IMAGE QUALITY EVALUATION OF ULTRASOUND IMAGES USING COMPUTER SIMULATION

by

A. Levent Kurtoğlu

Yıldız Technical University, Electrical Engineering,
Faculty of Electrical and Electronic Engineering, 2004

Submitted to the Institute of Biomedical Engineering
in partial fulfillment of the requirements
for the degree of
Master of Science
in
Biomedical Engineering

Boğaziçi University
2010

IMAGE QUALITY EVALUATION OF ULTRASOUND IMAGES USING COMPUTER SIMULATION

APPROVED BY:

Assoc. Prof. Dr. Albert Güveniş .....
(Thesis Advisor)

Prof. Dr. Yekta Ülgen .....

Assoc. Prof. Dr. İbrahim Adaletli .....

DATE OF APPROVAL: 07 January 2010

ACKNOWLEDGMENTS

Hereby, I would like to thank to several people because of their physical and motivational support and contributions throughout the completion of this thesis work:

My thesis advisor and a dear person Prof. Dr. Albert Güveniş, Prof. Dr. Sebuh Kuruoğlu, Istanbul University Cerrahpaşa Medical Faculty, Radiology Institute, Dr. Ali Özgümüş, my friends, Hilmi Murat Çakır, Efe Duyan, Sinem Balta, Gönül Uludağ and Mustafa Kellekçi, my father Ahmet Kurtoğlu and Emine Deniz Kurtoğlu, Prof. Dr. Haluk Çokuğraş, Istanbul University Cerrahpaşa Medical Faculty, my mother Prof. Dr. Fügen Çullu Çokuğraş, Istanbul University Cerrahpaşa Medical Faculty, and of course my grandmother Gülen Çullu for her trust in me and my grandfather Ziya Çullu who passed away but left his inspiring motivation as heritage to me.

ABSTRACT

IMAGE QUALITY EVALUATION OF ULTRASOUND IMAGES USING COMPUTER SIMULATION

Image quality, for scientific and medical purposes is defined as how well the desired information can be extracted from the image thus the principal research goal in medical imaging is the development of data acquisition and reconstruction procedures that can produce consistently good clinical images to be able to make precise and accurate diagnoses.

Parallel to other imaging modalities, ultrasound imaging made also massive breakthroughs in the last decade in terms of its image quality and archiving modalities abnormalities in the body are better observed with a region specific ultrasound probe, which works within a certain frequency border and the images yielded are subjected to certain filters to remove the noise, blur or clock that arise because of reasons such as body fat, location of the lesion or minor malfunctions in the hardware etc. For diseases with long term follow-up, the images are compressed and stored, for being able to examine later on or some images obtained in a distant part of the world are transmitted via internet for telemedicine applications.

This work intends to make an evaluation of medical images, using real and simulated ultrasound images compressed via lossy algorithms, to examine the feasibility of a simulation procedure for assessing compression algorithms, to investigate the performance of those and to make comparisons between the different sources of errors and the compression errors that effect the image quality.

Keywords: Image compression, ultrasound image quality, computer simulation

ÖZET

BİLGİSAYAR SİMÜLASYONU İLE ULTRASON GÖRÜNTÜLERİNİN KALİTE DEĞERLENDİRMESİ

Bilimsel ve tıbbi amaçlar için görüntü kalitesi, istenilen bilginin görüntüden ne derece iyi elde edilebildiği şeklinde tanımlanabilir. Bu bağlamda tıbbi görüntülemede amaç klinik açıdan başarılı teşhis koyulmasını sağlayacak görüntüye ulaşmaktır.

Ultrason ile görüntüleme, gerek donanımsal gerekse yazılımsal yönden, rahatsızlıklara özel yaklaşımları, görüntü kalitesi ve görüntülerin hafızada saklanabilmesi açısından, son dönemde büyük ilerleme göstermiştir. Günümüzde, rahatsızlıklar, bölgeye uygun, belirli bir frekans aralığında çalışan prob seçimi ile, yada ilgili görüntülerin farklı filtrelerden geçirilerek, hasta veya dış kaynaklı gürültü, bulanıklık ve buğunun kaldırılması suretiyle daha başarılı bir şekilde gözlenebilmektedir. Uzun dönemli takip gerektiren rahatsızlıklar, ileriki bir tarihte incelenebilmek üzere arşivlenebilmekte yada dünyanın herhangi bir yerinden internet aracalığıyla yollanan görüntüler, uzmanların görüşlerine sunulabilmektedir.

Bu çalısmanın amacı, gerçek ve simule edilmiş ultrason görüntülerinden faydalanarak, kayba yol açan arşivleme algoritmalarını, bahsi geçen simulasyon metodunun bu tür çalısmalar için ne derece güvenilir olabileceğini tespit etmek ve çeşitli hata kaynaklarının görüntü kalitesindeki etkilerini, sıkıştırma esnasında gerçekleşen hatalarla karşılaştırmaktır.

Anahtar Sözcükler: Görüntü sıkıştırma, ultrason görüntü kalitesi, bilgisayar simulasyonu.

TABLE OF CONTENTS

ACKNOWLEDGMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES ix
LIST OF TABLES xi
LIST OF SYMBOLS xii
LIST OF ABBREVIATIONS xiv
1. INTRODUCTION 1
2. GENERAL DESCRIPTIONS 3
2.1 Ultrasound Imaging 3
2.2 Simulation 8
3. KIDNEY 10
3.1 Polycystic Kidney 12
4. IMAGE COMPRESSION 16
4.1 Lossy Image Compression 17
4.1.1 Joint Photographic Experts Group (JPEG) 22
4.1.2 JPEG 2000 23
4.1.3 Joint Photographic Experts Group Lossless (JPEG-LS) 25
4.2 Lossless Image Compression 26
4.2.1 Huffman Coding 27
4.2.2 Run Length Encoding (RLE) Coding 29
4.2.3 Lempel Zil Welch (LZW) Coding 30
4.2.4 Tagged Image File Format (TIFF) 31
5. NOISE 32
5.1 Image Data Independent Noise 33
5.2 Image Data Dependant Noise 33
5.2.1 Detector Noise 34
5.2.2 Speckle Noise 34
6. QUALITY EVALUATION 35

6.1 Root Mean Squarred Error (RMSE) 36
6.2 Peak Signal Noise Ratio (PSNR) 37
6.3 Normalized Cross Correlation (NCC) 38
6.4 Average Difference (AD) 38
6.5 Structural Content (SC) 39
6.6 Maximum Difference (MD) 39
6.7 Size Validation (SV) 40
6.8 Normalized Absolute Error (NAE) 40
7. METHODS 41
7.1 Field II 41
7.2 Image Data Sets 42
7.2.1 Real Image Data Set 42
7.2.2 Simulated Image Data Set 43
7.3 Creating Different Scenarios 44
8. RESULTS and ANALYSIS 46
8.1 Comparison of Lossy Compression Algorithms 46
8.2 Comparison of Probe Types With Regard to Compression Ratios . . . 50
8.3 Comparison of Image Quality and Central Frequency With Regard to Compression Ratios 52
8.4 Comparison of Despeckled and Non-Despeckled Images in Terms of Compression Ratios 54
8.5 Comparison of Image Quality and the Influence of Attenuation 55
8.6 Comparison of a Gaussian Noise Source and the Quality Metrics of the Simulated Image Set 56
8.7 Comparison Between Real and Simulated Images Compressed With JPEG Algorithm 60
9. CONCLUSION 63
10. FUTURE WORK 65
APPENDIX A. KIDNEY SIMULATION by FIELD II 66
A.1 The Sampling Frequency 71
A.2 The Impulse Response of an Aperture 71
A.3 The Excitation Pulse of an Aperture 72

A.4 Apodization Time Line for an Aperture 73
A.5 Linear Array Transducer 74
A.6 Freeing the Storage Occupied by an Aperture 75
A.7 Calculation the Received Signal from a Collection of Scatterers 75
A.8 Calculation of Different Weight Tables for the Interpolation 76
APPENDIX B. CYST SIMULATION by FIELD II 79
APPENDIX C. PICTURE QUALITY MEASURES CALCULATION 84
C.1 Mean Square Error 86
C.2 Peak Signal to Noise Ratio 86
C.3 Normalized Cross Correlation Calculation 86
C.4 Average Difference Calculation 87
APPENDIX D. mmROI / ROI ALGORITHM 88
APPENDIX E. Attenuation Supplement by Field II 98
REFERENCES 99

LIST OF FIGURES

Figure 2.1	Real-time B-mode ultrasound imaging system.	5
Figure 2.2	Ultrasound image of a 13th week fetus. The markers at the border of the image indicate one centimeter [10].	7
Figure 3.1	The cut section of the Kidney [23].	12
Figure 3.2	CKD and the Public Health Agenda for Chronic Diseases [24].	13
Figure 3.3	Polycstic Kidney [26].	14
Figure 4.1	Principal components of a transform-based coding structure.	18
Figure 4.2	Example of R-D curve	19
Figure 4.3	Transform decomposition structures.	20
Figure 4.4	Nonuniform and uniform scalar quantization	21
Figure 4.5	JPEG 2000 Encoder / Decoder.	24
Figure 4.6	Neighboring samples used for prediction for LOCO coder.	26
Figure 4.7	PDS of the Lena image.	28
Figure 4.8	Huffman Code Generation for six symbols.	29
Figure 7.1	Cyst phantom / 15 Cyst images.	43
Figure 7.2	The kidney image	44
Figure 8.1	Comparison of MSE and CR among different lossy compression Algorithms	47
Figure 8.2	Comparison between PSNR and CRs among different compression algorithms	47
Figure 8.3	Comparison of MSE and CR using real images among different compression algorithms.	49
Figure 8.4	Comparison of PSNR and CR using real images among different compression algorithms.	49
Figure 8.5	Comparison between MSE and CR with regard to different probe types.	51
Figure 8.6	Comparison between PSNR and CR with regard to different probe selection.	51
Figure 8.7	Comparison between CRs on JPEG images with various Cf.	53

Figure 8.8	Comparison between PSNR and CR on JPEG Images with various Cf	53
Figure 8.9	Comparison between MSE and CR on JPEG images with regard to a despeckle algorithm	55
Figure 8.10	Comparison between PSNR and CR on JPEG images with regard to a despeckle algorithm.	56
Figure 8.11	The overall flow of MSE and CR on JPEG Images with Attenuation.	57
Figure 8.12	The overall flow of PSNR and CR on with Jpeg compressed images including attenuation.	57
Figure 8.13	Comparison between MSE and CR concerning a gaussian noise source.	58
Figure 8.14	Comparison between MSE and CR concerning a gaussian noise addition.	59
Figure 8.15	Comparison between MSE and CR of real and simulated image data sets.	61
Figure 8.16	Comparison between real and simulated images in terms of PSNR values.	61

LIST OF TABLES

Table 2.1	Typical attenuation values for human tissue.	6
Table 8.1	Quantitative Representation of lossy compression algorithms' performance using a simulated set of images.	46
Table 8.2	Quantitative Representation of lossy compression algorithms' performance using a real set of images.	48
Table 8.3	Quantitative representation of image quality metrics among probe types.	50
Table 8.4	Quantitative representation of image quality metrics with regard to different central frequency levels using a linear probe.	52
Table 8.5	Quality metrics of despeckled and non-despeckled images.	54
Table 8.6	Effects of attenuation on image quality metrics.	56
Table 8.7	Quantitative representation of image quality metrics on the image data set with a gaussian noise source.	58
Table 8.8	Quantitative Representation of quality metrics of real and simulated images compressed with JPEG Algorithm.	60

LIST OF SYMBOLS

X	The frequency domain X
T_s	Operator for a transformation process relative to a given filter s
x	Representing the time domain
S	The transform Filter of length I
$*$	Convolution operator
N	The number of data samples
MxN	Representing the dimensions of an image in matrix form
$S(k,l)$	One of the subbands of a MxN image decomposed during transformation
d_n	Denotes a set of decision levels
r_n	Representing the quantized response
$X_q[n]$	The scalar quantized symbol
$X[n]$	The transformed coefficient
q	The quantization step size
$[\,]$	Operator for rounding down to the nearest integer
x_{ext}	Representing a tile componenent of an equally sized block
a_b	Denotes a DWT coefficient
Δ_b	Step size of each DWT coefficient relative to a certain dynamic range
ε_b	The dynamic range of subband b
$\hat{x}$	Denotes the predicted pixel
μ_b	Mantissa representation related to the dynamic range of subband b
NW	Denotes the neighboring pixels
I	Representing the number of iterations
$i(m,n)$	Representation of a recorded image with mxn pixels
$t(m,n)$	Denotes the true image

$n(m,n)$	Representation of the noise in the image
σ_i^2	Denotes the variance of the true image
σ_n^2	Denotes the variance of the recorded image
$a^2x\alpha$	The Variance of multiplicative noise a^2xα
g	Denotes the gray level
$I(x,y)$	Represents the original image
$I_c(x,y)$	Represents the image that underwent lossy compression
MAX_i^2	Maximum possible pixel value of the image
$t(x,y)$	Represents the cross-correlation of a template
$f(x,y)$	Denotes a subimage
$x_{j,k}$	Represantation of the original image
$x_{j,k}^{'}$	Represantation of the distorted image
$F(j,k)$	Notation for the original image
$\hat{F}(j,k)$	Notation for the distorted image

LIST OF ABBREVIATIONS

μs	Microsecond
2-D	2 Dimensional
3-D	3 Dimensional
AD	Average Difference
ADPKD	Autosomal Dominant Polycystic Kidney Disease
A-Mode	Amplitude Mode
ARPKD	Autosomal Recessive Polycystic Kidney Disease
B-Mode	Brightness Mode
BP	Blood Pressure
bpp	gray level
C_f	Central Frequency of the Probe
CKD	Chronic Kidney Disease
cm	centimeter
CR	Compression Ratio
CT	Computer Tomography
dB	decibel
DCT	Discrete Cosine Transform
DNA	Deoxyribonucleic Acid
DPCM	Differential Pulse Code Modulation
DWT	Discrete Wavelet Transform
EBCOT	Optimized truncation of the embedded bitstreams
FIR	Finite Impulse Response
fMRI	Functional Magnetic Resonance Imaging
g	gram
GHz	Giga Hertz
IIR	Infinite Impulse Response
ISO	International Organization for Standardization
JPEG	Joint Photographic Experts Group

JPEG2000	Joint Photographic Experts Group 2000
JPEG-LS	Joint Photographic Experts Group Lossless
kHz	Kilo Hertz
L3	Lumbar spinal nerve 3
LOCO	Lossless Image Compression Algorithm
LZW	Lempel Zil Welch
M.Sc.	Master of Science
MATLAB	Numerical computing environment and fourth generation programming language
MB	Megabyte
MD	Mean Difference
MHz	Mega Hertz
mm	millimeter
M-Mode	Motion Mode
MRI	Magnetic Resonance Imaging
MSE	Mean Square Error
NAE	Normalized Absolute Error
NCC	Normalized Cross Correlation
PACS	Picture Archiving and Communication System
PCM	Pulse Code Modulation
PDS	Probability Distribution of Symbols
PET	Positron Emission Tomography
PH	Percentage of Hydrogen
ph.D.	Doctor of Philosophy
PKD	Polycystic Kidney Disease
PSNR	Peak Signal Noise Ratio
pxl	pixel
RAW	The image format used for the data as it comes directly off the CCD, with no in camera processing is perform
R-D	Rate versus Distortion
RGB	Red Green Blue

RLE	Run Length Encoding
RMSE	Root Mean Square Error
ROI	Region of Interest
SC	Structural Content
SNR	Signal to Noise Ratio
SV	Size Value
T12	Thoracic spinal nerve 12
TIFF	Tagged Image File Format
YCbCr	a way of encoding RGB information
YUV	luma-chrominance

1. INTRODUCTION

Imaging the interior anatomical structure and behavior of the human body have long been an elusive ambition of medical doctors and engineers. It was Leonardo Da Vinci's drawings that first visualized the body's functional mechanism as accurately as possible. Then, the development of x-ray lightened the candle and enabled the human-being to see a living human structure. This breakthrough opened the door for visual diagnosis and gave way to ideas of new inventions to be discovered later on such as ultrasound, *Magnetic Resonance Imaging* (MRI), *Computer Tomography* (CT), *Positron Emission Tomography* (PET) etc. which are an indispensable part of our lives, that we sometimes forget.

Following the advancement of medical imaging applications, the necessity to store the digital medical images gave rise to the inventions of new data compression algorithms and techniques, which resulted in a trade-off between the work of quality and the economical approach.

In this study, it has been aimed to evaluate this phenomena of the image quality of ultrasound images that are compressed to be stored in various *Compression Ratios* (CR). There, it has been made use of image simulations to examine the desired environment and real images are used to be able to make several comparisons among different situations.

The second chapter of the whole work introduces some general descriptions about the medical imaging background of the study. The anatomical background and the images dealt with are discussed in the third chapter. Following that the concept of image compression is described and explained under sub-branches in relevance with the work. As medical images are always prone to artefacts, possible source of data errors are presented under the topic of noise, in the fifth chapter. Afterwards, in order to be able to evaluate our procedure, the so called quality parameters are explained,

and thus the state of the art explanation of the background concept is completed.

The last chapters seven and eight described first the methods and the materials used in this study and then presented the results with diverse analyses. Finally, in the conclusion part, Chapter 9, the interpretations and deductions with regard to the yielded results and analyses are made. In the latest part of the work, Chapter 10, some future work possibilities are also submitted.

2. GENERAL DESCRIPTIONS

The thesis work is based upon two fundamental concepts. One of them is ultrasound imaging and the second one is computer simulation. Thus, in order to build up the experimental workflow and yield the intended results, one has to understand the medical imaging concept and the functional mechanism of ultrasound imaging first. Following that a general view about computer simulations is given and its relevant ongoing applications on the field are introduced.

2.1 Ultrasound Imaging

Medical imaging constitutes a key part of clinical diagnosis and it has experienced phenomenal growth within the last century. Improvements in the quality and type of information available from such images have extended the diagnostic accuracy and spectrum of new applications in the health care. Previously regarded as the domain of hospital radiology departments, recent technological advances such as CT, MRI, *Functional Magnetic Resonance Imaging* (fMRI), PET, PET-CT have expanded medical imaging into neurology, cardiology, gastroenterology and cancer centers etc [1].

Considering ultrasonic imaging, it was recognized long ago that the tissues of the body are inhomogeneous and that signals sent into them, like pulses of high-frequency sound, are reflected and scattered by those tissues [2]. But its potential as an imaging modality was realized in late 1940s when, utilizing sonar and radar technology developed during World War II. In the early 1950s, John Wild and John Reid in Minnesota developed a prototype *Brightness Mode* (B-mode) ultrasonic imaging instrument and were able to demonstrate the capability of ultrasound for imaging and characterization of cancerous tissues at frequencies as high as 15 MHz [3]. Ultrasonic imaging of the soft tissues of the body really began in the early 1970s. At that time, the technologies began to become available to capture and display the echoes backscattered by structures within the body as images, at first as static compound images and later

as real-time moving images. The development followed much the same sequence (and borrowed much of the terminology) as did radar and sonar, from initial crude single line of sight displays *Amplitude Mode* (A-mode) to recording these side by side to build up recordings over time to show *Motion Mode* (M-mode), to finally sweeping the transducer either mechanically or electronically over many directions and building up two-dimensional views (B-mode or 2D)

Ultrasound is simply any sound wave whose frequency is above the limit of human hearing, which is usually taken to be 20 kHz. In the context of imaging of the human body, since frequency and wavelength (and therefore resolution) are inversely related, the lowest frequency of sound commonly used is around 1 MHz, with a constant trend toward higher frequencies in order to obtain better resolution. Axial resolution is approximately one wavelength, and at 1 MHz, the wavelength is 1.5 mm in most soft tissues, so one must go to 1.5 MHz to achieve 1-mm resolution [4]. Attenuation of ultrasonic signals increases with frequency in soft tissues, and so a tradeoff must be made between the depth of penetration that must be achieved for a particular application and the highest frequency that can be used. Applications that require deep penetration (e.g., cardiology, abdominal, obstetrics) typically use frequencies in the 2- to 5-MHz range, while those applications which only require shallow penetration but high resolution (e.g., ophthalmology, peripheral vascular, testicular) use frequencies up to around 20 MHz. Intra-arterial imaging systems, requiring submillimeter resolution, use even higher frequencies of 20 to 50 MHz, and laboratory applications of ultrasonic microscopy use frequencies up to 100 or even 200 MHz to examine structures within individual cells [5]. The main units of a modern B-mode imaging system are shown in Figure 2.1.

A multi-element transducer is used for both transmitting and receiving the pulsed ultrasound field. The central frequency of the transducer can be from 2 to 15 MHz depending on the use. Often advanced composite materials are used in the transducer, and they can attain a relative bandwidth in excess of 100%. The resolution is, thus, on the order of one to three wavelengths. The mean speed of sound in the tissue investigated varies from 1446 m/s (fat) to 1566 m/s (spleen) [6]-[7], and an averaged

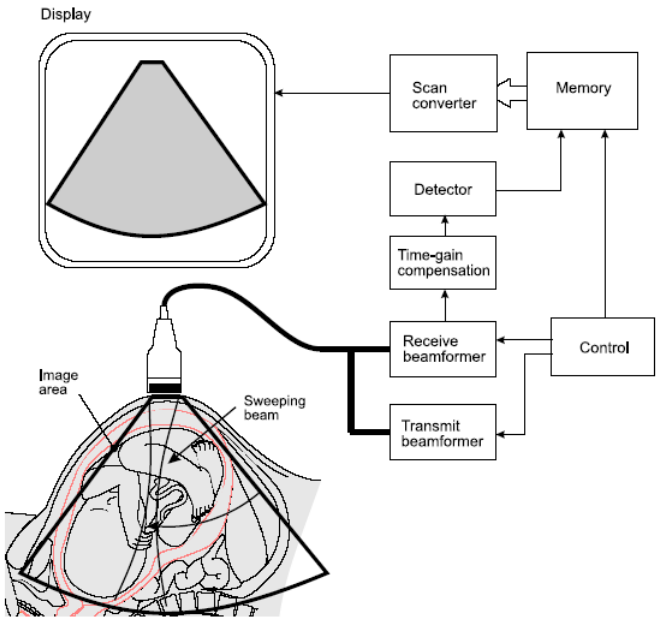


Figure 2.1 Real-time B-mode ultrasound imaging system.

value of 1540 m/s is used in the scanners. This gives an wavelength of 0.308 mm and a resolution in the axial direction of 0.3 to 1 mm. The emission of the beam is controlled electronically and is for a phased array system swept over the *Region of Interest* (ROI) in a polar scan. A single focus can be used in transmit, and the user can select the depth of the focus. The reflected and scattered field is then received by the transducer again and amplified by the time gain compensation amplifier. This compensates for the loss in amplitude due to the attenuation experienced during propagation of the sound field in the tissue. Typical attenuation values are shown in Table 2.1.

A typical value used, when designing a scanner, is 0.7 dB/[MHz·cm], indicating that the attenuation increases exponentially with both depth and frequency. This is the one-way attenuation and a 5 MHz wave measured at a depth of 10 cm would, thus, be attenuated 70 dB. After amplification the signals from all transducer elements are passed to the electronic beamformer, that focuses the received beam. For low-end scanners this is done through analog delay lines, whereas more modern high-end scanners employ digital signal processing on a sampled version of the signal from all elements. Hereby a continuous focus can be attained giving a very high resolution

Table 2.1
Typical attenuation values for human tissue [8].

Tissue	**Attenuation dB/[MHz·cm]**
Liver	0.6 - 0.9
Kidney	0.8 - 1.0
Spleen	0.5 - 1.0
Fat	1.0 - 2.0
Blood	0.17 - 0.24
Plasma	0.01
Bone	16.0 - 23.0

image. Often the beamformer can handle 64 to 192 transducer elements, and this is the typical element count in modern scanners. There is a continuous effort to expand the number of channels to improve image resolution and contrast. The beamformed signal is envelope detected and stored in a memory bank. A scan conversion is then performed to finally show the ultrasound image on a gray-level display in real time. The images can cover an area of 15 by 15 cm, and a single pulse-echo line then takes $2 * 0.15/1540 = 195\ \mu s$ to acquire. Since an image consists of roughly 100 polar lines, this gives a frame rate of 51 images a second. Often smaller images are selected to increase the frame rate, especially for blood velocity imaging [9]. A typical ultrasound image is shown in Figure 2.2 [10] for a fetus in the 13th week.

The head, mouth, legs and the spine is clearly identified in the image. It is also seen that the image has a grainy appearance, and that there are no clear demarcations or reflection between the placenta and the amniotic fluid surrounding the fetus. There are, thus, no distinct reflections from plane boundaries as they seldom exist in the body. It is the ultrasound field scattered by the constituents of the tissue that is displayed in medical ultrasound, and the medical scanners are optimized to display the scattered signal. The scattered field emanates from small changes in density, compressibility, and absorption from the connective tissue, cells, and fibrous tissue. These structures are

Figure 2.2 Ultrasound image of a 13th week fetus. The markers at the border of the image indicate one centimeter [10].

much smaller than one wavelength of the ultrasound, and the resulting speckle pattern displayed does not directly reveal physical structure. It is rather the constructive and destructive interference of scattered signals from all the small structures. So it is not possible to visualize and diagnose microstructure, but the strength of the signal is an indication of pathology. A strong signal from liver tissue, making a bright image, is, e.g., an indication of a fatty or cirrhotic liver. As the scattered wave emanates from numerous contributors, it is appropriate to characterize it in statistical terms. The amplitude distribution follows a Gaussian distribution [11], and is, thus, fully characterized by its mean and variance. The mean value is zero since the scattered signal is generated by differences in the tissue from the mean acoustic properties. Since the backscattered signal depends on the constructive and destructive interference of waves from numerous small tissue structures, it is not meaningful to talk about the reflection strength of the individual structures. Rather, it is the deviations from the mean density and speed of sound within the tissue and the composition of the tissue that determine the strength of the returned signal. The magnitude of the returned signal is, therefore, described in terms of the power of the scattered signal. Since the small structures re-radiate waves in all directions and the scattering structures might

be ordered in some direction, the returned power will, in general, be dependent on the relative position between the ultrasound emitter and receiver. Such a medium is called anisotropic, examples of which are muscle and kidney tissue. By comparison, liver tissue is a fairly isotropic scattering medium, when its major vessels are excluded, and so is blood.

2.2 Simulation

Let us suppose that we wish to see how a system behaves under certain stimuli, but it is inappropriate or even impossible to carry out the required experiment. If a valid model of the system is available, one can perform an experiment on the model by using a computer to see how the system would have reacted. This is called simulation. Simulation is thus an inexpensive and safe way to experiment with the related system. Clearly, the value of the simulation results depends completely on the quality or the validity of the model of the system [12].

Although the definition clearly states what a simulation is, the question, why we basically carry out computer simulations, remains unreplied. The answer is that it might not be possible, appropriate, convenient, or desirable to perform particular experiments on the system (e.g., it cannot be done at all, it is to difficult to realize, it is too expensive, it is too dangerous, it is not ethical, or it would take too long to obtain results). Therefore, we need an alternative way of experimenting with the system. Simulation offers such an alternative that overcomes the above mentioned limitations and provides us with the information that is useful in relation to our modeling.

The so to be approached pyramid of applications with simulations may be considered from low scales to high scales: The simulation models oriented to molecular and cellular aspects *(low scales)* and the systematic applications oriented to physiological models and medical support applications *(high scales)*.

Simulation models, particularly those oriented to parallel computation and su-

percomputing [13] have been successfully applied in molecular modeling since the 1970s. Their main objective is the discovery of new drugs [14]-[15]. Simulation models have also been applied as support to research in cellular biology, for example, in the study of microtubule stability and cellular metabolism. The mathematical modeling of cellular metabolism refers to the description of the cellular behavior under either different environmental conditions, given a genetic background, or different genetic backgrounds, given an environmental condition [16]. Another use of computer models at the cell scale is the study of cellular processes' chemical kinetics, as, for example, enzyme-catalyzed reactions, protein-protein interactions, or protein-DNA *(Deoxyribo Nucleic Acid)* binding [17]. Moving forward to higher scales, we can cite 3D computational hemodynamic models [18] which are fundamental tools for the development of new blood interacting devices, such as stent grafts, and in the research of cardiovascular pathologies, on account of the relationship between blood flow patterns and the cardiovascular disease.

Narrowing down within the simulation models, one faces the fact that finding analytic solutions to virtually every "practical problem" encountered gets harder in terms of high-tech technological imaging systems and the time-cost efficiency dilemma they cause. Among the various types, MRI, CT, PET and ultrasound imaging constitute the major research areas to create the appropriate artificial environment in order to be able to evaluate the "theoretical" and "practical" problems confronted on the examining field.

3. KIDNEY

As this thesis study deals primarily with quality evaluation of polycystic kidney images, it has been regarded as indispensable to give some broad information about the human kidney's anatomical structure, functional mechanism and the nature of *Polycystic Kidney Disease* (PKD).

Kidneys are seen in many types of animals, including vertebrates and some invertebrates. They are located behind the abdominal cavity, with regard to the anatomical reference in the retroperitoneum and consist of various distinct cell types such as glomerulus parietal cell, proximal tubule brush border cell, Loop of Henle thin segment cell etc. and are a vital part of the urinary system. The kidneys receive blood from the paired renal arteries, and drain into the paired renal veins and each kidney excretes urine into a ureter, itself a paired structure that empties into the urinary bladder.

The primary function of a kidney is the urine production but it also has several secondary functions in terms of homeostatic balancing. These may be mentioned as follow:

- The regulation of electrolytes, *Percentage of Hydrogen* (PH) balance and *Blood Pressure* (BP)
- Reabsorption if Glucose and Amino acids
- Production of hormones including vitamin D, renin and erythroprotein

Kidney physiology is studied under the terminology "Renal Physiology", while "nephrology" is the medical specialty concerned with diseases of the kidney. Diseases of the kidney are diverse, but individuals with kidney disease frequently display characteristic clinical features. Common clinical presentations include the nephritic and

nephrotic syndromes, acute kidney failure, *Chronic Kidney Disease* (CKD), urinary tract infection, nephrolithiasis, and urinary tract obstruction [19].

As mentioned above, located behind the abdominal cavity, with regard to the anatomical reference in the retroperitoneum, one on each side of the spine; they are approximately at the vertebral level *Thoracic Spinal Nerve 12* (T12) to *Lumbar Spinal Nerve 3* (L3) [20]. The right kidney resides just below the diaphragm and posterior to the liver, and the left one below the diaphragm and posterior to the spleen. The adrenal gland resting on top of the kidney is called the the suprarenal gland. Due to the asymmetry within the abdominal cavity caused by the liver, the right kidney is typically located slightly lower when compared to the left one. On the other hand the left one stands more medially [21]-[22]. The cranial parts of the kidneys are partially protected by the eleventh and twelfth ribs, plus each whole kidney and adrenal gland are surrounded by the peri-pararenal fat and the renal fascia. An adult kidney weighs between 125 and 170 g in males and between 115 and 155 g in females [20]. The left kidney is usually a little larger than the right one.

Considering the structure, kidneys are bean-shaped structures. Each kidney has concave and convex surfaces. The concave surface, the renal hilum, is the point at which the renal artery enters the organ, and the renal vein and ureter leaves. The kidney is surrounded by fibrous tissue, the renal capsule, which is itself surrounded by perinephric fat, renal fascia of Gerota, and paranephric fat. The anterior border of these tissues is the peritoneum, while the posterir border is the transversalis fascia.

The substance, or parenchyma, of the kidney is divided into two major compartments: The overlayer part is called the renal cortex and the renal medulla resides below that. These structures look like the shape of 8 to 18 cone-shaped renal lobes, each containing renal cortex surrounding a portion of medulla renal pyramid [22]. Between renal pyramids, which are composed of medulla, are projections of renal columns of Bertin.

Nephrons, the urine-producing functional structures of the kidney, span the

cortex and medulla. The initial filtering portions of the nephron, the renal corpuscles, are located in the cortex and each sends a renal tubule that passes from the cortex deep into the medullary pyramids. Part of the renal cortex, a medullary ray is a collection of renal tubules that drain into a single collecting duct Figure 3.1 [23].

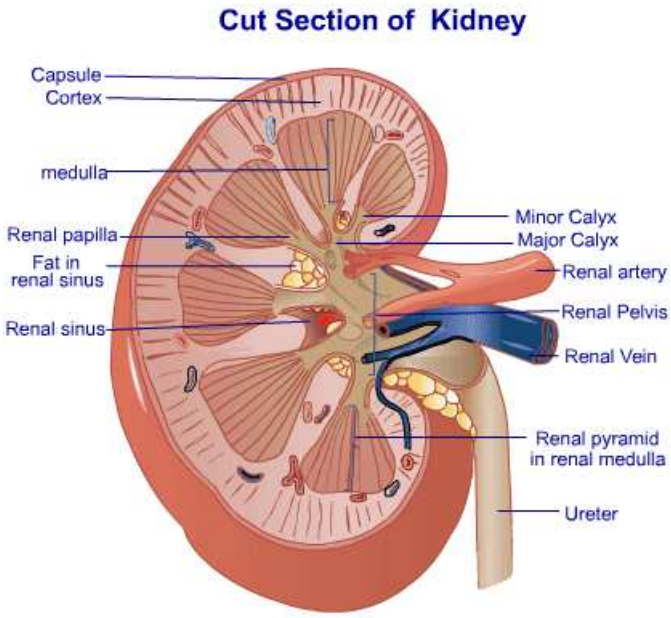


Figure 3.1 The cut section of the Kidney [23].

The tip, or papilla, of each pyramid empties urine into a minor calyx, minor calyces empty into major calyces, and major calyces empty into the renal pelvis.

The blood supply to the kidneys is provided by the left and right renal arteries, which branch directly from the abdominal aorta. Despite their relatively small size, the kidneys receive approximately 20 % of the cardiac output [20].

3.1 Polycystic Kidney

The human kidney is subject to various diseases and disorders which may have been arised because of distinct sources such as congenital or acquired reasons. Among the very important ones are the Diabetic nephropathy, Lupus nephritis, Renal failure, Renal agenesis and PKD etc. Each of these and the ones, that are not counted here

just because of irrelevance but absolutely not their level of importance, do have also subbranches and sortes thus a huge pile of names are constituting the list of possible sicknesses. Overally CKDs are a growing and thearetining problem of the human race Figure 3.2 [24], and the merely solution "an early diagnosis" is overseen due to neglegtion of patients, carelessness of the diagnosing personal, economical problems or failure of the diagnosing equipment.

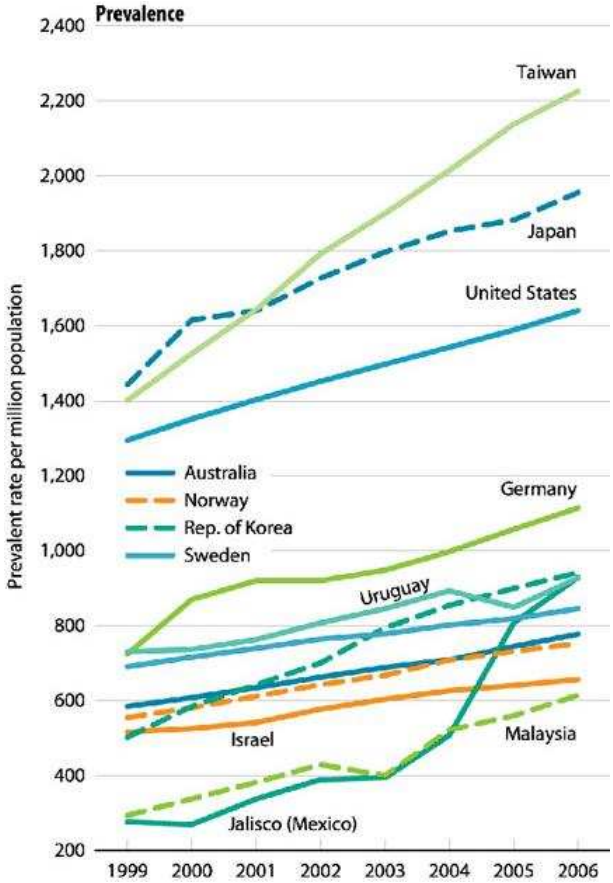


Figure 3.2 CKD and the Public Health Agenda for Chronic Diseases [24].

Although there are several types of PKDs [31], PKD usually refers to a genetic or inherited disease that is sometimes called "adult PKD" as because it normally appears in adult life. A less common type of PKD occurs primarily in babies and children. In PKD, cysts, or fluid-filled pouches Figure 3.3 [26], are found primarily in the kidney but they can also affect other organs, including the liver, pancreas, spleen and ovary. Outpouchings may occur in the walls of the large intestine and in the walls of blood vessels in the brain, where they may cause aneurysms. They may also be found in the abdominal wall, causing hernias. In addition, the valves of the heart may be involved,

becoming floppy and resulting in a heart murmur in some patients.

PKD is the most common life-threatening genetic disease, 12.5 million people worldwide [27]. It is found in all races and occurs equally in men and women. The adult type of PKD (also called autosomal dominant PKD or *Autosomal Dominant Polycystic Kidney Disease* (ADPKD)) is passed from parent to child by an autosomal dominant type of inheritance. This means that only one copy of the abnormal gene is needed to cause the disease. Therefore, if one parent has the disease, each child has a 50-50 chance of developing the disease. The risk is the same for every child, regardless of how many children develop the disease. Boys and girls have the same chance of inheriting the disease. The less common form of PKD (also called autosomal recessive PKD or *Autosomal Recessive Polycystic Kidney Disease* (ARPKD)) is passed by an autosomal recessive pattern of inheritance. This means that both parents must carry the abnormal gene, and both must pass the gene to the child in order for the child to develop the disease. In this situation, every child has a 25 percent chance of developing the disease in a family that is at risk.

Figure 3.3 Polycstic Kidney [26].

Whereas high PM [28], headaches, a fluttering/pounding in the chest as well as chest pain are considered to be possible symptoms to indicate a PKD, ultrasound

still is the most reliable, inexpensive and non-invasive screening technique to diagnose PKD. Someone at risk for PKD who is older than 30 and has a normal ultrasound of the kidneys probably does not have PKD. At present, PKD cannot be diagnosed by a single blood test. However, in some situations where it is important to have a diagnosis (for example, if a family member wants to donate a kidney to an affected parent or sibling, ultrasound and CT scans are normal), special blood tests on at least three family members can be done to get a diagnosis in the at-risk individual. This form of testing is called gene linkage analysis.

At present, there is no cure for PKD. However, a lot of research is being done [29]-[30]-[31]. Many studies suggest that some treatments may slow the rate of kidney disease in PKD, but further research is needed before these treatments can be used with patients. Other studies are improving our understanding of the genetic basis of PKD, but considering the sensitivity of ultrasound, it still remains as the best and most reliable early diagnosis procedure for the detection of ADPKD is 100 % for subjects 30 years or older with a positive family history. Diagnostic criteria require one or more cysts in one kidney and at least one cyst in the contralateral kidney in young subjects, but four or more in subjects older than 60 years because of the increased frequency of benign simple cysts. Most often, the diagnosis is made from a positive family history and imaging studies showing large kidneys with multiple bilateral cysts and possibly liver cysts.

4. IMAGE COMPRESSION

Image compression is an important and emerging technological tool in the modern digital world. Over the years, the advancements within the digital media technology have also led to the proliferation of digital picture compression systems. This is particularly noticeable in the entertainment industry, consumer electronics, and security/surveillance systems. However, data compression is becoming increasingly important in medical imaging applications as well, due to the increased popularity of digital biomedical imaging systems with regard to their indispensable help to diagnose, the constant improvement of image resolution itself, and the practical need for online sharing of information through networks.

Image compression in broad terms may be defined as the pattern recognition and file optimization of the digital data within a digital image that contains information about color, brightness and the positions of the pixels. It came about following the invention of analog television broadcasting. Initial methods of limiting signal bandwidth for transmission were relatively simple. They included subsampling to lower picture resolution and/or interlacing of television pictures into alternate fields in alternate picture frames. With the introduction of color television, subsampling was extended to the color channels as well [32]. The digital image is considered to be as a natural migration from the analog picture. Hence, it is felt that digital picture compression is, in many ways, a natural evolution of analog compression. While this is true in some sense, the nature of digital signals and of analog signals is quite different. Consequently, the methods for compressing digital and analog pictures are distinct from one another.

Specifically medical services today rely heavily on imaging technologies, including X-ray, computed tomography, magnetic resonance imaging MRI, ultrasound and nuclear medicine examinations. As these examinations become more and more common the volume of acquired data also increases to the point of becoming excessive. In a typical large hospital it is estimated that the annual volume of image data is over

one million images (250-300MB/patient/visit) requiring more than two terabytes of storage capacity. Thus, systems such as *Picture Archiving and Communication Systems* (PACS) have been proposed as tools for converting the data in digital form and for handling the resulting amount of information [33]. Image compression plays an important role in the archiving and transmission of medical images.

Image data compression can be classified into two broad categories: Lossy (Information sacrificing) and lossless (information preserving) [34]. As lossy compression of images results in a tradeoff between compression and quality, they are not widely used for both clinical and legal reasons. However standard and newer lossless compression algorithms such as *Joint Photographic Experts Group 2000* (JPEG2000) and wavelet-based compression can yield images statistically identical diagnostic results compared with using the original images without any loss [35]-[36]. Therefore lossless image coding was considered to be relatively important for medical image compression because any information loss or error caused by the image compression process could affect clinical diagnostic decision [37].

The topic is still a hot one in terms of choosing the right compression algorithm for the right purposes within safe trade-off boundaries [38]-[39]-[40] and as the thesis work also dealt with some of these compression methods in terms of their various characteristics, the following section will mainly be about the core specifications of these techniques. In addition the compression algorithms that are used are going to be introduced more closely.

4.1 Lossy Image Compression

The necessity to attain a higher CR that surpasses that of lossless coding templates to tolerate some information loss gave rise to the development of lossy image compression. Transform-based coding is employed generally for this purpose. It consists of a transformation operator followed by quantization and bitstream coding Figure 4.1.

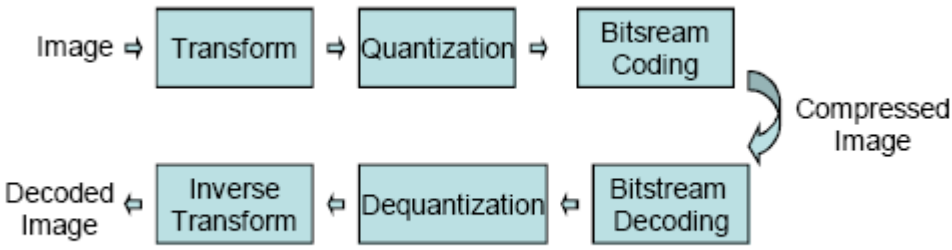


Figure 4.1 Principal components of a transform-based coding structure.

It degrades information integrity to a certain extent. However, any information lost is offset by a higher CR. Therefore, lossy compression is a balancing act between information quality and compression performance as dictated by the *Rate versus Distortion* (R-D) curves of pictures Figure 4.2 [41].

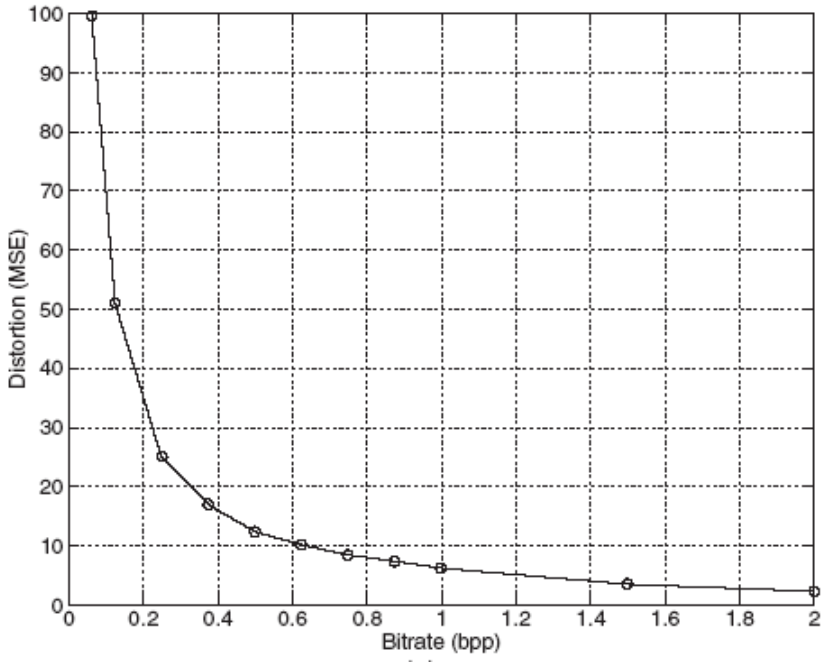
To sum up the procedures mentioned in the scheme Figure 4.1, the transformation operation is intended to rearrange data, in images, in a manner that facilitates compression. The most common form of data transformation applied in image coding is frequency based. Nonfrequency transforms such as fractal [42] have also been studied. Equation 4.1 shows how a frequency transform projects data from the time domain, x, to the frequency domain, X, relative to a given filter set, s:

$$X = T_s(x) \tag{4.1}$$

The transformation process, T, is carried out through a convolution operator $(*)$ defined by

$$X = T_s(x) = S * x = \sum_{i=1}^{N} S[i]x[n-1] \tag{4.2}$$

where S is the transform Filter of length I, $n = 1, 2, ..., N$, with N the number of data samples as stated in Equation 4.2. In regard to filter length, most signal processing



(a)

(b)

Figure 4.2 Example of R-D curve (a) for the Lena image (b) coded at different bit rates with the JPEG2000 coder [41] As the bit rate increases, the distortion decreases. An increase in bit rate corresponds to a lower CR

applications, including picture compression, typically employ *Finite Impulse Response* (FIR) filters. FIR filters have a finite number of filter taps, or Finite Filter length. This is desirable for practicability. The alternative *Infinite Impulse Response* (IIR) filters, although they are superior to FIR filters in terms of frequency selectivity [43] require in finite sampling. In finite sampling may be carried out through recursive filtering [44].

For invertible systems with forward (analysis) and inverse (synthesis) transformation, the synthesis IIR operation can be realized only with signals of finite samples.

The process of frequency transformation decomposes image data into different subband images. In this respect, frequency transformation is referred to as subband transform. Subband transforms are classified according to their decomposition structure and filters. The decomposition structure determines the way in which transform coefficients are arranged. The nature of this arrangement may be subband [45], block based [46], or hierarchical [47] Figure 4.3. Hierarchical decomposition is also known as multiresolution or wavelet transform. Image transformation is generally two-dimensional, since images are $2D$ data. But with the advent of new medical imaging modalities, which deal with the $3D$, there is also a natural extension from $2D$ to $3D$ transform to account for the temporal dimension. However, due to the prevailing coding philosophy [48] most coders adhere to the $2D$ transform for an individual picture frame.

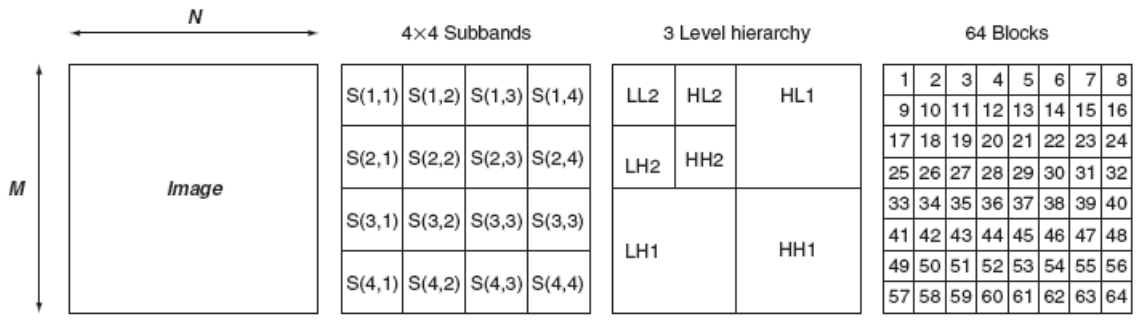


Figure 4.3 Transform decomposition structures. (a) An image of MxN dimension. (b) A $4x4$ subband decomposition. Each band $S(k,l)$ with k, $l = 1,2,3,4$ has $M/4xN/4$ transform coefficients. (c) A two-level dyadic hierarchical decomposition. Each successive level is a quarter of the resolution of the previous. (d) Block-based transform with 64 blocks. Each block is of $M/8$ and $N/8$ dimension.

The transformation is followed by the quantization which aims to limit the possible range of symbols to be coded to achieve an acceptable level of compression. The quantization operation in image coders effectively remaps the transform coefficient from a larger to a smaller set of discrete numbers. Quantizers may be scalar or vector based. In addition, scalar quantizers may be uniform or nonuniform Figure 4.4.

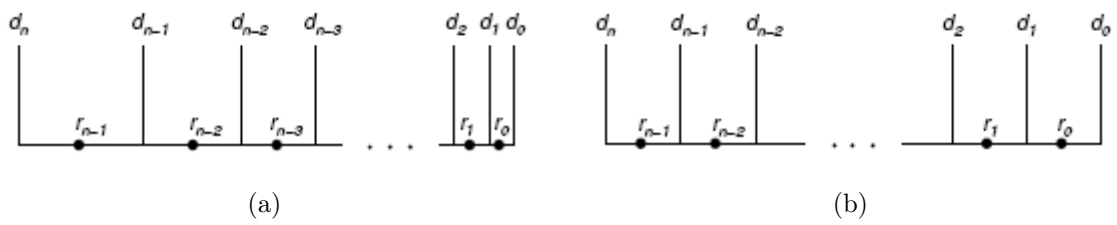


Figure 4.4 Nonuniform (a) and uniform (b) scalar quantization. Each set of decision levels, d, corresponds to a quantized response, r. For uniform quantization, the decision levels are equal distances apart. Nonuniform quantization has variable distance intervals between decision levels.

Scalar quantization is usually carried out through the division operation.

$$X_q[n] = \left\lfloor \frac{X[n]}{q} \right\rfloor \tag{4.3}$$

where $X_q[n]$ is the scalar quantized symbol, $X[n]$ is the transformed coefficient, q is the quantization step size, and $\lfloor\ \rfloor$ denotes rounding down to the nearest integer. The further methodology for designing optimal quantizers is discussed by Lloyd [49] and Max [50].

The last stage of a transform coder is called bit streaming and concerned with two things: first, entropy coding of the quantized transformed data; second, the efficient arrangement of an entropy-coded data stream. Entropy coding is a reference term for information lossless compression at or near the data entropy. Its general operation consists of two stages: *modeling* and *coding*. Modeling is performed to identify and describe data redundancies. The coding phase then encodes the information in data, is based on the description of data derived during the modeling phase, by assigning a distinct code to each symbol [51]. For bitstream coding, it is common to adapt various combinations of coding techniques to suit the nature of the data being coded. For example, the JPEG still image coder employs run-length [52] coding prior to Huffman coding in an effort to reduce the number of quantized coefficients to be coded. Similarly, for the JPEG2000 coder, run-length coding has also been utilized, albeit conditionally.

4.1.1 Joint Photographic Experts Group (JPEG)

JPEG is an acronym for *Joint Photographic Experts Group*. It refers to the definition of a still-image compression algorithm established by the JPEG committee under the auspices of the *International Standards Organization* (ISO). As it is the most widely preferred image compression technique, much work is done about it's evaluation in terms of performance. Thus JPEG still preserves the prevailing interest to be researched upon [53]. The JPEG Compression Standard has three levels of definition:

- Baseline system
- Extended system
- Special lossless function

The JPEG baseline, which is made use of in this work, is a lossy image coding using *Discrete Cosine Transform* (DCT) scalar quantization with run-length and Huffman codes for entropy coding [54]. Basically it consists of four stages: *a transformation stage*, *a lossy quantization stage* and *two lossless coding stages*. The transformation directs the information energy into the first few transform coefficients, the quantizer causes a controlled loss of information and the lossless coding stages further compress the data. In the transformation stage, JPEG uses a two dimensional 8 x 8 DCT on an 8 x 8 block of pixels. The two dimensional DCT can be obtained by performing a one dimensional DCT on the columns and a one dimensional DCT on the rows of the 8 by 8 block. The DCT coefficient values are regarded as the relative amounts of the two dimensional spatial frequencies contained in the 64 point input signal. In the quantization stage the DCT coefficients are quantized to reduce their magnitude and to increase the number of zero value coefficients. The JPEG baseline model uses the uniform mid-step quantizer with a different step size for each DCT coefficient. The step sizes of the quantizers for each DCT coefficient are given in a 64 element quantization matrix, which must be specified by the application or the user as input

to the encoder. The quantizer is the lossy stage in the JPEG coding scheme. Coarse quantization results in images which look blocky and fine quantization gives images of better visual quality with less blockings, but with lower CRs. In the lossless coding stage JPEG rearranges the quantized DCT coefficients into a zigzag pattern, with the lowest frequencies first and the highest, frequencies last. This zigzag pattern is used to increase the run-length of zero coefficients found in the block. The image' DC coefficients often vary only slightly between blocks. The coding of the DC coefficient exploits this property through *Differential Pulse Code Modulation* (DPCM). This technique codes the difference between the quantized DC coefficient of the current block and the quantized DC coefficient of the previous block. The quantized AC coefficients usually contain runs of consecutive zeroes. A coding advantage is obtained by using a run-length technique. The block codes from the DPCM and the run-length models are further reduced using entropy coding. There the Huffman coder is used to compress the data closer to the symbol entropy [55].

4.1.2 JPEG 2000

JPEG 2000 is designed to compliment the current JPEG and not to replace it. It is based on advancing technologies and is designed for new applications such as color facsimile, digital imaging, remote sensing, medical imagery, digital libraries/archives, internet, e-commerce etc. JPEG 2000 is wavelet based and can be implanted in fixed point, floating point and integer formats. It has both reversible (lossless) and irreversible (lossy) modes which includes corresponding component transformations.

The encoding/decoding principles may be explained roughly with the scheme Figure 4.5 below.

To describe the procedure, first all component values are dc shifted by subtracting the same quantity, if they are unsigned. (e.g. bpp 0-255 levels, subtract from each pxl. Two techniques are used then for the component transformation. One of them is called the irreversible component transformation and is able to perform lossy or lossless

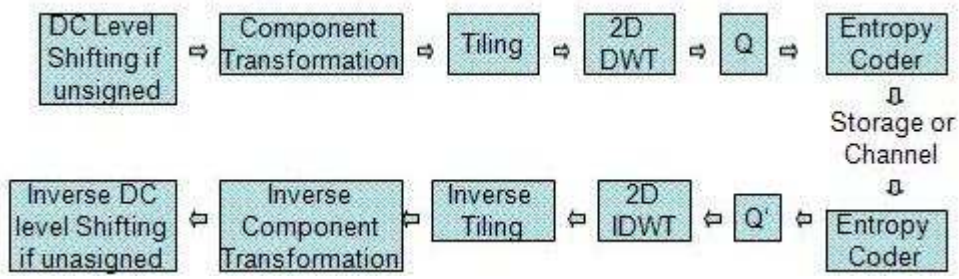


Figure 4.5 JPEG 2000 Encoder / Decoder.

coding. The second one is used for only lossless coding. For this reversible component transformation "*Red Green Blue* (RGB), *luma-chrominance* (YUV), *a way of encoding RGB information* (YCbCr) etc." shall have the same sampling parameters and the same depth. After the image is divided into regular no overlapping equal size blocks except at the right and lower boundaries, all operations such as mixing, quantization, entropy coding etc., are performed independently on each file. These equal sized blocks reduce memory requirements and enable decoding specific parts of the compressed image. Following that the tile components (equal sized blocks) are mapped into domain using irreversible (Daubechies 9/7 filter [56]) and reversible (5/3 filter) transformations. Dyadic decomposition of the images using these filters can be implemented in two modes i.e. convolution based and lifting based. For the lossless case, the lifting based filtering for the 5/3 analysis filter is obtained as follows in Equation 4.4:

$$y(2n+1) = x_{ext}(2n+1) - \left\lfloor \frac{x_{ext}(2n) + x_{ext}(2n+2)}{2} \right\rfloor \tag{4.4a}$$

$$y(2n) = x_{ext}(2n) + \left\lfloor \frac{y(2n-1) + y(2n+1) + 2}{4} \right\rfloor \tag{4.4b}$$

All the *Discrete Wavelet Transform* (DWT) coefficients $a_b(u, v)$ -except for the lossless case (5/3 reversible filter)- are then quantized as stated in Equation 4.5:

$$Q_b(u,v) = sign\Big(a_b(u,v)\Big)\left\lfloor \frac{a_b(u,v)}{\Delta_b} \right\rfloor \tag{4.5}$$

where the step size Δ_b (Equation 4.6) is represented relative to the dynamic range of subband b by the exponent ε_b and mantissa μ_b as

$$\Delta_b = 2^{R_b - \varepsilon_b}\left[\frac{1+\mu_b}{2^{1!}}\right] \tag{4.6}$$

Rectangular arrays of code blocks are responsible for collecting the quantized coefficients of subbands. There, the individual bit planes of the coefficients in a code block are entropy coded using context based arithmetic coding. Maxshift [57] or scaling method enable to certain *Regions of Interests* (ROI) to be coded with a higher quality compared to the background. In JPEG 2000, markers are added to the bitstream to provide quality and resolution scabilities, error resilience and progressive lossy to lossless coding. The entire basic coding operation is based on embedded block coding with *Optimized Truncation of the Embedded Bitstreams* (EBCOT). Thus, the bitstream is organized as succession of layers. For each code block, a separate bitstream is generated independently. Truncation points to each code block are allocated using rate distortion criterion. By approximately truncating each code block's embedded bitstream, a specific target bitrate (distortion or quality) can be achieved [58].

4.1.3 Joint Photographic Experts Group Lossless (JPEG-LS)

Joint Photographic Experts Group Lossless (JPEG-LS) [59] is a simple and efficient baseline algorithm which consists of two independent and distinct stages and it is based on based on the *Lossless Image Compression Algorithm* (LOCO) [60] coding engine. JPEG-LS was developed with the aim of providing a low-complexity "near-

lossless" image compression standard that could offer better compression efficiency than lossless JPEG. This mode is an addition to the standard, and in the baseline form of JPEG (not using arithmetic coding) the algorithm used was not really close to 'state of the art' techniques [61].

It relies on the four causal neighboring pixels for prediction and employs a conditional predictor given in Equation 4.7,

$$\hat{x} = \begin{cases} min(W, N), & NW \geq max(W, N) \\ max(W, N), & NW \leq min(W, N) \\ W + N - NW, & \text{otherwise} \end{cases} \tag{4.7}$$

where $\hat{x}$ is the predicted pixel, with W, N, and NW the neighboring pixels as illustrated in Figure 4.6. The residue is the difference between the original pixel, x, and the predicted pixel, $\hat{x}$.

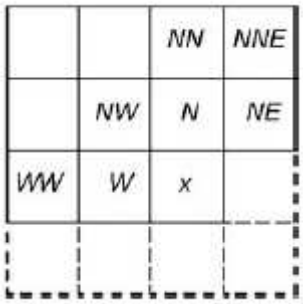


Figure 4.6 Neighboring samples used for prediction for LOCO coder.

4.2 Lossless Image Compression

Lossless image coding, basically, is similar to entropy coding of which general operation consists of two stages: *modeling* and *coding*. Modeling is performed to identify and describe data redundancies. It is carried out through statistical analysis of the data set to be able to capture the *Probability Distribution of Symbols* (PDS)

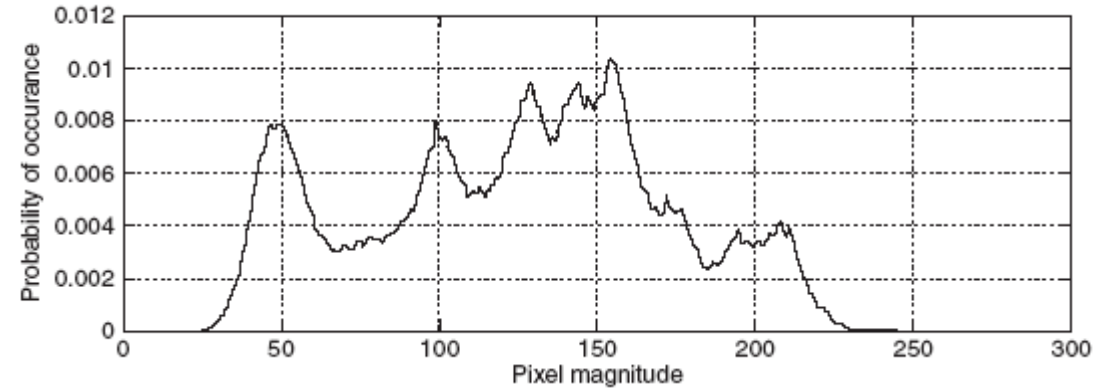
"*Pulse Code Modulation* (PCM)" [62] in the set. The coding phase then encodes the information in data, based on the description of data derived during the modeling phase, by assigning a distinct code to each symbol [51]. The size of each code, measured in number of bits, depends on the probability of true coming of its respective symbol. Generally, the most common symbol will have the smallest code size, while the least common symbol will have the largest code size. The manner in which these codes are generated is dependent on the coding algorithm. However, image data, being 2D in nature, generally have strong correlations between adjacent pixels. Consequently, it is practical to employ predictive coding prior to entropy coding to further improve compression performance. Utilizing predictive coding such as DPCM [63]-[64] has the effect of reshaping the PDS. For natural images, predictive coding leads to sharper PDS, usually centered about the zero prediction with a Gaussian or a Laplacian profile. This relates to better compression of data, since the probability distribution is concentrated on fewer numbers of symbols, as shown in Figure 4.7.

Among the various types of lossless compression techniques, there are 3 main algorithms, which are also made use of in the previous studies [39] performed to evaluate the effects of lossless image coding. These are as follows,

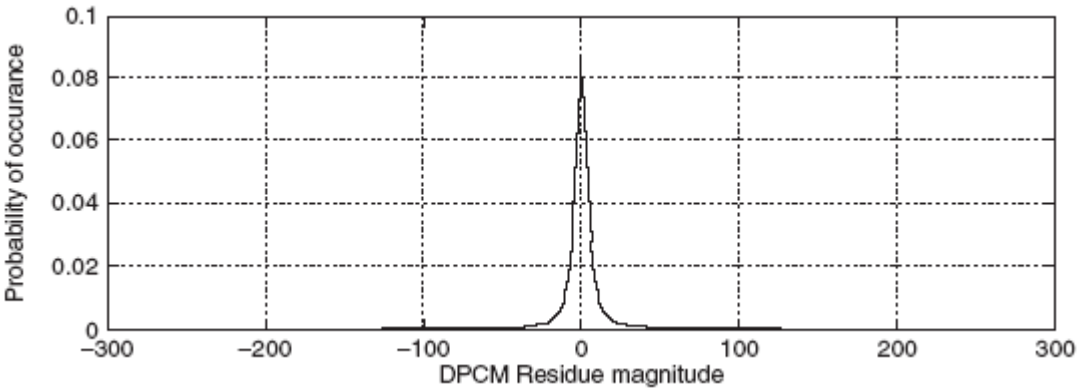
1. Huffman Coding
2. *Run Length Encoding* (RLE) Coding
3. *Lempel Zil Welch* (LZW) "TIFF"

4.2.1 Huffman Coding

The most common of the lossless compression algorithms is the Huffman code [65], which generates its alphabet of codewords through recursive sorting of source symbols, of a given probability distribution, into a binary tree Figure 4.8. In each iteration, the Huffman coding algorithm performs the following operations:



(a)



(b)

Figure 4.7 PDS of the Lena image. (a) PCM (entropy: 7.4456 bpp). (b) Row-wise DPCM (entropy: 5.0475 bpp).

- Sorting the PDS in a descending order.
- Coupling of two symbols with the lowest probability of occurrence in order to build a tree branch and to give rise to a actual symbol look-up table. For every tree branch each of the two merged symbols is assigned to a binary digit. Respectively 0 for the last and 1 for the second last. Each successful iteration will then provide higher building blocks of binary codeword.
- The final step of the iteration is its root. When there remains only one symbol which leads to no more formation of a tree branch. Thus, the number of iterations, i, required to generate the entire tree is one less than the total number of symbols, n, i.e., or $i = n - 1$

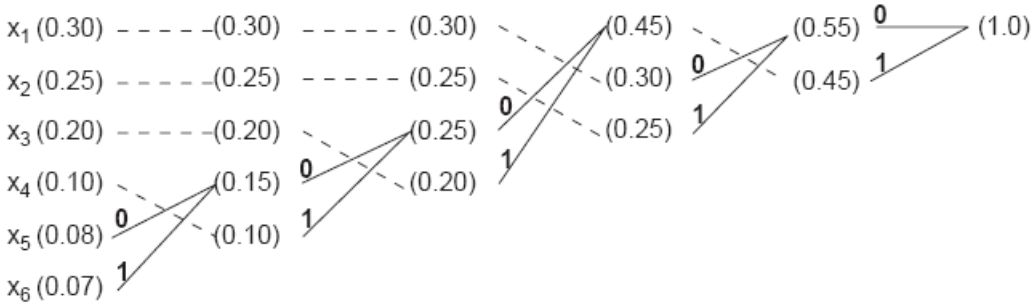


Figure 4.8 Huffman Code Generation for six symbols.

4.2.2 Run Length Encoding (RLE) Coding

The RLE scheme is created for the compression of data that contain large amount of runs (that is, sequences in which the same data value occurs in many consecutive data elements) for areas with equal color values, e.g. a black background, and store image areas as a sequence of rows. These rows can be efficiently compressed using the RLE scheme, which is a very simple and fast compression scheme [66]. It simply replaces runs of repeated symbols by the symbol of the run and the length of the run. However, if the sequence length is only 1, the RLE scheme increases the required storage space for this pixel. Therefore, a special character (e.g., "!") is required to mark whether the following number represents a frequency of repetition or the gray value of the neighbored pixel. In order to guarantee that the compressed image does not increase in data volume, only runs of four or more symbols are converted. Just as stated in various literature sources, an example would help to explain the definition more clearly: the sequence

$$"ABCAAABCCCCCAAAAAAAAAAAAAAAAAAAAABBBBC" \quad (37 characters)$$

converts into

$$"ABCAAABC!1A!20B!0C" \quad (17 characters)$$

Here, the special character also defines that the gray value is repeated four times. If a special character is not available in the alphabet, one can -accepting an increase of storage space- use the four-sequence of thy symbol itself. Then, the example above reads to the decoder

$$"ABCAAABCCCC5AAAA20BBBB4C" \quad (23 characters)$$

In these examples, the compression rate is 2.18 and 1.61, respectively.

4.2.3 Lempel Zil Welch (LZW) Coding

In contrast to the RLE scheme, LZW schemes compress not only areas with equal gray or color values but also more complex patterns. As a result, LZW schemes achieve stronger compression rates but have a lower speed than RLE schemes. It is named after its developers, A. Lempel and J. Ziv [67], with later modifications by Terry A. Welch. LZW Compression algorithm is considered to be the foremost technique for general purpose data compression due to its simplicity and versatility. Typically, LZW is used to compress text, executable code, and similar data files to about one-half their original size.

The most popular example in the literature to explain how it is works is as follows, a dictionary is initialized to contain the single-character strings corresponding to all the possible input characters (and nothing else). The algorithm works by scanning through the input string for successively longer substrings until it finds one that is not in the dictionary. When such a string is found, it is added to the dictionary, and the index for the string less the last character (i.e., the longest substring that is in the dictionary) is retrieved from the dictionary and sent to output. The last input character is then used as the next starting point to scan for substrings.

In this way -successively- longer strings are registered in the dictionary and made

available for subsequent encoding as single output values. The algorithm works best on data with repeated patterns, so the initial parts of a message will see little compression. As the message grows, however, the CR tends asymptotically to the maximum.

4.2.4 Tagged Image File Format (TIFF)

Tagged Image File Format (TIFF) [68] describes image data that typically comes from scanners, frame grabbers, and paint -and photo- retouching programs. TIFF is not a printer language or page description language. The purpose of TIFF is to describe and store raster image data. A primary goal of TIFF is to provide a rich environment within which applications can exchange image data. This richness is required to take advantage of the varying capabilities of scanners and other imaging devices. TIFF is capable of describing bi-level, grayscale, palette-color, and full-color image data in several color spaces, includes a number of compression schemes that allow developers to choose the best space or time tradeoff for their applications, it is not tied to specific scanners, printers, or computer display hardware, it does not favor particular operating systems, file systems, compilers, or processors. The universal lossless data compression algorithm LZW [69] enables TIFF images to be compressed losslessly.

TIFF is intended to be independent of specific operating systems, filing systems, compilers, and processors. The only significant assumption is that the storage medium supports something like a "file", defined as a sequence of 8-bit bytes, where the bytes are numbered from 0 to N. The largest possible TIFF file is 4 GB in length. Since TIFF uses pointers (byte offsets) quite liberally, a TIFF file is most easily read from a random access device such as a hard disk or flexible diskette, although it should be possible to read and write TIFF files on magnetic tape [70].

5. NOISE

Medical imaging techniques -particularly ultrasound- plays a vital role in the early detection of kidney diseases such as polycystic kidney and is a proven-suitable way of approach for diagnoses and progenies alike. As in other imaging modalities the accurate detection of ROI is in ultrasound imaging also of vital importance. But it the result of reflection, refraction and deflection of ultrasound waves from different types of tissues with different acoustic impedances, usually, cause the contrast in an ultrasound image to be very low and the boundaries between ROI and background to be fuzzy [71]. If one also counts in the noise, which is defined generally as to be any measurement that is not part of the phenomena of interest is, and the wedges, for the physician it is getting harder to make accurate diagnosis.

Medical Images -as also ultrasound is- are prone to noise [72]. While analyzing an image, noise usually is regarded to be as the departure of the ideal signal. It arises as a result of unmodelled or unmodellable [73] processes going on during the generation and the receiving of the real signal itself -namely the ultrasound waves-. The noise is not a part of the ideal signal and may be caused by a wide range of sources. For instance, variation in the detector sensitivity, environmental transitions, the discrete nature of radiation, transmission or quantization errors, etc. It is also possible to treat irrelevant scene details as if they are image noises, for example, surface reflectance textures.

If we consider an ultrasound image as a two dimensional function $f(x, y)$, where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity or grey level of the image at that point [74], data sets collected by image sensor are generally contaminated by noise. The ROI in the image can be degraded by the impact of imperfect instrument, the problem with data acquisition process or interfering natural phenomena. Noise is generally grouped into two categories, image independent noise and image data dependent noise. And ultra-

sound imaging -specifically- is prone to the both groups, whereas in this work they will be named but only the ones, that are made use of, will be explained further.

5.1 Image Data Independent Noise

Image Data Independent Noise is described by an additive noise model, where the recorded image, $i(m,n)$ is the sum of the true image $t(m,n)$ and the noise $n(m,n)$ [74]-[75].

$$i(m,n) = t(m,n) + n(m,n) \tag{5.1}$$

The noise $n(m,n)$ is often zero-mean and described by its variance σ_n^2 . In fact, the impact of the noise on the image is often described by the *Signal to Noise Ratio* (SNR) [76], which is given by

$$SNR = \sqrt{\frac{\sigma_i^2}{\sigma_n^2} - 1} \tag{5.2}$$

5.2 Image Data Dependant Noise

Data-dependent noise (e.g. arising when monochromatic radiation is scattered from a surface whose roughness is of the order of a wavelength, causing wave interference which results in image speckle). Among the various Detector noise, Speckle Noise, Salt and Pepper Noise and Poisson Noise may be counted as the primary ones. In relation to the following work detector noise and speckle noise are decisive.

5.2.1 Detector Noise

In theory, detector noise is another sort of Gaussian noise, which occurs in all recorded images to a certain extent. It is due to the discrete nature of radiation which arise by the fact that each imaging system is recording an image by counting photons. Allowing some assumptions (which are valid for many applications) this noise can be modeled with an independent, additive model, where the noise has a zero-mean Gaussian distribution described by its standard deviation (σ) or variance [77]. This means that each pixel in the noisy image is the sum of the true pixel value and a random, Gaussian distributed noise value.

5.2.2 Speckle Noise

Another common form of noise is the data dropout noise generally referred to as speckle noise. This noise is, in fact, caused by errors in data transmission [78]. The corrupted pixels are either set to the maximum value, which is something like a snow in image or have single bits flipped over. Speckle noise has the characteristic of multiplicative noise [79]. It follows a gamma distribution and is given as

$$F(g) = \left[\frac{g^{\infty-1}}{(\infty-1)!a^{\infty}} e^{-\frac{g}{a}} \right] \tag{5.3}$$

6. QUALITY EVALUATION

As talked about before, image compression can provide increases in transmission speed and in the quantity of images stored on a given memory. Lossless compression, in which an original image is perfectly recoverable from the compressed format, can be used controversy. However, its gains, thus the compromise between the quality and the size is limited, ranging from 2:1 to 4:1. Hence, serious compression must be performed lossy, which leads to a approximational recovery from the compressed format [80]-[81].

The terminology of quality evaluation in terms of an approximation is quite tricky. There is no doubt that the usefulness of image compression depends critically on the quality of the processed images. Quality is an attribute with diverse definitions and interpretations, depending heavily on the use to what the images will be put. Sometimes, in order to consider a compressed image as "high quality", it should be visually indistinguishable from the original. This is usually referred as "transparent quality" or "perceptually lossless" since the use of the compression on the image is transparent to the viewer. Although upon first consideration, this concept of visually indistinguishable would seem quite logical and a simple definition of a relative-seen quality threshold that everyone could agree upon, two images are considered only then visually indistinguishable, when the difference between them is perceived by at least two persons that look at them in a row. For example, a pair of medical images viewed by regular people may appear identical, whereas a radiologist trained in viewing those images might detect differences [82].

A number of studies have evaluated the effects of lossy and lossless image compression methods based on their either diagnostic or statistical quality metrics in terms of the use in medical imaging [83]-[84]. Apart some parameters, such as image dimension, gray level etc., which are considered to be as pre-conditions for the image-data sets used throughout this study, this thesis work primarily deals with the image quality of self-generated images and their compression characteristics with different algorithms.

Thus the deduced list of relevant quality parameters from the previous work and evaluations are listed as follows:

1. *Root Mean Squared Error* (RMSE)
2. *Peak to Signal Noise Ratio* (PSNR)
3. *Normalized Cross Correlation* (NCC)
4. *Average Difference* (AD)
5. *Structural Content* (SC)
6. *Maximum Difference* (MD)
7. *Size Validation* (SV)
8. *Normalized Absolute Error* (NAE)

6.1 Root Mean Squarred Error (RMSE)

Probably the most common metric of image quality for images that have undergone compression in the literature is the RMSE between the original image and the image that underwent lossy image compression [85].

Basically, the mean square error or MSE of an estimator is a tool to quantify the difference between a copy and the true value of the quantity being estimated. In statistic it is defined as a risk function, corresponding to the expected value of the squared error loss or quadratic loss thus MSE measures the average of the square of the "error". The error is the amount by which the reproduced image or the copy differs from the quantity to be estimated. The difference between them may occur because of randomness, a certain level of noise, loss in the reproduction algorithm or because the copy is not merely not capable of accounting for information that could produce a more accurate estimate [86]. Mathematically it is expressed as:

$$RMSE = \sqrt{\frac{1}{XY}\sum_{x=0}^{X}\sum_{y=1}^{Y}\left[I\big(x,y\big) - I_c\big(x,y\big)\right]^2} \tag{6.1}$$

where $I(x, y)$ is the original image and $I_c(x, y)$ is the image that underwent lossy compression. X and Y are the sizes of the image. As a matter of fact one has to state there also that RMSE does not take into account the properties of the human visual system or a visual task.

6.2 Peak Signal Noise Ratio (PSNR)

The PSNR, also noted as PSNR, is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. It is usually expressed in logarithmic level as many signals have a very wide dynamic range. The mathematical expression for PSNR is:

$$PSNR = 20.\log_{10}\left(\frac{MAX_i^2}{\sqrt{MSE}}\right) \tag{6.2}$$

The signal in this case is the original image, and the noise is which is considered primarily the error introduced by the lossy compression. In literature, while comparing compression codecs, PSNR 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 [87].

6.3 Normalized Cross Correlation (NCC)

Within image processing applications, the images can be first normalized where the brightness of the image and template do vary due to lighting and exposure conditions. Basically this is done at every step by subtracting the mean and dividing by the standard deviation of the image. That is, the cross-correlation of a template, $t(x, y)$ with a subimage $f(x, y)$ is:

$$NCC = 1\frac{1}{n-1}\sum_{x,y}\frac{\left(f(x,y)-\bar{f}\right)\left(t(x,y)-\bar{t}\right)}{\sigma_f\sigma_t} \tag{6.3}$$

where n is the number of pixels in $t(x, y)$ and $f(x, y)$.

6.4 Average Difference (AD)

The AD is a tool to define the ratio between noise, that arises because of an input of an external/internal noise, the compression algorithm itself or a filter which is used to improve the image quality but causes a side-effect which gives rise to a loss in the image and the original image. It may be denoted as:

$$AV = \sum_{j=1}^{M}\sum_{k=1}^{N}\frac{\left(x_{j,k}-x'_{j,k}\right)}{MN} \tag{6.4}$$

where $x_{j,k}$ stands for the original image and $x'_{j,k}$ for the distorted image respectively. MN is the matrix expression of the original image.

6.5 Structural Content (SC)

SC is an image correlation measure, which gives the similarity between two digital images in terms of a correlation function [88]. It basically measures the similarity between two images, hence in this sense they are complementary to the difference-based measures such as MSE or PSNR. The mathematical expression for SC is stated as follows:

$$SC = \frac{\sum_{j=1}^{M}\sum_{k=1}^{N}\left[F(j,k)\right]^2}{\sum_{j=1}^{M}\sum_{k=1}^{N}\left[\hat{F}(j,k)\right]^2} \tag{6.5}$$

whereas $F(j,k)$ indicates the multispectral pixel vectors of the original image and $\hat{F}(j,k)$ of the distorted image respectively.

6.6 Maximum Difference (MD)

The MD is another pixel-difference based quality metric which is used in signal and image processing. The mathematical expression for that is given below:

$$MD = Max\left\{\left|F(j,k) - \hat{F}(j,k)\right|\right\} \tag{6.6}$$

whereas $F(j,k)$ denotes the original image and $\hat{F}(j,k)$ the distorted image accordingly.

6.7 Size Validation (SV)

SV is merely made to compare the amount of data in terms of kBs, MBs etc. of the same image' original and distorted versions. It has no significant value for quantifying or stating whether one image is better than the other but it is another transparent proof for the loss or unwanted additive information in the distorted images when compared to the original ones.

6.8 Normalized Absolute Error (NAE)

NAE is a combinational quality metric of both pixel based and image correlation evaluation tools. Mathematically, the normalized absolute error between a reference and degraded image is represented as:

$$NAE = \frac{\sum_{j=1}^{M}\sum_{k=1}^{N}\left|O\Big\{F(j,k)\Big\} - O\Big\{\hat{F}(j,k)\Big\}\right|}{\sum_{j=1}^{M}\sum_{k=1}^{N}\left|O\Big\{F(j,k)\Big\}\right|} \tag{6.7}$$

where $F(j,k)$ denotes the reference image and $\hat{F}(j,k)$ the degraded image.

7. METHODS

7.1 Field II

Modern ultrasound scanners use a number of schemes for attaining high resolution and high contrast images. Multi-element transducers are used to steer the ultrasound beam and to use multiple foci zones whereas apodization is used for reducing side lobe levels and thereby increase the dynamic range of the image. Both focusing and apodization are dynamic and change as a function of depth in tissue or corresponding time. Also many array transducer geometries exist from small phased array probes for use in cardiac imaging to convex linear arrays suitable for the abdomen. The optimization of these transducers and their use is eased by employing software scanning. Here the transducer, phantom, and image processing are simulated by a computer and the different choices can easily be studied. The problems are the many types of arrays and the dynamic nature of the image formation [89].

The Field simulation program, which is used throughout this work as the simulation tool was developed by Professor Ph.D., Dr. Techn., M.Sc. Jørgen Arendt Jensen and his team at the Technical University of Denmark, is based on a previously made general simulation program for all types of ultrasound transducers [90]. This program has been used successfully at a number of universities and has yielded accurate results, when compared to measurements [91]. It uses the Tupholme-Stepanishen approach [92]-[93] of spatial impulse responses, that assumes linear propagation. It can handle any type of transducer geometry and excitation and all ultrasound fields can be calculated by this method. In this program the transducer surface is split into rectangles and a far field approximation used for making the calculation fast. This has made it possible to simulate images from computer phantoms in about 24 hours. The program is based on a menu interface that makes it flexible to use at the price of less flexibility. Now, with its based programming including in C written commands, the new version of Field for defining transducers, setting their properties and calculating fields of these transducers

is predicted to be faster and more flexible.

The prime application of the Field program is to simulate the image of an ultrasound scanner. This necessitates that multiple foci zones can be taken into account and that dynamic apodization can be used. The two concepts are introduced through time lines. The focus time line holds information about the dynamic behavior of the focusing. Each focal zone in characterized by a time point and a delay value for each transducer element. The time point indicates the time after pulse emission when these delay values are used. The same approach is used for the apodization time line, which assigns an apodization value for each transducer element. Multiple transducers can be handled by the program. Commands for defining linear, phased, and $2D$ matrix arrays are given. The commands return an identifier for the array, which can be passed to the routines for field calculation. Thereby different transducers can be used on the same scatterers and the effect of different choices can readily be evaluated. Commands are also found for setting the excitation waveform of the transducer and the electro-mechanical impulse response. Commands for calculating the emitted, the pulse-echo, and the scattered fields are given. Thereby the transducers can be evaluated and images for computer phantoms can be found.

7.2 Image Data Sets

7.2.1 Real Image Data Set

The real data image set consists of 10 ultrasound images totally. They are retrieved with a convex probe with the central frequency band of 2-5 MHz and with a sectoral probe with central frequency figure of 5-8 MHz at Istanbul University Cerrahpasa Medical Faculty Institute of Radiology. All of the images are obtained using the Siemens Sono Elegra Systems [94] ultrasound device. Parallel to the aim of the study, the images contain polycystic kidney properties with a patient age variability 5-39 years. The whole set of images were in TIFF format, a lossless compression tool comparable to the RAW. Their sizes are $768 * 576$ pixels, As some of the images were

containing some blur and were not able to be recognized in the first diagnostic approach, presumably due to the environmental facts, physcophyisological daily aspects of the radiologists or a not fully correct way of performance etc. with the consensus of the radiologists 5 of the images are picked up to be used in the study.

7.2.2 Simulated Image Data Set

The simulated data image sets are subcategorized under various subtitles so that the evaluation and analysis were able to be performed for different scenarios. The first 2 simulations are made to yield the cysts and kidney images. The cyst phantom is a collection of point targets, five cyst regions, and five highly scattering regions Figure 7.1.

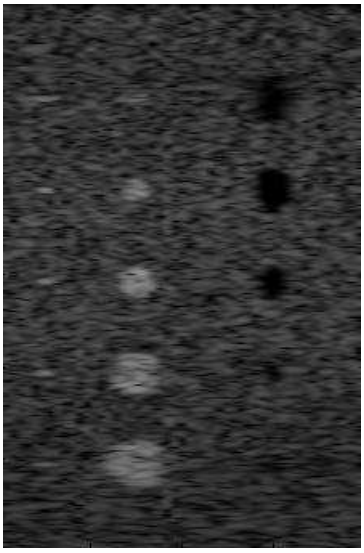


Figure 7.1 Cyst phantom / 15 Cyst images.

This is used for creating different contrast-lesion detection capabilities of the ultrasound imaging system. The scatterers in the phantom are generated by finding their random position within a 60x40x15 mm cube, and then ascribe a Gaussian distributed amplitude to each scatterer. If the scatterer resides within a cyst region, the amplitude is set to zero. Within the highly scattering region the amplitude is multiplied by 10. The point targets have a fixed amplitude of 100, compared to the standard deviation of the Gaussian distributions of 1 (Appendix B). Following that the cyst regions inside

the stated cube are retracted using the mmOI ROI (Appendix D) algorithm. Afterwards they are picked up randomly-containing 3 pieces in each set and placed into the kidney image according to the literary findings and empirical information received from physicians, hence creating 8 different scenarios for a polycystic image.

With a similar concept and using the simulation program Field II, a phantom for a left kidney in a longitudinal scan has been made (Figure 7.2). 1.000,000 scatterers were randomly distributed within the phantom, and with a Gaussian distributed scatter amplitude with a standard deviation determined by the scatter map. The phantom was scanned with a 7 MHz 128 element phased array transducer with lambda/2 spacing and Hanning apodization. A single transmit focus 60 mm from the transducer was used, and focusing during reception is at 5 to 150 mm in 1 mm increments. The images consist of 128 lines with 0.7 degrees between lines. Thus the whole simulated image set is made up of 10 arbitrarily picked up images among 85 different possibilities. (Appendix A)

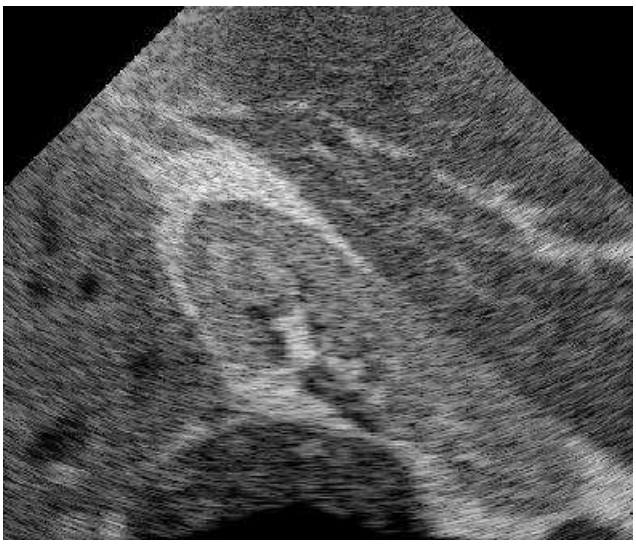


Figure 7.2 The kidney image (Scale: 1/5).

7.3 Creating Different Scenarios

To achieve a proper comparison level, throughout the case scenarios it has been used a set of fixed 5 real and 10 simulated image-sets. According to the scenario the

image data set was either containing a mixture of real and simulated images or merely made up of real or simulated images due to the regulation of the desired parameter's evaluation (Appendix C).

Before beginning with the evaluation, 2 set curves of the image-sets are yielded in order to be able to compare future results with an original starting point. Both the real and the simulated image data sets -separately- are compressed by lossy algorithms (JPEG, JLS, JPEG2000) within predefined ratios (5, 10, 15, 20) in Irfanview (V.4.25) [95].

First scenario was to compress a set of 15 images mixed (real+simulated) with certain ratios (5, 10, 15, 20) and evaluate the performance of the lossy compression algorithms (JPEG, JLS, JPEG2000) in terms of image quality measuring tools respectively. A second one is to add an overall 1 per cent Gaussian noise onto the whole image, compress it with JPEG compression algorithm and analyze the result as a consequence of the contribution of 1 % Gaussian noise (via Photoshop CS3 V8.0) which is comparable to various artifacts in ultrasound images. The effects of nowadays widely used despeckle algorithms on ultrasound images during compression is taken into account via using the despeckle tool of Photoshop CS3 V8.0 and thus building up a comparison for despeckled and non-despeckled images in terms of image compression with JPEG. Following that the image data sets are created for investigating the attitude of the compressed image quality by changing the attenuation level up to 1,5 dB/cm*MHz (Appendix E), the central frequency (1, 5, 10 MHz.) or the probe type (convex/linear) itself while holding 2 of the mentioned parameters stable thus without any adjustments.

8. RESULTS and ANALYSIS

By establishing a set of 5 real images and another set of 10 simulated images, the study was able to present a clear representation and deduction in terms of the outcomes and effect of various reasons on the quality of the ultrasound images. Thus in this section of the work different scenarios will be focused on and explained with regard to relevance of the scope of the thesis study.

8.1 Comparison of Lossy Compression Algorithms

The investigated lossy compression algorithms are JPEG, JPEG-LS and JPEG 2000. A set of 10 simulated images and 5 real images are picked up separately among 10 real and 85 simulated polycystic images. The results below show the quality metrics of the corresponding image set before and after a compression rate of 0, 5, 10, 15 and 20 per cent respectively.

Table 8.1
Quantitative Representation of lossy compression algorithms' performance using a simulated set of images.

Compression Ratio	**JPEG**		**JPEG2000**		**JLS**	
	MSE	**PSNR**	**MSE**	**PSNR**	**MSE**	**PSNR**
0	0,19	42,17	0	99	0	99
5	0,46	39,06	0,12	61,60	0,12	92,43
10	1,13	38,93	1,13	52,53	0,21	61,49
15	2,71	38,64	2,70	44,57	0,76	50,43
20	9,72	37,39	6,48	40,21	7,78	39,78

As in similar findings the metrics are always given in decibels (dB). The images are obtained without attenuation, with a linear probe simulation having a central frequency level (Cf) of 5 MHz. There, the difference of the performance is obvious

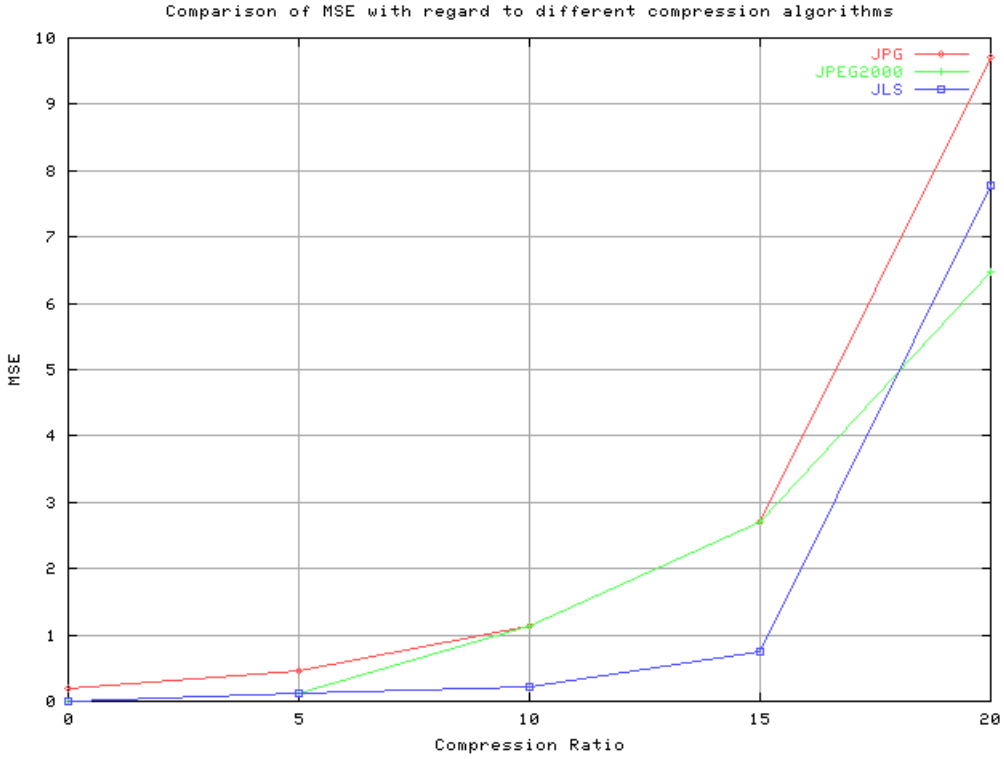


Figure 8.1 Comparison of MSE and CR among different lossy compression Algorithms

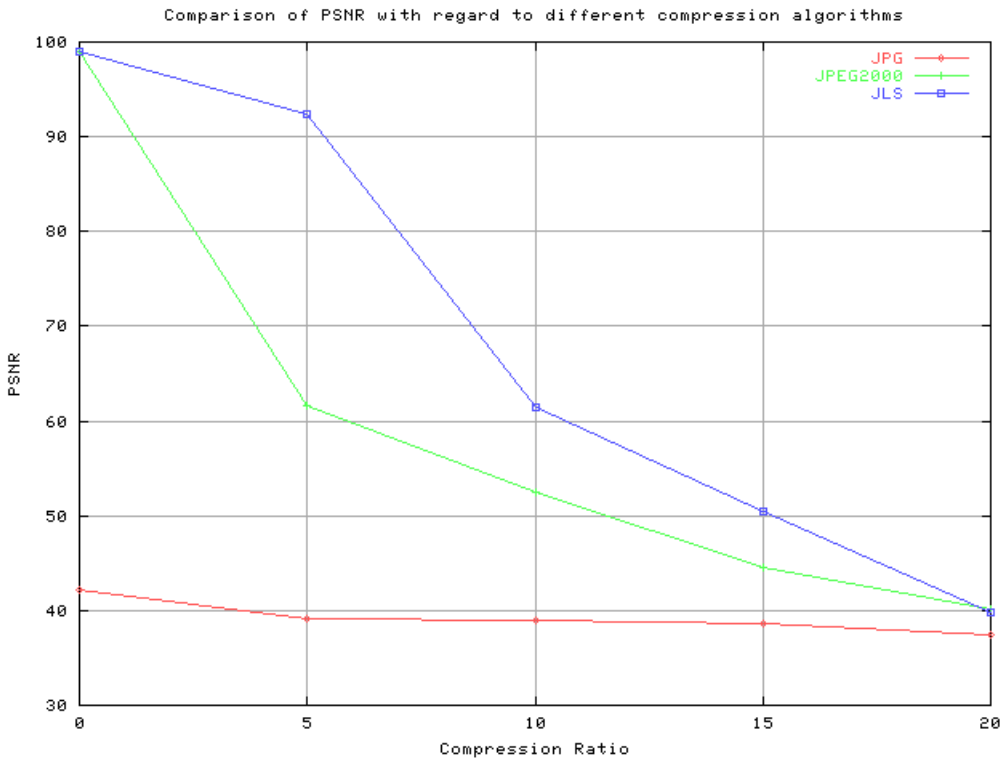


Figure 8.2 Comparison between PSNR and CRs among different compression algorithms

concerning the quantitative representation of lossy compression algorithms. A criteria of 30 dB is considered to be the lowest value for a good PSNR. The Jpeg Algorithm which is most commonly used in commercial and in ultrasound image archiving differs right at the beginning drastically from the other 2 compression tools, which is totally dependant of its quantization and encoding method as it was explained previously. Within lower CRs the other two compression algorithms JPEG 2000 and JLS are differing from each other, favoring JLS. Whereas the increasing ratio of compression leads that JPEG 2000 outperforms JLS. Anyhow, none of the compression algorithms sink below the critical level of image quality, which was defined as 30 dB.

Similar findings show up also, when the comparison is executed via the real set of images as shown below.

Table 8.2
Quantitative Representation of lossy compression algorithms' performance using a real set of images.

Compression Ratio	**JPEG**		**JPEG2000**		**JLS**	
	MSE	**PSNR**	**MSE**	**PSNR**	**MSE**	**PSNR**
0	0,17	48,15	0	99	0	99
5	0,40	44,59	0,10	70,33	0,10	96,32
10	0,99	44,45	0,99	59,97	0,18	70,22
15	2,37	44,11	2,36	50,89	0,67	57,59
20	8,51	42,68	5,68	45,90	6,82	45,42

The real set of images consist of 5 images obtained via a linear probe with a Cf of 5 MHz.

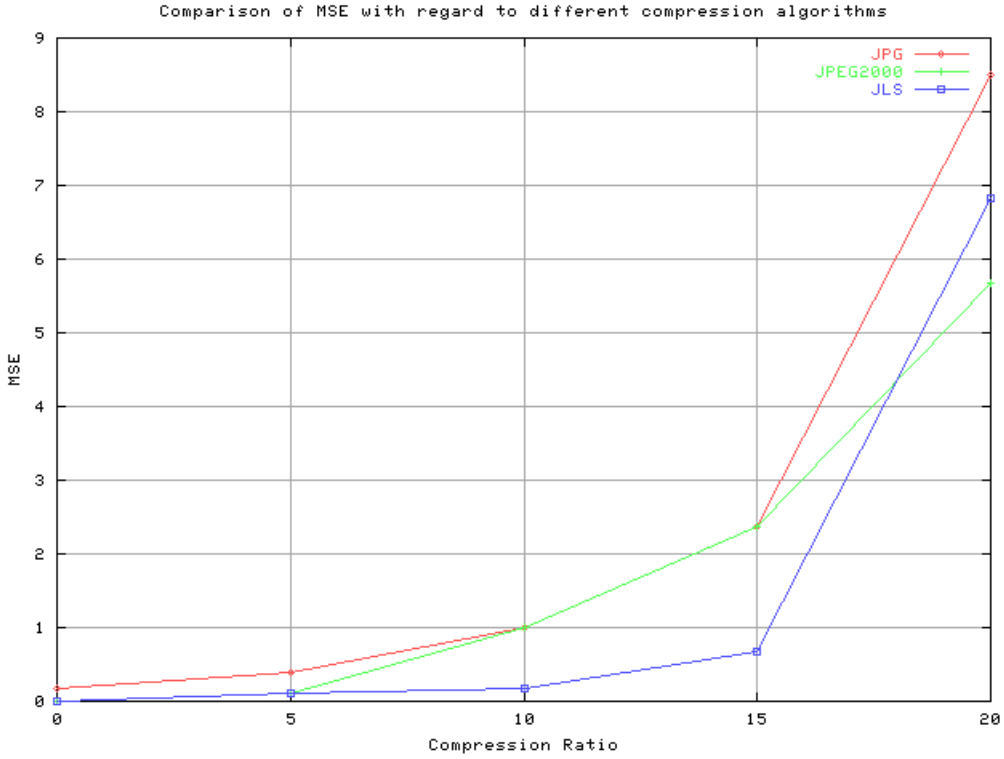


Figure 8.3 Comparison of MSE and CR using real images among different compression algorithms.

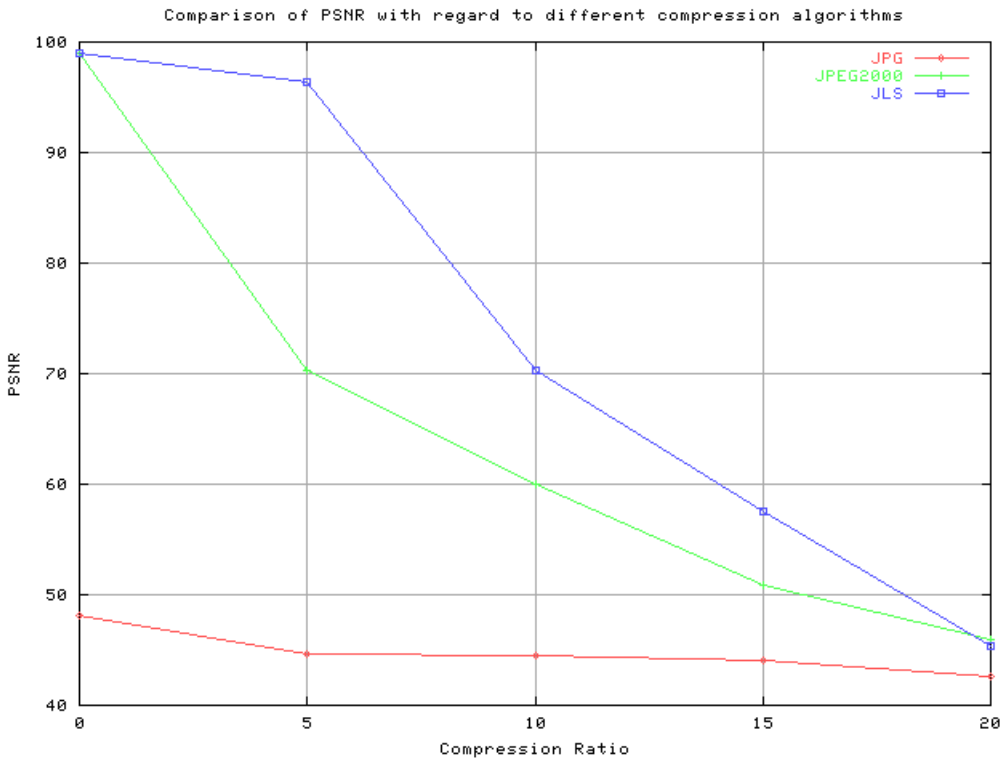


Figure 8.4 Comparison of PSNR and CR using real images among different compression algorithms.

8.2 Comparison of Probe Types With Regard to Compression Ratios

It is clear that the velocity and level of penetration of the sound wave depend of the power thus the properties of the probe. In this case by simulating the change of the probe's physical structure, the study intended to find out the effect of the probe's type on the image quality. The related quantitative results are given below for the same simulated image data set of 10 images.

Table 8.3
Quantitative representation of image quality metrics among probe types.

Compression Ratio	**Convex**		**Linear**	
	MSE	**PSNR**	**MSE**	**PSNR**
0	0,1965	42,1261	0,1920	42,1700
5	0,4666	39,0547	0,4562	39,0567
10	1,1590	38,9247	1,1330	38,9295
15	2,7710	38,6252	2,7090	38,6366
20	9,9420	37,3492	9,7210	37,3872

As stated in the table as well as shown graphically in the diagrams respectively the convex probe's- and the linear probe's MSE and PSNR scores are range of 0-15 per cent, is followed with a quadruple effect, stating a relatively massive loss on the image quality. Whereas looking at the PSNR numerical figures, the values stay within the defined safe boundaries of image quality.

As for the choice of the probe, there is no clear distinction between the figures, whereas it needs to be mentioned that the increasing level of compression seems to be leading in favor of a better linear probe's performance in terms of yielding a better image quality which drastically rise up with the increasing JPEG compression ratio.

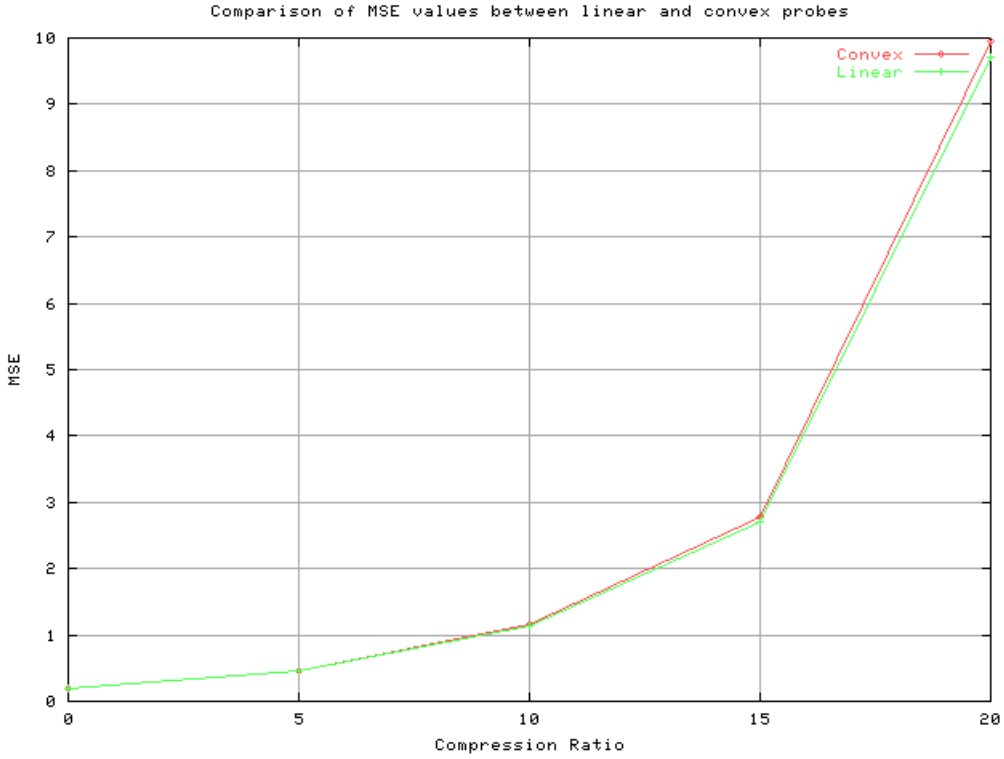


Figure 8.5 Comparison between MSE and CR with regard to different probe types.

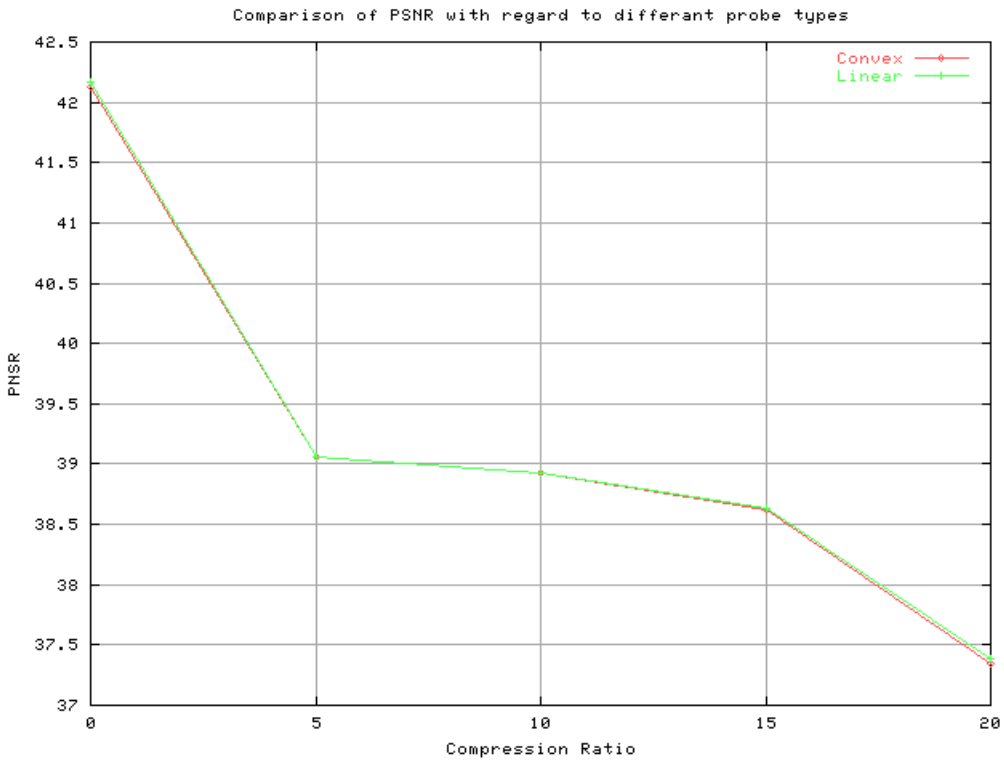


Figure 8.6 Comparison between PSNR and CR with regard to different probe selection.

8.3 Comparison of Image Quality and Central Frequency With Regard to Compression Ratios

As an outcome of the physical properties of the probe type, the central frequency of the probe itself is also responsible on the ultrasound energy, thus the quality fluctuation of the image. The results are obtained with a simulation of the simulated image set with the a linear probe, adding no additional attenuation but changing the probe's central frequency (Cf) and compressing at different levels using JPEG.

Table 8.4
Quantitative representation of image quality metrics with regard to different central frequency levels using a linear probe.

Compression Ratio	**1 Mhz**		**5 Mhz**		**10 Mhz**	
	MSE	**PSNR**	**MSE**	**PSNR**	**MSE**	**PSNR**
0	0,1996	42,0957	0,1929	42,1616	0,1956	42,1349
5	0,4724	39,0536	0,4562	39,0567	0,7669	38,9982
10	1,1690	38,9228	1,1290	38,9303	1,1770	38,9213
15	2,8180	38,6165	2,7210	38,6344	2,7510	38,6289
20	10,1000	37,3222	9,7590	37,3807	9,8970	37,3570

As it is represented in the data table Table 8.4 and in the diagram Figure 8.7 respectively the MSE values increase parallel to each other, whereas two remarkable points need to mentioned there. One of them is the dispersive behavior of the highest frequency used within the compression rates 2, 5-10 per cent, which leads to the result, that higher frequencies with a low compression level lead to poor image quality of more than 50 %. The second one would be the overall leading performance level of 5 MHz central frequency. Furthermore the graphs also suppose that higher compression levels might drag the performances of various central frequencies to divaricate from each other. Nevertheless the images manage to stay within the pre-defined image quality boundaries 30-50 dBs.

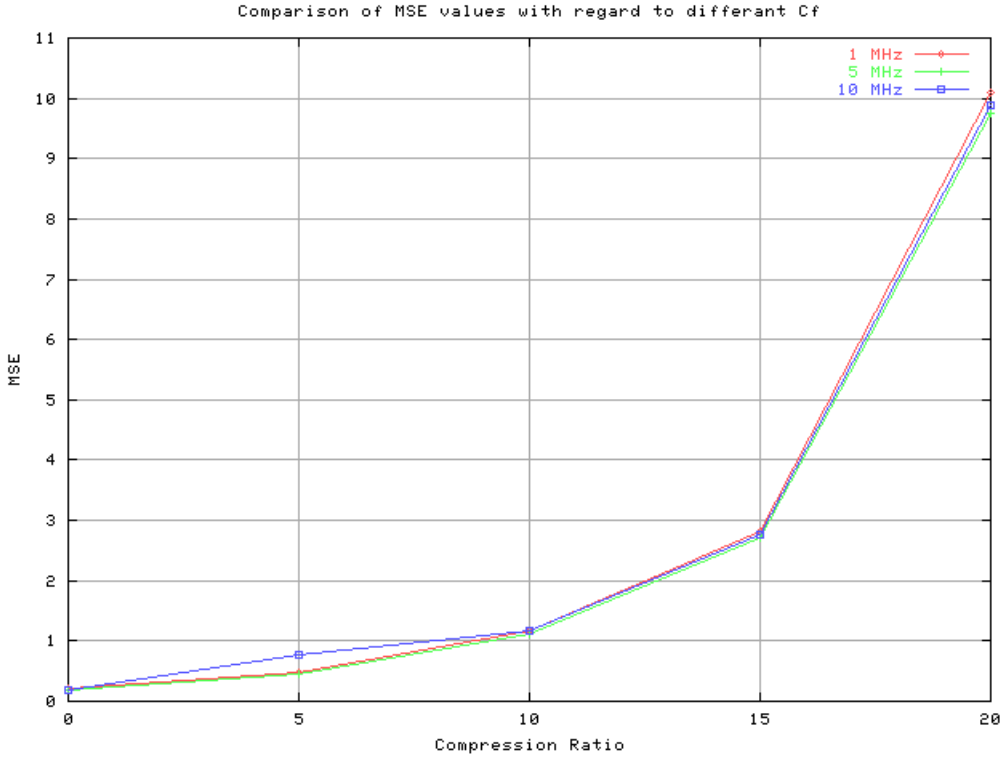


Figure 8.7 Comparison between CRs on JPEG images with various Cf.

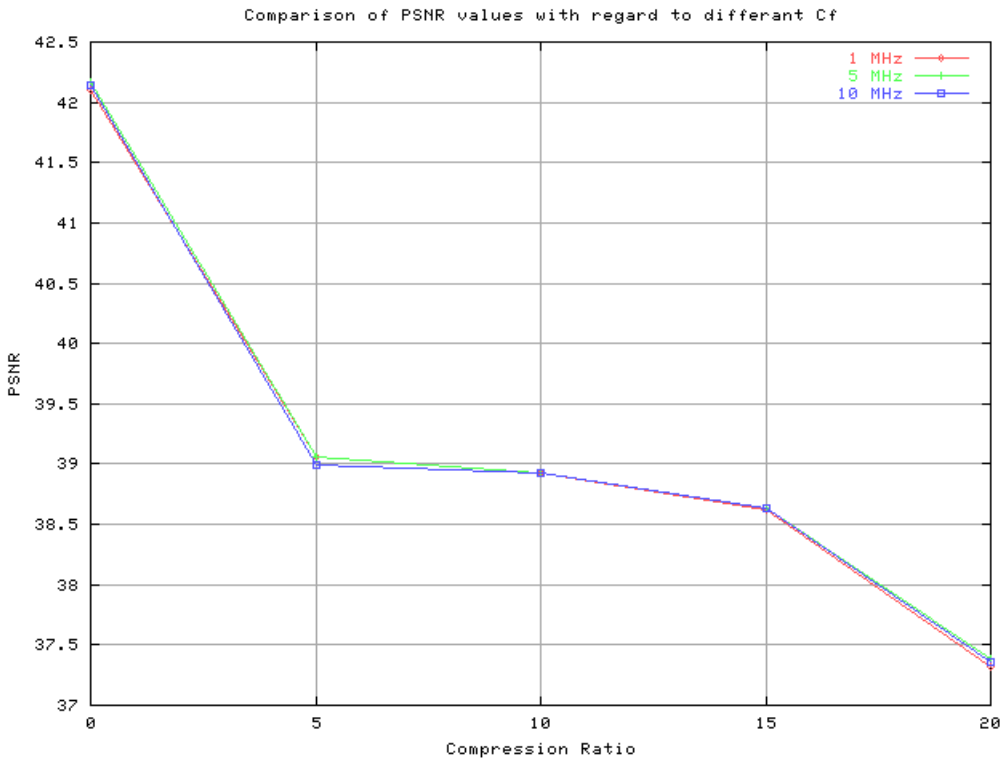


Figure 8.8 Comparison between PSNR and CR on JPEG Images with various Cf

8.4 Comparison of Despeckled and Non-Despeckled Images in Terms of Compression Ratios

In this scenario the simulated data set of 10 images are compressed using JPEG Algorithm. Prior to compression the same data set underwent either a despeckling algorithm via Photoshop CS3 or not. Thus the yielded results are shown in the table below.

Table 8.5
Quality metrics of despeckled and non-despeckled images.

Compression Ratio	Non-Despeckled		Despeckled	
	MSE	PSNR	MSE	PSNR
0	0,19	42,17	0,17	42,42
5	0,46	39,06	0,39	39,07
10	1,13	38,93	0,98	38,96
15	2,71	38,64	2,35	38,70
20	9,72	37,39	8,42	37,61

As seen in Figure 8.6 above and Figure 8.6 below, at the beginning of compression the quality values of the image set reflect a flow of course with a slight but clear difference of up to 2 per cent in favor of the despeckled images. Within the range of 5 to 12, 5 per cent parallel to the increasing CR the quality metrics are closing the gap concerning either the MSE or the PSNR. After 15 per cent of compression another branching begins, favoring the images despeckled before undergoing a compression process.

The overall course of both quality metrics do not change due to a despeckling improvement and the PSNR values stay within the defined range.

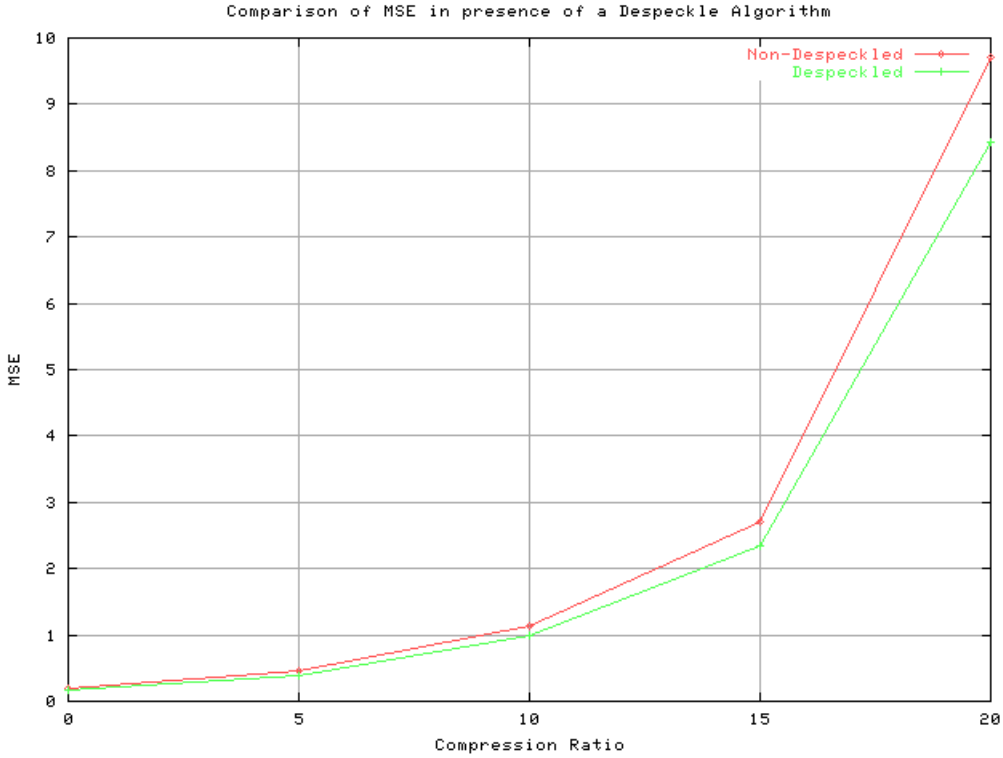


Figure 8.9 Comparison between MSE and CR on JPEG images with regard to a despeckle algorithm

8.5 Comparison of Image Quality and the Influence of Attenuation

The simulated set of images is prone to 1,5 dB / cm*MHz (Appendix E). As previously the data for a linear with the central frequency of 5 MHz is used for comparison.

As expected the attenuation effected directly the image quality proportionally in terms of CRs i.e. the image data set without attenuation outperformed the image data set including a certain level of attenuation. There was up to 10 per cent loss of image quality with rising CR in terms of evaluation metrics MSE and PSNR. The anomaly, seen at the beginning, is evaluated also with other simulated image data sets and it has been seen that it keeps on happening. Thus it may be regarded as simply as an exception or as an exceptional response of JPEG compression algorithm towards attenuation. Last but not least the images stayed within the pre-defined boundaries.

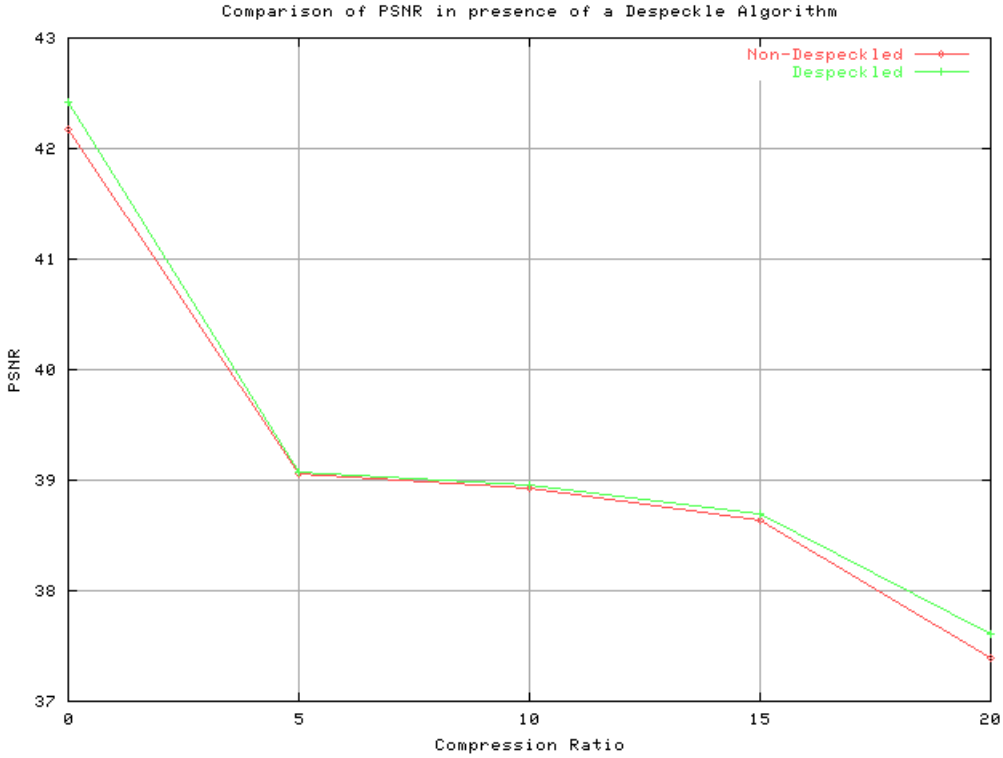


Figure 8.10 Comparison between PSNR and CR on JPEG images with regard to a despeckle algorithm.

Table 8.6
Effects of attenuation on image quality metrics.

CR	MSE	PSNR
0	0,1922	42,6582
5	0,4612	38,6097
10	1,2670	38,4818
15	2,9030	38,1922
20	9,7280	36,9572

8.6 Comparison of a Gaussian Noise Source and the Quality Metrics of the Simulated Image Set

The simulated image set is subjected to a 1 % per cent Gaussian noise source via Photoshop CS3 before undergoing a JPEG compression and compared with the

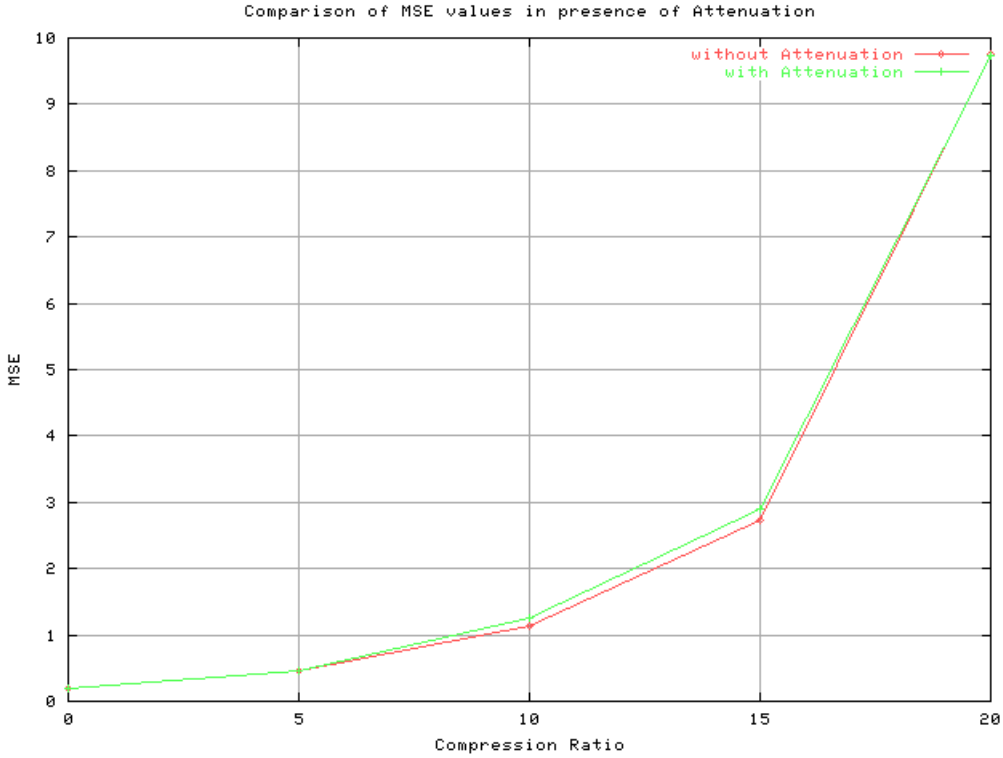


Figure 8.11 The overall flow of MSE and CR on JPEG Images with Attenuation.

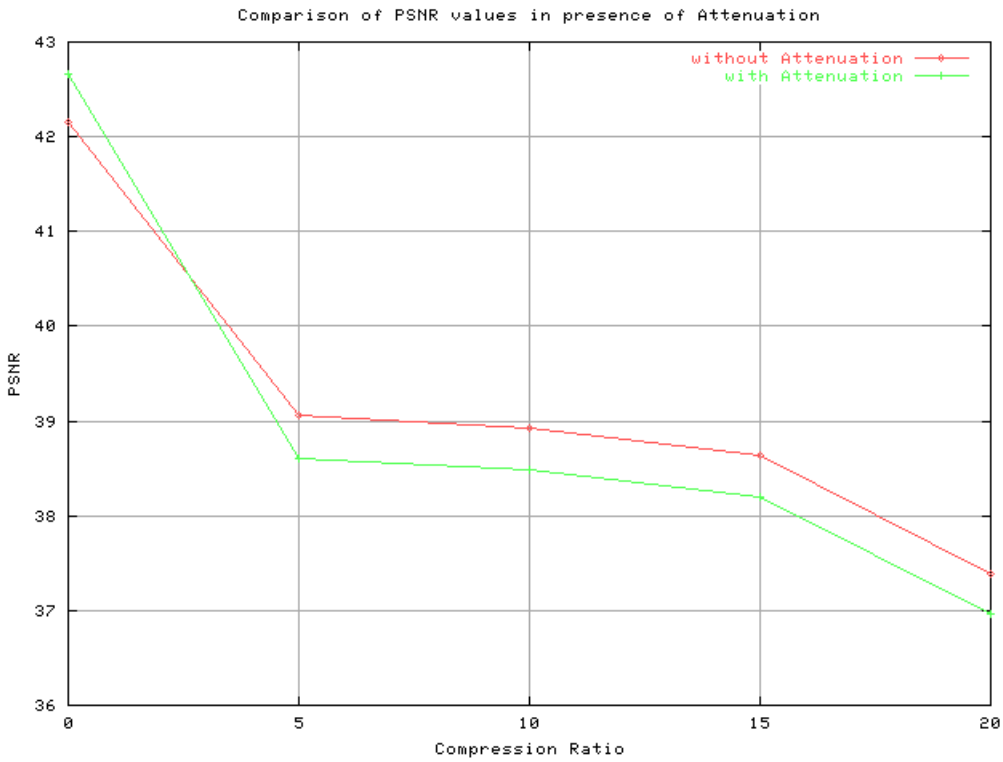


Figure 8.12 The overall flow of PSNR and CR on with Jpeg compressed images including attenuation.

values that did not include any added noise. The numerical presentation of the results are shown as follows.

Table 8.7
Quantitative representation of image quality metrics on the image data set with a gaussian noise source.

CR	MSE	PSNR
0	0,3591	40,6042
5	0,8520	38,9822
10	2,1070	38,7480
15	5,0600	38,2079
20	18,1600	35,9999

For a graphical demonstration and comparison, the data of the simulated image set with a linear probe of 5 MHz central frequency yielded previously is compared to the values above.

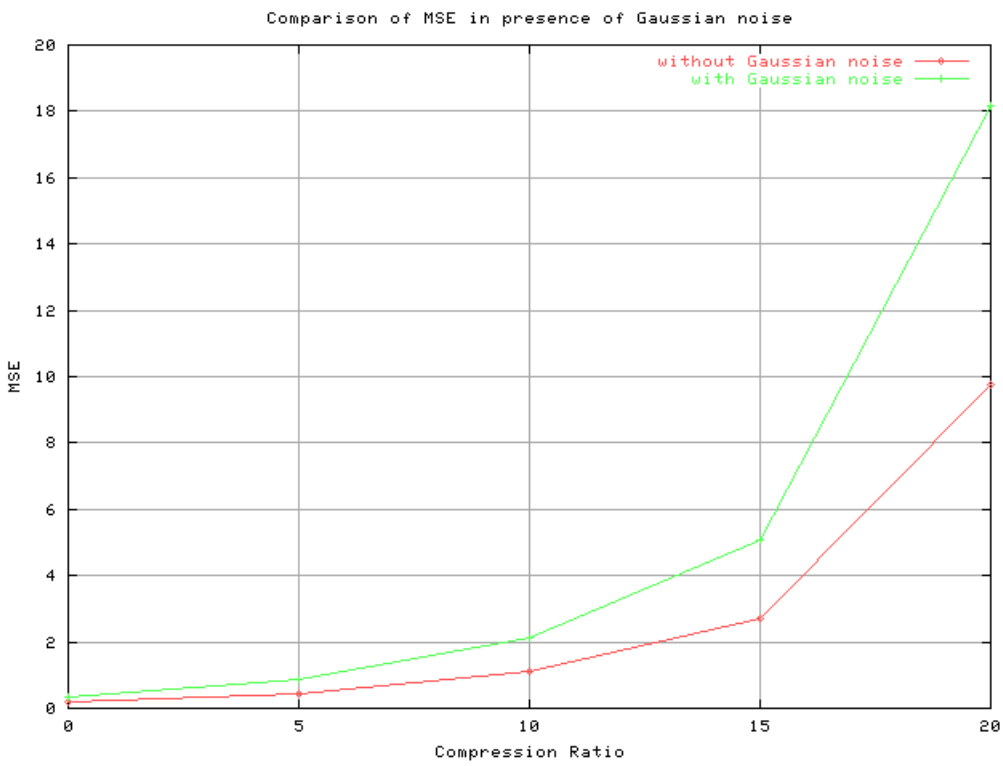


Figure 8.13 Comparison between MSE and CR concerning a gaussian noise source.

As predicted the graphs show how drastically an addition of a Gaussian noise source changes the image quality in terms of the evaluation parameters MSE and PSNR respectively. Parallel to the increasing CR the MSE score increases with a rising slope an the PSNR score decreases dramatically. The at the beginning (0 % Compression) 2 per cent gap rises continually and reaches up to 19 per cent loss of image quality.

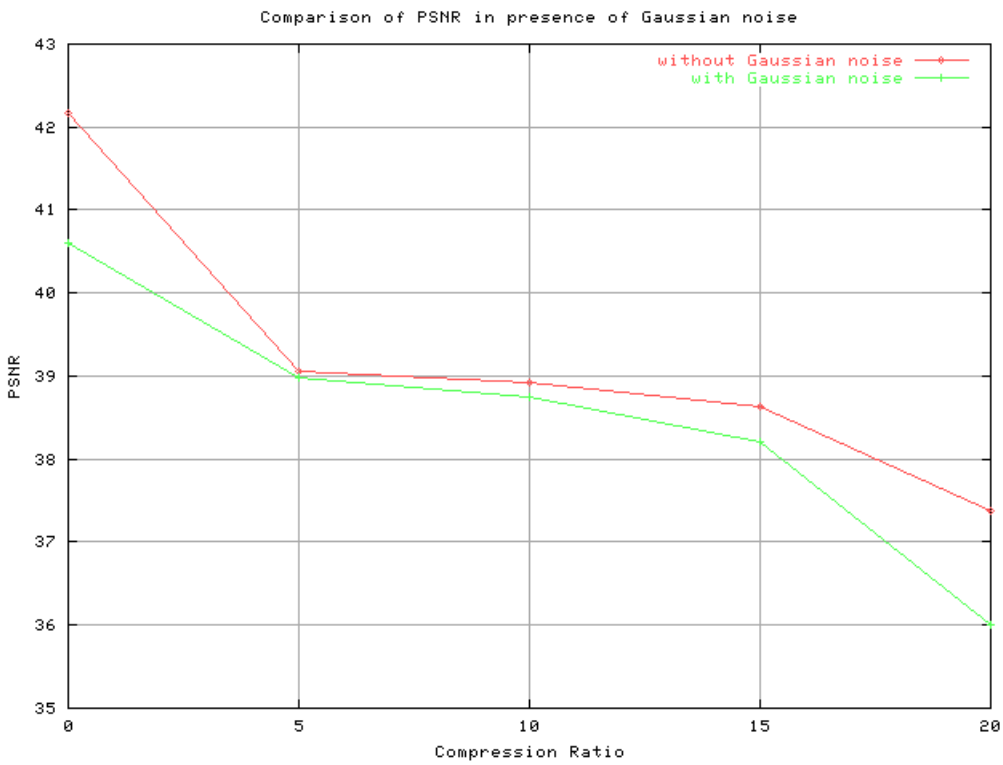


Figure 8.14 Comparison between MSE and CR concerning a gaussian noise addition.

Surprisingly there was also a point where both image sets had the same PSNR scores when they were compressed within 5-7 per cent. This behavior couldn't be explained and reexamined with other simulated images, whereas no certain repeatability has been reported. Finally, although the drastic decrease on the image quality in terms of quality metrics, the gaussian noise image data set has stayed within the pre-defined evaluation boundaries (30-50 dB).

8.7 Comparison Between Real and Simulated Images Compressed With JPEG Algorithm

This case based on the comparison of the real image data set and simulated image data set. Both sets are the same polycystic kidney images used throughout the study. 5 real images, yielded via linear probe at 5 MHz constituted the real image data set and 10 simulated images at the same technical specifications made up the simulated image data set. The attenuation level for the simulated ones was set to zero as usual, whereas there was no healthy information about the attenuation level of the images, acquired in real. The corresponding numerical presentation is shown below.

Table 8.8
Quantitative Representation of quality metrics of real and simulated images compressed with JPEG Algorithm

Compression Ratio	**Simulated**		**Real**	
	MSE	**PSNR**	**MSE**	**PSNR**
0	0,19	42,17	0,17	48,15
5	0,46	39,06	0,40	44,59
10	1,13	38,93	0,99	44,45
15	2,71	38,64	2,37	44,11
20	9,72	37,39	8,51	42,68

After both image sets are compressed via JPEG at different levels, quality evaluation resulted in the below and below drawn graphics. As it is obvious from the Figure 8.15 and Figure 8.16 as well, overally the real image data set outperforms at all points the simulated image data set.

In terms of quality parameters the simulated image data set, even without any compression is not able to reach the final values of the real images after 20 per cent of compression. Both curves have a parallel flow along the compression steps, which suggests the characteristic effect of JPEG compression on image quality. The increasing

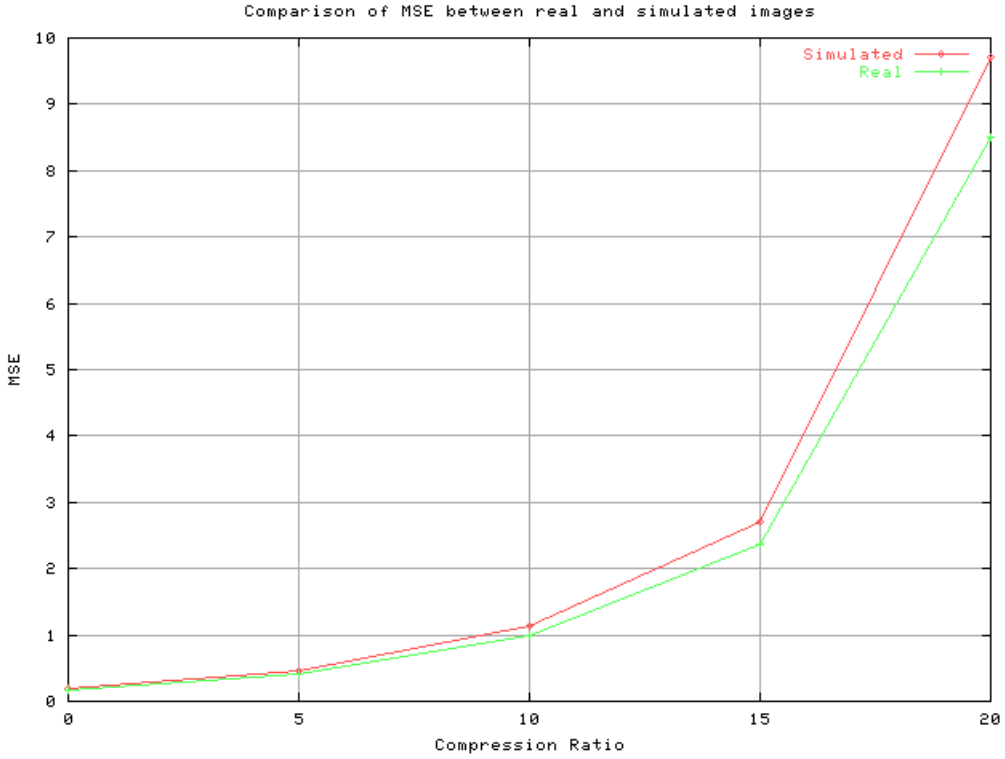


Figure 8.15 Comparison between MSE and CR of real and simulated image data sets.

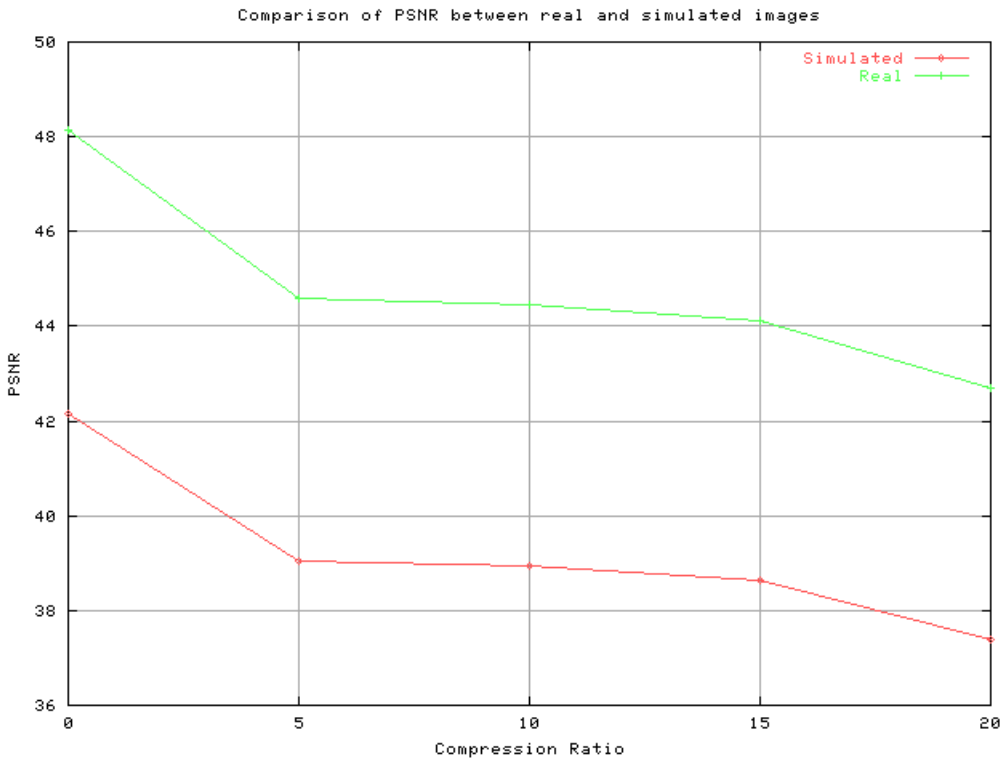


Figure 8.16 Comparison between real and simulated images in terms of PSNR values.

JPEG compression rate influences the simulated images to approach to the pre-defined threshold line of 30dB and predicts a rising rate of compression more than 25 % will pull down the PSNR score below this line. But in general, the simulation tool Field II seems to be performing well in terms of being able to generate images between 30-50 dBs.

9. CONCLUSION

No doubt that medical imaging, with its all affairs, is an emerging and indispensable technology tool that helps physicians on diagnosis. However, as nowadays healthcare professionals confront problems concerning challenges among quality, diagnostic efficiency and moreover performance, it has not to be forgotten that one of the most spread and widely used medical diagnostic technology, ultrasound, as the rest, face also some trade-offs as a cause of loss of image quality due to lossy compression algorithms and various sources of environmental and internal errors. Thus the list of these theoretical problems is so long and extensive, it is neither timely nor physically possible to examine them. Hence this thesis work goaled the intention shortening the time needed to create different set-ups by using computer simulated images, comparing the image quality in terms of lossy compression algorithms JPG, JPEG 2000 [89] and JLS, and the effects of various external error sources on medical image quality using the mostly benefited evaluation parameters.

To mention it first, the simulation tool, Field II, was able to create polycystic kidney images similar to the real ones yielded. Although the simulation procedure is not to be considered, in terms of computational time needed, that efficient, the images obtained were more than enough to simulate the intended set-ups for possible external errors, that are not always clinically reachable.

Secondly, using real and simulated images a comparison among the lossy compression algorithms could be made and as a combined result, the study showed that the JPEG compression algorithm has been outperformed by JPEG 2000 and JLS, whereas JLS and JPEG 2000 took the lead interchangeably there in terms of generating least loss with regard to quality parameters, which is also confirming with the previous literature [39].

Last but not least and probably the most interesting part of the work was that

by using scenarios dealing with creating different error sources each time the study showed that there is an oversighted trade-off between choosing the right compression algorithm and/but not being able to avoid other external errors such as a gaussian noise source, attenuation or diversities arising from the physical structure and settings of the related ultrasound device.

10. FUTURE WORK

Medical technology, thus medical devices, services and demands of patients as well as of physicians are on an unpreventable rise with a nearly exponential slope. On the other hand, the time, spent for innovations, advances and in this case clinical trials for image quality assessment procedures, is of vital value as it directly equals to money. Hence among the many suggestions, this thesis work also tries to give insight to, evaluating various modalities of ultrasound imaging by legitimate simulation methods such as Field II and combining that with a proper model observer diagnostic accuracy assessment would be a continuation for this study [84].

Expanding the idea by enhancing the rate of different error sources, such as applying a gaussian error just on a lesion and not on the whole image, and thus achieving a frequency level of the error source' effects on image quality in an appropriate setting would also direct developers to deal with them.

Last but least, as tele-radiology and tele-medicine are emerging technologies as well, quality evaluation of images [39] with regard to transmission rates and compression techniques and moreover using sabermetrics and data-mining realizing a related diagnostic accuracy estimation would be another time and thus money saving work proposals.

APPENDIX A. KIDNEY SIMULATION by FIELD II

```
%  Start up the Field simulation system
%  manipulated by A.Levent KURTOGLU,May 15,2008

p=path;
path(p, 'C:\Users\yako\Desktop\M.Sc.Thesis Docu\...
           Programs regarding thesis\kidney_example')

field_init(0)

%  Make the scatterer file

N=1e6;
[phantom_positions, phantom_amplitudes] = human_kidney_phantom (N);

%  Save the data

save pht_data.mat phantom_positions phantom_amplitudes
%  Phased array B-mode scan of a human kidney
%
%  This script assumes that the field_init procedure has been called
%  Here the field simulation is performed and the data is stored
%  in rf-files; one for each rf-line done. The data must then
%  subsequently be processed to yield the image. The data for the
%  scatteres are read from the file pht_data.mat, so that the procedure
%  can be started again or run for a number of workstations.
%
%  Example by Joergen Arendt Jensen and Peter Munk,
%  Version 1.1, April 1, 1998, JAJ.
%  manipulated by A.Levent KURTOGLU,May 15,2008


%  Ver. 1.1: 1/4-98: Procedure xdc_focus_center inserted to use the new
%                    focusing scheme for the Field II program
```

```
%  Generate the transducer apertures for send and receive
f0=7e6;                  %  Transducer center frequency [Hz]
fs=100e6;                %  Sampling frequency [Hz]
c=1540;                  %  Speed of sound [m/s]
lambda=c/f0;             %  Wavelength [m]
width=lambda/2;          %  Width of element
element_height=5/1000;   %  Height of element [m]
kerf=lambda/10;          %  Kerf [m]
focus=[0 0 90]/800;      %  Fixed focal point [m]
N_elements=128;          %  Number of physical elements

%  Set the sampling frequency
set_sampling(fs);
set_field ('show_times', 5);
set_field ('att',25*100);
set_field ('Freq_att',0.5*100/1e6);
set_field ('att_f0',3e6);
set_field ('use_att',1);



% Generate aperture for emission
xmit_aperture=xdc_linear_array (N_elements, width, element_height, ...
                                kerf, 1, 5, focus);

% Set the impulse response and excitation of the xmit aperture
impulse_response=sin(2*pi*f0*(0:1/fs:2/f0));
impulse_response=impulse_response.*hanning(max(size(impulse_response)))';
xdc_impulse (xmit_aperture, impulse_response);

excitation=sin(2*pi*f0*(0:1/fs:2/f0));
xdc_excitation (xmit_aperture, excitation);

% Generate aperture for reception
receive_aperture=xdc_linear_array (N_elements, width, element_height, ...
                                   kerf, 1, 5, focus);

% Set the impulse response for the receive aperture
```

```
xdc_impulse (receive_aperture, impulse_response);

% Load the computer phantom
load pht_data

% Set the different focal zones for reception
focal_zones=[5:1:150]'/1000;
Nf=max(size(focal_zones));
focus_times=(focal_zones-10/1000)/1540;
z_focus=60/1000;          %  Transmit focus

% Set the apodization
apo=hanning(N_elements)';
xdc_apodization(xmit_aperture, 0, apo);
xdc_apodization(receive_aperture, 0, apo);

% Do phased array imaging
no_lines=128;                  %  Number of lines in image
image_width=90/180*pi;         %  Size of image sector [rad]
dtheta=image_width/no_lines;   %  Increment for image

% Do imaging line by line
for i=1:128

  if ~exist(['rf_data/rf_ln',num2str(i),'.mat'])

    cmd=['save rf_data/rf_ln',num2str(i),'.mat i']
    eval(cmd)

  % Set the focus for this direction
    theta=(i-1-no_lines/2)*dtheta;
    xdc_focus (xmit_aperture, 0, [z_focus*sin(theta) 0 z_focus*cos(theta)]);
    xdc_focus (receive_aperture, focus_times, [focal_zones*sin(theta) ...
               zeros(max(size(focal_zones)),1) focal_zones*cos(theta)]);

    % Calculate the received response
    [rf_data, tstart]=calc_scat(xmit_aperture, receive_aperture, ...
                                phantom_positions, phantom_amplitudes);
```

```
    % Store the result
    cmd=['save rf_data/rf_ln',num2str(i),'.mat rf_data tstart']
    eval(cmd)
    end
  end

% Free space for apertures
xdc_free (xmit_aperture)
xdc_free (receive_aperture)
% Make the image interpolation for the polar scan
% Set initial parameters

D=10;       % Sampling frequency decimation factor
fs=100e6/D; % Sampling frequency  [Hz]
c=1540;     % Speed of sound [m]
no_lines=128;                  % Number of lines in image
image_width=90/180*pi;         % Size of image sector [rad]
dtheta=image_width/no_lines;   % Increment for image

% Read the data and adjust it in time
min_sample=0;
for i=1:no_lines

  % Load the result
  cmd=['load rf_data/rf_ln',num2str(i),'.mat']
  eval(cmd)

  % Find the envelope
  rf_env=abs(hilbert([zeros(round(tstart*fs-min_sample),1); rf_data]));
  env(1:max(size(rf_env)),i)=rf_env;
  end

% Do logarithmic compression to 40 dB
dB_Range=50;
env=env-min(min(env));
log_env=20*log10(env(1:D:max(size(env)),:)/max(max(env)));
log_env=255/dB_Range*(log_env+dB_Range);
```

```
% Get the data into the proper format
start_depth=0.02;   % Depth for start of image in meters
image_size=0.105;   % Size of image in meters
skipped_samples=0;
samples=max(size(log_env));

start_of_data=(skipped_samples+1)/fs*c/2;        % Depth for start of data in meters
end_of_data=(skipped_samples+samples+1)/fs*c/2;  % Depth for end of data in meters
delta_r=c/2*1/fs;                                % Sampling interval for data in meters

theta_start=-no_lines/2*dtheta;   % Angle for first line in image

Nz=1024;                       % Size of image in pixels
Nx=1024;                       % Size of image in pixels
scaling=1;                     % Scaling factor form envelope to image

[N,M]=size(log_env);
D=floor(N/1024);
env_disp=uint8(255*log_env(1:D:N,:)/max(max(log_env)));

% Make the tables for the interpolation
make_tables(start_depth, image_size,          ...
            start_of_data, delta_r*D, round(samples/D),  ...
            -theta_start, -dtheta, no_lines, scaling, Nz, Nx);

% Perform the interpolation
img_data=make_interpolation (env_disp);

% Display the image
image(((1:Nz)/Nz-0.5)*image_size*1000,((1:Nx)/Nx*image_size+start_depth)*1000,img_data)
axis('image')
set(gca,'FontSize',14)
colormap(gray(256))
xlabel('Lateral distance [mm]')
ylabel('Axial distance [mm]')
axis([-50 50 20 105])
```

A.1 The Sampling Frequency

```
% Set the sampling frequency the system uses.
%
% Remember that the pulses used in all apertures must
% be reset for the new sampling frequency to take effect.
%
% Calling:  set_sampling (fs);
%
% Parameters:  fs - The new sampling frequency.
%
% Return:  nothing.
%
% Version 1.0, December 7, 1995 by Joergen Arendt Jensen
% manipulated by A.Levent KURTOGLU,May 15,2008


function res=set_sampling(fs)

% Call the C-part of the program to initialize it
  Mat_field (5020,fs);
```

A.2 The Impulse Response of an Aperture

```
% Procedure for setting the impulse response of an aperture
%
% Calling:  xdc_impulse (Th,pulse);
%
% Parameters:  Th - Pointer to the transducer aperture.
%              pulse - Impulse response of aperture as row vector
%
% Return:    None
%
% Version 1.01, May 20, 1997 by Joergen Arendt Jensen
% manipulated by A.Levent KURTOGLU,May 15,2008
```

```
function res=xdc_impulse(Th, pulse)

% Test that pulse is of right dimension
  [n,m]=size(pulse);
  if (n ~= 1)
    error ('Pulse must be a row vector');
    end

% Call the C-part of the program to show aperture
  Mat_field(1050,Th,pulse);
```

A.3 The Excitation Pulse of an Aperture

```
% Procedure for setting the excitation pulse of an aperture
%
% Calling: xdc_excitation (Th,pulse);
%
% Parameters: Th - Pointer to the transducer aperture.
%             pulse - Excitation pulse of aperture as row vector
%
% Return: None
%
%  Version 1.0, November 27, 1995 by Joergen Arendt Jensen
%  manipulated by A.Levent KURTOGLU,May 15,2008


function res=xdc_excitation(Th, pulse)

% Test that pulse is of right dimension
  [n,m]=size(pulse);
  if (n ~= 1)
    error('Pulse must be a row vector');
    end

% Call the C-part of the program to show aperture
  Mat_field(1051,Th,pulse);
```

A.4 Apodization Time Line for an Aperture

```
% Procedure for creating an apodization time line for an aperture
%
% Calling: xdc_apodization(Th, times, values);
%
% Parameters: Th     - Pointer to the transducer aperture.
%             times  - Time after which the associated apodization is valid.
%             values - Apodization values. Matrix with one row for each
%                      time value and a number of columns equal to the
%                      number of physical elements in the aperture.
%
% Return: none
%
% Version 1.01, June 19, 1998 by Joergen Arendt Jensen
% manipulated by A.Levent KURTOGLU,May 15,2008


function res=xdc_apodization(Th,times,values)

% Check the times vector
  [m1,n]=size(times);
  if (n ~= 1)
    error('Times vector must have one column');
    end

  [m2,n]=size(values);

% Check both arrays
  if (m1 ~= m2)
    error('There must be the same number of rows for times and values');
    end

% Call the C-part of the program to insert apodization
  Mat_field(1070,Th,times,values);
```

A.5 Linear Array Transducer

```
% Procedure for creating a linear array transducer
%
% Calling: Th=xdc_linear_array (no_elements, width, height, kerf, ...
                                 no_sub_x, no_sub_y, focus);
%
% Parameters: no_elements -  Number of physical elements.
%             width       -  Width in x-direction of elements.
%             height      -  Width in y-direction of elements.
%             kerf        -  Width in x-direction between elements.
%             no_sub_x    -  Number of sub-divisions in x-direction of elements.
%             no_sub_y    -  Number of sub-divisions in y-direction of elements.
%             focus[]     -  Fixed focus for array (x,y,z). Vector with three elements.
%
% Return: A handle Th as a pointer to this transducer aperture.
%
% Version 1.0, November 20, 1995 by Joergen Arendt Jensen
% manipulated by A.Levent KURTOGLU,May 15,2008


function Th=xdc_linear_array(no_elements, width, height, ...
                             kerf, no_sub_x, no_sub_y, focus)

% Check that all parameters are valid
  if (no_elements<1)
    error('Field error: Illegal number of physical transducer elements')
    end

  if (width<=0) | (height<=0)
    error('Field error: Width or height is negativ or zero')
    end

  if (kerf<0)
    error('Field error: Kerf is negativ')
    end

  if (no_sub_x<1) | (no_sub_y<1)
```

```
    error('Field error: Number of mathematical elements must 1 or more')
    end

  if (min(size(focus))~=1) | (max(size(focus))~=3)
    error('Field error: Focus must be a vector with three elements')
    end

% Call the C-part of the program to create aperture
  Th=Mat_field(1001,no_elements, width, height, kerf, no_sub_x, no_sub_y, focus);
```

A.6 Freeing the Storage Occupied by an Aperture

```
% Procedure for freeing the storage occupied by an aperture
%
% Calling: xdc_free(Th);
%
% Parameters: Th - Pointer to the transducer aperture.
%
% Return: None
%
% Version 1.0, November 28, 1995 by Joergen Arendt Jensen
% manipulated by A.Levent KURTOGLU,May 15,2008

function res=xdc_free(Th)

% Call the C-part of the program to show aperture
  Mat_field(1040,Th);
```

A.7 Calculation the Received Signal from a Collection of Scatterers

```
% Procedure for calculating the received signal from a collection of scatterers.
%
% Calling: [scat, start_time]=calc_scat(Th1, Th2, points, amplitudes);
```

```
%
% Parameters: Th1        - Pointer to the transmit aperture.
%             Th2        - Pointer to the receive aperture.
%             points     - Scatterers. Vector with three columns (x,y,z)
%                          and one row for each scatterer.
%             amplitudes - Scattering amplitudes. Row vector with one
%                          entry for each scatterer.
%
% Return: scat       - Received voltage trace.
%         start_time - The time for the first sample in scat.
%
% Version 1.0, November 28, 1995 by Joergen Arendt Jensen
% manipulated by A.Levent KURTOGLU,May 15,2008

function[scat, start_time]=calc_scat(Th1, Th2, points, amplitudes)

% Check the point array
  [m1,n]=size(points);
  if (n ~= 3)
    error('Points array must have three columns');
    end
  [m2,n]=size(amplitudes);
  if (m1 ~= m2)
    error('There must be the same number of rows for points and amplitudes arrays');
    end

% Call the C-part of the program to show aperture
  [scat, start_time]=Mat_field(4005,Th1,Th2,points, amplitudes);
```

A.8 Calculation of Different Weight Tables for the Interpolation

```
% Function for calculating the different weight tables
% for the interpolation. The tables are calculated according
% to the parameters in this function and are stored in
% the internal C-code.
```

```
%
% Input parameters:
%
% start_depth - Depth for start of image in meters
% image_size  - Size of image in meters
%
% start_of_data - Depth for start of data in meters
% delta_r       - Sampling interval for data in meters
% N_samples     - Number of samples in one envelope line
%
% theta_start - Angle for first line in image
% delta_theta - Angle between individual lines
% N_lines     - Number of acquired lines
%
% scaling - Scaling factor from envelope to image
% Nz      - Size of image in pixels
% Nx      - Size of image in pixels
%
% Output: Nothing, everything is stored in the C program
%
% Calling: make_tables(start_depth, image_size, ...
%                      start_of_data, delta_r, N_samples, ...
%                      theta_start, delta_theta, N_lines, ...
%                      scaling, Nz, Nx);
%
% Version 1.0, 14/2-1999, JAJ
% Version 1.1, 11/9-2001, JAJ: Help text corrected
% manipulated by A.Levent KURTOGLU,May 15,2008


function res=make_tables(start_depth, image_size, ...
                         start_of_data, delta_r, N_samples, ...
                         theta_start, delta_theta, N_lines, ...
                         scaling, Nz, Nx)

% Call the appropriate function
fast_int(1,start_depth, image_size, ...
         start_of_data, delta_r, N_samples, ...
```

```
        theta_start, delta_theta, N_lines, ...
        scaling, Nz, Nx);

% Return nothing
ret=0;
```

APPENDIX B. CYST SIMULATION by FIELD II

```
% Start up the Field simulation system
%
% Version 1.1, 1/4-98, JAJ
% manipulated by A.Levent KURTOGLU,May 15,2008


path(path, 'C:\Users\yako\Desktop\M.Sc.Thesis Docu\ ...
               Programs regarding thesis/Field_II_PC5')

field_init(0)

% Make the scatteres for a simulation and store
% it in a file for later simulation use

% Joergen Arendt Jensen, Feb. 26, 1997
% manipulated by A.Levent KURTOGLU,May 15,2008


[phantom_positions, phantom_amplitudes]=cyst_pht(100000);
save pht_data.mat phantom_positions phantom_amplitudes

% This script shows how a linear array B-mode system scans an image
%
% This script assumes that the field_init procedure has been called
% Here the field simulation is performed and the data is stored
% in rf-files; one for each rf-line done. The data must then
% subsequently be processed to yield the image. The data for the
% scatteres are read from the file pht_data.mat, so that the procedure
% can be started again or run for a number of workstations.
% Ver. 1.1: 1/4-98: Procedure xdc_focus_center inserted to use the new
% focusing scheme for the Field II program

% Generate the transducer apertures for send and receive
f0=3.5e6;              %  Transducer center frequency [Hz]
```

```
fs=100e6;              %  Sampling frequency [Hz]
c=1540;                %  Speed of sound [m/s]
lambda=c/f0;           %  Wavelength [m]
width=lambda;          %  Width of element
element_height=5/1000; %  Height of element [m]
kerf=0.05/1000;        %  Kerf [m]
focus=[0 0 70]/1000;   %  Fixed focal point [m]
N_elements=192;        %  Number of physical elements
N_active=64;           %  Number of active elements

% Do not use triangles
set_field('use_triangles',0);

% Set the sampling frequency
set_sampling(fs);

% Generate aperture for emission
xmit_aperture=xdc_linear_array(N_elements, width, element_height, kerf, 1, 10,focus);

% Set the impulse response and excitation of the xmit aperture
impulse_response=sin(2*pi*f0*(0:1/fs:2/f0));
impulse_response=impulse_response.*hanning(max(size(impulse_response)))';
xdc_impulse(xmit_aperture, impulse_response);

excitation=sin(2*pi*f0*(0:1/fs:2/f0));
xdc_excitation (xmit_aperture, excitation);

% Generate aperture for reception
receive_aperture=xdc_linear_array(N_elements, width, element_height, kerf, 1, 10,focus);

% Set the impulse response for the receive aperture
xdc_impulse(receive_aperture, impulse_response);

% Load the computer phantom
load pht_data

% Set the different focal zones for reception
focal_zones=[30:20:200]'/1000;
```

```
Nf=max(size(focal_zones));
focus_times=(focal_zones-10/1000)/1540;
z_focus=60/1000; %  Transmit focus

% Set the apodization
apo=hanning(N_active)';

% Do linear array imaging
no_lines=50;              %  Number of lines in image
image_width=40/1000;      %  Size of image sector
d_x=image_width/no_lines; %  Increment for image

% Do imaging line by line
%for i=[1:no_lines]
for i=[24]
i
   % The the imaging direction
   x=-image_width/2 +(i-1)*d_x;

   % Set the focus for this direction with the proper reference point
   xdc_center_focus(xmit_aperture, [x 0 0]);
   xdc_focus(xmit_aperture, 0, [x 0 z_focus]);
   xdc_center_focus(receive_aperture, [x 0 0]);
   xdc_focus(receive_aperture, focus_times, [x*ones(Nf,1), zeros(Nf,1), focal_zones]);

   % Calculate the apodization
   N_pre=round(x/(width+kerf) + N_elements/2 - N_active/2);
   N_post=N_elements - N_pre - N_active;
   apo_vector=[zeros(1,N_pre) apo zeros(1,N_post)];
   xdc_apodization(xmit_aperture, 0, apo_vector);
   xdc_apodization(receive_aperture, 0, apo_vector);

   % Calculate the received response
   [rf_data, tstart]=calc_scat(xmit_aperture, receive_aperture, ...
                     phantom_positions, phantom_amplitudes);

   % Store the result
   cmd=['save sim_cyst/rf_ln',num2str(i),'.mat rf_data tstart']
```

```
   eval(cmd)
   end

% Free space for apertures
xdc_free(xmit_aperture)
xdc_free(receive_aperture)

% Compress the data to show 60 dB of
% dynamic range for the cyst phantom image
%
% version 1.3 by Joergen Arendt Jensen, April 1, 1998.
% manipulated by A.Levent KURTOGLU,May 15,2008

f0=3.5e6;                 %  Transducer center frequency [Hz]
fs=100e6;                 %  Sampling frequency [Hz]
c=1540;                   %  Speed of sound [m/s]
no_lines=50;              %  Number of lines in image
image_width=40/1000;      %  Size of image sector
d_x=image_width/no_lines; %  Increment for image

% Read the data and adjust it in time
min_sample=0;
for i=1:no_lines

  % Load the result
  cmd=['load sim_cyst/rf_ln',num2str(i),'.mat']
  eval(cmd)

  % Find the envelope
  rf_env=abs(hilbert([zeros(round(tstart*fs-min_sample),1); rf_data]));
  env(1:max(size(rf_env)),i)=rf_env;
  end

% Do logarithmic compression
D=10;  % Sampling frequency decimation factor

log_env=env(1:D:max(size(env)),:)/max(max(env));
log_env=log(log_env+0.01);
```

```
log_env=log_env-min(min(log_env));
log_env=64*log_env/max(max(log_env));

% Make an interpolated image
ID=20;
[n,m]=size(log_env)
new_env=zeros(n,m*ID);
for i=1:n
  if (rem(i,100) == 0)
    i
    end
  new_env(i,:)=abs(interp(log_env(i,:),ID));
  end
[n,m]=size(new_env)

fn=fs/D;
clf
image(((1:(ID*no_lines-1))*d_x/ID-no_lines*d_x/2)*1000, ...
      ((1:n)/fn+min_sample/fs)*1540/2*1000,new_env)
xlabel('Lateral distance [mm]')
ylabel('Axial distance [mm]')
colormap(gray(64))
brighten(0.2)
axis('image')
axis([-20 20 35 90])
```

APPENDIX C. PICTURE QUALITY MEASURES CALCULATION

```
% Program for Image / Picture Quality Measures Calculation
% The program calculates the difference Image/Picture Quality Measures

% Clear Memory & Command Window
clc;
clear all;
close all;

% Read Original & Distorted Images
origImg=imread('.\OriginalImages\1A.tif');
distImg=imread('.\DistortedImages\1A.jpg');

% If the input image is rgb, convert it to gray image
noOfDim=ndims(origImg);
if(noOfDim == 3)
    origImg=rgb2gray(origImg);
end

noOfDim=ndims(distImg);
if(noOfDim == 3)
    distImg=rgb2gray(distImg);
end

% Size Validation
origSiz=size(origImg);
distSiz=size(distImg);
sizErr=isequal(origSiz, distSiz);
if(sizErr == 0)
    disp('Error: Original Image & Distorted Image should be of same dimensions');
    return;
end

% Mean Square Error
```

```
MSE=MeanSquareError(origImg, distImg);
disp('Mean Square Error = ');
disp(MSE);

% Peak Signal to Noise Ratio
PSNR=PeakSignaltoNoiseRatio(origImg, distImg);
disp('Peak Signal to Noise Ratio = ');
disp(PSNR);

% Normalized Cross-Correlation
NK=NormalizedCrossCorrelation(origImg, distImg);
disp('MNormalized Cross-Correlation  = ');
disp(NK);

% Average Difference
AD=AverageDifference(origImg, distImg);
disp('Average Difference  = ');
disp(AD);

% Structural Content
SC=StructuralContent(origImg, distImg);
disp('Structural Content  = ');
disp(SC);

% Maximum Difference
MD=MaximumDifference(origImg, distImg);
disp('Maximum Difference = ');
disp(MD);

% Normalized Absolute Error
NAE=NormalizedAbsoluteError(origImg, distImg);
disp('Normalized Absolute Error = ');
disp(NAE);
```

C.1 Mean Square Error

```
% Program for Mean Square Error Calculation
function MSE=MeanSquareError(origImg, distImg)

origImg=double(origImg);
distImg=double(distImg);

[M N]=size(origImg);
error=origImg - distImg;
MSE=sum(sum(error .* error))/(M*N);
```

C.2 Peak Signal to Noise Ratio

```
% Program for Peak Signal to Noise Ratio Calculation
function PSNR=PeakSignaltoNoiseRatio(origImg, distImg)

origImg=double(origImg);
distImg=double(distImg);

[M N]=size(origImg);
error=origImg - distImg;
MSE=sum(sum(error .* error))/(M*N);

if(MSE > 0)
    PSNR=10*log(255*255/MSE)/log(10);
else
    PSNR=99;
end
```

C.3 Normalized Cross Correlation Calculation

```
% Program for Normalized Cross Correlation Calculation
function NK=NormalizedCrossCorrelation(origImg, distImg)
```

```
origImg=double(origImg);
distImg=double(distImg);

NK=sum(sum(origImg .* distImg))/sum(sum(origImg .* origImg));
```

C.4 Average Difference Calculation

```
% Program for Average Difference Calculation
function AD=AverageDifference(origImg, distImg)

origImg=double(origImg);
distImg=double(distImg);

[M N]=size(origImg);
error=origImg - distImg;

AD=sum(sum(error))/(M*N);
```

APPENDIX D. mmROI / ROI ALGORITHM

```
function [roi, im]=mmROI

% 1) Goal: Interactively process MULTIPLE images with MULTIPLE ROIs
%    (so-called mmROI), which returns ROI mean, std, min, max, median, area
%    and center(X,Y), and plots the mean/std values along the image series.
%
% 2) Usage: [roi, im] = mmROI; (please don't forget to add a semicolon ";"
%    at the end of this command. Otherwise, all image data will be showing on
%    screen!)
%    a) The statistic data are in roi structures, which may be save into a
%      text file (optional). If you want to see the details, you may type,
%      for example, roi.mean to show all mean values; roi.mean(1, 1, 1)
%      to display the mean value of the 1st image, the 1st ROI and the red
%      color; roi.std(3, 1, :) to show std values of all image 1, the 1st
%      ROI and the blue color, etc.
%    b) The image data are in a stack (im). You may use immovie(im) to play
%       a movie or montage(im) to show all images in one figure.
%
% 3) Limitation: all images MUST have exactly identical size. Otherwise, an
%    error will take place and the program will be terminated. The reason is
%    that all image data were loaded into a stack matrix, im(:,:,:,imNumber).
%
% 4) This can be treated as an upgraded version of previous multiple ROI
%    program for a single image (mathworks file excahnge ID: 4462). The most
%    important feature of this new mmROI is that it permits to open multiple
%    images and to process multiple ROIs interactively. I believe that it is
%    much more valuable to those who are interested in studying the dynamic
%    phenomenon via the image techniques.
%
% 5) The multiple file opening is based on javax.swing.JFileChooser, so
%    this mmROI program should work for matlab with jave version 1.3.1 or
%    newer (although it was created on Matlab, v6.5.1, R13SP1). If it dose not
%    work for an older version of matlab, you may download "uigetfiles.dll"
%    (mathworks file exchange ID: 331) into your matlab directory, and
```

```
%    substitute those codes in charge of file opening as stated in the program.
%
% 6) An input dialog box will pop up, asking you to input the total number
%    of ROIs (default value 1), and to select a working image (Figure No. XX)
%    in which you are going to draw your ROIs (default is set to the 1st
%    image). These ROIs will be applied to all images opened. Some users may
%    want to draw different ROIS in each images, you may easily modify the
%    program by yourself (If you need help, you may contact me).
%
% 7) ROI selection is based upon ROIPOLY, therefore, image process toolbox
%    is needed. Use normal button clicks to add vertices to the polygon.
%    Pressing <BACKSPACE> or <DELETE> removes the previously selected vertex.
%    A shift-click, right-click, or double-click adds a final vertex to the
%    selection and then starts the fill; pressing <RETURN> finishes the
%    selection without adding a vertex. For more details, please see HELP ROIPOLY.
%
% 8) ROI coordinates were automatically saved into a text file in your
%    current directory ("roiXY.txt"). This maybe be needed if you want
%    to process further the images with exactly identical ROI polygon, such as
%    poly2mask, etc. If you want to keep it, please find roiXY.txt, and backup
%    before performing any new mmROI. Otherwise, it will be overwriten.
%
% 10) The ROI mean/std values were plotted along the image series. Each ROI
%     has a new plot with three lines that are corresponding to red, green and
%     blue color, respectively.
%
% Shanrong Zhang
% Department of Radiology
% University of Washington
%
% email: zhangs@u.washington.edu
%
% update info (04/15/2005): fix bug so that it is usable on Version R14SP2

% Multiple file selections based upon javax.swing.JFileChooser
% Manipulated by A.Levent Kurtoglu on June 20,2008

import com.mathworks.mwswing.MJFileChooser;
```

```
import com.mathworks.toolbox.images.ImformatsFileFilter;

filechooser=javax.swing.JFileChooser(pwd);
filechooser.setMultiSelectionEnabled(true);
filechooser.setFileSelectionMode(filechooser.FILES_ONLY);

% Parse formats from IMFORMATS
formats=imformats;
nformats=length(formats);
desc=cell(nformats,1);
[desc{:}]=deal(formats.description);
ext=cell(nformats,1);
[ext{:}]=deal(formats.ext);

% Create a filter that includes all extensions
ext_all=cell(0);
for i = 1:nformats
    ext_i=ext{i};
    ext_all(end+1: end+numel(ext_i))=ext_i(:);
end

[ext{end+1,1}]=ext_all;
[desc{end+1,1}]='All image files';

% Make a vector of String arrays
extVector=java.util.Vector(nformats);
for i = 1:nformats+1
    extVector.add(i-1,ext{i})
end

% Push formats into ImformatsFileFilter so instances of
% ImformatsFileFilter will be based on IMFORMATS.
ImformatsFileFilter.initializeFormats(nformats,desc,extVector);

% Create all_images_filter
all_images_filter=...
ImformatsFileFilter(ImformatsFileFilter.ACCEPT_ALL_IMFORMATS);
filechooser.addChoosableFileFilter(all_images_filter);
```

```
% Add one ChoosableFileFilter for each format in IMFORMATS
for i = 1:nformats
    filechooser.addChoosableFileFilter(ImformatsFileFilter(i-1))
end

% Put accept all files at end
accept_all_filter=filechooser.getAcceptAllFileFilter;
filechooser.removeChoosableFileFilter(accept_all_filter);
filechooser.addChoosableFileFilter(accept_all_filter);

% Make default be all_images_filter
filechooser.setFileFilter(all_images_filter);
returnVal=filechooser.showOpenDialog(com.mathworks.mwswing.MJFrame);

if (returnVal == MJFileChooser.APPROVE_OPTION)
    pathname=[char(filechooser.getCurrentDirectory.getPath), ...
              java.io.File.separatorChar];
    selectedfiles=filechooser.getSelectedFiles;
    imNumber=size(selectedfiles);
    for n = 1:imNumber
        filenames(n)=selectedfiles(n).getName;
    end
    filenames=char(filenames);
else
    pathname=pwd;
    filenames=0;
end

if filenames==0;
    return;
end

% If you have downloaded "uigetfiles.m" (witten by me based on java
% interface) in your directory, the above codes can be substituted by only
% one line:
%
% [filenames, pathname] = uigetfiles;
```

```
%
% Alternatively, if you had downloaded "uigetfiles.dll and uigtefiles.m"
% (MEX written in C language), you may substitute the above codes by one line:
%
% [filenames, pathname]=uigetfiles('*.*, 'Please Select image files');
% filenames=char(filenames);  % to converte cell array into char array

for imIndex = 1:imNumber
    im(:,:,:,imIndex)=imread([pathname, filenames(imIndex,:)]);
    figure;
    imHandle=imshow(im(:,:,:,imIndex));
    title(filenames(imIndex,:));
    axis image;
    axis off;
end

prompt={'Total ROI Number:','Perform ROI selection on which image (Figure No. XX):'};
dlg_title='Inputs for mmROI function';
num_lines=1;
def={'1','1'};
inputs=str2num(char(inputdlg(prompt, dlg_title, num_lines, def)));

roiNumber=inputs(1);
workingImage=inputs(2);

% generate a jet colormap according to roiNumber
clrMap=jet(roiNumber);
rndprm=randperm(roiNumber);

hold on;

tfid=fopen('roiXY.txt', 'w+');

for roiIndex = 1:roiNumber
    figure(workingImage);
    [x, y, BW, xi, yi] = roipoly;
    xmingrid = max(x(1), floor(min(xi)));
    xmaxgrid = min(x(2),  ceil(max(xi)));
```

```
    ymingrid = max(y(1), floor(min(yi)));
    ymaxgrid = min(y(2),  ceil(max(yi)));
    xgrid = xmingrid : xmaxgrid;
    ygrid = ymingrid : ymaxgrid;
    [X, Y] = meshgrid(xgrid, ygrid);
    inPolygon = inpolygon(X, Y, xi, yi);
    Xin = X(inPolygon);
    Yin = Y(inPolygon);

    % Save each roi coordinates into a text file "roiXY.txt", so that one
    % can easily use "poly2mask" for further regional image process with
    % exactly identical ROIs created here.
    fprintf(tfid, 'ROI "%s":\n', num2str(roiIndex));
    fprintf(tfid, 'Xi = \t');
    fprintf(tfid, '%10.2f\t', xi);
    fprintf(tfid, '\n');
    fprintf(tfid, 'Yi = \t');
    fprintf(tfid, '%10.2f\t', yi);
    fprintf(tfid, '\n');

    roi.area(:,roiIndex) = polyarea(xi,yi);
    roi.center(:,roiIndex) = [mean(Xin(:)), mean(Yin(:))];

    for imIndex = 1:imNumber
        roi.mean(:,roiIndex,imIndex)=mean(impixel(x,y,im(:,:,:,imIndex),xi,yi));
        roi.std(:,roiIndex,imIndex)=std(impixel(x,y,im(:,:,:,imIndex),xi,yi));
        roi.min(:,roiIndex,imIndex)=min(impixel(x,y,im(:,:,:,imIndex),xi,yi));
        roi.max(:,roiIndex,imIndex)=max(impixel(x,y,im(:,:,:,imIndex),xi,yi));
        roi.median(:,roiIndex,imIndex)=median(impixel(x,y,im(:,:,:,imIndex),xi,yi));

        figure(imIndex);
        hold on;
        plot(xi,yi,'Color',clrMap(rndprm(roiIndex), :),'LineWidth',1);
        text(roi.center(1,roiIndex), roi.center(2,roiIndex), num2str(roiIndex),...
             'Color', clrMap(rndprm(roiIndex), :), 'FontWeight','Bold');
    end
end
```

```
fclose(tfid);

% Save ROIs to a file (optional)
SaveOutput = questdlg('Do you want to save ROI outputs ?');
switch SaveOutput
    case 'Yes'
        [outFilename, outPathname]=uiputfile('*.txt', 'Please select an output file');
        if outFilename == 0
            disp(' Cancel by user !');
        else
            ofid=fopen([outPathname outFilename], 'w+');
            fprintf(ofid, '%-20s\t %-25s\n', 'Date\time = ', datestr(now));
            fprintf(ofid, '\n');
            fprintf(ofid, 'Images used in this ROI process: \n');
            fprintf(ofid, '\t"%s"\n', pathname(1,:));
            fprintf(ofid, '     Index\t\tfilenames \n');
            for imIndex = 1:1:imNumber
                fprintf(ofid, '%10.0f\t\t', imIndex);
                fprintf(ofid, '%-100s\n', filenames(imIndex,:));
            end
        end
    otherwise
        outFilename = 0;
end

warning off MATLAB:colon:operandsNotRealScalar;
imIndex = 1:imNumber;

% Plot of mean/std values along the image series. Each ROI plots as a new
% figure with three lines of red, green and blue, respectively.
for roiIndex=1:roiNumber
    redMean     = reshape(roi.mean(1,roiIndex,:),   [1, imNumber]);
    redStd      = reshape(roi.std(1,roiIndex,:),    [1, imNumber]);
    redMin      = reshape(roi.min(1,roiIndex,:),    [1, imNumber]);
    redMax      = reshape(roi.max(1,roiIndex,:),    [1, imNumber]);
    redMedian   = reshape(roi.median(1,roiIndex,:), [1, imNumber]);

    greenMean   = reshape(roi.mean(2,roiIndex,:),   [1, imNumber]);
```

```
greenStd    = reshape(roi.std(2,roiIndex,:),    [1, imNumber]);
greenMin    = reshape(roi.min(2,roiIndex,:),    [1, imNumber]);
greenMax    = reshape(roi.max(2,roiIndex,:),    [1, imNumber]);
greenMedian = reshape(roi.median(2,roiIndex,:), [1, imNumber]);

blueMean    = reshape(roi.mean(3,roiIndex,:),   [1, imNumber]);
blueStd     = reshape(roi.std(3,roiIndex,:),    [1, imNumber]);
blueMin     = reshape(roi.min(3,roiIndex,:),    [1, imNumber]);
blueMax     = reshape(roi.max(3,roiIndex,:),    [1, imNumber]);
blueMedian  = reshape(roi.median(3,roiIndex,:), [1, imNumber]);

figure;
title(sprintf('MEAN/STD plot of ROI "%s" along the image series', ...
      num2str(roiIndex)));
xlabel('The image Series');
ylabel('Mean/Std of ROI');
hold on;
errorbar(  redMean,   redStd, '-or');
errorbar(greenMean, greenStd, '-sg');
errorbar( blueMean,  blueStd, '-db');

% write ROI statistics into file (optional)
if outFilename ~= 0
    fprintf(ofid, '\n');
    fprintf(ofid, 'ROI "%s" statistic data: \n\n', num2str(roiIndex));

    fprintf(ofid, '%20s\t', 'ROI Area = ');
    fprintf(ofid, '%10.2f\t', roi.area(:,roiIndex));
    fprintf(ofid, '\n');

    fprintf(ofid, '%20s\t', 'ROI center (X,Y) = ');
    fprintf(ofid, '%10.2f\t', roi.center(:,roiIndex));
    fprintf(ofid, '\n\n');

    fprintf(ofid, '%20s\t', 'Image Index = ');
    fprintf(ofid, '%10.0f\t', imIndex);
    fprintf(ofid, '\n\n');
```

```
fprintf(ofid, '%20s\t', 'redMean = ');
fprintf(ofid, '%10.2f\t', redMean);
fprintf(ofid, '\n');

fprintf(ofid, '%20s\t', 'greenMean = ');
fprintf(ofid, '%10.2f\t', greenMean);
fprintf(ofid, '\n');

fprintf(ofid, '%20s\t', 'blueMean = ');
fprintf(ofid, '%10.2f\t', blueMean);
fprintf(ofid, '\n\n');

fprintf(ofid, '%20s\t', 'redStd = ');
fprintf(ofid, '%10.2f\t', redStd);
fprintf(ofid, '\n');

fprintf(ofid, '%20s\t', 'greenStd = ');
fprintf(ofid, '%10.2f\t', greenStd);
fprintf(ofid, '\n');

fprintf(ofid, '%20s\t', 'blueStd = ');
fprintf(ofid, '%10.2f\t', blueStd);
fprintf(ofid, '\n\n');

fprintf(ofid, '%20s\t', 'redMin = ');
fprintf(ofid, '%10.2f\t', redMin);
fprintf(ofid, '\n');

fprintf(ofid, '%20s\t', 'greenMin = ');
fprintf(ofid, '%10.2f\t', greenMin);
fprintf(ofid, '\n');

fprintf(ofid, '%20s\t', 'blueMin = ');
fprintf(ofid, '%10.2f\t', blueMin);
fprintf(ofid, '\n\n');

fprintf(ofid, '%20s\t', 'redMax = ');
fprintf(ofid, '%10.2f\t', redMax);
```

```
        fprintf(ofid, '\n');

        fprintf(ofid, '%20s\t', 'greenMax = ');
        fprintf(ofid, '%10.2f\t', greenMax);
        fprintf(ofid, '\n');

        fprintf(ofid, '%20s\t', 'blueMax = ');
        fprintf(ofid, '%10.2f\t', blueMax);
        fprintf(ofid, '\n\n');

        fprintf(ofid, '%20s\t', 'redMedian = ');
        fprintf(ofid, '%10.2f\t', redMedian);
        fprintf(ofid, '\n');

        fprintf(ofid, '%20s\t', 'greenMedian = ');
        fprintf(ofid, '%10.2f\t', greenMedian);
        fprintf(ofid, '\n');

        fprintf(ofid, '%20s\t', 'blueMedian = ');
        fprintf(ofid, '%10.2f\t', blueMedian);
        fprintf(ofid, '\n\n');
    end
end

if outFilename ~= 0
    disp(sprintf('Done! ROI statistic data were output to "%s",', outFilename));
    disp('ROI coordinates (X,Y) were saved in "roiXY.txt",');
    disp('but the images were not saved yet!');
    fclose(ofid);
else
    disp('Done, ROI coordinates (X,Y) were saved in "roiXY.txt",');
    disp('but their statistics and the images were not saved!');
end
% end of code
```

APPENDIX E. Attenuation Supplement by Field II

```
% manipulated by A.Levent KURTOGLU,May 15,2008
% This part of the Program adds attenuation to the image

set_field ('att',25*100);
set_field ('Freq_att',0.5*100/1e6);
set_field ('att_f0',3e6);
set_field ('use_att',1);
```

REFERENCES

1. Bronzino, J. D., *The Biomedical Engineering HandBook, 2nd Ed*, pp. 1116-1118, CRC Press LLC, 2000, ISBN: 0-8493-0461-X

2. Lindsay, R. B., "The story of acoustics". Journal of Acoustics Society Am., Vol. 39(4), pp. 629-644, 1965.

3. Shung, K. K., "Diagnostic Ultrasound Imaging and Blood Flow Measurements", *Taylor and Francis*, CRC Press, 2006, ISBN: 0-8247-4096-3, 5-25.

4. Moore, J., Zouridakis, G., *Biomedical Technology and Devices Handbook*, pp. 31-33, CRC Press, 2004, ISBN: 0-8493-1140-3.

5. Webster, J. G., *Encyclopedia of Medical Devices and Instrumentation*, Second Edition, Vol. 6, pp. 453-473, Wiley Interscience, 2006, ISBN: 13 978-0-471-26358-6.

6. Stepanishen, P.R., "Transient Radiation from pistons in an infinite baffle", *Journal of Acoustics Society Am.*, Vol. 49, pp. 1629-1638, 1971.

7. Pierce, A. D., *Acoustics: An Introduction to Physical Principles and Applications.* Acoustical Society of America, New York, 1989.

8. Haney M.J., O'brien, W.D., "Temperature Dependance of ultrasonic propagation in bilologic materials, In tissue Characterization with Ultrasound", Greenleaf, pp:15-55, Boca Raton: CRC Press, 1986.

9. Jensen, J. A., *Estimation of Blood Velocities Using Ultrasound: A Signal Processing Approach,* Cambridge University Press, New York, 1996.

10. Available: [visited on November 2008] http://nexradiology.blogspot.com/

11. Jensen, J. A., "A model for the propagation and scattering of ultrasound in tissue", *Journal of Acoustics Am,* Vol. 89, pp. 182-191, 1991.

12. Feng, D. D., *Biomedical Information Technology,* Elsevier, pp. 115-135, 2008.

13. Laxminarayan, S., Michelson, L., "Perspectives in biomedical Supercomputing", *IEEE Eng.Med.Bio.Mag.,* Vol. 7, pp. 12-15, 1988.

14. Gund, T. M., "Supercomputer Applications in Molecular Modeling", *IEEE Eng. Med. Biol. Mag.,* Vol. 7, pp. 21-26, 1988.

15. Lisle, C., Parsons, R., "The Design and Applications of a Recursive Molecular Modeling Framework.", *IEEE Trans. Inform. Technol. Biomed.,* Vol. 4, pp. 159-164, 2000.

16. Gombert, A. K., Nielsen, J., "Mathematical Modelling of Metabolism.", *Curr. Opin. Biotechnol.,* Vol. 11, pp. 180-186, 2000.

17. Rizzi, M., Baltes, M., Theobald, U., Reuss, M., "In vivo analysis of metabolic dynamics in Saccharomyces cerevisiae", *II. mathematical model. Biotechnol. Bioeng.,* Vol. 55, pp. 592-608, 1997.

18. Wang, K. C., Dutton, R. W., Taylor, C. A., "Improving Geometric Model Construction for Blood Flow Modeling.", *IEEE Eng. Med. Biol. Mag.,* Vol. 18, pp. 33-39, 1999.

19. Vinay Kumar, Abul K. Abbas, Nelson Fausto Kumar, V., Abbas, A. K., Nelson, F., "Robbins and Cotran Pathologic Basis of Disease", *Elsevier-Saunders,* St. Louis MO., 2005, ISBN: 0-7216-0187-1.

20. Walter F., *PhD, Medical Physiology: A Cellular And Molecular Approach,* Elsevier/ Saunders, Boron, 2004, ISBN: 1-4160-2328-3.

21. Available: http://www.indexedvisuals.com/scripts/ivstock/pic.asp?id=118-100

22. Available: http://www.bioportfolio.com/indepth/Kidney.html

23. Available:http://galileo.phys.virginia.edu/classes/304/kidney.gif

24. Collins, A. J., CDRG presentation, "CKD and the Public Health Agenda for Chronic Diseases, Figure 12.1", Vol. 2, USRDS, March 2009.

25. Huang, E., Samaniego-Picota, M., McCune, T., Melancon, J.K., Montgomery, R. A., Ugarte, R., Kraus, E., Womer, K., Rabb, H. and Watnick, T., *Polycystic Kidney Disease, Transplantation,* pp. 133-135, 2009.

26. Available: http://www.pkdcure.org/Portals/0/files/images/12523.jpg

27. Available: http://www.pkdcure.org/

28. Ecder, T., Schrier, R. W., "Cardiovascular Abnormalities in Autosomal Dominant Polycystic Kidney Disease", *Nat. Rev. Nephrol,* Vol. 5, pp. 221-228, 2009.

29. Rizk, D., Jurkovits, C., Veledar, E., Bagby, S., Baumgarten, D. A., Rahbari-Oskoui, F., Steinman, T., Chapman, A. B., "Quality of Life in Autosomal Dominant Polycystic Kidney disease Patients not yet on Dialysis", *Clinical Journal of American Society of Nephrology*, Vol. 4, pp. 560-566, 2009.

30. Harris, P. C., "2008 Homer W. Smith Award: Insights into the Pathogenesis of Polycystic Kidney Disease from Gene Discovery", *Journal of American Society of Nephrology,* Vol. 20, pp. 1188-1198, 2009.

31. Huang, E., Samaniego-Picota, M., McCune, T., Melancon, J.K., Montgomery, R. A., Ugarte, R., Kraus, E.,Womer, K., Rabb, H. andWatnick, T., *DNA Testing for Live Kidney Donors at Risk for Autosomal Dominant Polycystic Kidney Disease, Transplantation,* Vol. 87, pp. 135-137, 2009.

32. ITU-R Recommendation BT.601-5, Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide-Screen 16:9 Aspect Ratios, International Telecommunication Union-Telecommunications Standardization Sector (ITU-T), 1995.

33. Michael, G., Strintzis, A., "Review of Compression Methods for Medical Images in PACS", *International Journal of Medical Informatics*, Vol. 52, pp. 159-165, 1998.

34. Gonzalez, R. C., Woods, R. E., *Digital Image Processing,2nd ed.,* Pearson-Prentice Hall, 2002.

35. Persons, K. R., Pallison, P. P., Manduca, A., Charboneau, W. J., "Ultrasound grayscale image compression with JPEG andWavelet techniques", *Journal of Digital Imaging*, Vol. 13, pp. 25-32, 2000.

36. Bradley J., Erickson M.D, "Irreversible Compression of Medical Images", *Journal of Digital Imaging*, Vol.15/1, pp. 5-14, 2002.

37. Sayre J., Aberle D., Boechat I., "The Effect of Data Compression on Diagnostic Accuracy in Digital Hand and Chest Radiography", *Proceedings of SPIE,* Vol. 1653, pp. 232-240, 1992.

38. Ratnakar, V., Livny, M., "RD-OPT:An Efficient Algorithm For Optimizing DCT Quantization Tables", *Proceedings of Data Compression Conference*, pp. 332-341, 1995.

39. Delgorge, C., Rosenberger, C., Vieyres, P., Poisson, G., JPEG 2000, "An Adapted Compression Method for Ultrasound Images? A comparative study ", SCI, Orlando, 2002.

40. Fidlera, A., Skaleric, U., Likar, B., "The Impact of Image Information on Compressibility and Degradation in Medical Image Compression", *Med.Phys,* Vol. 33/8, pp. 2832-2838, 2006.

41. ISO/IEC JTC 1/SC 29. Information Technology-JPEG 2000 Image Coding System-Part 1: Core Coding System, ISO/IEC 15444-1, 2000.

42. Dansereau, R., Kinsner, W., "Perceptual Image Compression Through Fractal Surface Interpolation and Wavelet Compression", WESCANEX 97: *Communications, Power and Computing, Conference Proceedings, IEEE,* pp. 94-99, 1997.

43. Vetterli, M., Kovacevic, J., *Wavelets and Subband Coding,* Prentice Hall, 1995.

44. Oppenheim, A. V., Schafer, R. W., Buck, J. R., *Discrete-Time Signal Processing,* Prentice Hall, 1999.

45. Woods, J. W., O'Neil, S. D., "Subband Coding of Images", *IEEE Trans. Acoustics, Speech and Signal Processing.,* Vol. 34/5, pp. 1278-1288, 1986.

46. Rao, K.R., Yip, P., *Discrete Cosine Transform:Algorithms, Advantages, and Applications,* Academic Press, Boston, 1990.

47. Mallat, S. G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", *IEEE Trans. Pattern Anal. Mach. Intel.,* Vol. 11/7, pp. 674-693, 1989.

48. Bary G., Haskell, A., Puri, A., Netravali, N., *Digital Video: An Introduction to MPEG-2,* Chapman and Hall, New York, 1997.

49. Lloyd, S. P., "Least Square Quantization in PCM", *IEEE Trans. Inform. Theory,* Vol. 28/2, pp. 127-135, 1982.

50. Max, J., "Quantizing for Minimum Distortion", *IRE Trans. Inform. Theory,* Vol. 6/1, pp. 7-12, 1960.

51. Netravali, A. N., Haskell, B. G., *Digital Pictures,Representation, Compression and Standards,* 2nd ed. Plenum Press, New York, 1995.

52. Jayant, N. S., Noll, P., *Digital Coding of Waveform: Principles and Applications to Speech and Video,* Prentice Hall, 1984.

53. Livny, M., Ratnakar, V., "Quality Controlled Compression of Sets of Images", *International Workshop on Multimedia Database Management Systems IEEE,* pp. 109-114, 1996.

54. Mitchell, J. L., W. Pennebaker, B., *JPEG-Still Image Data Compression Standart, 8th Ed.,* Kluwer Academic Publishers, 2004.

55. Chaddha, N., Agrawal, A., Gupta, A., Meng, T. H.Y., "Variable Compression Using JPEG", *ICMS,* pp. 562-569, 1994.

56. Daubechies, I., "Orthonormal Vases of Compactly Supported Wavelets", *Communication on Pure and Applied Mathematics,* Vol. 41, pp. 909-996, 1988.

57. Brodal, G. S., Frogioni, D., Marchetti-Spaccamela, A., "Algorithm Engineering", *5th International Workshop,* WAE 2001 Aarhus, Vol. 2141, pp. 13-25, 2001.

58. Rao, K.R., Huh, Y., "JPEG 2000, Proceedings of International Symposium on Intelligent Multimedia", *Video and Speech Processing,* pp.20-23, Hong Kong, 2002.

59. ISO/IEC JTC 1/SC 29. Information Technology–Lossless and Near-Lossless Compression of Continuous-Tone Still Images: Baseline. ISO/IEC 14495-1:1999, 1999

60. Weinberger, M., Seroussi, G., Sapiro, G., "The LOCO lossless image compression algorithm: Principles and standardization into JPEG-LS.", *IEEE Trans. Image Proc.,* Vol. 9/8, pp. 1309-1324, 2000.

61. Available: http://www.jpeg.org/jpeg/jpegls.html

62. Goodall, W. M., "Television by Pulse Coding Modulation", *Bell Systems Technical Journal,* Vol. 28, pp. 33-49, 1951.

63. Cutler, C. C., *Differential Quantization of Communication Signals,* US Patent No. 2 605 361, 1952.

64. Harrison, C. W., "Experiment with Linear Prediction in Television", *Bell Systems Technical Journal,* Vol. 29, pp. 764-783, 1952.

65. Huffman, D.A., "Method for the Construction of Minimum Redundancy Codes", *Proceedings of the IRE,* Vol. 40/9, pp. 1098-1101, 1952.

66. Sayood, K., *Introduction to Data Compression, Academic Press, 2nd ed.,* 2000.

67. Andleigh, P. K., Kiran, T., *Multimedia System Design,* Prentice Hall, Englewood Cliffs, pp. 52-123, New Jersey, 1996.

68. TIFF Revision 6.0 Final ,June 3, 1992, Avalable:http://partners.adobe.com/ public/developer/en/tiff/TIFF6.pdf

69. Ziv, J., Lempel, A., "Compression of Individual Sequences Via Variable-Rate Coding", *IEEE Transactions on Information Theory,* Vol. 24/5, pp. 530-536, 1978.

70. Available: http://cool.conservation-us.org/bytopic/imaging/std/tiff5.html

71. Abbott, J. G., Thur, F.L., "Acoustic speckle: Theory and Experimental Analysis", *Ultrason. Imag.,* Vol. 1, pp. 303-324, 1979.

72. Guo, H., Odegard, J.E., Lang, M., Gopinath, R. A., Selesnick, I. W., Burrus, C.S., "Wavelet Based Speckle Reduction with Application to SAR Based ATD/R", *First International Conf. on Image Processing,* vol. 1, pp. 75-79, 1994.

73. Maini, R., Sohal, J. S., "Performance Evaluation of Prewitt Edge Detector for Noisy Images", *GVIP Journal,* Vol. 6/3, pp. 39-46, 2006.

74. Gonzalez, R. C., Woods, R. E., *Digital Image Processing,* Addison-Wesley, 1992.

75. Jain, A.K., *Fundamentals of digital image processing, Englewood cliffs,* NJ Prentice-Hall, 1989.

76. Young, I. T., Gerbrands, J. J., van Vliet, L. J., "Image Processing Fundamentals", *Statistics, Signal to Noise Ratio,* 2001.

77. Nowak, R. D., "Wavelet Based Rician Noise Removal", *IEEE Transaction on image processing,* Vol. 8, No. 10, pp. 1480, 1999.

78. Gagnon, L., Smaili, F. D., "Speckle Noise Reduction of Airborne SAR Images with Symmetric Daubechies Wavelets", *SPIE Proc,* No: 2759, pp. 1424, 1996.

79. Gagnon, L., "Wavelet Filtering of Speckle Noise-Some Numerical Result", *Proceeding of the Conference Vision Interface,* Trois-Riveres, 1999.

80. Gray, R. M., Olshen, R. A., Ikeda, D., Cosman, P. C., Perlmutter, S., Nash, C., Perlmutter, K., "Evaluating Quality and Utility in Digital Mammography", *International Conference on Image Processing (ICIP'95)*, Vol. 2, pp. 2005, 1995.

81. Ke, L., Marcellin, M. W., "Near-Lossless Image Compression: Minimum-Entropy, Constrained-Error DPCM", *International Conference on Image Processing*, vol. 1, pp. 298, 1995.

82. Bankman, I. N., PhD, *Handbook of Medical Imaging*, New York, Academic Press, 2000.

83. Rigolin, V. H., Robiolio, P. A., Spero, L. A., Harrawood, B. P., Morris, K. G., Fortin, D. F., Baker, W. A., Bashore, T. M., Cusma, J. T., "Compression of Digital Coronary Angiograms Does Not Affect Visual or Quantitative Assessment of Coronary Artery Stenosis Severity", *Am. J. of Card.,* Vol. 78, pp. 131-135, 1996.

84. Eckstein, M. P., Bartroff, J. L., Abbey, C. K., Whiting, J. S., Bochud, F. O., "Automated Computer Evaluation and Optimization of Image Compression of x-ray Coronary Angiograms for Signal Known Exactly Detection Tasks", *Optical Society of America,* vol. 11, No. 5, 2003.

85. Goldberg, M., Panchanathan, S., Wang, L. A., "Comparison of Lossy Techniques for Digitized Radiographic Images, Medical Imaging IV: Image Capture, Formatting and Display", *Proc. SPIE,* pp. 269-281, 1993.

86. Lehmann, E. L ,*Casella George, Theory of point estimation Springer Texts in Statistics, 2nd ed,* Springer-Verlag, New York, 1998.

87. Bushberg, J. T., *The Essential Physics of Medical Imaging, 2nd ed.,* Lippincott Williams and Wilkins, Philadelphia, pp. 280, 2006.

88. Eskicioglu, A. M., "Application of Multidimensional Quality Measures to Reconstructed Medical Images", *Opt.Eng,* Vol. 35 (3), pp. 778-785, 1996.

89. Jensen, J. A., "Field: A Program for Simulating Ultrasound Images", *Medical and Biological Engineering and Computing,* vol. 34, Supplement 1, pp. 351-353, 1996.

90. Jensen, J. A., Svendsen, N. B., "Calculation of Pressure Fields from Arbitrarly Shaped, Apodized, and Excited Ultrasound Transducers", *IEEE Trans. Ultrason.,* Ferroelec., Freq. Contr., Vol. 39, pp. 262-267, 1992.

91. Jensen, J. A., "A model for the Propagation and Scattering of Ultrasound in Tissue", *Journal of Acoust.Soc. Am.,* Vol. 89, pp. 182-191, 1991.

92. Tupholme, G. E., "Generation of Acoustic Pulses by Baffled Plane Pistons", *Mathematika,* Vol. 16, pp. 2009-224, 1969.

93. Stephanishen, P.R., "Transient Radiation from Pistons in an Infinite Planar Baffle", *Journal of Acoust.Soc. Am.,* Vol. 49, pp. 1629-1638, 1971.

94. *http://www.medical.siemens.com/siemens/en$_G$LOBAL/rg_marcom_FBAs/files/brochures/DICOM/us/sonoline/DCS_Elegra_70.pdf*

95. Available: http://www.irfanview.de