this window size (3×3) is the best with PSNR=17.27 Figure-37a, because it is greater than other PSNR result and noisy Ultrasound Image which is (11.77).

[1 1]	[1 2]	[1 3]	[1 4]	[1 5]	[1 6]
-14.89	13.57	14.96	15.57	15.94	16.03
[2 1]	[2 2]	[2 3]	[2 4]	[2 5]	[2 6]
13.44	14.90	15.88	16.09	16.20	16.14
[3 1]	[3 2]	[3 3]	[3 4]	[3 5]	[3 6]
15.02	16.19	17.27	17.02	17.11	16.85
[4 1]	[4 2]	[4 3]	[4 4]	[4 5]	[4 6]
15.24	15.90	16.59	16.33	16.30	16.11
[5 1]	[5 2]	[5 3]	[5 4]	[5 5]	[5 6]
15.43	15.90	16.50	16.24	16.21	15.98
[6 1]	[6 2]	[6 3]	[6 4]	[6 5]	[6 6]
15.04	15.36	15.72	15.50	15.49	15.31

(a)



(b)

Figure-37. (a) PSNR results of de-noised Ultrasound Image by Wiener filter with windows size from (1×1) to (6×6), (b) De-noised Ultrasound Image by Wiener filter PSNR= 17.27.

Table-1. PSNR results of applying Median, Wiener and Gaussian filter on 4 ultrasound images (US1, US2, US3) and 4 MRI images (MRI1, MRI2, MRI3).

	Ultrasound Image			MRI		
	US1	US2	US3	MRI1	MRI2	MRI3
	PSNR	PSNR	PSNR	PSNR	PSNR	PSNR
	11.78dB	11.34dB	11.33dB	9.19dB	11.65dB	9.28dB
Median Filter	14.98dB	15.11dB	15.31dB	14.01dB	16.07dB	12.27dB
Wiener Filter	17.27dB	16.73dB	16.63dB	15.35dB	18.63dB	14.15dB
Gaussian Filter	17.37dB	17.01dB	16.85dB	15.53dB	18.88dB	14.26dB

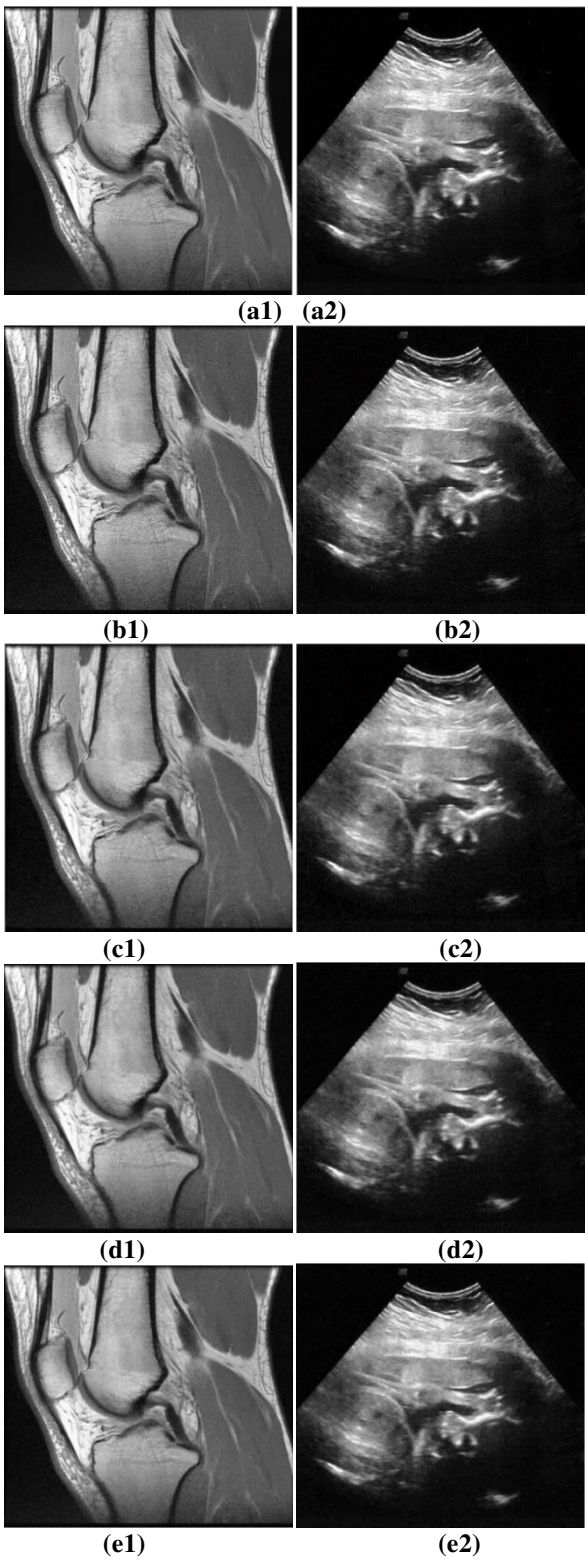


Figure-38. (a1) Original MRI, (a2) Original ultrasound image, (b1) Noisy MRI by White Gaussian Noise, (b2) Noisy ultrasound image by White Gaussian Noise, (c1) De-noised MRI by Median filter, (c2) De-noised ultrasound image by Median filter, (d1) De-noised MRI by Gaussian filter, (d2) De-noised ultrasound image by Gaussian filter, (e1) De-noised MRI by Wiener filter, (e2) De-noised ultrasound image by Wiener filter.

By referring to Table-1 and comparing PSNR results of noisy images with PSNR results of de-noised images, also by looking to Figure-38 and comparing noisy MRI (b1) with de-noised MRI by Median, wiener and Gaussian filter (c1), (d1) and (e1) it can observe that the noise in the de-noised MRI is reduced and the image become better, then comparing noisy ultrasound image (b2) with de-noised ultrasound image by Median, wiener and Gaussian filter (c2), (d2) and (e2) it can observe that the noise in the de-noised ultrasound image is reduced and the image become better.

CONCLUSIONS

This study shows one of the medical images de-noising techniques by applying three filters (Medan, Gaussian and Wiener filter) on the two modalities (MRI and ultrasound Image). Consequently, after calculating and comparing Peak Signal to Noise Ratio (PSNR) values between a noisy image with every due-noised image. It can be observed that Gaussian filter is reliable for both of images MRI and ultrasound image.

REFERENCES

[1] Y. Wang and H. Zhou. 2006. Total variation wavelet-based medical image de-noising. School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332-0430, USA.

[2] S. Sudha, G. Suresh and R Sukanesh. 2009. Speckle noise reduction in ultrasound images using context-based adaptive wavelet thresholding. IETE Journal. 55: 135-143.

[3] A. Achim and A. Bezerianos. 2001. Novel Bayesian Multiscale Method for Speckle Removal in Medical Ultrasound Images. IEEE Transactions on Medical Imaging. 20(8): 772-783.

[4] A. M Shire and F. C. Seman. 2013. Effects of dielectric substrate on performance of UWB Archimedean Spiral Antenna. Space Science and Communication (IconSpace). pp. 412-415.

[5] D. Healy and J. Weaver. 1992. Two applications of wavelet transforms in magnetic resonance imaging. IEEE Trans. Info. Theory. 38(2): 840-860.

[6] A. M. Shire and F. C. Seman. Analysis of the Active Region of Archimedean Spiral Antenna. Chapter 24, Notes Electrical Eng., *et al.* Advanced Computer and Communication Engineering Technology. 315: 231-239. Springer International Publishing Switzerland.

[7] M. Okumura, O. Takamasa, and S. Tsukagoshi. 2006. New Method of Evaluating Edge-preserving Adaptive Filters for Computed Tomography (CT). Digital Phantom Method. Jpn J. Radiol Technol. 62(7): 971-978.

[8] H. Raymond Chan, Chung-WaHo, and M. Nikolova. 2005. Salt-and-Pepper noise removal by median-type noise detectors and detail-preserving regularization. IEEE Trans Image Process. 14: 1479-1485.

[9] R. Fisher, S. Perkins, A. Walker, E. Wolfart. 2003. Gaussian smoothing, Hypermedia image processing reference (HIPR2). Available from: URL: http:// homepages .inf. ed.ac.uk / rbf/ HIPR2/gsmooth.htm.

[10] S. Kumar, P. Kumar, M. Gupta and A. K. Nagawat. 2010. Performance Comparison of Median and Wiener Filter in Image De-noising. International Journal of Computer Applications. 12(4): 27-31.

[11] Kazubek. 2003. Wavelet domain image de-noising by thresholding and Wiener filtering. M. Signal Processing Letters IEEE. 10(11): 265, Vol.3.

[12] Q. Huynh-Thu and M. Ghanbari. 2008. Scope of validity of PSNR in image/video quality assessment. Electronics Letters. 44 (13): 800-801.

[13] N. Thomos, V. Boulgouris, M. G. Strintzis. 2006. Optimized Transmission of JPEG2000 Streams Over Wireless Channels. IEEE Transactions on Image Processing. 15(1).

[14] P. Hsiao1, S. Chou and F. Huang. 2007. Generic 2-D Gaussian Smoothing Filter for Noisy Image Processing. National University of Kaohsiung, Kaohsiung, Taiwan.

[15] H. Yoshino, C. Dong, Y. Washizawa and Y. Yamashita. 2010. Kernel Wiener Filter and its Application to Pattern Recognition. IEEE Transactions on neural networks.

[16] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. Sullivan. 2003. Rate-constrained coder control and comparison of video coding standards. IEEE Trans. Circuits Syst. Video Technol. 13(7): 688-703.