Institute of Chemical Technology, Prague
Department of Computing and Control Engineering

BIOMEDICAL SIGNAL AND IMAGE PROCESSING

Ph.D. Thesis

Victor Musoko

Chemical and Process Engineering
Technical Cybernetics

Prague **July 2005**

Declaration

The research work described in this dissertation was done by the author from September 2002 till July 2005 at the Department of Computing and Control Engineering, Institute of Chemical Technology, Prague. The author do hereby declare that the contents of the thesis except where indicated are entirely original and are a result of the research work he has carried out.

Victor Musoko

Acknowledgements

I would like to take this opportunity to thank all those who have contributed to this thesis. First and foremost my big thanks go to my supervisor Prof. Aleš Procházka, who gave me the opportunity to pursue the research. I appreciated his guidance, encouragement and the helpful discussions. The work presented here would certainly not have been accomplished without his influence and support. From my supervisor, I have learned a great deal about problem solving and presentation of the results of my research work.
Last but not least, I would also like to express my deep thanks to Ing. Aleš Pavelka, for his tireless help at the time it was needed most and Assoc. Prof. Jaromír Kukal, for introducing me to the world of programming in C/C++ and for his important discussions. Thanks in general to all my colleagues in the Department of Computing and Control Engineering.

Abstract

The main goal of the thesis is to show the de-noising algorithms based upon the discrete wavelet transform (DWT) that can be applied successfully to enhance noisy multidimensional magnetic resonance (MR) data sets i.e two-dimensional (2-D) image slices and three-dimensional (3-D) image volumes. Noise removal or de-noising is an important task in image processing used to recover a signal that has been corrupted by noise. Random noise that is present in MR images is generated by electronic components in the instrumentation. The thesis present both 2-D image decomposition, thresholding and reconstruction and the 3-D de-noising of MR image volumes using the DWT as a new approach which can be used in the processing of biomedical images. A novel use of the complex wavelet transform for the study and application in the de-noising of MR images is presented in the main part of the thesis. Segmentation of image textures using a watershed transform and a wavelet based feature extraction for classification of image textures by a competitive neural network will be shown.

Further topic of interest to be presented in the thesis is the visualization of 2-D MR slices and 3-D image volumes using some MATLAB functions. This makes it possible to visualize images without the need of special glasses especially for 3-D image volumes or using special expensive software programs which will need some bit of expertise. The application of the proposed algorithms is mainly in the area of magnetic resonance imaging (MRI) as an imaging technique used primarily in medical field to produce high quality images of the soft tissues of the human body. An insight to the visualization of MRI data sets i.e. 2-D image slices or 3-D image volumes is of paramount importance to the medical doctors.

The thesis presents the theory of the fundamental mathematical tools (discrete Fourier transform (DFT) and DWT) that are used for the analysis and processing of biomedical images. DWT plays an increasingly important role in the de-noising of MR images. 3-D digital image processing, and in particular 3-D DWT, is a rapidly developing research area with applications in many scientific fields such as biomedicine, seismology, remote sensing, material science, etc. The 3-D DWT algorithms are implemented as an extension of the existing 2-D algorithms. The performance of the de-noising algorithms are quantitatively assessed using different criteria namely the mean square error (MSE), peak signal-to-noise ratio (PSNR) and the visual appearance. The results are discussed in accordance to the type of noise and wavelets implemented. The properties of wavelets make them special

in that they have a good time and frequency localization which make them ideal for the processing of non-stationary signals like the biomedical signals (EEG, ECG,..) and images (MR). The traditional Fourier transform only provides the spectral information of a signal and thus it is not suitable for the analysis of non-stationary signals.

A novel complex wavelet transform (CWT) which was introduced by Dr. Nick Kingsbury of Cambridge University is analyzed and implemented in the main part of the thesis. The description of the dual tree implementation of CWT is followed by its analysis and discussions devoted to its advantages over the classical wavelet transform. This enhanced transform is then applied to MRI data analysis. Experimental results show that complex wavelet de-noising algorithm can powerfully enhance the PSNR in noisy MRI data sets.

The further part of the thesis devoted to the description of basic principles of a watershed transform for segmentation of MR images. After its verification for simulated textures it is used for segmentation of a human knee MR image. The anatomical regions of the knee which includes the muscle, bone and tissue can be easily distinguished by this algorithm. Segmentation is an important field in medical applications and can be used for disease diagnosis e.g. detecting brain tumor cells.

Texture analysis of artificial textures based on image wavelet decomposition is considered as a pre-processing method for the classification of the textures. The wavelet features are obtained by using the mean or the standard deviations of the wavelet coefficients. For the classification of the textures these feature vectors form the inputs to a competitive neural network. The work also presents own algorithms for class boundaries evaluation. Texture analysis is used in a variety of applications, including remote sensing, satellite imaging, medical image processing, etc.

Finally, I conclude and give suggestions for future research work. The thesis also gives a review of the de-noising and visualization of biomedical images on the web using the Matlab Web Server (MWS).

Keywords

Time-Frequency and Time-Scale Signal Analysis, Discrete Wavelet Transform, Complex Wavelet Transform, Image De-noising, Biomedical Image Processing, Watershed Algorithm, Segmentation, Feature Extraction, Image Visualization

Abstrakt

Cílem disertační práce je návrh a analýza algoritmů založených na aplikaci diskrétní wavelet transformace (DWT) pro potlačení rušivých složek a zvýraznění vícerozměrných souborů dat. Aplikační část práce je přitom věnovaná obrazům magnetické resonance (MR) a jejich zpracování v případě dvourozměrné (2-D) vrstvy a třírozměrného (3-D) obrazu. Potlačování rušivých složek je důležitou úlohou zpracování obrazů pro oddělení signálu od šumu. Šum obrazů magnetické resonance se přitom skládá z náhodných signálů generovaných elektronickými komponentami systému. Práce presentuje jak dvourozměrnou dekompozici, prahování a následující rekonstrukci obrazů tak i potlačování rušivých složek ve třírozměrném prostoru s využitím DWT jako nové metody, která muže být aplikována na zpracování biomedicínských obrazů. Nová uplatnění komplexní wavelet transformace implementující duální strom (DT CWT) pro studie a aplikace na potlačování rušivých složek obrazů MR je prezentována v hlavní části práce. Segmentace obrazových textur pomocí rozvoďové (watershed) transformace a jejich klasifikace pomocí vzorů získaných z wavelet transformace tvoří další část práce. Vzory jsou přitom využity jako vstupy do samoorganizující se neuronovou síť.

Dalším tématem práce je vizualizace 2-D vrstev a 3-D obrazů pomoci systému MATLAB. Tato cesta umožňuje zobrazit obraz bez používání speciálních technických pomůcek nutných pro vizualizaci 3-D obrazů nebo používání speciálních a často velice složitých komerčních programů. Aplikace navržených algoritmů je zejména v oblasti zobrazování výsledků získaných pomocí magnetické rezonance (MRI) jako moderní vyšetřovací metody používané hlavně v lékařské praxi pro získávání vysoce kvalitních obrazů vnitřních orgánů lidského těla. Vizualizace sad MRI dat je přitom důležitá zejména pro lékařskou diagnostiku.

Práce prezentuje v teoretické části základní matematické metody analýzy a zpracování biomedicínských obrazů zahrnující diskrétní Fourierovu transformaci (DFT) a zejména diskrétní wavelet transformaci (DWT), která se stále významněji uplatňuje při analýze a potlačování rušivých složek signálů. Práce přitom zahrnuje i třírozměrné zpracování obrazů pomocí DWT (3-D DWT), které je z algoritmického hlediska zobecněním metod dvourozměrných a tvoří výzkumnou oblast, který má aplikace v různých vědních oborech včetně biomedicíny, seismologie, analýzy životního prostředí ap. Výsledky numerických experimentů jsou dále hodnoceny z hlediska kvality obrazů na základě vyhodnocování středních kvadratických chyb (MSE) a maximálních hodnot poměrů signál - šum (PSNR)

s ohledem na typ šumu a implementované wavelet funkce. Při rozborech jsou přitom využité vlastnosti wavelet funkcí zahrnující zejména jejich dobrou časovou a frekvenční lokalizaci. Tato vlastnost je zejména důležitá pro zpracování nestacionárních signálů, které se vyskytují v biomedicíně a zahrnují EEG a EKG signály a dále biomedicínské obrazy. V práci je ukázáno i porovnání s klasickou Fourierovou transformací, která poskytuje jen spektrální informace o celém signálu a neumožňuje časovou lokalizaci nestacionárních komponent pozorovaných dat.

Komplexní wavelet transformace implementující duální strom jako nová transformace, kterou navrhl Dr. Nick Kingsbury z Cambridge university, je analyzována a implementována v hlavní části práce. Práce popisuje teorii této transformace s následující analýzou a diskusí jejích výhod ve srovnání s klasickou WT. Tato rozšířená transformace je dále aplikována na analýzu a vyhodnocování MRI dat. Experimentální výsledky ukazují že potlačování rušivých složek signálů pomocí CWT metody výrazně zvyšuje hodnoty PSNR reálných MRI souborů.

Další část práce je věnována popisu základních principů rozvoďové transformace použité pro segmentaci MR obrazů. Po jejím ověření na simulovaných datech jsou navržené algoritmy následně užity pro segmentaci MR obrazu kolena. Anatomické oblasti kolena, které zahrnují svaly, kosti a tkáně jsou snadno touto transformací odlišeny. Segmentace je přitom velmi důležitá i v dalších biomedicínských aplikacích a může být používána k diagnóze nemocí například při detekování mozkových nádorů.

Analýza umělých textur pomocí wavelet dekompozice obrazu je v práce představena jako metoda pro předzpracování obrazů z hlediska jejich následné klasifikace. Vlastnosti obrazových segmentů jsou přitom získány ze středních hodnot nebo směrodatných odchylek wavelet koeficientů. Tyto vzory tvoří sloupcové vektory pro vstup do samoorganizující neuronové sítě z hlediska jejich klasifikace. Navržené algoritmy zahrnují rovněž výpočet hranic jednotlivých tříd. Analýza textur nachází přitom uplatnění v různých aplikacích, které zahrnují dálkové snímání, družicová pozorování, zpracování obrazů a další.

V závěru práce jsou formulovány návrhy pro další výzkumnou práci v uvedené oblasti. Disertační práce poskytuje navíc i popis užití Matlab Web Server (MWS) pro vzdálené zpracování a vizualizaci biomedicínských obrazů přes webové rozhraní.

Klíčová slova

Diskrétní wavelet transformace - dekompozice a rekonstrukce obrazů - komplexní wavelet transformace - potlačování rušivých složek obrazů - zpracování biomedicínských dat - segmentace obrazů - specifikace vlastností - klasifikace - umělé neuronové sítě - vizualizace

Contents

List of Figures

List of Tables

List of the Used Abbreviations

1-D	one-dimensional
2-D	two-dimensional
3-D	three-dimensional
CWT	Continuous Wavelet Transform
DFT	Discrete Fourier Transform
DICOM	Digital Imaging and Communications in Medicine
DSP	Digital Signal Processing
DT CWT	Dual Tree Complex Wavelet Transform
DWT	Discrete Wavelet Transform
FIR	Finite Impulse Response
FT	Fourier Transform
IDFT	Inverse Discrete Fourier Transform
MAD	Median Absolute Deviation
MAE	Mean of Absolute Error
MR	Magnetic Resonance
MRI	Magnetic Resonance Imaging
MSE	Mean Square Error
MWS	Matlab Web Server
PET	Positron Emission Tomography
PSNR	Peak Signal to Noise Ratio
STFT	Short Time Fourier Transform
WT	Wavelet Transform

1
Introduction

Digital signal processing (DSP) describes the science that tries to analyze, generate and manipulate measured real world signals with the help of a digital computer. These signals can be anything that is a collection of numbers, or measurements and the most commonly used signals include images, audio (such as digitally recorded speech and music) and medical and seismic data. In most digital signal processing applications, the frequency content of the signal is very important. The Fourier transform (FT) is probably the most popular transform used to obtain the frequency spectrum of a signal.

Noise removal or de-noising is an important task in image processing. Image enhancement is a collection of techniques that improve the quality of the given image, that is making certain features of the image easier to see or reducing the noise. In general, the results of the noise removal have a strong influence on the quality of the image processing techniques.

Noise generated by electronic components in instrumentation is a common type of random signal that is present in much biomedical data even though contemporary electronic design minimizes this noise. Often those components of a signal which are not understood are classified as *noise*. The ultimate basis for deciding what constitutes noise should be derived from considerations about the experimental or clinical measurements and the source of a a signal [6]. Ideally when a priori knowledge for judging whether certain components of a signal represent the desired measurement or not is known then the signal processing method is chosen to enhance the desired signal and reduce undesired signal components. In some cases this information may not be known and it may be necessary to examine the results of the signal processing steps to assess whether the output signal exhibits some apparent separation into desired and noise components.

The field of imaging provides many examples of both biomedical images and biomedical image processing. Magnetic resonance imaging (MRI) is excellent for showing abnormalities of the brain such as: stroke, hemorrhage, tumor, multiple sclerosis or lesions. In the MRI basic signals are currents induced in a coil caused by the movement of molecular dipoles as the molecules resume a condition of random orientation after having been aligned by the imposed magnetic field. Signal processing is required to detect and decode them, which is done in terms of the spatial locations of the dipoles (which is related to

the type of tissue in which they are located). Much of the associated signal processing is based on Fourier transform. Since MRI utilizes two-dimensional Fourier transforms the basic concepts are the same.

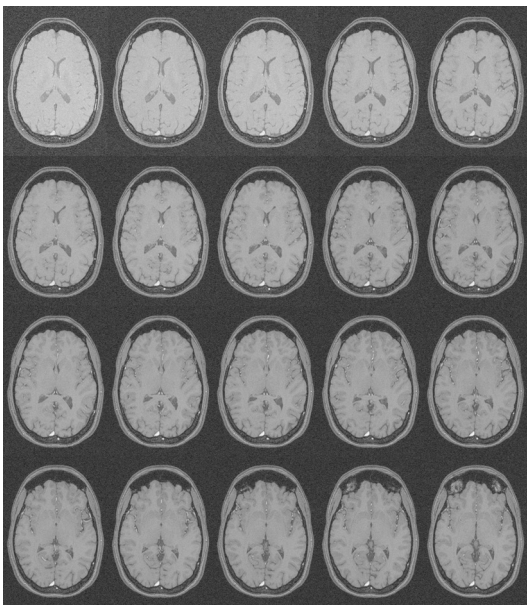


FIGURE 1.1. Sequence of MR image slices

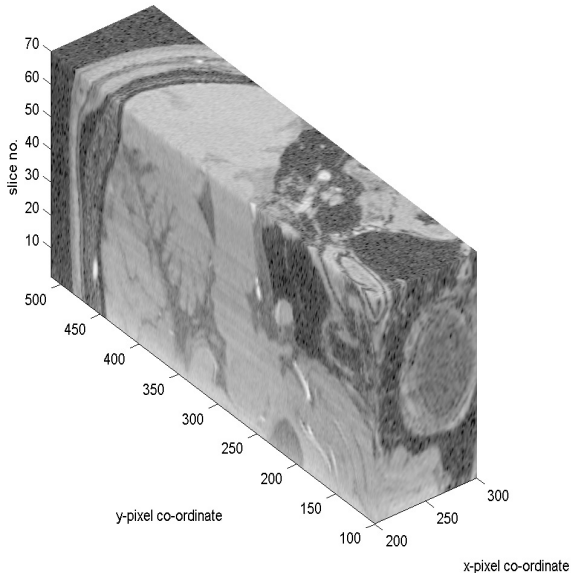


FIGURE 1.2. 3-D model visualization

In addition, once an image is constructed it is often desirable to process it to enhance the visibility of certain features such as the edges of a tumor. Noise in MR images consists of random signals that do not come from the tissues but from other sources in the machine and environment that do not contribute to the tissue differentiation. The noise of an image gives it a grainy appearance and mainly the noise is evenly spread and more uniform. There are many advanced methods of image processing involving techniques some of which are going to be encountered in the Chapter 3. Image filters are designed for noise reduction in the MR images and better edge definition. Indeed, even though other imaging modalities such as positron emission tomography (PET), ultrasound, and X-ray utilize different physical principles for acquisition of the image, the signal processing methods for enhancing images are similar.

Image processing can be defined as the manipulation of an image for the purpose of either extracting information from the image or producing an alternative representation of the image. There are numerous specific motivations for image processing but many fall into the following categories: (i) to remove unwanted signal components that are corrupting the image and (ii) to extract information by rendering it in a more obvious or more useful form.

1.1 Objectives of the Thesis

The main goal of the thesis is to show the de-noising algorithms based upon the discrete wavelet transform (DWT) that can be applied successfully to enhance noisy multidimensional magnetic resonance (MR) data sets i.e two-dimensional (2-D) image slices and three-dimensional (3-D) image volumes. Segmentation and classification of image textures are considered as new methods which might find their use in the classification of MR images. The work and results presented in the thesis have been also published in [71, 58, 61, 59, 57, 60].

The thesis proposes different methods for processing of biomedical image data: the traditional Fourier transform and the wavelet transform. Recently a dual tree complex wavelet transform (DT CWT) was developed and has added advantages over classical methods, these include shift invariance and improved directionality. DT CWT was introduced by Dr. Nick Kingsbury of Cambridge University and is now widely used by a sizeable research groups at Brooklyn University [75] and Rice University [27, 28].

In the thesis, wavelet transform is used for multiscale signal analysis. The de-noising algorithms apply a chosen wavelet on the wavelet decomposition and for the reconstruction of MRI images. DWT reduces the noise effectively, preserving the edge details of the image. Examples are provided to show the de-noising results and the experimental results of the high signal-to-noise rate could be obtained to make a comparison of the different wavelets used. Applications of the DWT [90] in the medical imaging field include noise reduction, image enhancement, and segmentation, image reconstruction. Experiments are done on 2-D and 3-D MRI data sets. Each part starts with algorithms for the 2-D case (image slices) and then continues with generalizing the algorithms to handle the 3-D case (image volumes). 3-D DWT algorithm is implemented as an extension of the existing two-dimensional (2-D) algorithms.

An efficient and accurate watershed algorithm was developed by Vincent and Soille [94] who used an immersion based approach to calculate the watershed lines. The method is tested on the segmentation of simulated and MR image textures. Texture analysis is used in a variety of applications, including satellite imaging and medical image processing. Šonka [95] stresses that texture is scale dependent, therefore a multi-scale or multiresolution analysis of an image is required if texture is going to be analysed. We have proposed the use of image wavelet decomposition [21, 73], using wavelet coefficients at selected levels to describe image features. Classification of image segments into a given number of classes using segment features is done by using a competitive neural network.

1.2 Organization of the Thesis

This thesis is organized into eight chapters which can be summarized as follows:

- **Chapter 1:** A review of the problems to be addressed in this thesis and define the goals of the work. Finally we present the organization of the thesis.
- **Chapter 2:** A brief introduction to the technology behind the magnetic resonance imaging process in producing 2-D images (MRI slices) and the subsequent visualization and presentation of three-dimensional 3-D models.
- **Chapter 3:** The mathematical background used for the image processing. Theory of mathematical tools used for image processing is explained. Time-scale and time-frequency signal analysis methods which include the discrete Fourier transform (DFT), discrete wavelet transform.
- **Chapter 4:** Applications of discrete wavelet transform and a presentation of the experimental results of the de-noising algorithms for the noisy 2-D and 3-D data sets.
- **Chapter 5:** An insight to the theory of dual tree complex wavelet transform. Application of the resulting wavelet algorithm in the analysis and de-noising of magnetic resonance biomedical images.
- **Chapter 6:** The principle of the watershed algorithm. A brief review of image segmentation and feature extraction. The use of discrete wavelet transform in multiresolution study of texture.
- **Chapter 7:** Pattern recognition and classification by using competitive neural networks.
- **Chapter 8:** Presents the general conclusions of the thesis and propose possible improvements and directions of future research work.

Finally in the appendices, Appendix B presents a description of image coding showing how different types of images are displayed. Appendix C introduces the implementation of the Matlab Web Server (MLWS) in remote data processing. Appendix D provides all the MATLAB programs (scripts) used in the thesis.

2
Biomedical Image Visualization

Progress in engineering and medicine demands new ways for visualization of large complex data sets. The first step in biomedical image visualization is to obtain the MRI data. In the thesis I have used MRI scan data taken at the Královské Vinohrady Hospital where we have a collaboration work with some of the staff there. To test the visualizing programs MRI data set of the brain has been used. The data is usually acquired in a file format called DICOM. We will first explain a little bit of additional information about the MR technology and the DICOM standard.

2.1 Magnetic Resonance Imaging

Magnetic resonance imaging (MRI) scan is an imaging technique used primarily in medical field to produce high quality images of the soft tissues of the human body. Using brain images acquired by MRI often allows physicians and engineers to analyze the brain without the need for invasive surgery. Other types of imaging modalities which exist include ultrasound imaging, X-ray imaging, computed tomography (CT).

MRI combines a powerful magnet with an advanced computer system and radio waves to produce accurate, detailed pictures of organs and tissues in order to diagnose a variety of medical conditions. There are two types of MRI exams namely the *high-field* MRI and *low-field* open MRI. The difference is in that high-field MRI produces a highest quality image in the shortest time allowing a most accurate diagnosis to be made. Since MRI can give high quality clear pictures of soft-tissue structures near and around bones, it is the most sensitive exam for brain, spinal and joint problems. MRI is widely used to diagnose sports related injuries, especially those affecting the knee. The images allow the physician to see even very small tears and injuries to ligaments and muscles.

MR Technology

Magnetic resonance imaging (MRI) is based on the absorption and emission of energy in the radio frequency range of the electromagnetic spectrum. Radio waves are directed at protons, the nuclei of hydrogen atoms, in a strong magnetic field. The protons are first excited and then relaxed emitting radio signals, which can be computer-processed to form an image. In the body, protons are most abundant in the hydrogen atoms of water so that

an MRI image shows differences in the water content and distribution in various body tissues. Even different types of tissue within the same organ, such as the gray and white matter of the brain, can be easily distinguished.

The DICOM Standard

Digital Imaging and Communications in Medicine (DICOM) Services offers an interface for transmitting medical images and information in the DICOM industrial standard. As a result, most medical image files (CT, MRI, PET, and Nuclear Medicine) are now received in DICOM file format.

```
info = dicominfo('MRI-12-Brain.dcm')

info =

                        Filename: 'MRI-12-Brain.dcm'
                    FileModDate: '10-Dec-2003 12:26:08'
                        FileSize: 222096
                          Format: 'DICOM'
                   FormatVersion: 3
                           Width: 326
                          Height: 326
                        BitDepth: 12
                       ColorType: 'grayscale'
                                .
                                .
                        Modality: 'MR'
                    Manufacturer: 'Elscint'
                InstitutionName: 'FNKV Praha 10  2T Prestige'
                                .
                                .
                    PatientsName: [1x1 struct]
                       PatientID: '760801/3589'
                                .
                                .
            MRAcquisitionType: '2D'
                 SequenceName: 'FSE'
               SliceThickness: 3
                ImagedNucleus: 'H'
                                .
                                .
```

FIGURE 2.1. DICOM information

A single DICOM file contains both a header (which stores information about the patient's name, the type of scan - the modality used to create the data, image dimensions,

etc), as well as the image data (which can contain information in three dimensions). The dicomread MATLAB function can process the metadata fields defined by the DICOM specification (Fig. 2.1).

2.2 Two-dimensional Medical Imaging Visualization

MRI acquisition of MRI images helps the biomedical engineers to fully analyze different aspects of the brain thereby reducing the need for surgery. With appropriate image analysis techniques, a biomedical engineer can use one small set of MR images and manipulate them to analyze some interesting facets of the brain. To load MR images of the brain into MATLAB and perform the necessary image analysis specifically the task will require us to carry out the following steps:

- Loading the MRI data set file.
- Displaying a cross sectional view of all MR image frames in one figure (Fig. 2.2).

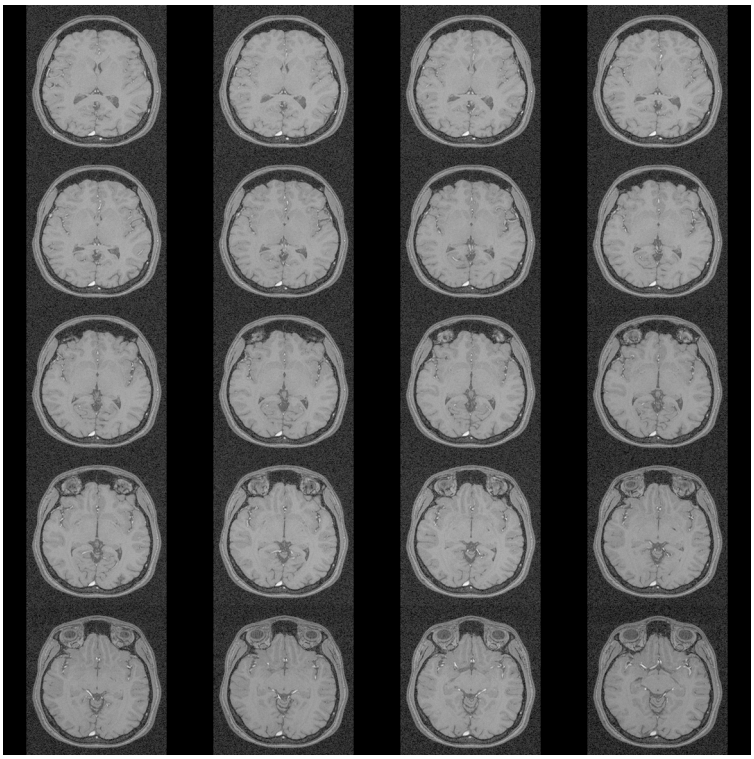


FIGURE 2.2. A sequence of MR image slices

- Isolating a frame of interest (e.g. slice no. 40) and display it as an individual figure. 2-D representation of the slice is done using the `imshow` function (Fig. 2.3).

FIGURE 2.3. MR image slice

The images are called slices because they look like you have cut the brain in many slices to see what's inside. Using MATLAB `movie` function it's possible to show all frames of MRI data as a movie. Obtain sagittal slice image of brain and show as individual image.

Another major advantage of MRI is its ability to image in any plane unlike computer tomography (CT) which is limited to one plane, the axial plane. An MRI system can create *axial* (upper to lower) images as well as images in the *sagittal* (right to left) plane and *coronal* (front to back) cross sectional view of the brain MR image (Fig. 2.4).

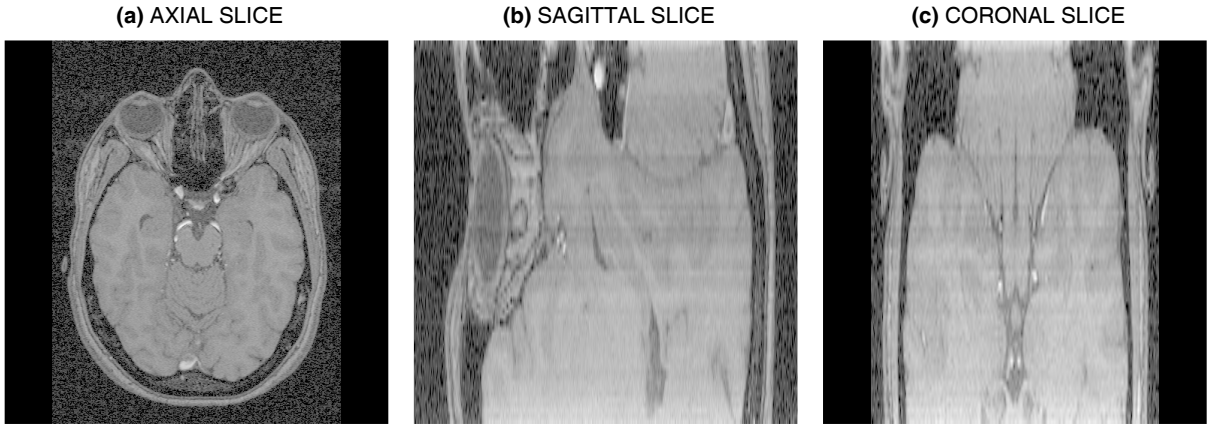


FIGURE 2.4. MR image of the head brain (a) axial slice, (b) sagittal slice, and (c) coronal slice

Sometimes it is necessary to interpolate the image data so that we have a smooth grayscale image or when an image has been sampled, we can fill in the missing samples by doing interpolation. Resampling the data to increase the resolution can be done using

the linear, spline or cubic spline interpolation (Fig. 2.5). In medical imaging this helps to have a quality clear picture when for example we are zooming in for region of interest.

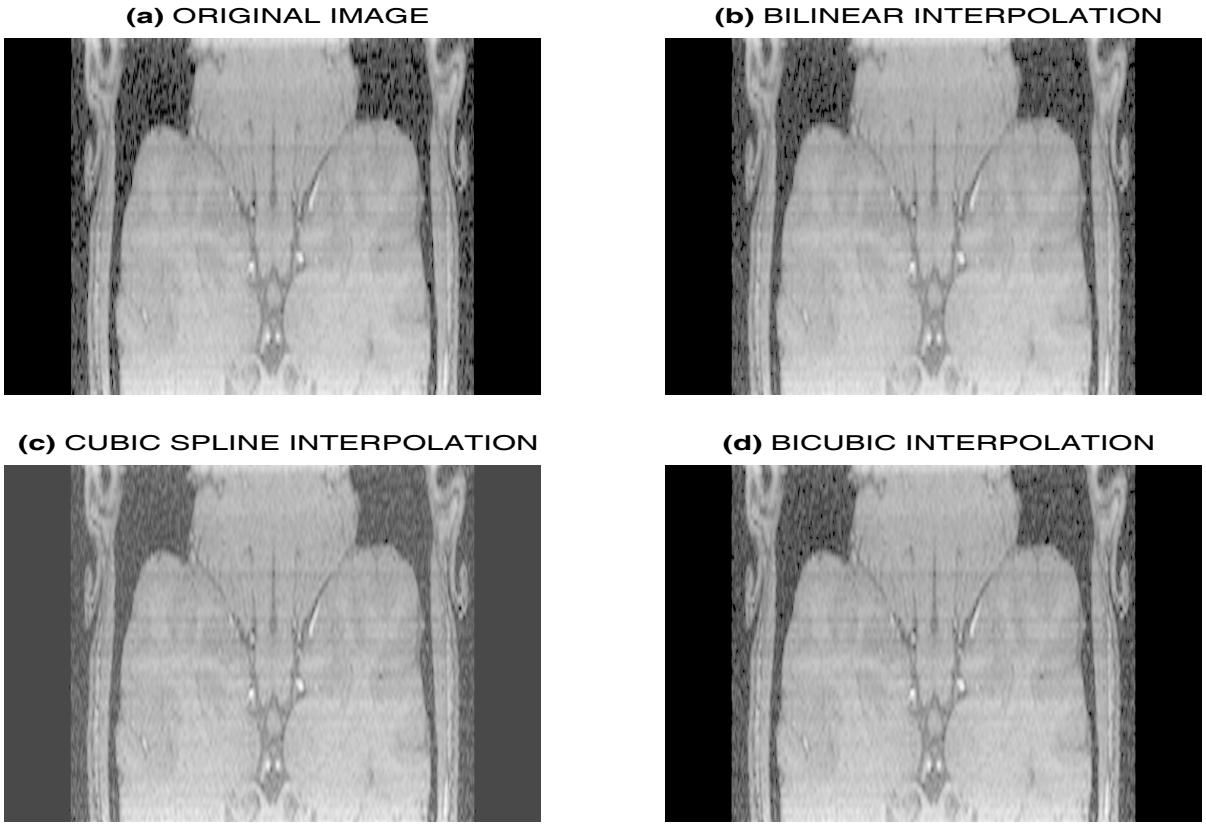


FIGURE 2.5. Interpolation of the (a) axial MR image of the head brain using (b) linear interpolation, (c) spline interpolation, and (d) cubic interpolation

2.3 Three-dimensional Biomedical Image Visualization

Three dimensional imaging [17] is now widely available and used often to aid in the comprehension and application of volumetric MR data to diagnosis, planning and therapy. Especially in clinical neurosurgery [29] 3-D visualization would benefit the planning and surgical treatment immensely. Models of the image data can be visualized by volume or contour surface rendering and can yield quantitative information. Assessment of three dimensional anatomical images (SPECT, PET, MRI and CT) is useful to determine the extent of lesions or tumors in the brain or other structures of interest. There are two different options available for the 3-D visualization of the data. One would be to use a commercially available program created specifically for this purpose. The second which is going to be applied here is to use MATLAB codes and functions to do the visualization.

Viewing the 3-D data

To create a 3-D model two-dimensional data sets (MRI slices) are stacked together to form a volume of data. The data is then addressed with three-dimensional co-ordinates (x, y, z) and oriented to choose a viewing direction. The three-dimensional array values are represented as *voxels* (three-dimensional versions of pixels) $f[m, n, k]$ where $m = 1, \ldots, M$ represents the x-pixel co-ordinates, $n = 1, \ldots, N$ represents the y-pixel co-ordinates and $k = 1, \ldots, K$ is the index of the corresponding slices. Fig. 2.6 shows a subset of volume data extracted after stacking together seventy MRI brain slices. The image volume of the brain MRI scan is displayed as an isosurface using the volume visualization functions in MATLAB including the lighting and perspective camera projection.

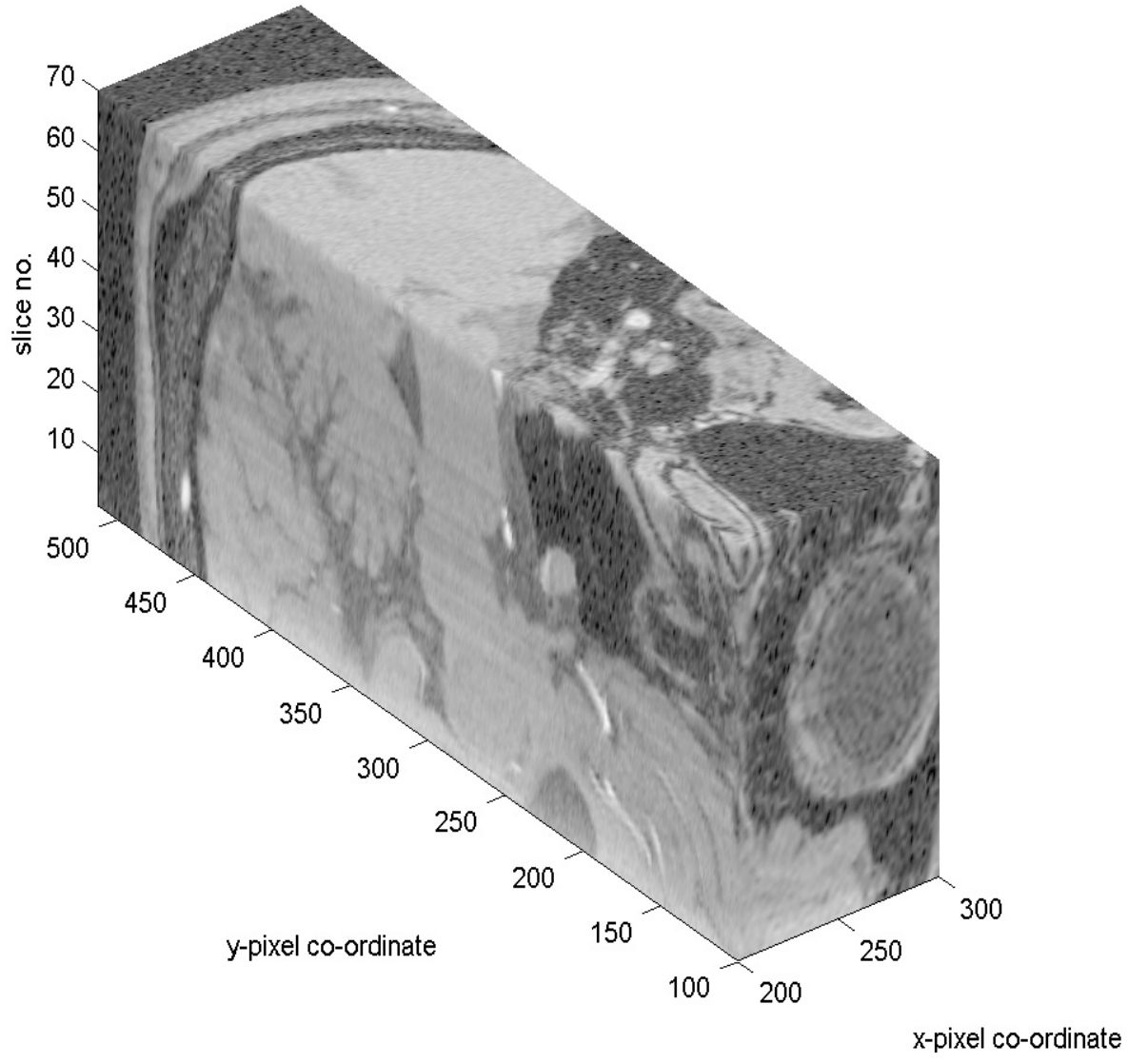


FIGURE 2.6. Extracted 3-D model of the brain and its visualization rendered as a 3-D view of the stack of images

2.4 Discussions

The developed 3-D visualization proved to operate efficiently with the MR images and it enables a clearer depiction of the complex brain anatomy. Its versatility enabled the integration of different objects (e.g., MRI data of the head and brain structures), different volume cuts, transparency, and multimodal visualization. Such properties enable efficient representation of images aiming to medical utility: tumors can be presented in 3-D together with surrounding MRI data.

A multimodal visualization MATLAB code was tested with the MR image data sets. To reduce computational time of 3-D reconstruction of the MR image slices a volume of interest (VOI) of size $100 \times 400 \times 70$ voxels was extracted and rendered into 3-D model producing an excellent 3-D image subvolume (Fig. 2.6). The most important subject of biomedical visualization is to learn and understand how the software code accomplishes the visualization, and also being able to modify the program to suit the user's specifications. This is not normally possible when using some commercial software programs but by using MATLAB and its good image processing tools, modifications and corrections of the code can be done to suit requirements which might be needed. A MATLAB code shown in Appendix D was written for a 3-D rendering program to analyze MRI scan data, and using this code modifications can be made where necessary like changing the colormap or displaying the image volume in different orientations.

3
Mathematical Methods of Image Processing

The chapter deals with the mathematical and theoretical background of the methods used in image processing namely the Fourier transform and wavelet transform. Discrete one-dimensional or multi-dimensional transforms represent mathematical tools for efficient signal and image analysis. Fourier analysis has an enormous impact in engineering, science, and mathematics. For example most medical blood pressure transducers are designed based on Fourier analysis. The list of engineering systems designed based on Fourier analysis is just endless.

The mathematical theory of wavelets is not very old, yet the wavelet transform has already become a fundamental tool in signal and image processing. It has been recognized that a global Fourier transform gives good information of the spectrum of the signal however unlike the wavelet transform it cannot easily detect high frequency bursts. Non-stationary signals which are unpredictable like a speech signal or an EEG signal can be easily analyzed using wavelets which necessitates the notion of frequency analysis that is local in time (time-scale analysis). High frequency bursts for instance cannot be read off or detected easily when using Fourier transform.

An important aspect of these transforms is the chance to extract relevant information from a signal or the underlying process, which is actually present but hidden in its complex representation.

3.1 Time-Frequency Analysis

Time-frequency analysis plays a central role in signal analysis. The Fourier Transform (FT) is only suitable for stationary signals, i.e., signals whose frequency content does not change with time. Fourier analysis is not well suited to describing local changes in frequency content because the frequency components defined by the Fourier transform have infinite (i.e. global) time support. FT just gives us the frequency components of a signal.

3.1.1 Discrete Fourier Transform

Discrete Fourier transform (DFT) plays a central role in the implementation of many signal and image processing algorithms. DFT is a mathematical transform which resolves

a time series $x(n)$ into the sum of an average component and a series of sinusoids with different amplitudes and frequencies. To compute the frequency content of a signal (or the frequency response of a system) we use the DTFT

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n} \tag{3.1}$$

It is easy to find an analytic expression for $X(e^{j\omega})$ for all ω if $x(n)$ has a closed-form analytic expression. However, what if we have a discrete signal $x(n)$ from a real-world environment? Two problems which arise with the above are:

- to compute $X(e^{j\omega_0})$ for any ω_0, we need to compute an infinite sum, this means we need to wait forever before we get an answer for any frequency.
- It's quite impossible to compute $X(e^{j\omega})$ for all $\omega \in [0, 2\pi)$ because there is an uncountably infinite number of points in that range.

There are two simple solutions to this: first, restrict $x(n)$ to be a finite-length sequence of N samples numbered from 0 to $N-1$, i.e. $x(n)$ is nonzero only between $n = 0$ and $n = N-1$, second instead of computing $X(e^{j\omega})$ for all frequencies, compute it only for a finite number of points on $[0, 2\pi)$

The fact that $x(n)$ is a finite-length sequence implies that the DTFT can be rewritten as

$$X(e^{j\omega}) = \sum_{n=0}^{N-1} x(n)\, e^{-j\omega n} \tag{3.2}$$

Computing the DTFT for only a finite number of frequency points means that we can further simplify Eq. (3.2) to

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x(n)\, e^{-j\omega_k n} \tag{3.3}$$

where $\omega_k = \frac{2\pi k}{N}$ are the frequency samples; if we assume that there are N samples, then $k = 0, 1, \ldots, N-1$. That is, we are evaluating $X(e^{j\omega})$ only at the frequency values $\omega_k = \frac{2\pi k}{N}$ for $k = 0, 1, \ldots, N-1$. The resulting expression is

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x(n)\, e^{-j\frac{2\pi kn}{N}} \tag{3.4}$$

The N-point discrete Fourier transform (DFT), $X(k)$, of an N-point discrete-time sequence, $x(n)$, is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N} \tag{3.5}$$

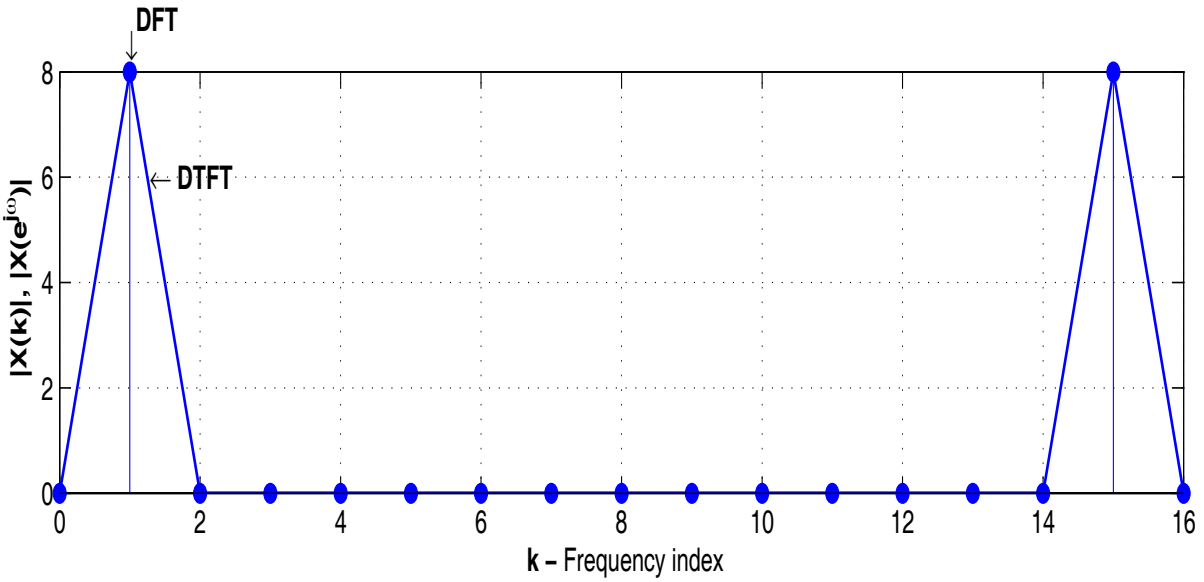


FIGURE 3.1. Comparison of DTFT and DFT of a sinusoidal signal sampled at 1Hz.

for $k = 0, 1, \ldots, N-1$. The DFT, $X[k]$, is just a sampled version of the DTFT, $X(e^{j\omega})$, at $\omega = \frac{2\pi k}{N}$. Another way to think of it is in terms of evaluating the z-transform, $X(z)$, at N points on the unit circle, $z = e^{\frac{j2\pi k}{N}}$ (Fig. 3.2).

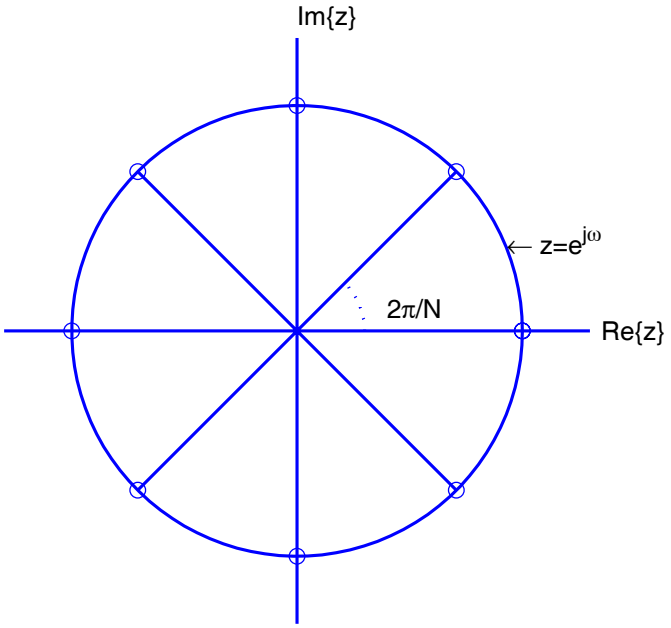


FIGURE 3.2. Z-transform the unit circle in the z-plane, showing (dots) the locations $z = e^{j\omega}$ at which $X(e^{j\omega})|_{\omega\,=\,2\pi k/N}, \quad 0 \le k \le N-1, \ N = 8$.

Sampling Theorem

To obtain a discrete-time signal $x(n)$: sample a continuous-time signal $x(t)$ every T seconds. Given a sampling period of T seconds, the sampling frequency $f_s = \frac{1}{T}$ Hz, and

the record length of an N-sample sequence is NT seconds. By sampling we throw out a lot of information all values of $x(t)$ between sampling points are lost. Under what conditions can we reconstruct the original signal $x(t)$ from its samples? Suppose $x(t)$ is bandlimited, such that

$$X(\omega) = 0 \quad \text{for} \quad |\omega| > \omega_m$$

Then $x(t)$ is uniquely determined by its samples $\{x(nT)\}$ if

$$\omega_s > 2\omega_m$$

where $\omega_s = 2\pi/T$ and this is called the *Nyquist sampling theorem*

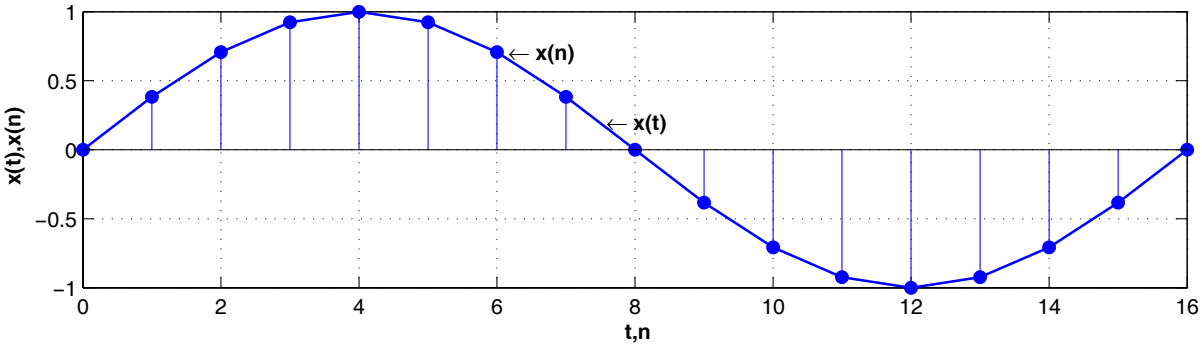


FIGURE 3.3. Sampled discrete signal $x(n)$ from a continuous time signal $x(t)$

Inverse DFT

An important property of the discrete transform is that unlike the continuous case we need not be concerned about the existence of the DFT or its inverse. The DFT and its inverse always exist since we are only dealing with finite values. These comments are directly applicable to two-dimensional and higher functions. For digital image processing, existence of either the discrete transform or its inverse is not an issue. Now that we have the frequency-domain samples $X(k)$, how do we recover the time-domain signal $x(n)$? The inverse of the DFT is defined as:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, e^{j2\pi kn/N} \quad n = 0, 1, \ldots, N-1 \tag{3.6}$$

Note that both $x(n)$ and $X(k)$ are actually periodic in N:

$$X(k+N) = \sum_{n=0}^{N-1} x(n)\, e^{-j\frac{2\pi(k+N)n}{N}} = X(k)$$

$$x(n+N) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, e^{j\frac{2\pi(k+N)n}{N}} = x(n)$$

The fact that $x(n)$ is periodic is another example of time-frequency duality: sampling in the time domain results in a periodic DTFT and sampling in the frequency domain also results in a a periodic time-domain sequence. But $x(n)$ is a finite-length sequence, so how can it be periodic? The answer to this is the discrete Fourier Series (DFS). Consider a periodic sequence $\tilde{x}(n)$ with period N:

$$\tilde{x}(n) = \sum_{r=-\infty}^{\infty} x(n + rN)$$

This sequence is formed by concatenating an infinite number of our original N-sample $x(n)$ sequences. another way to think of it is via the modulo notation:

$$\tilde{x}(n) = x[n \bmod N] = x[(n)_N]$$

Continuous-time periodic sequences can be represented in terms of their Fourier series components, the same is true for discrete-time periodic sequences:

$$\tilde{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(k)\, e^{jk\omega_0 n} \tag{3.7}$$

$\omega_0 = \frac{2\pi}{N}$ is the fundamental frequency. $\tilde{X}(k)$ are the discrete Fourier series (DFS) coefficients given by

$$\tilde{X}(k) = \sum_{n=0}^{N-1} \tilde{x}(n)\, e^{-j2\pi kn/N} \tag{3.8}$$

Note that $\tilde{x}(n)$ is defined for all n and $\tilde{X}(k)$ is defined for all k. Also note that unlike the continuous-time Fourier series, the upper limit of the summation in the formula for $\tilde{x}(n)$ is finite: this is because of the periodicity of $e^{j\frac{2\pi n}{N}}$.

But its obvious that the equations for the DFS and IDFS are identical to those for DFT and IDFT. The mathematics is the same, but the interpretation is different, DFT takes an N-sample time-domain sequence and produces an N-sample frequency-domain sequence. DFS takes a periodic (with period N) time-domain sequence and produces a periodic N-sample frequency-domain sequence. One way to think about it is that the DFT operates on one period of an underlying periodic time-domain sequence and returns one period of an underlying periodic frequency-domain sequence.

From Eq. (3.5) it can be seen that the components of the Fourier transform are complex quantities. So it's necessary to express $X(k)$ in a polar co-ordinate form.

$$X(k) = |X(k)|\, e^{-j\phi(k)} \tag{3.9}$$

where

$$|X(k)| = \left(R^2(k) + I^2(k)\right) \tag{3.10}$$

is the *magnitude* or *spectrum* of the Fourier transform and

$$\phi(k) = \tan^{-1}\left(\frac{I(k)}{R(k)}\right) \tag{3.11}$$

is the *phase angle* or *phase spectrum* of the transform. In Eqs. (3.10) and (3.11), $R(k)$ and $I(k)$ are the real and imaginary parts of $X(k)$ respectively. However in terms of image enhancement we are most concerned primarily with properties of the spectrum. Another quantity commonly used is the *power spectrum* or *spectral density* simply defined as the square of the Fourier magnitude

$$P(k) = |X(k)|^2 \tag{3.12}$$

From Eq. (3.5) it can be seen that the computation of $X(k)$ requires N^2 complex multiplications, thus the DFT is an $O(N^2)$ process. An algorithm was developed by Tukey and Cooley in 1965 called the Fast Fourier Transform (FFT) [16] that speeds up the process by computing the DFT using $O(N \log N)$ operations. Note that the FFT is just a faster algorithm for computing the DFT it does not produce a different result.

3.1.2 Short-Time Fourier Transform

To analyze non-stationary signals such as image and speech it's necessary to have a good time and frequency localization. To solve this problem, the Short-Time Fourier Transform (STFT) was introduced by Dennis Gabor (1946). In STFT, the non-stationary signal is divided into small portions, which are assumed to be stationary [70, 87]. This is done using a window function of a chosen width, which is shifted and multiplied with the signal to obtain the small stationary signals. The Fourier Transform is then applied to each of these segments mapping a signal into a two-dimensional function of time and frequency (*time-frequency plane*). STFT thus provides some information about both when and at what frequencies a signal event occurs.

Consider a sinusoidal signal (Fig. 3.4(a)) consisting of two frequencies, one frequency $f1 = 0.1$ Hz existing over an interval $T = 256s$ and the second a frequency $f2 = 0.35$ Hz existing over another interval $T = 256s$. Using DFT we will only get the information that the signal has two components of 0.1 Hz and 0.35 Hz but won't say when they start and stop but the STFT will do it i.e. time and frequency realization (Fig. 3.4(c) and (d)). A graphical display of the magnitude of the STFT, is called the spectrogram of the signal. To assume stationarity, the window is supposed to be narrow, which results in a poor frequency resolution, i.e., it is difficult to know the exact frequency components that exist in the signal; only the band of frequencies that exist is obtained. If the width

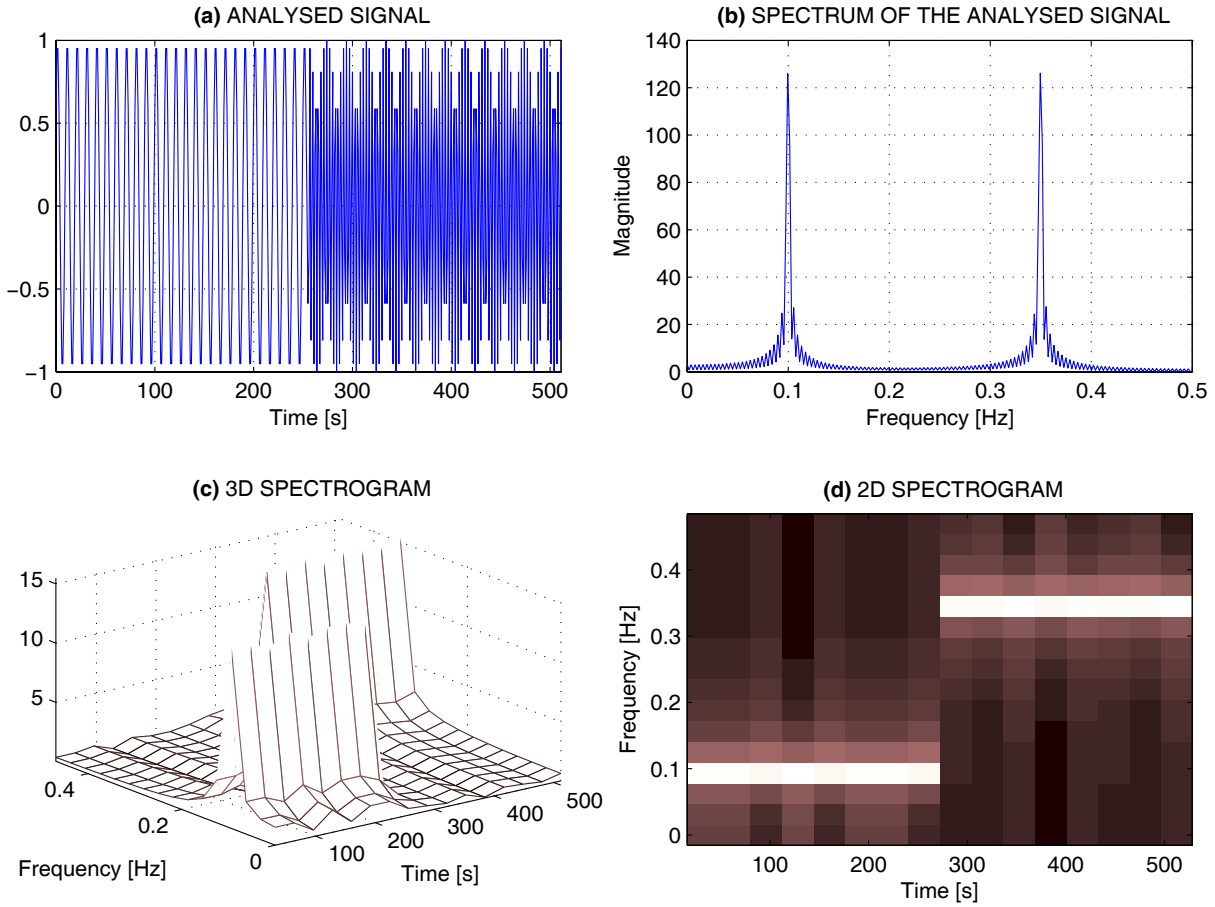


FIGURE 3.4. Signal analysis presenting (a) the original signal, (b) its spectrum, (c) 3-D spectrogram using a window length of 16 samples, and (d) 2-D spectrogram

of the window is increased, frequency resolution improves but time resolution becomes poor, i.e., it is difficult to know what frequencies occur at which time intervals. Also, choosing a wide window may violate the condition of stationarity. Thus, achieving good time and frequency resolution simultaneously is impossible as summarized in Tab.3.1. Consequently, depending on the application, a compromise on the window size has to be

TABLE 3.1. Time and Frequency resolution

Narrow Window	Good time resolution	Poor frequency resolution
Wide Window	Poor time resolution	Good frequency resolution

made. Once the window function is decided, the frequency and time resolutions are fixed for all frequencies and all times. This is a major drawback of the STFT and an alternative to this method is the wavelet transform. The most important feature of wavelet transforms is that they analyze different frequency components of a signal with different resolutions.

3.1.3 Two-dimensional Discrete Fourier Transform

Two-dimensional discrete Fourier transform is used for the processing of images. Basis functions are sinusoids with frequency u in one direction times sinusoids with frequency v in the other. For an $M \times N$ image $f[m, n]$, these basis functions can be replaced for computational purposes by complex exponentials $e^{i2\pi um/M}$ and $e^{i2\pi vn/N}$ to evaluate the discrete Fourier transform. It is taken by applying the one-dimensional transform to each row, and then to each column. For an $M \times N$ image, the 2-D DFT is usually defined as:

$$F(u, v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)\, e^{-i2\pi(um/M + vn/N)} \tag{3.13}$$

and its inverse transform is

$$f(m, n) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v)\, e^{i2\pi(um + vn)} \tag{3.14}$$

where $F[u, v]$ in the frequency domain is the spectrum of the image and $m = 0, 1, \ldots, M-1, n = 0, 1, \ldots, N-1, u = 0, 1, \ldots, M-1, v = 0, 1, \ldots, N-1$ are all discrete variables. As in the 1-D case the Fourier spectrum, phase angle and power spectrum for 2-D discrete transform are defined as:

$$|F(u, v)| = \left(R^2(u, v) + I^2(u, v)\right) \tag{3.15}$$

$$\phi(u, v) = \tan^{-1}\left(\frac{I(u, v)}{R(u, v)}\right) \tag{3.16}$$

$$P(u, v) = |F(u, v)|^2 \tag{3.17}$$

where $R(u, v)$ and $I(u, v)$ are the real and imaginary parts of (u, v) respectively. u, v are integers. The value of the transform at $(u, v) = (0, 0)$ from Eq. (3.13) is

$$F(0, 0) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) \tag{3.18}$$

which is the mean of all pixels in the image, thus the Fourier transform at the origin is equal to the mean of the gray level of the image. Because both frequencies are zero at the origin, $F(0, 0)$ is sometimes called the *dc* component of the spectrum.

Properties of 2-D DFT

There are many properties of the DFT that give insight into the content of the frequency domain representation of a signal and allow us to manipulate signals in one domain or the other. The 2-D DFT has many properties that are useful in image processing and a few interesting properties of these include:

Translation

A simple relationship that can be derived for shifting an image in one domain or the other. The Fourier transform pair has the following translation properties:

$$f(m,n)\, e^{-i2\pi(u_0 m/M + v_0 n/N)} \Leftrightarrow F(u-u_0, v-v_0) \tag{3.19}$$

and

$$f(m-m_0, n-n_0) \Leftrightarrow F(u,v)\, e^{-i2\pi(um_0/M + vn_0/N)} \tag{3.20}$$

Scaling

Shrinking in one domain causes expansion in the other for the 2-D DFT. This means that as an object grows in an image, the corresponding features in the frequency domain will expand. The equation governing this is:

$$f(am, bn) \Leftrightarrow \frac{1}{|ab|}\, F\left(\frac{u}{a}, \frac{v}{b}\right) \tag{3.21}$$

From Fourier theory we know that compression in time is equivalent to stretching the spectrum and shifting it upwards:

Periodicity and conjugate symmetry

The discrete Fourier transform has the following periodicity properties with period N:

$$F(u,v) = F(u+M, v) = F(u, v+N) = F(u+M, v+N) \tag{3.22}$$

The inverse transform is also periodic:

$$f(m,n) = F(m+M, n) = F(m, n+N) = F(m+M, n+N) \tag{3.23}$$

The idea of conjugate symmetry is

$$F(u,v) = F^*(-u,-v) \tag{3.24}$$

where $*$ indicates the standard conjugate operation on a complex number. The spectrum is also symmetric about the origin:

$$|F(u,v)| = |F(-u,-v)| \tag{3.25}$$

These properties allow for a rearrangement for the magnitude spectrum, either before or after the transform, so that one whole period is viewed with its origin shifted to the frequency point $(N/2, N/2)$.

Separability

The 2-D Fourier transform is *linearly separable* i.e. the Fourier transform of a two-dimensional image is the Fourier transform of the rows followed by the Fourier transform of the resulting columns (or vice versa) as shown below.

$$\begin{aligned} F(u,v) &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\; e^{-i2\pi(um/M\,+\,vn/N)} \\ &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\; e^{-i2\pi um/M}\; e^{-i2\pi vn/N} \\ &= \sum_{m=0}^{M-1}\left[\sum_{n=0}^{N-1} f(m,n)\; e^{-i2\pi um/M}\right] e^{-i2\pi vn/N} \end{aligned}$$

The fast Fourier transform (FFT) [16] is an efficient algorithm to calculate the DFT. For the 2-D DFT a total of $O(N^4)$ operations are required and by using the FFT the number of operations can be reduced to an order of $O(N^2 \log N)$. N and M are commonly powers of 2 (for the FFT). Therefore the overall complexity of a 2-D FFT is $O(N^2 \log N)$, where N^2 equals the number of pixels in the image.

Thus an $M \times N$ image has an $M \times N$ set of (complex) Fourier coefficients. To implement this transform, we would like an analog of the FFT, which will let us quickly compute the coefficients of the transform. In fact, we can do better. The two dimensional DFT is separable into two one dimensional DFTs which can be implemented with an FFT algorithm.

Convolution Theorem

The Fourier convolution theorem states that convolution in one domain is multiplication in the other and vice versa. The discrete convolution of two functions $f(x,y)$ and $h(x,y)$ of size $M \times N$ denoted by $f(x,y) * h(x,y)$ and is defined by Eq. (3.26)

$$f(x,y) * h(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\; h(x-m, y-n) \tag{3.26}$$

can think of f as representing an image and h represents a filter. The convolution theorem consists of the following relationships between the two functions and their Fourier transforms:

$$f(x,y) * h(x,y) \Leftrightarrow F(u,v)\; H(u,v) \tag{3.27}$$

and

$$f(x,y)\; h(x,y) \Leftrightarrow F(u,v) * H(u,v) \tag{3.28}$$

Frequency Domain Filtering

One of the applications of the convolution theorem is filtering in the frequency domain. Filtering may be achieved in the transform domain by first computing the DFT, applying a filter which modifies the transform values, and then applying an inverse transform. The image $f(x, y)$ is first transformed to give $F(u, v)$ and finally the transform is modified using the equation below

$$G(u, v) = H(u, v)\, F(u, v)$$

where $H(u, v)$ is the filter function. The filtered transformed is then inverse-transformed to give the filtered image $g(x, y)$. An illustration of this application is represented in Fig. 3.5.

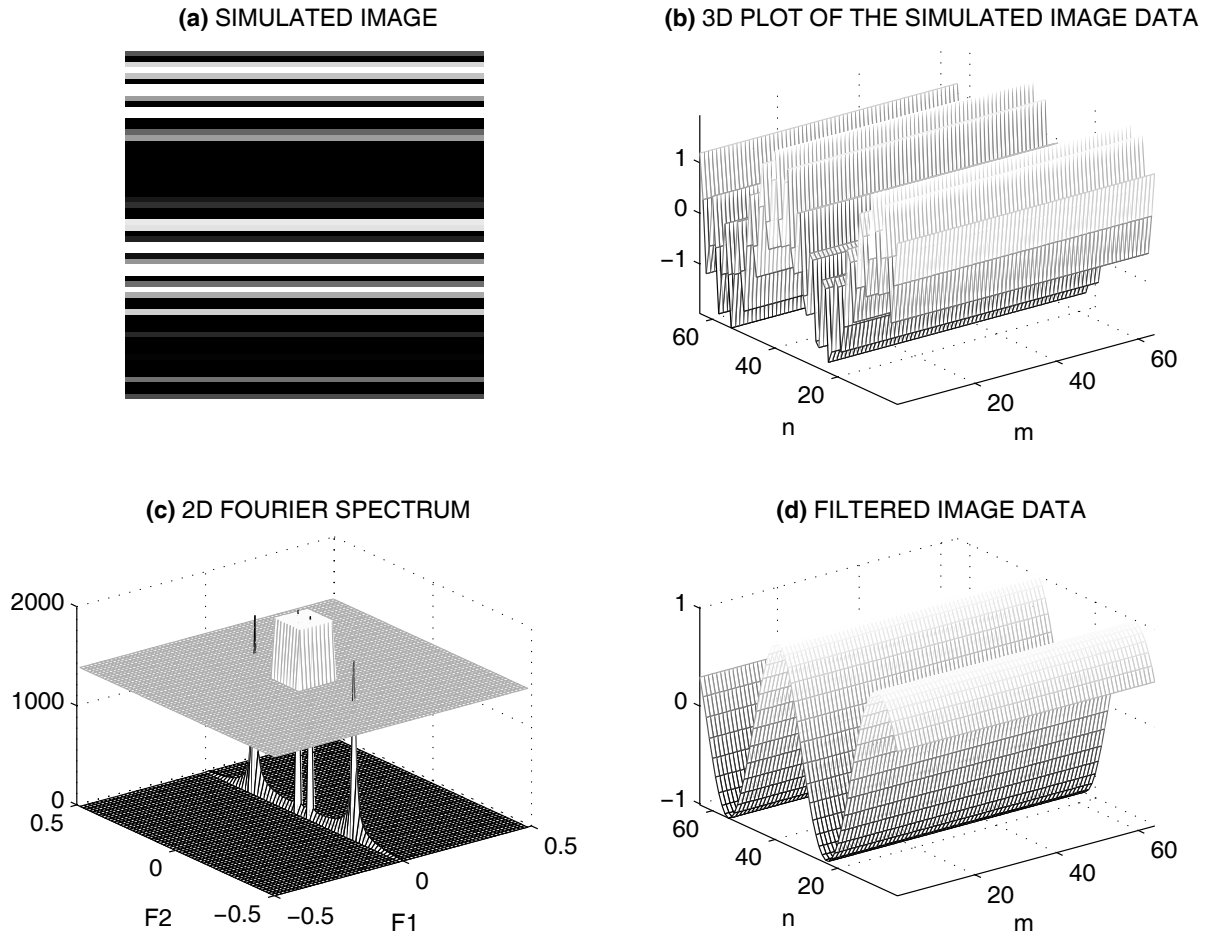


FIGURE 3.5. Application of the 2-D DFT for the frequency domain filtering presenting (a) simulated image, (b) three-dimensional plot of the image data, (c) DFT of both the two-dimensional signal and the box filter function, and (d) low frequency image data in time domain obtained from the inverse DFT. Scalar product of the filter and signal in frequency domain stand for convolution in time domain

3.2 Time-Scale Analysis

The continuous wavelet transform (CWT) provides a time-scale description similar to that of the STFT with a few important differences: frequency is related to scale and the CWT is able to resolve both time and scale (frequency) events better than the STFT. By time localization we mean the ability to clearly identify signal events which show up during a short time interval, such as an abrupt impulse. To distinguish such transient behavior of a signal [92] it's necessary to have some basis functions which are very short (high frequency) and long (low frequency) depending on the resolution of the frequency analysis.

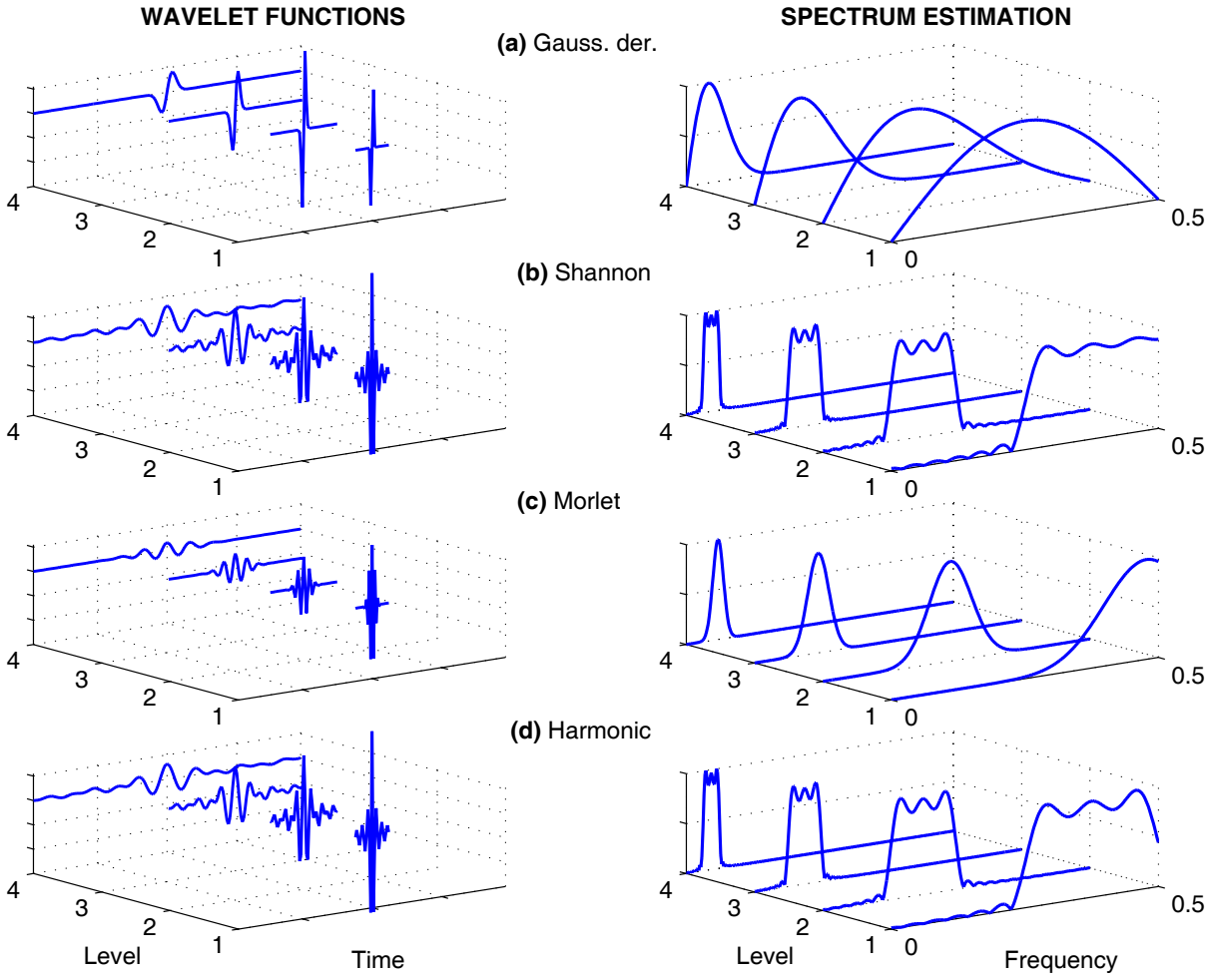


FIGURE 3.6. Wavelet functions in time and frequency domain

This can be achieved with the wavelet transform, where the basis functions are obtained from a single prototype wavelet (Eq. (3.29)) (mother wavelet) by translation and dilation (contraction)

$$W_{a,b}(t) = \frac{1}{\sqrt{a}}\, W\left(\frac{t-b}{a}\right) \tag{3.29}$$

where $a \in R^+, b \in R$. For large a, the basis function becomes a stretched version of the prototype wavelet which is a low frequency function, while for small a the basis function becomes a contracted wavelet which is a high frequency function. The continuous wavelet transform (CWT) [50] is then defined as the convolution of $x(t)$ with a wavelet function, $W(t)$, shifted in time by a translation parameter b and a dilation parameter a (Eq. (3.30))

$$X_W(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} W\left(\frac{t-b}{a}\right) x(t)dt \tag{3.30}$$

At high frequencies the CWT is sharper in time while at low frequencies the CWT is sharper in frequency. $W(t)$ denotes the mother wavelet. The parameter a represents the scale index that is the reciprocal of the frequency. The parameter b indicates the time shifting (or translation). Fig. 3.6 illustrates some of the commonly used wavelet functions. Generation of the wavelet family by translation (parameter b) and dilation (parameter a). The graph of $W\left(\frac{t-b}{a}\right)$ is obtained by stretching the graph of $W(t)$ by the factor a, called the scale, and shifting in time by b. Wavelet series are thus constructed with two parameters scale and translation, these parameters make it possible to analyze a signal behavior at a dense set of time locations and with respect to a vast range of scales, thus providing the ability to zoom in on the transient behavior of the signal. Suppose a

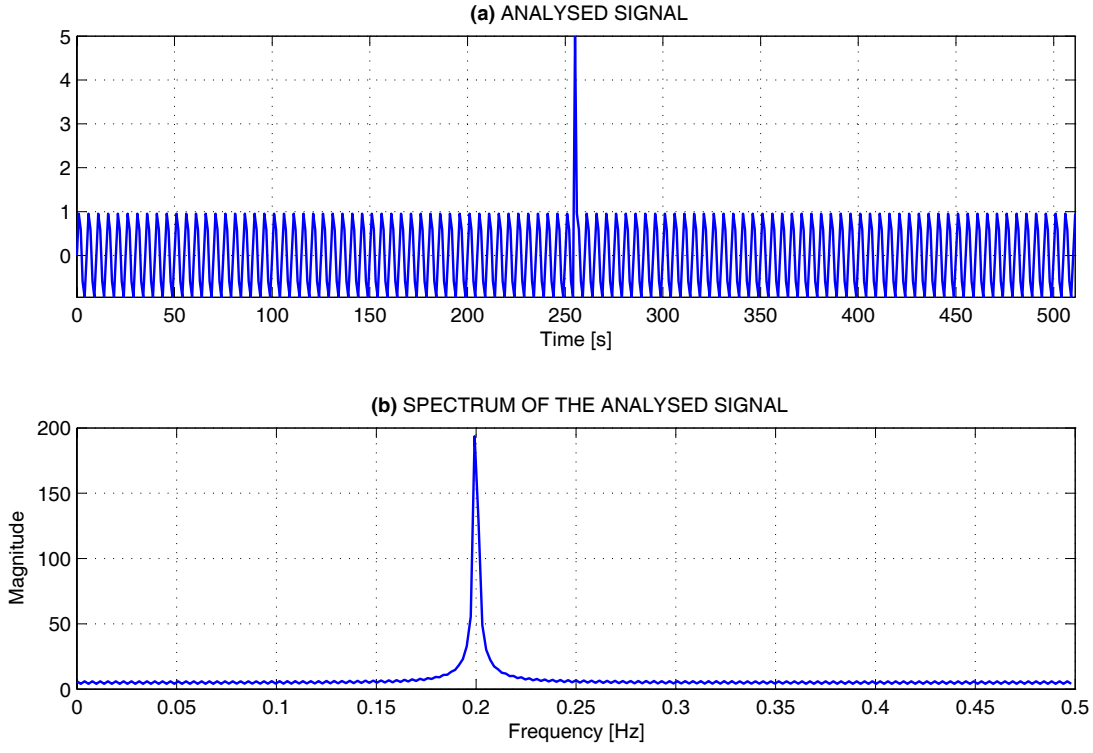


FIGURE 3.7. DFT analysis of time series presenting (a) signal with an impulse at time $t = 256$ and (b) its spectrum pointing to its frequency $f = 0.2$ Hz. The signal impulse is not detected

sinusoidal signal (Fig. 3.7(a)) of length 512 seconds with an impulse at 256 s, using the DFT we will only get the information that the signal has a frequency component of 0.2 Hz (Fig. 3.7(b)) but won't say when the impulse occurred. The STFT (Fig. 3.8(a)) doesn't tell us exactly where is the impulse but looking at the spectrograms of the DWT (Fig. 3.8(c)) and CWT (Fig. 3.8(e)) it's possible to find exactly when the impulse occurred.

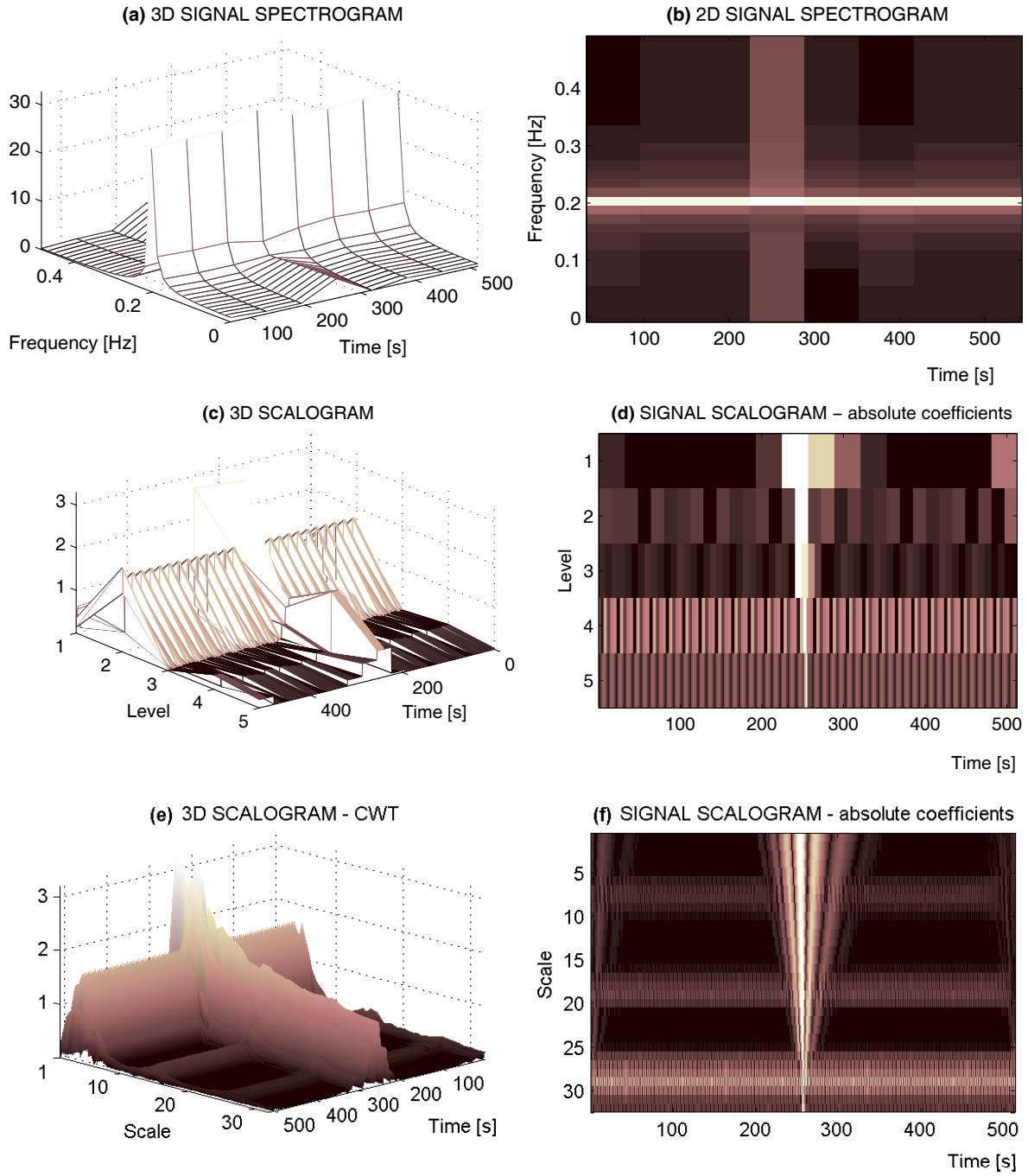


FIGURE 3.8. Comparison of the STFT, DWT and the CWT in detecting an impulse added to a sinusoidal signal (a),(b) STFT does not detect the impulse accurately as can be seen from its spectrograms, (c),(d) DWT shows that as the level increases the time/frequency resolution increases, and (e),(f) CWT clearly illustrates that the higher the scale the better the the detection of the impulse (see scalograms)

The scalogram is defined as the square magnitude of the CWT coefficients, $|W(a,b)|^2$. The scalogram can be computed at any scale and the position can also be shifted continuously over the entire time domain of the signal being analyzed. In contrast to STFT, which uses a single analysis window, the wavelet transform uses short windows at high frequencies and long windows at low frequencies. This results in multi-resolution analysis by which the signal is analyzed with different resolutions at different frequencies, i.e., both frequency resolution and time resolution vary in the time-frequency.

3.2.1 Discrete Wavelet Transform

By restricting to a discrete set of parameters we get the Discrete Wavelet Transform (DWT) [64] which corresponds to an orthogonal basis of functions all derived from a single function called the mother wavelet.

CWT is redundant since the parameters (a,b) are continuous thus it's necessary to discretize the grid on the time-scale plane corresponding to a discrete set of continuous basis functions. This lead us to a question: how can we discretize the wavelet in Eq. (3.29)?

$$W_{j,k}(t) = \frac{1}{\sqrt{a_j}}\, W\left(\frac{t-b_k}{a_j}\right) \tag{3.31}$$

In theory $a_j = a_0^j$ and $b_k = kb_0a_0^j$ where $j,k \in Z, \quad a_0 > 1, \quad b_0 \neq 0$. The discrete form of the wavelet is shown in Eq. (3.31), where j controls the dilation and k controls the translation. Two popular choices for the discrete wavelet parameters a_0 and b_0 are 2 and 1 respectively, a configuration that is known as the *dyadic grid* arrangement, Eq. (3.31) can be written as

$$\begin{aligned} W_{j,k}(t) &= a_0^{-j/2}.\, W(a_0^{-j}\, t - kb_0) \\ &= 2^{-j/2}.\, W(2^{-j}\, t - k) \end{aligned}$$

This is discretization on a dyadic grid which occurs for $a_0 = 2,\ b_0 = 1$. The $a_0^{-j/2}$ term scales the signal. Discrete dyadic wavelets are usually selected to be orthonormal, which means that the wavelets are both orthogonal and normalized to have unit energy. The discrete wavelet transform (DWT) [50] is another way to decompose a time series into a sequence of components with different scales. The original signal can be reconstructed from these components. There is no redundancy in the output from the DWT (Fig. 3.8(d)) unlike the CWT (Fig. 3.8(f)) which generates a redundant two-dimensional scalogram.

Wavelet analysis is simply the process of decomposing a signal into shifted and scaled versions of a mother (initial) wavelet. An important property of wavelet analysis is perfect reconstruction, which is the process of reassembling a decomposed signal or image into

its original form without loss of information. For decomposition and reconstruction two types of basis functions normally used are:

- Scaling function $\Phi_{jk}(t)$

$$\Phi_{jk}(t) = 2^{-\frac{j}{2}}\ \Phi_0(2^{-j}\ t - k) \tag{3.32}$$

- Wavelet $W_{jk}(t)$

$$W_{jk}(t) = 2^{-\frac{j}{2}}\ \Psi_0(2^{-j}\ t - k) \tag{3.33}$$

where m stands for dilation or compression and k is the translation index. Every basis function W is orthogonal to every basis function Φ. Wavelets are functions defined over a finite interval and have an average value of zero.

An example of a simple wavelet function is called the Haar wavelet. Historically the Haar function was the original wavelet but with poor approximation. In translation the word *wavelet is a small wave*. In Haar's case it is a square wave. The square wave $W(t)$ has compact support and it comes from a FIR filter with finite length. The words *compact support* mean that the interval here $[0, 1]$ is closed and outside $W(t)$ is zero (bounded interval). The Haar mother wavelet $W(t)$ and scaling function $\Phi(t)$ are defined as follows:

$$\Phi(t) = \begin{cases} 1 & 0 \leq t \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$W(t) = \begin{cases} 1 & 0 \leq t \leq \frac{1}{2} \\ -1 & \frac{1}{2} \leq t \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

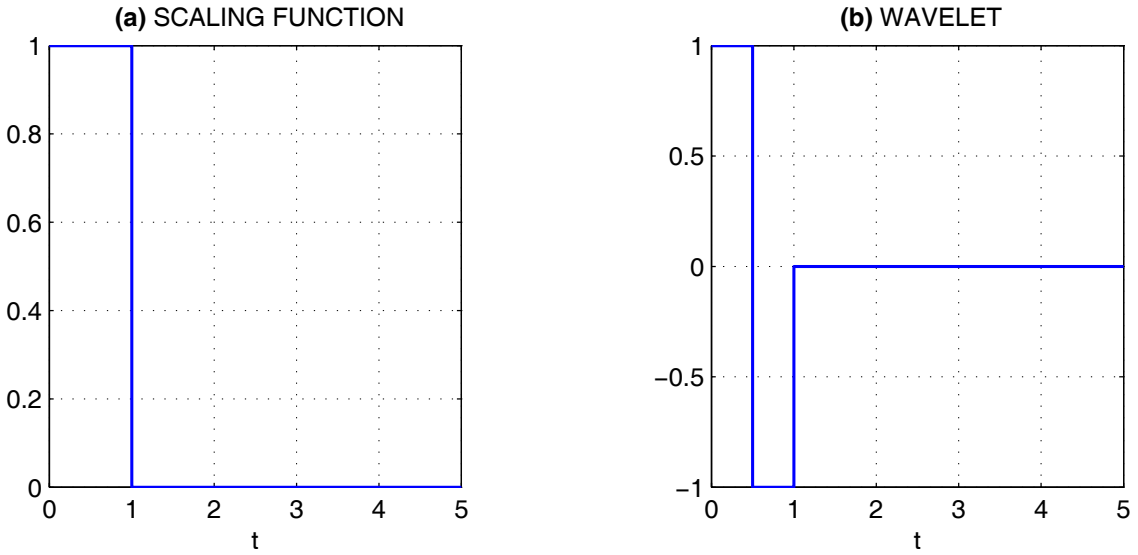


FIGURE 3.9. An example of (a) Haar scaling function and (b) Haar wavelet

Filter Bank Decomposition

The discrete wavelet transform (DWT) [42] is commonly implemented using dyadic multirate filter banks, which are sets of filters that divide a signal frequency band into subbands. At each scale in DWT, the approximation coefficients are generated from a low-pass filter and are associated with the low frequency trend while the detail coefficients are output from a high-pass filter and capture the high frequency components of the time series

The inverse discrete wavelet transform (IDWT) reconstructs a signal from the approximation and detail coefficients derived from decomposition. The IDWT differs from the DWT in that it requires upsampling and filtering, in that order. Upsampling, also known as interpolating, means the insertion of zeros between samples in a signal. The right side of Fig. 3.11 shows an example of reconstruction.

In Fig. 3.10, L and H represent the scaling function and wavelet function respectively. The wavelet filter coefficients are obtained from alternating flip of scaling filter coefficients. Short filter results in faster computation of convolutions. A pair of filters: a low-pass filter L and a high-pass filter H, split a signal's bandwidth in two halves. This provides the coefficients $c_j(k)$ and $d_j(k)$ for the decomposition of the signal into its scaling function and wavelet function components.

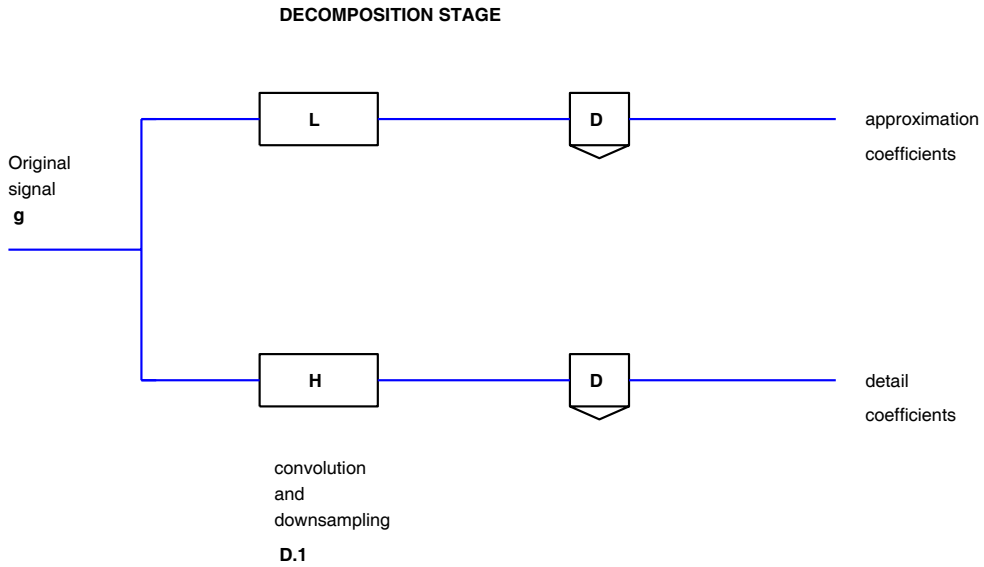


FIGURE 3.10. One-dimensional signal decomposition

The 1-D forward wavelet transform of a discrete-time signal $x(n)$ $(n = 0, 1, \ldots, N)$ is performed by convolving signal $x(n)$ with both a half-band low-pass filter L and high-pass filter H and downsampling by two.

$$c(n) = \sum_{n=0}^{L-1} h_0(k)\, x(n-k) \qquad d(n) = \sum_{n=0}^{L-1} h_1(k)\, x(n-k) \tag{3.34}$$

where $c(n)$ represent the approximation coefficients for $n = 0, 1, 2 \ldots, N-1$ and $d(n)$ are the detail coefficients, h_0 and h_1 , are coefficients of the discrete-time filters L and H respectively

$$\{h_0(n)\}_{n=0}^{L-1} = (h_0(0),\ h_0(1),\ \ldots,\ h_0(L-1))$$

$$\{h_1(n)\}_{n=0}^{L-1} = (h_1(0),\ h_1(1),\ \ldots,\ h_1(L-1))$$

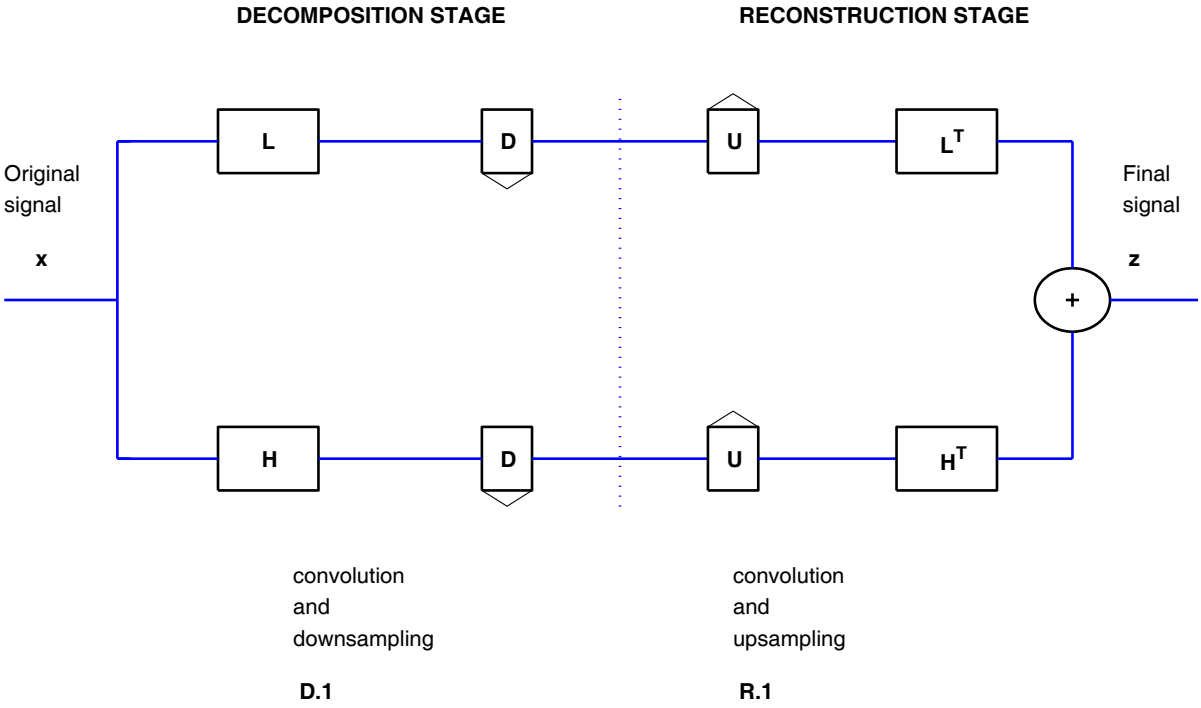


FIGURE 3.11. Signal decomposition and reconstruction

The connection between filter banks and wavelets is that the highpass filter leads to $W(t)$ and the lowpass filter leads to a scaling function $\Phi(t)$. A filter bank is a set of filters, the analysis bank often has two filters, lowpass H_0 and highpass H_1. They split the input signal into frequency bands. The filtered outputs from both filters give a double signal length. To overcome this we have to downsample or decimate. To compensate for losing half the components in downsampling we multiply the downsampled signal $y(2n)$ by $\sqrt{2}$. This normalizing factor is usually included with the filter bank, so that

$$lowpass: \quad H_0(\omega) \text{ changes to } C(\omega) = \sqrt{2}\, H_0(\omega)$$

$$highpass: \quad H_1(\omega) \text{ changes to } D(\omega) = \sqrt{2}\, H_1(\omega)$$

For the lowpass coefficients of a Haar filter $h(0) = h(1) = \frac{1}{2}$ for the original coefficients and introducing $c(0), c(1)$ for the normalized coefficients of C we have:

$$c(0) = c(1) = \frac{\sqrt{2}}{2} = \frac{1}{\sqrt{2}}$$

Similarly for the highpass coefficients $\frac{1}{2}$ and $-\frac{1}{2}$ are multiplied by $\sqrt{2}$, in D:

$$d(0) = \frac{\sqrt{2}}{2} = \frac{1}{\sqrt{2}} \text{ and } d(1) = -\frac{\sqrt{2}}{2} = -\frac{1}{\sqrt{2}}$$

The response at frequency $\omega = 0$ is $C = \sqrt{2}$ rather than 1. Decimation of filters in the time domain is carried out in such a way that downsampling follows the filter C, in operating on x. The combination of filtering by C and decimation by 2 is represented by a *rectangular matrix* L that no longer has constant diagonals.

$$L = (\downarrow 2)\, C = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & & & \\ & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & \\ & & & & \cdot & \cdot \\ & & & & \cdot & \cdot \end{bmatrix}$$

The entries are $c(0)$ and $c(1)$ but half the rows have gone (downsampling removes every second row). Similarly the decimated highpass filter is represented by a rectangular matrix B:

$$B = (\downarrow 2)\, D = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & & & \\ & & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & \\ & & & & \cdot & \cdot \\ & & & & \cdot & \cdot \end{bmatrix}$$

Putting the lowpass L and the highpass B into one matrix. The rectangular L and B fit into a square matrix:

$$\begin{bmatrix} (\downarrow 2)\, C \\ (\downarrow 2)\, D \end{bmatrix} = \begin{bmatrix} L \\ B \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & & & & \\ & & 1 & 1 & & \\ & & & & \cdot & \cdot \\ & & & & \cdot & \cdot \\ -1 & 1 & & & & \\ & & -1 & 1 & & \\ & & & & \cdot & \cdot \\ & & & & \cdot & \cdot \end{bmatrix}$$

This matrix represents the whole analysis bank. All rows are unit vectors (because of the division by $\sqrt{2}$ - the normalizing factor). The row vectors are mutually orthogonal and at the same time the columns are also *orthogonal unit vectors*.

The combined square matrix is invertible. The inverse is the transpose (the transpose of downsampling is upsampling $(\downarrow 2)^T = (\uparrow 2)$) - Chapter 3 [83]. Upsampling places zeros

into the odd-numbered components, downsampling removes odd-numbered components:

$$\begin{bmatrix} L \\ B \end{bmatrix}^{-1} = \begin{bmatrix} L^T & B^T \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & & -1 & & \\ 1 & & 1 & & \\ & 1 & & -1 & \\ & 1 & & 1 & \\ & & . & & . \\ & & . & & . \end{bmatrix}$$

The second matrix $[L^T\ B^T]$ represents the *synthesis bank*. this is an orthogonal filter bank, because $inverse = transpose$. The synthesis bank is the transpose of the analysis bank. When one follows the other we have perfect reconstruction

$$\begin{bmatrix} L^T & B^T \end{bmatrix} \begin{bmatrix} L \\ B \end{bmatrix} = L^T\, L + B^T\, B = I$$

Perfect reconstruction is an important property of wavelet filter banks and as shown the *Haar filter bank* has orthogonal filters which usually have the following properties:

- Coefficients of the high pass filter are an alternating flip of the lowpass filter

$$g(k) = (-1)^k\, h(N-k)$$

- The filters are not symmetric, h and g have an even length
- The synthesis filters are mirror filters to analysis filters

$$H = (h_0,\quad h_1,\ h_2,\quad h_3)\quad H^T = (\quad h_3,\ h_2,\quad h_1,\ h_0)$$

$$G = (h_3,\ -h_2,\ h_1,\ -h_0)\quad G^T = (\ -h_0,\ h_1,\ -h_2,\ h_3)$$

  In other words synthesis filters are transposes of analysis filters

Wavelet filters have finite length and this a property which makes them perform local analysis, or the examination of a localized area of a larger signal.

Multi-resolution Analysis or Multi-level Decomposition

Termed multilevel decomposition, this process can be repeated, with successive approximations (the output of the low-pass filter in the first bank) being decomposed in turn, so that one signal is broken down into a number of components. This is called the Mallat algorithm or Mallat-tree decomposition.

A three-level decomposition is shown in Fig. 3.12. In this illustration, $a3$ represents the approximation coefficients, while $d3$, $d2$ and $d1$ represent the detail coefficients resulting from the three-level decomposition. After each decomposition, we employ decimation by two to remove every other sample and, therefore, reduce the amount of data present.

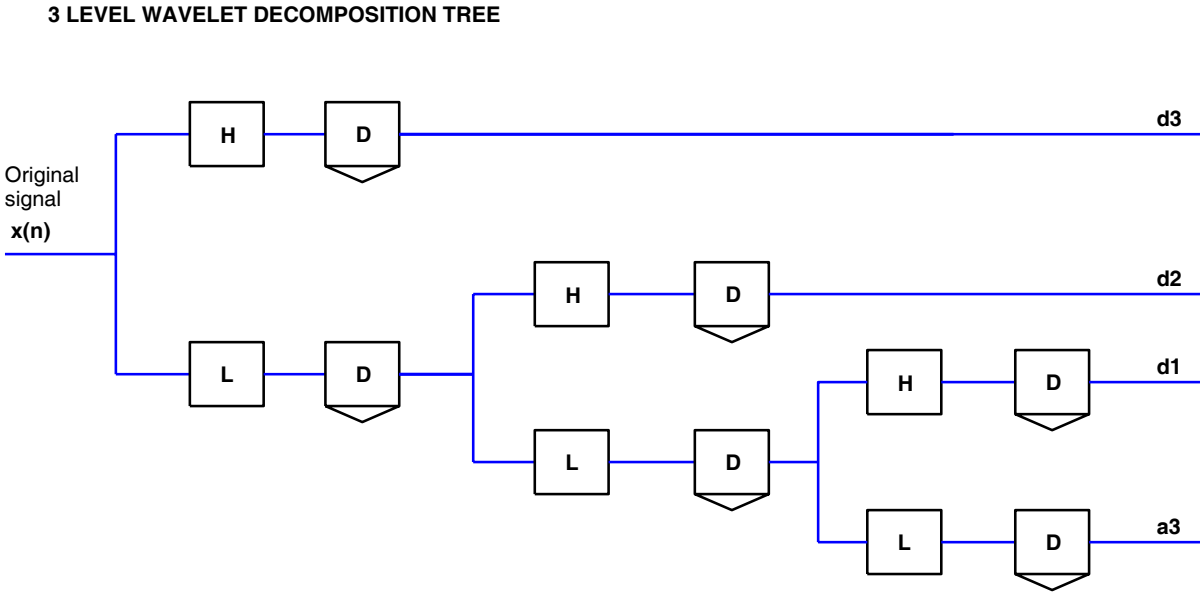


FIGURE 3.12. Multi-level decomposition

An example of a simulated signal decomposition into three levels using a Daubechies wavelet function is illustrated in Fig. 3.13. The scalogram of the resulting decomposition is also shown.

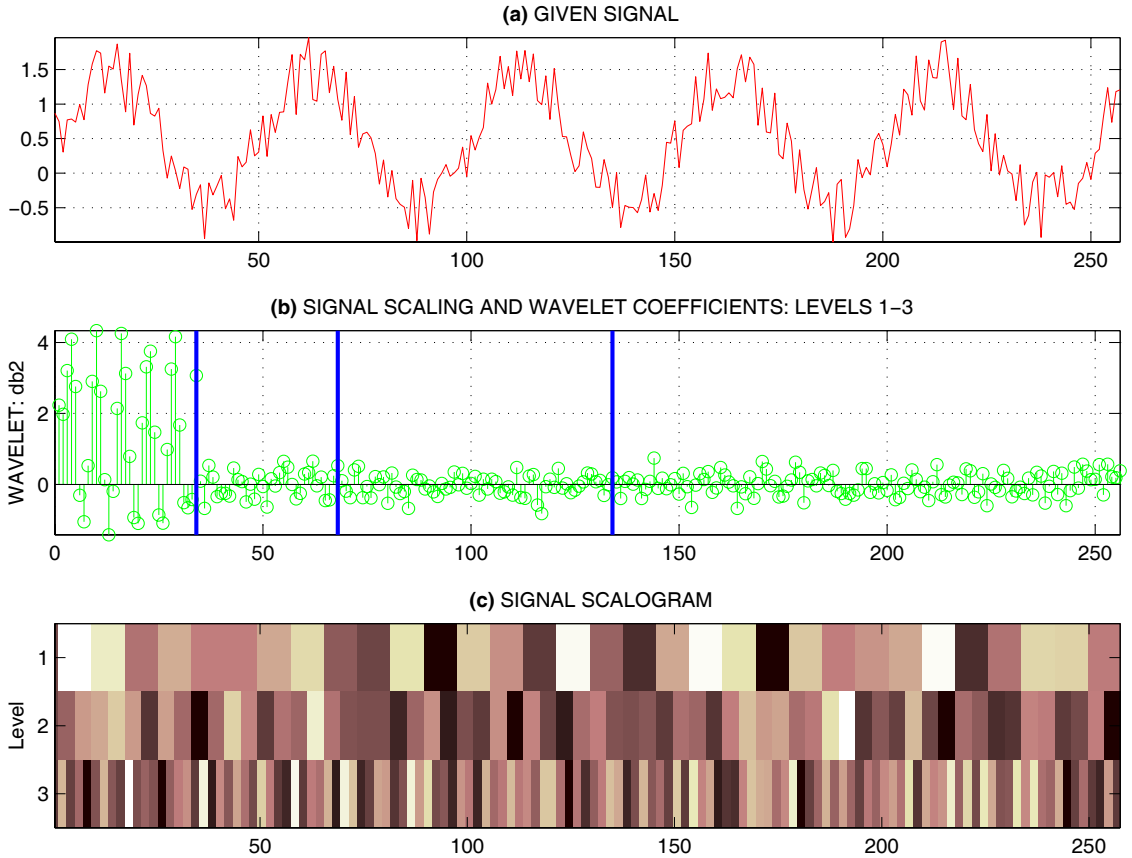


FIGURE 3.13. Wavelet decomposition of (a) simulated sinusoidal signal, (b) the wavelet coefficients, and (c) the resulting signal scalogram

Dilation Equation

The construction of wavelets [82] begins with the scaling function $\Phi(x)$. The dilation equation or refinement equation connects $\Phi(x)$ to translates of $\Phi(2x)$

$$\Phi(x) = \sum_{k=0}^{N-1} c_k \, \Phi(2x - k) \tag{3.35}$$

This is the *dilation equation*, it is a two scale equation involving x and $2x$ where x represents the time variable. For example the coefficients for Haar are $c_0 = c_1 = 1$, the box function is the sum of two half-width boxes. Then for the wavelet function $W(x)$ it is a combination of the same translates. The coefficients for Haar wavelet $W(x) = \Phi(2x) - \Phi(2x-1)$ are 1 and -1. It can be easily seen that $W(x)$ uses the same coefficients as $\Phi(x)$ but in reverse order and with alternating signs,

$$W(x) = \sum_{k=0}^{N-1} (-1)^k c_k \, \Phi(2x - k) \tag{3.36}$$

This construction makes $W(x)$ orthogonal to $\Phi(x)$ and its translates. The key is that every vector $c_0,\ c_1,\ c_2,\ c_3$ is automatically orthogonal to $c_0,\ -c_2,\ c_1,\ -c_0$. Then a question arises of how to solve the dilation equation. Two principal methods exist one is by Fourier transform and the other is by matrix products [82].

In order to solve the basic recursion equation Eq. (3.35), an iterative algorithm has been proposed that will generate the successive approximations to $\Phi(x)$. If the algorithm converges to a fixed point then that fixed point is a solution to Eq. (3.35), the iterations are defined by

$$\Phi^{(k+1)}(x) = \sum_{k=0}^{N-1} c_k \, \Phi^{(k)}(2x - k) \tag{3.37}$$

for the k^{th} iteration where an initial $\Phi^{(0)}(x)$ must be given. Transforming the dilation equation into frequency domain [83] (Chapter 6) instead of x and $2x$ the transform involves ω and $\omega/2$. The dilation equation becomes

$$\hat{\Phi}(\omega) = H\left(\frac{\omega}{2}\right) \hat{\Phi}\left(\frac{\omega}{2}\right)$$

Iterating this equation it connects $\omega/2$ to $\omega/4$:

$$\hat{\Phi}(\omega) = H\left(\frac{\omega}{2}\right) \left[H\left(\frac{\omega}{4}\right) \hat{\Phi}\left(\frac{\omega}{4}\right)\right]$$

After N-iterations, this becomes

$$\hat{\Phi}(\omega) = H\left(\frac{\omega}{2}\right) H\left(\frac{\omega}{4}\right) \cdots H\left(\frac{\omega}{2^N}\right) \hat{\Phi}\left(\frac{\omega}{2^N}\right)$$

In the limits as $N \rightarrow \infty$, we have a formula for the solution $\hat{\Phi}(\omega)$. Note that $(\frac{\omega}{2^N})$ is approaching zero and $\hat{\phi}(0) = \int \Phi(x)\ dx$ is the area under the graph of $\Phi(x)$ which is equal to one. The formal limit of the iteration leads to the infinite product for $\hat{\Phi}$:

$$\hat{\Phi}(\omega) = \prod_{j=1}^{\infty} H\left(\frac{\omega}{2^j}\right)$$

Fig. 3.14 shows how the fourth order Daubechies scaling function with four coefficients is approximated by applying successive iterations to an initial box funtion. This is achieved by using Eq. (3.37) which converges to a reliably $\Phi(x)$ after six iteration steps. From this scaling function, the wavelet can also be generated using Eq. (3.36).

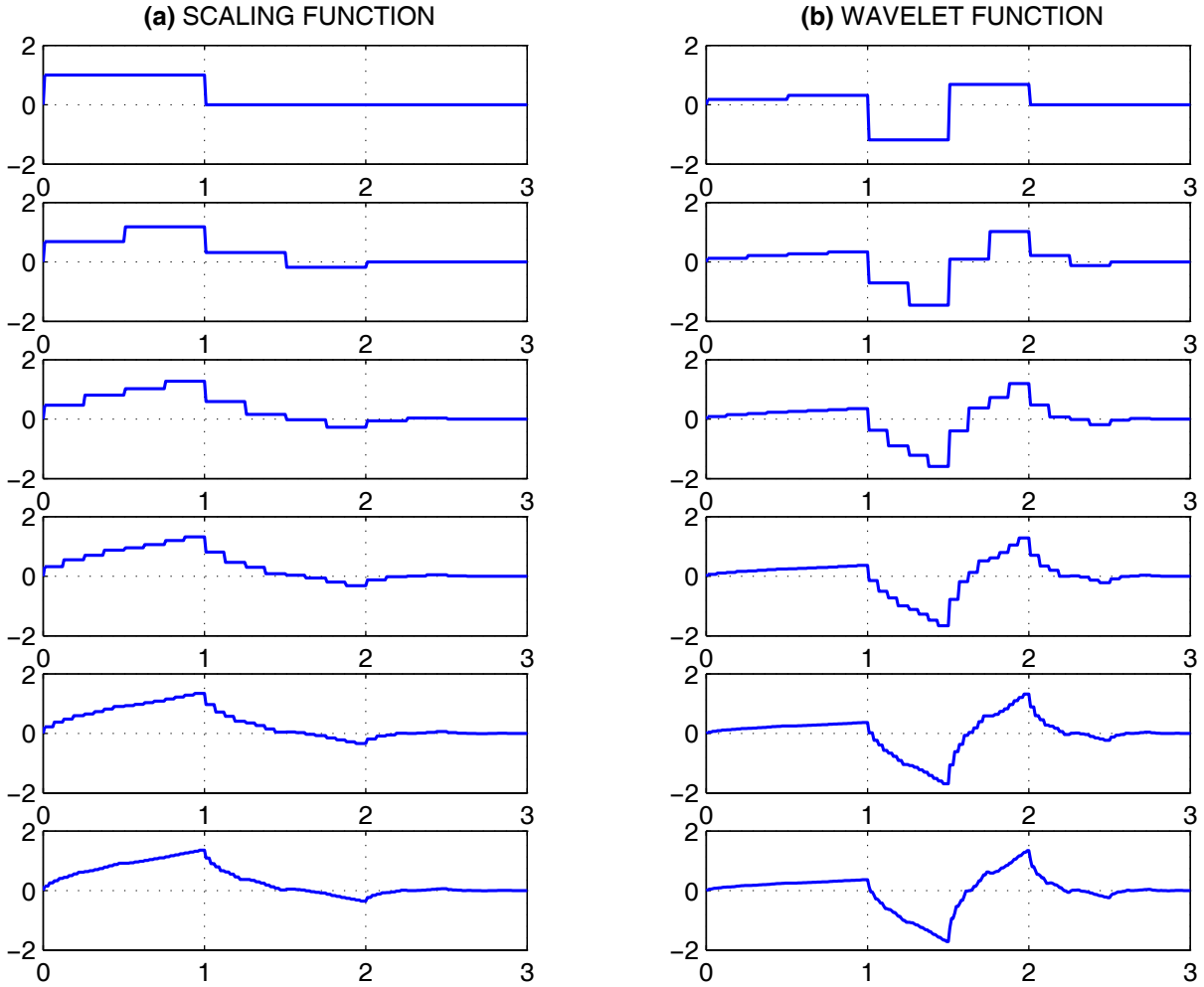


FIGURE 3.14. Recurrent solution of the dilation equation for (a) the scaling function and (b) the wavelet function

3.2.2 Two-dimensional Discrete Wavelet Transform

Digital images are 2-D signals that require a two-dimensional wavelet transform. The 2-D DWT analyzes an image across rows and columns in such a way as to separate horizontal, vertical and diagonal details. In the first stage [91] the rows of an $N \times N$ are filtered using a high pass and low pass filters. This filtering is done using 1-D convolution with the coefficients $h_0(k)$ and $h_1(k)$, since each row of the image is a one-dimensional signal. This is followed by downsampling with a factor of 2 which removes every odd-numbered sample in the filtered result this has the effect of removing every other column of the $N \times N$ block giving an $N \times (N/2)$ image.

In the second stage 1-D convolution with $h_0(k)$ and $h_1(k)$ is applied to the columns of the filtered image. Downsampling removes each odd-numbered sample in each column of the now twice-filtered result which results in the removal of every other row. Each of the branches in the tree is shown in the Fig. 3.15 therefore produces an $(N/2) \times (N/2)$ subimage. This leads at each level to 4 different subbands $HH,\ HL,\ LH$ and LL . The LL is filtered again to get the next level representation, Fig. 3.15 summarizes the transform for a one level decomposition.

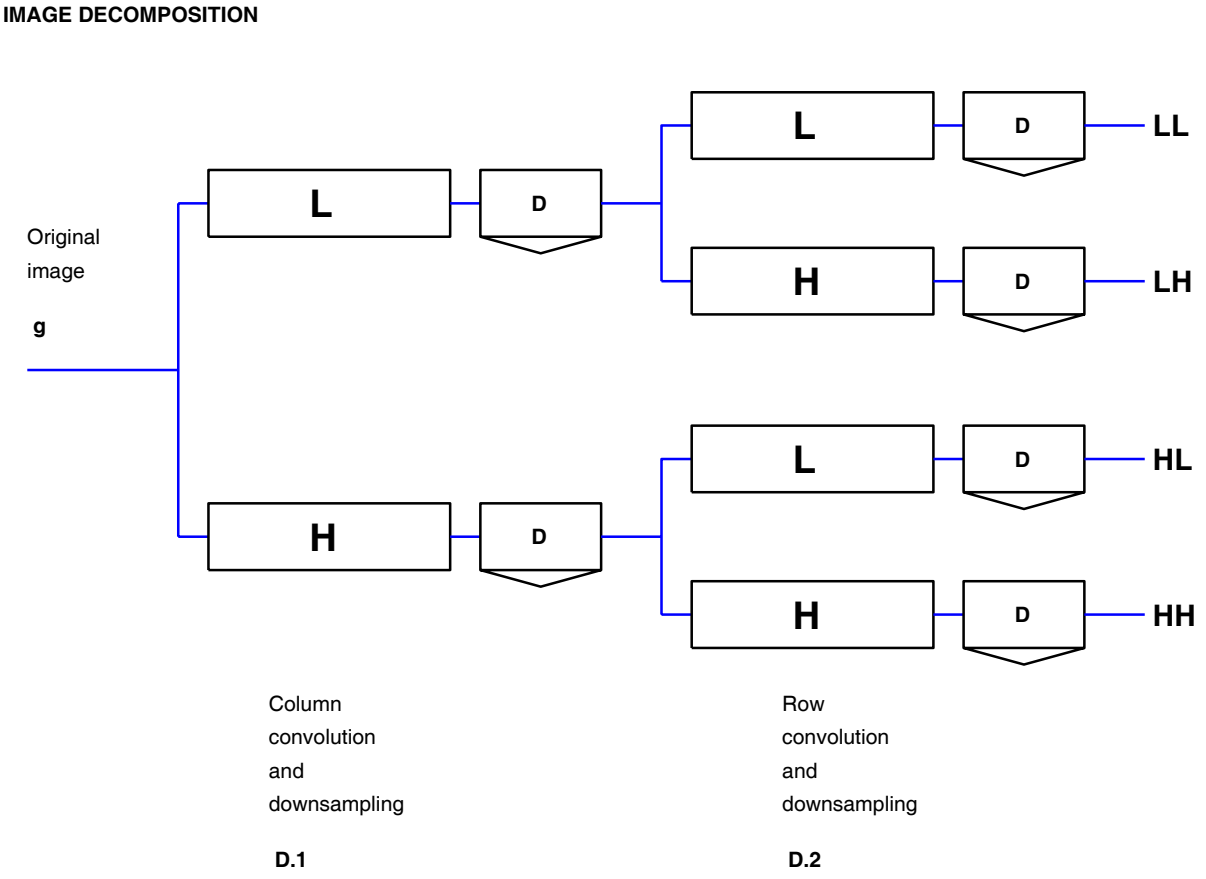


FIGURE 3.15. A one-level two-dimensional DWT decomposition

The role of each branch in the analysis of an $N \times N$ image can be explained as follows:

- Initial low pass filtering of the rows blurs the image values along each row followed by low pass filtering along the columns which result in a low pass *approximation* of the whole image.

- Low pass filtering of the rows followed by high pass filtering of the columns highlights the changes that occur between the rows - *horizontal details*

- Initial high pass filtering of the original rows of the image highlights the changes between elements in any given low. Subsequent low pass filtering of the columns blurs the changes that may occur between the rows thus providing the *vertical details*

- High pass filtering of the rows followed by high pass filtering of the columns only changes that are neither horizontal are emphasized. This sequence gives the *diagonal details* of the original image.

The $(N/2) \times (N/2)$ low pass approximation is subjected to the same process as the original image resulting in four $(N/4) \times (N/4)$ subimage: a low pass part, horizontal, vertical and diagonal details. Analysis can continue until the subimages obtained contain a single pixel only. Fig. 3.16 shows a two level decomposition of a DWT using the $db4$ wavelet on a 128×128 MR image.

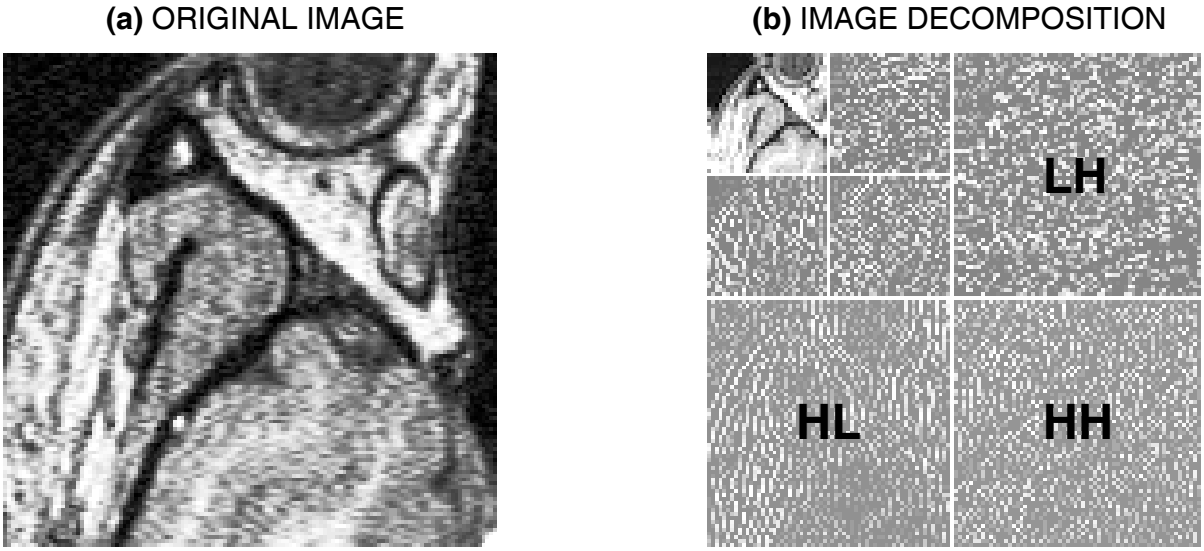


FIGURE 3.16. DWT image analysis presenting (a) the original image and (b) its decomposition into the second level

Reconstruction

To reconstruct the image from its 2-D DWT subimages (LH, HL,HH) the details are recombined with the low pass approximation using upsampling and convolution as shown in Fig. 3.17. Upsampling refers to the insertion of of a zero row after each existing row or a zero column after each existing column. In the first stage the columns of the upsampled subimages are convolved with the impulse responses $h_0^T(k)$ and $h_1^T(k)$ and in the second stage the rows of the upsampled sums are convolved with the same impulse responses.

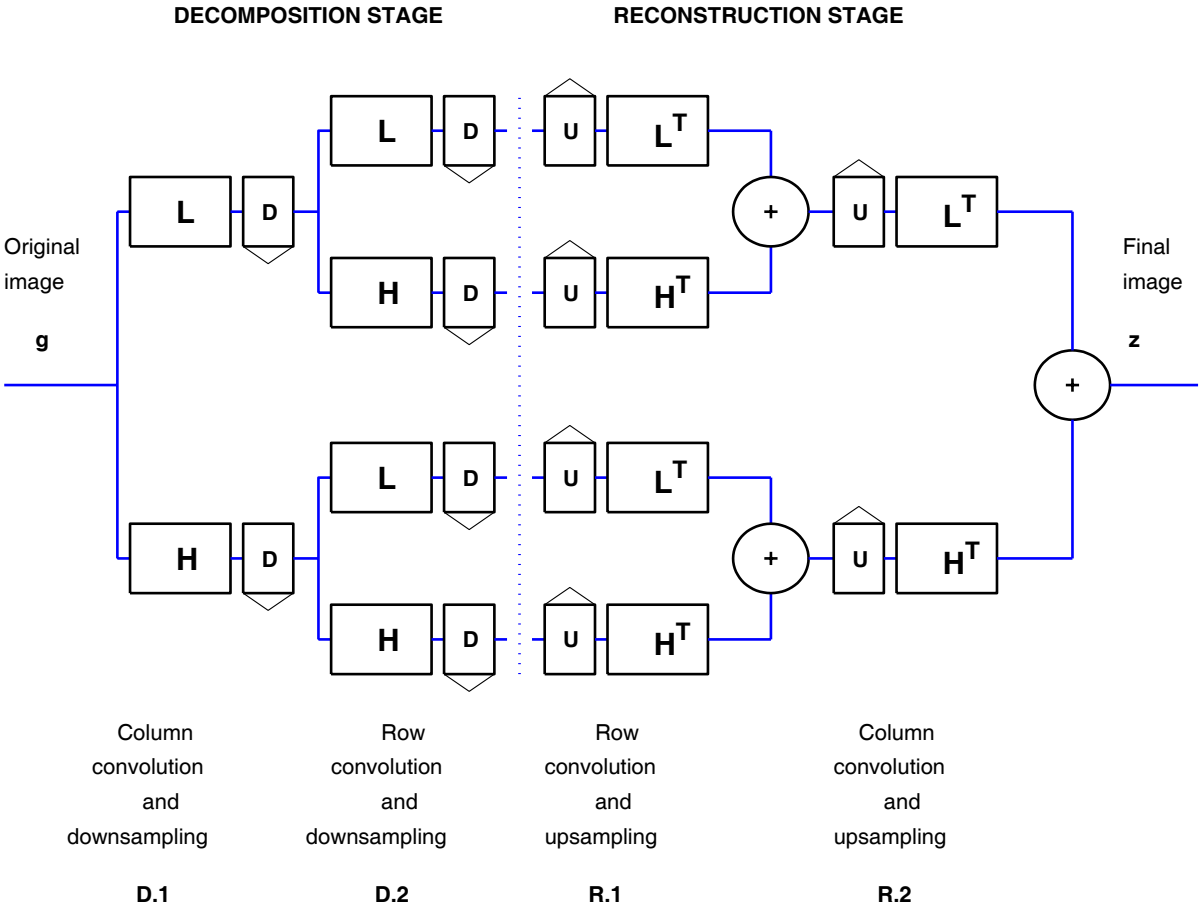


FIGURE 3.17. Two-dimensional DWT reconstruction

3.2.3 Three-dimensional Discrete Wavelet Transform

Discrete Wavelet Transform (DWT) [66] is a separable, sub-band transform. Although nonseparable wavelets can also be used for multidimensional signals, such filters are much harder to design than are separable filters. As a result, their use has been limited in image processing applications. 3-D wavelets can be constructed as separable products of 1-D wavelets by successively applying a 1-D analyzing wavelet in three spatial directions (x, y, z).

Fig. 3.18 shows a separable 3-D decomposition [42] of an image volume. The volume $F(x, y, z)$ is firstly filtered along the x-dimension, resulting in a low-pass image $L(x, y, z)$ and a high-pass image $H(x, y, z)$. Since the size of L and H along the x-dimension is now half that of $F(x, y, z)$, down-sampling of the filtered volume in the x-dimension by two can be done without loss of information. The down-sampling is done by dropping each odd filtered value. Both L and H are then filtered along the y-dimension, resulting in four decomposed sub-volumes: LL, LH, HL and HH. Once again, we can downsample the sub-volumes by two, this time along the z-dimension. Then each of these four sub-volumes are then filtered along the z-dimension, resulting in eight sub-volumes: *LLL, LLH, LHL, LHH, HLL, HLH, HHL* and *HHH* (Fig 3.18).

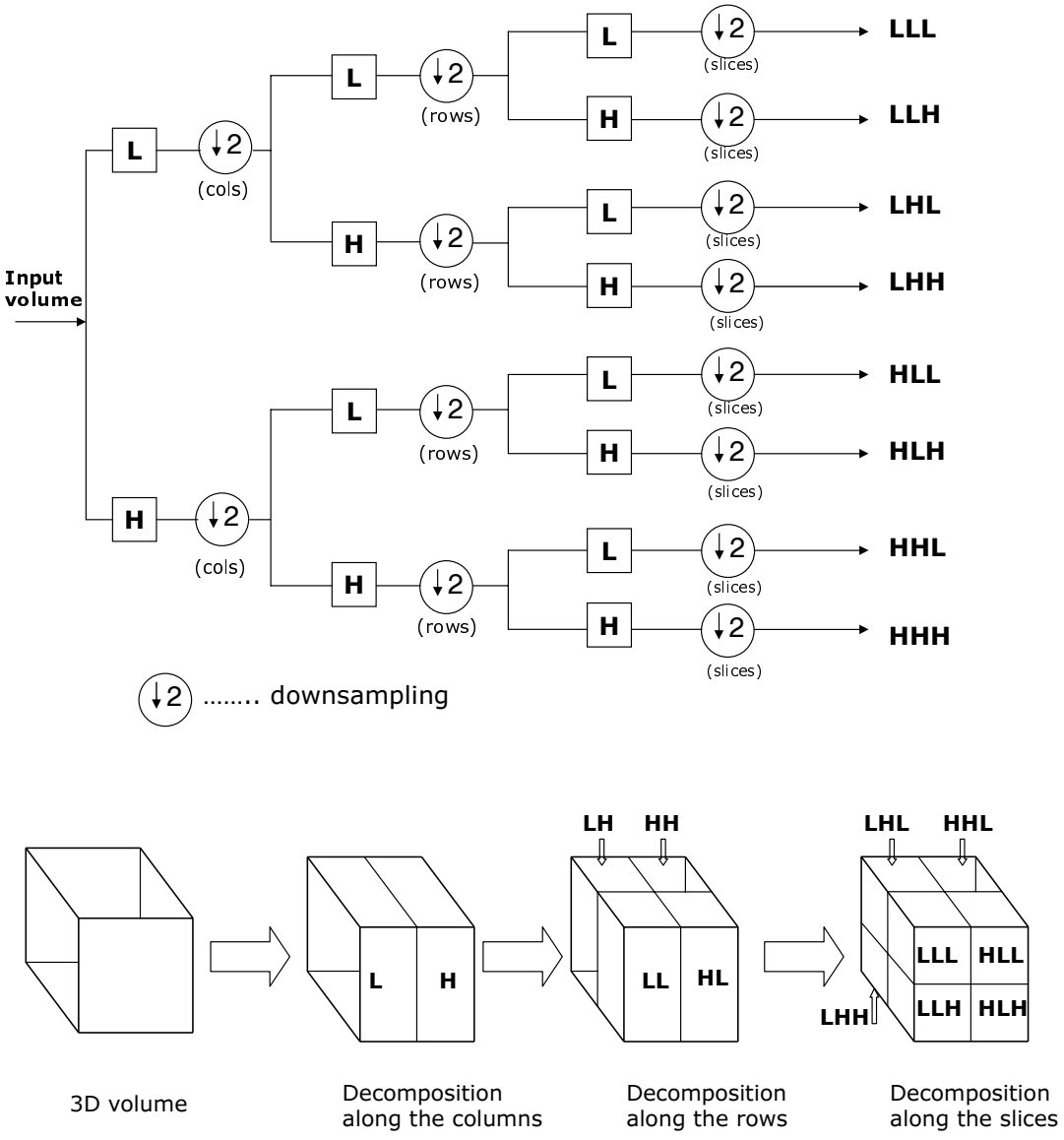


FIGURE 3.18. Three-dimensional DWT decomposition

Reconstruction

After modifying the coefficients we can apply the inverse transform by convolving with the respective low-pass and high-pass synthesis filters as described in [83] and [92]. The reconstruction for a one-level three dimensional discrete wavelet transform (3-D DWT) is illustrated by the following resulting structure which is presented in Fig. 3.19.

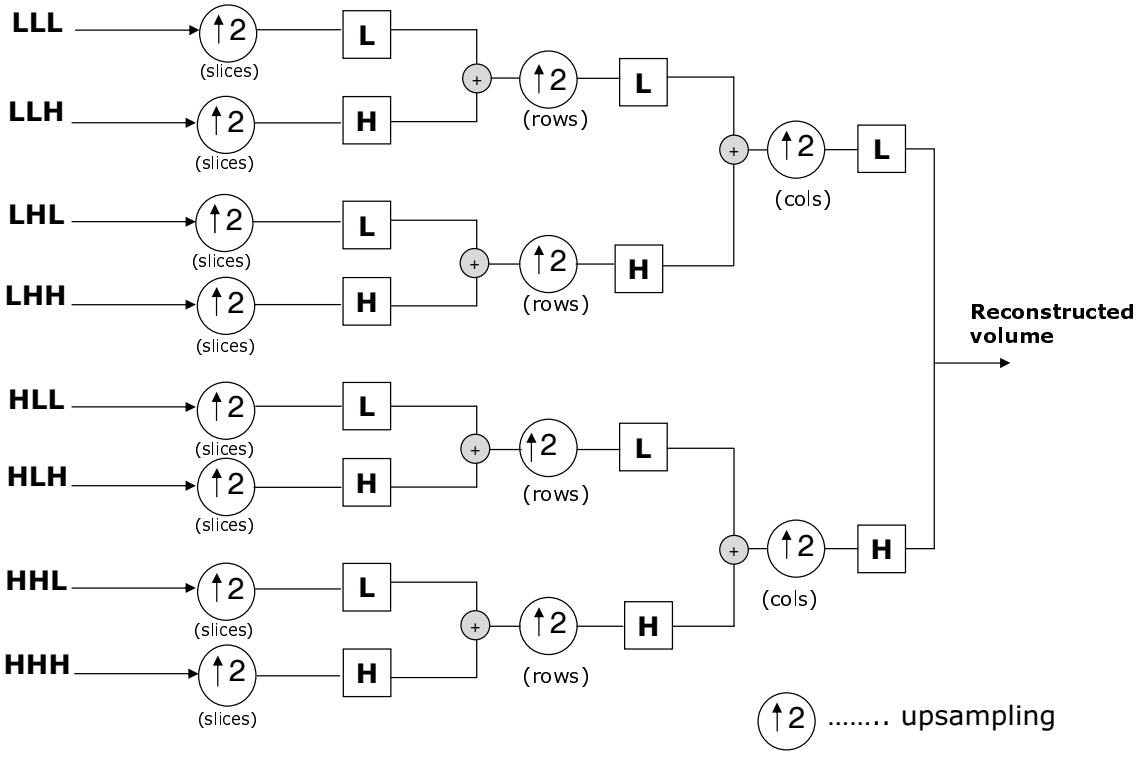


FIGURE 3.19. Three-dimensional DWT reconstruction

4
Discrete Wavelet Transform Applications

Wavelet transform plays an increasingly important role in the medical signal analysis and biomedical image processing. Wavelet transform analysis has been applied to a wide variety of biomedical signals including: the ECG, EEG, MR images, heart sounds, breath sounds, respiratory patterns, blood pressure trends, DNA sequences and other biomedical applications. Various useful applications of wavelet transform decomposition and reconstruction include:

- de-noising - decomposing an image into a low frequency approximation image and a set of high frequency detailed images, performing thresholding and then reconstructing the image from the thresholded coefficients [54].
- resolution enhancement - resampling image enhancement can significantly improve the quality of the images
- reconstruction of missing regions or objects by using iterative wavelet transform
- image compression - digital images and digital video require tremendous amounts of storage. Compression can significantly reduce both storage needs and transmission times e.g. JPEG2000 and FBI fingerprint compression is built upon wavelets.
- edge detection - The spatial features of the image indicating the edges obtained by multi-resolution wavelet transform of the high frequency details.
- feature extraction
- 3-D object processing - analysis and processing of 3-D medical data sets.

Noise

Noise in MR images consists of random signals that do not come from the tissues but from other sources in the machine and environment that do not contribute to the tissue differentiation. The noise of an image gives it a grainy appearance. Mainly the noise is evenly spread and more uniform. There are two ways to corrupt an image with noise. A noise image can be simply added to the original image (additive noise), or the noise values can be multiplied by the original intensities (multiplicative noise).

Image independent noise is often described by an additive noise model, where the noise image $f(i,j)$ is the sum of the true image $s(i,j)$ and the noise $n(i,j)$:

$$f(i,j) = s(i,j) + n(i,j)$$

In many cases, additive noise is evenly distributed over the frequency domain (i.e. white noise), whereas an image contains mostly low frequency information. Hence, the noise is dominant for high frequencies and its effects can be reduced using some kind of lowpass filter.

The classification of noise is based upon the shape of the probability density function or histogram for the discrete case of the noise. The first type of noise to be presented is uniform noise. Fig. 4.1 shows a histogram of a uniform noise distribution. This can be modelled as

$$p(n) = \begin{cases} \frac{1}{2\sigma\sqrt{3}} & \text{for } |n| \leq \sigma\sqrt{3} \\ 0 & \text{else} \end{cases} \tag{4.1}$$

A uniform distribution shows that there are about the same number of counts for each value bin.

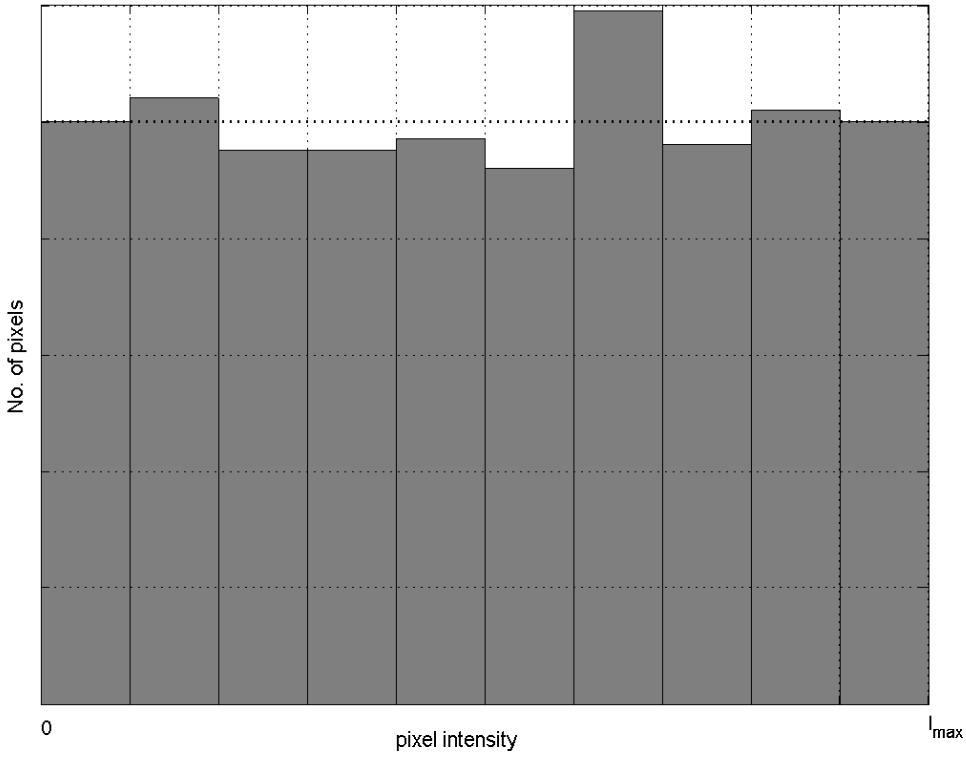


FIGURE 4.1. Uniform distribution histogram for 1000 values generated using the *rand* MATLAB function

The most common type of noise and the one that is mostly encountered is the Gaussian noise. Gaussian distribution is assumed to be symmetrical about a mean of zero. The

standard deviation (σ) is a measure of the amount of spread around the central peak. At low standard deviations, the central bins are concentrated near the mean and the peak is very tall and sharp. At high deviations the peak is lower and values are more evenly distributed to outlying bins. The probability of larger and larger deviations can be seen to decrease rapidly. Its probability density function (pdf) is defined according to the equation stated below (Eq. 4.2):

$$p(n) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{n^2}{2\sigma^2}} \tag{4.2}$$

The frequency spectrum of such a signal is flat, that is, it has equal values at all frequencies. Gaussian noise is evenly distributed across the entire range of frequencies. A histogram of Gaussian noise is shown in Fig. 4.2.

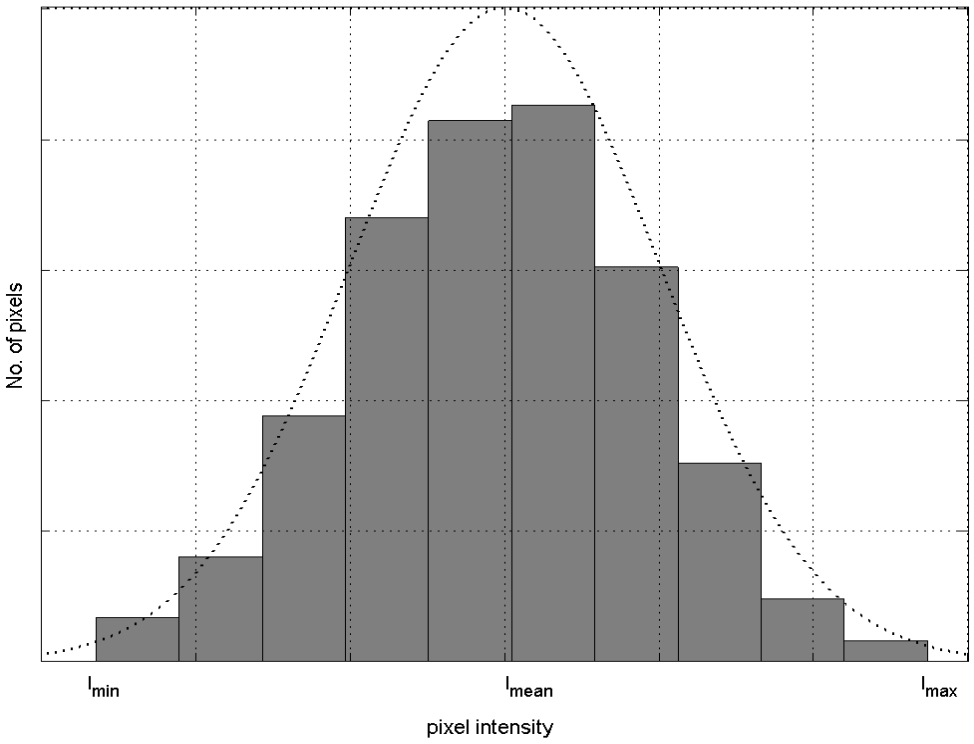


FIGURE 4.2. Gaussian distribution histogram for 1000 values generated by using the MATLAB software

Another common form of noise is data drop-out noise (commonly referred to as intensity spikes, speckle or salt and pepper noise). Here, the noise is caused by errors in the data transmission. The corrupted pixels are either set to the maximum value (which looks like snow in the image) or have single bits flipped over. In some cases, single pixels are set alternatively to zero or to the maximum value, giving the image a 'salt and pepper' like appearance. Unaffected pixels always remain unchanged. The noise is usually quantified

by the percentage of pixels which are corrupted. Salt-and-pepper noise takes on two gray levels, I_p and I_s. Fig. 4.3 shows a histogram for salt-and-pepper noise

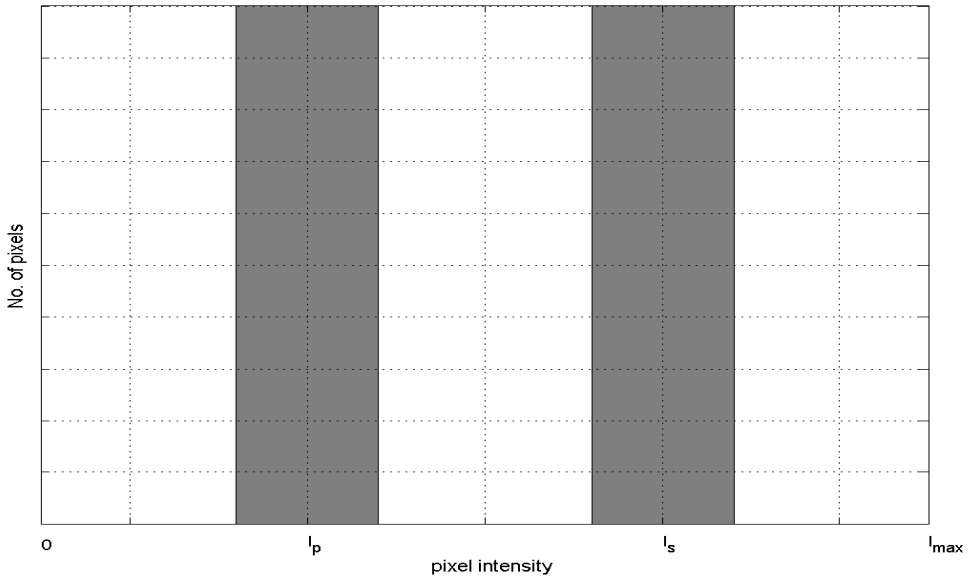


FIGURE 4.3. Salt and pepper distribution histogram

4.1 Image De-noising

The reduction of noise present in images is an important aspect of image processing. De-noising is a procedure to recover a signal that has been corrupted by noise. After discrete wavelet decomposition the resulting coefficients can be modified to eliminate undesirable signal components. To implement wavelet thresholding a *wavelet shrinkage* method for de-noising the image has been verified. The proposed algorithm to be used is summarized in Algorithm 1 and it consists of the following steps:

Algorithm 1: Wavelet image de-noising

- Choice of a wavelet (e.g. Haar, symmlet, etc) and number of levels or scales for the decomposition. Computation of the forward wavelet transform of the noisy image.
- Estimation of a threshold
- Choice of a shrinkage rule and application of the threshold to the detail coefficients. This can be accomplished by ***hard*** (Eq. (4.3)) or ***soft*** thresholding (Eq. (4.4))
- Application of the inverse transform (wavelet reconstruction) using the modified (thresholded) coefficients

4.1.1 Thresholding

Thresholding is a technique used for signal and image de-noising. The shrinkage rule define how we apply the threshold. There are two main approaches which are:

- Hard thresholding (Fig. 4.4(b)) deletes all coefficients that are smaller than the threshold λ and keeps the others unchanged. The hard thresholding is defined as follows:

$$\overline{c}_h(k) = \begin{cases} \text{sign } c(k)\ (|c(k)|) & \text{if } |c(k)| > \lambda \\ 0 & \text{if } |c(k)| \leq \lambda \end{cases} \tag{4.3}$$

  where λ is the threshold and the coefficients that are above the threshold are the only ones to be considered. The coefficients whose absolute values are lower than the threshold are set to zero.

- Soft thresholding (Fig. 4.4(c)) deletes the coefficients under the threshold, but scales the ones that are left. The general soft shrinkage rule is defined by:

$$\overline{c}_s(k) = \begin{cases} \text{sign } c(k)\ (|c(k)| - \lambda) & \text{if } |c(k)| > \lambda \\ 0 & \text{if } |c(k)| \leq \lambda \end{cases} \tag{4.4}$$

For an illustration of what has been described above a linear signal is thresholded according to the methods described using a threshold λ of 0.5 (Fig. 4.4).

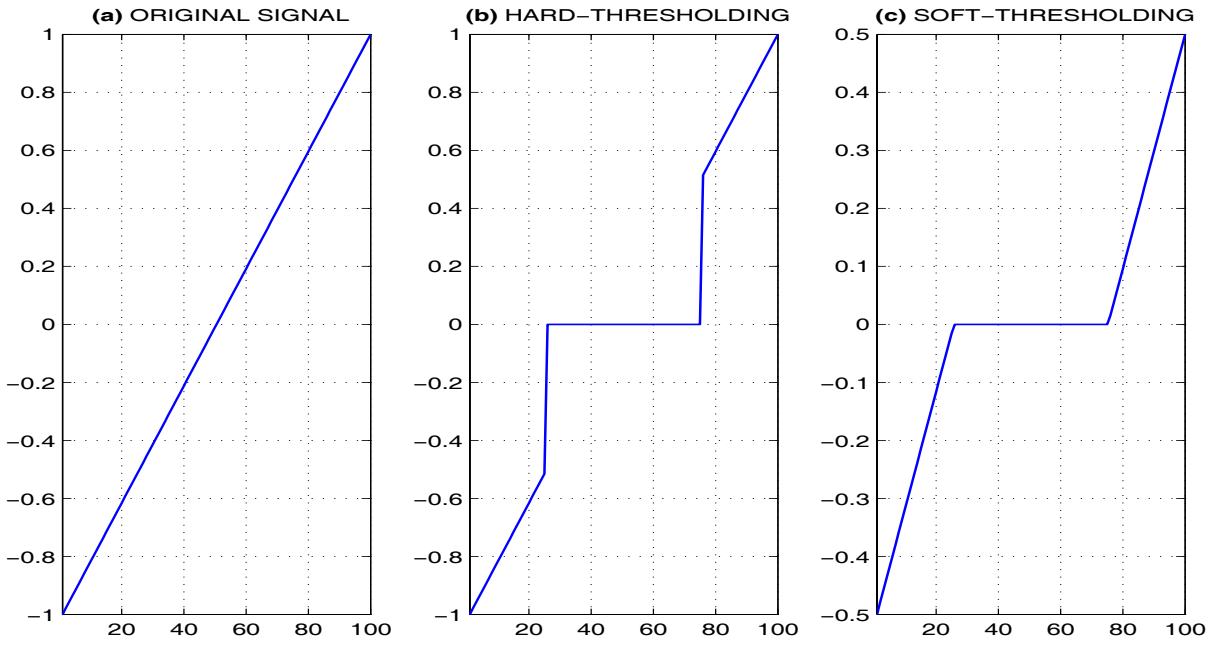


FIGURE 4.4. An example of (a) linear signal thresholded using, (b) hard-thresholding, and (c) soft-thresholding

Global Threshold

The global threshold method derived by Donoho is given by Eq. (4.5) has a universal threshold:

$$\lambda \;=\; \sigma\sqrt{2\log(N)} \tag{4.5}$$

where N is the size of the coefficient arrays and σ^2 is the noise variance of the signal samples.

Level Dependent Threshold

Level dependent thresholding method is done by using Eq. (4.6). Estimation of the noise standard deviation σ_k is done by using the robust median estimator in the highest sub-band of the wavelet transform

$$\lambda_k \;=\; \sigma_k\sqrt{2\log(N)} \tag{4.6}$$

where the scaled MAD noise estimator is computed by:

$$\sigma_k = \frac{MAD_k}{0.6745} = \frac{(median(|\omega_i|))_k}{0.6745}$$

where MAD is the median absolute deviation of the magnitudes of all the coefficients at the finest decomposition scale and ω_i are the coefficients for each given sub-band, the factor 0.6745 in the denominator rescales the numerator so that σ_k is also a suitable estimator. The threshold estimation method is repeated for each sub-band separately, because the sub-bands exhibit significantly different characteristics.

Optimal Threshold Estimation

Estimate the mean square error function to that compute the error of the output to minimize the function, the minimum MSE serves as a solution to the optimal threshold. A function of the threshold value which is minimized is defined in Eq. (4.7).

$$G(\lambda) = MSE(\lambda) = \frac{1}{N}||y - y_\lambda||^2 \tag{4.7}$$

If y_λ is the output of the threshold algorithm with a threshold value λ and y is the vector of the clean signal, the remaining noise on this result equals $e_\lambda = y_\lambda - y$. As the notation indicates, the MSE is a function of the treshold value λ. Find the optimal value of λ that minimizes $MSE(\lambda)$ and the convergence of the algorithm.

4.1.2 Measures of Image Quality

One of the issues of de-noising is the measure of the reconstruction error. In order to separate the noise and image components from a single observation of a degraded image it is necessary to assume or have knowledge about the statistical properties of the noise. To get the measure of the wavelet filter performance, the experimental results are evaluated according to three error criteria namely, the *mean square error* (MSE), the *mean absolute error* (MAE) and the *peak signal to noise ratio* (PSNR). For most quality assessment methods, the error criterion takes the form of a Minkowski norm [97] which is defined as follows:

$$E(\{e_{m,n}\}) = \left(\sum_m \sum_n |e_{m,n}|^\beta \right)^{\frac{1}{\beta}} \tag{4.8}$$

where $\{e_{m,n}\}$ is the error (difference) between the reference and de-noised image and β is a constant exponent typically chosen to lie between 1 and 4 for image error metrics.

The goal of de-noising is starting from a noisy image to produce the best possible estimate $\tilde{y}(m,n)$ of the original image $y(m,n)$. The measure of success in de-noising is usually an error measure $E(\hat{y}(m,n), y(m,n))$ between the original $y(m,n)$ and the estimate $\hat{y}(m,n)$.

The *mean square error* (MSE) function is commonly used because it has a simple mathematical structure that is easy to compute and it is differentiable implying that a minimum can be sought. For a discrete image signal $y(m,n)$ and its approximation (estimate) $\tilde{y}(m,n)$ where $m,n = 0,1,\ldots,N-1$ the MSE is defined as

$$MSE = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(y(m,n) - \tilde{y}(m,n) \right)^2 \tag{4.9}$$

where $y(m,n)$ and $\tilde{y}(m,n)$ represent the original image and the de-noised image respectively. The criterion *root mean squared error* (RMSE) is the square root of MSE, that means for RMSE $\beta = 2$ (Eq. (4.8)).

Another most common and simplest measures of image quality, is the the peak signal to noise ratio (PSNR) which is given by:

$$PSNR = 10 \log_{10} \left(\frac{I_{max}^2}{MSE} \right) \tag{4.10}$$

where I_{max} is the maximum intensity value, typical PSNR values range between 20 and 40. They are usually reported to two decimal points (e.g. 25.47). An improvement in of the PSNR magnitude will increase the visual appearance of the image. PSNR is typically expressed in decibels (dB). For comparison with the noisy image the greater the ratio, the

easier it is to identify and subsequently isolate and eliminate the source of noise. A PSNR of zero indicates that the desired signal is virtually indistinguishable from the unwanted noise. PSNR is a good measure for comparing restoration results for the same image, but between-image comparisons of PSNR are meaningless. One image with 20 dB PSNR may look much better than another image with 30 dB PSNR.

Another criterion measure include: *Mean of absolute error* (MAE) which is given by the Eq. (4.11)

$$MAE = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y(m,n) - \hat{y}(m,n)| \tag{4.11}$$

The goal of de-noising is to find an estimate image such that MAE is minimum. Other error measures such as the *maximum of absolute error* (MAX) and *median of absolute error* (MED) can all be derived from Eq. (4.8). The proposed error criteria described above can also be extended to three-dimensional image data and computed likewise.

4.2 Experimental Results

For our test experiments we have considered an additive noise with a uniform distribution which has been used to corrupt our simulated and real MR test image objects. Artificially adding noise to an image allows us to test and assess the performance of various wavelet functions. To reduce computational time a region of interest is cropped (extracted) for the de-noising (Fig. 4.5).

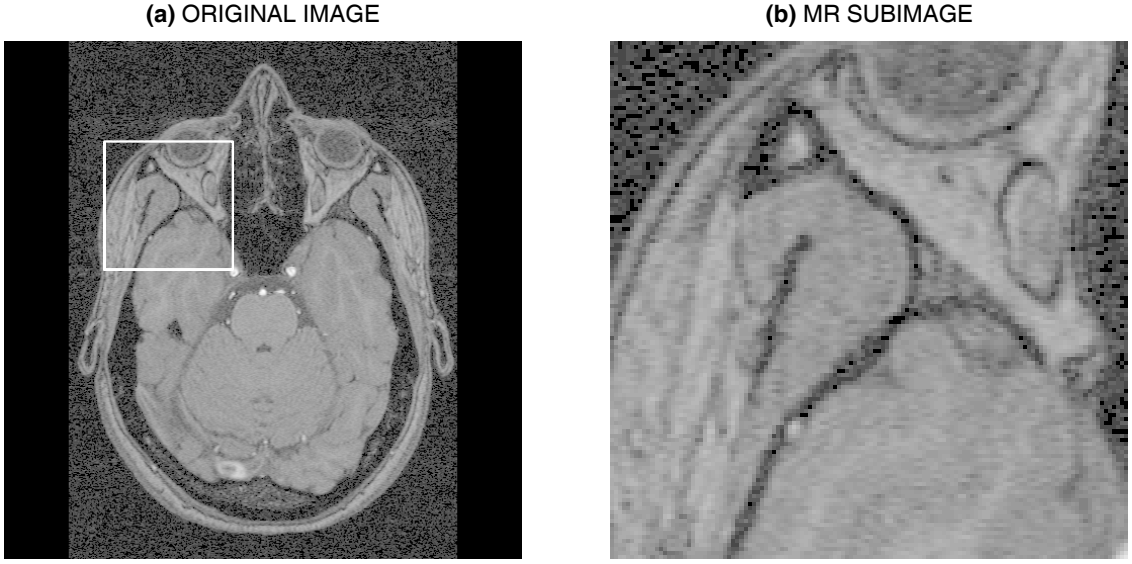


FIGURE 4.5. (a) Original MR image slice and (b) the chosen subimage for the de-noising application

Algorithm Implementation: We used MATLAB to implement the de-noising algorithm. Matlab has a wavelet toolbox and functions which are very convenient to do the DWT. A usual way to de-noise is to find a processed image such that it minimizes mean square error MSE, MAE and increases the value of the PSNR.

4.2.1 Results from Simulated Realistic Data - 2-D

A different number of wavelets have been chosen and are applied both for one-level and two-level decomposition. The original and the noisy simulated images are shown in Fig. 4.6.

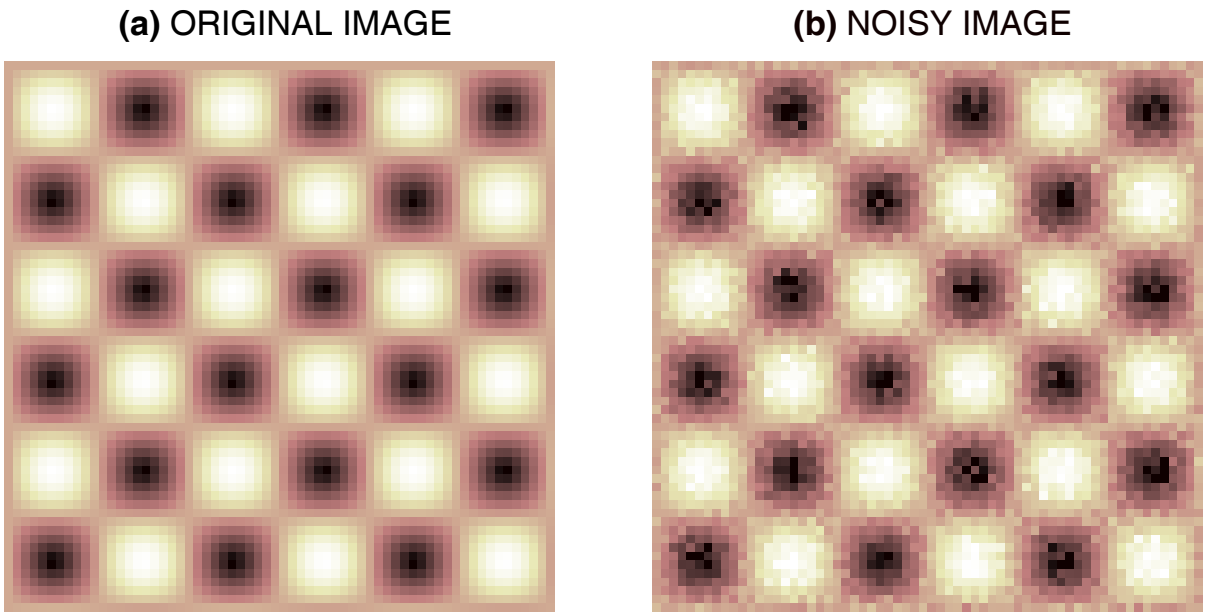


FIGURE 4.6. (a) Original image and (b) noisy image

For comparison of the five different wavelet functions, the quantitative de-noising results of the simulated images obtained by using global, level-dependent and optimal thresholding are shown in Tabs. 4.1, 4.2 and 4.3 respectively. The MSE, MAE, PSNR error criteria are the ones which have been used to assess the performance of the wavelet functions. Their numerical results are summarized in the tables.

The threshold values shown in red are obtained using global (Fig. 4.7), level-dependent (Fig. 4.8) and optimum thresholding (Fig. 4.9) for each decomposition level, which gave us good noise suppression.

The visual quality of the de-noised images obtained by using various wavelet functions at different levels are shown in Figs. 4.10 and 4.11. From the comparison results it can be observed, that the *db*4 wavelet decomposition gives greatly improved de-noising results.

TABLE 4.1. Qualitative analysis (SIMULATED image) - Global thresholding

Type of wavelet	Level 1			Level 2		
	MSE	**MAE**	**PSNR** [dB]	**MSE**	**MAE**	**PSNR** [dB]
Noisy image	0.01076	0.51493	19.68	0.01076	0.51493	19.68
Haar	0.00387	0.05231	24.12	0.01055	0.08362	19.77
db2	0.00116	0.02493	29.35	0.00320	0.04206	24.96
db4	**0.00059**	**0.01914**	**32.28**	**0.00067**	**0.01795**	**31.71**
sym2	0.00116	0.02493	29.35	0.00320	0.04206	24.96
sym4	0.00071	0.01884	31.46	0.00076	0.01667	31.22
bior1.1	0.00387	0.05231	24.12	0.01055	0.08362	19.77

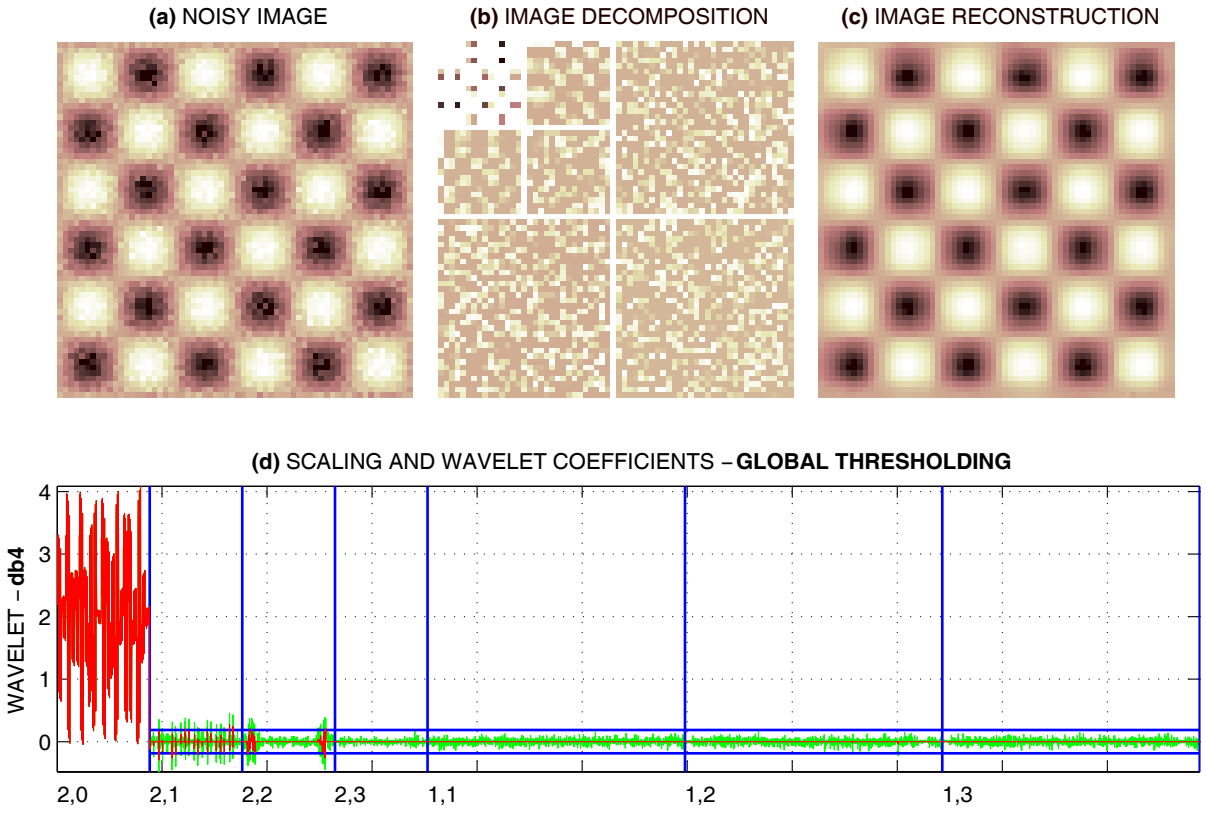


FIGURE 4.7. Discrete wavelet transform of the (a) noisy image using a *db*4 wavelet function, (b) the approximation image (low-frequency component) is in the top-left corner of the transform display, the other subimages contain the high frequency details, (d) global thresholding of the subband coefficients, and (c) shows the de-noised image, obtained by taking the inverse thresholded coefficients

TABLE 4.2. Qualitative analysis (SIMULATED image) - Level-dependent thresholding

Type of wavelet	Level 1			Level 2		
	MSE	**MAE**	**PSNR** [dB]	**MSE**	**MAE**	**PSNR** [dB]
Noisy image	0.01076	0.51493	19.68	0.01076	0.51493	19.68
Haar	0.00389	0.05242	24.11	0.01293	0.09437	18.89
db2	0.00116	0.02492	29.35	0.00320	0.04209	24.95
db4	**0.00059**	**0.01910**	**32.32**	**0.00059**	**0.01733**	**32.28**
sym2	0.00116	0.02492	29.35	0.00320	0.04209	24.95
sym4	0.00071	0.01879	31.46	0.00071	0.01630	31.46
bior1.1	0.00389	0.05242	24.11	0.01293	0.09437	18.89

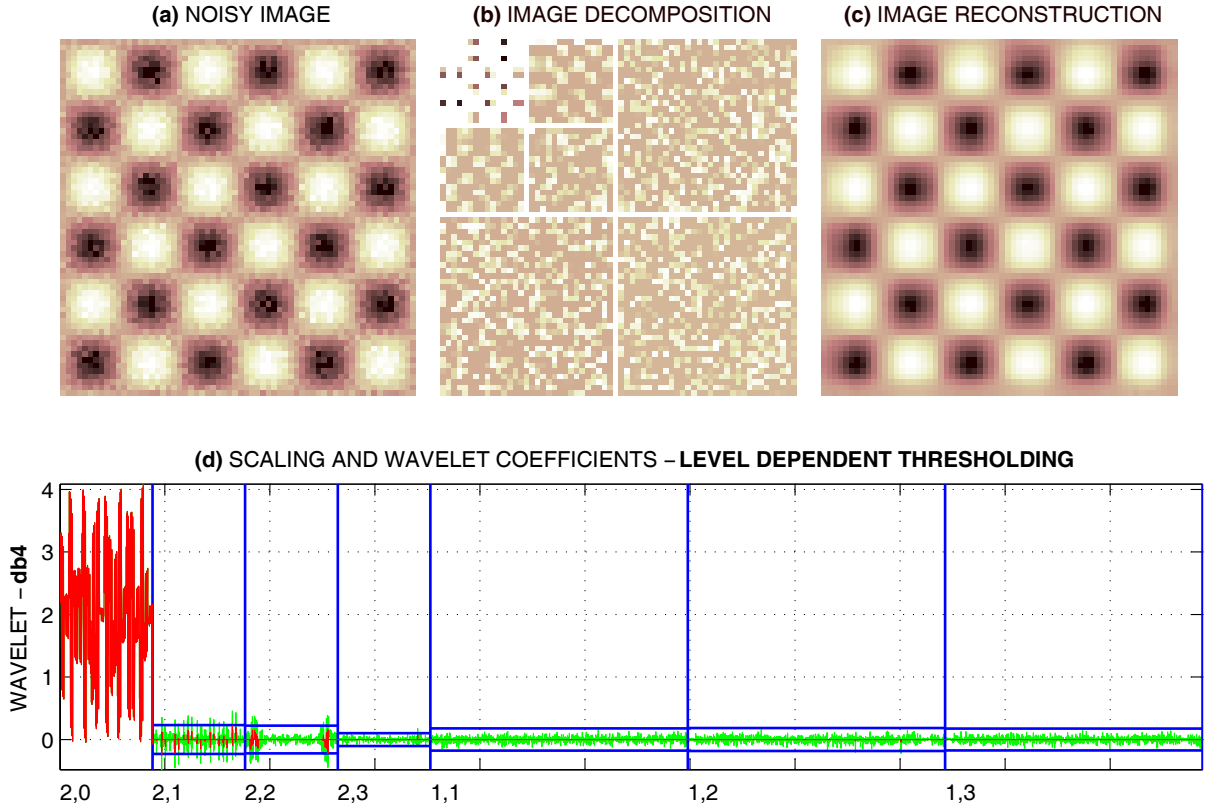


FIGURE 4.8. Image de-noising of (a) noisy image using a *db*4 wavelet function, (b) the decomposed image showing the approximation image (top-left corner) and the high frequency subimages details, (d) level dependent thresholding of the subband coefficients, and (c) the de-noised image

TABLE 4.3. Qualitative analysis (SIMULATED image) - Optimal thresholding

Type of wavelet	Level 1			Level 2		
	MSE	**MAE**	**PSNR** [dB]	**MSE**	**MAE**	**PSNR** [dB]
Noisy image	0.01076	0.51493	19.68	0.01076	0.51493	19.68
Haar	0.00376	0.05067	24.25	0.00513	0.05670	22.90
db2	0.00116	0.02492	29.35	0.00265	0.03912	25.76
db4	**0.00059**	**0.01909**	**32.33**	**0.00063**	**0.01781**	**32.00**
sym2	0.00116	0.02492	29.35	0.00265	0.03912	25.76
sym4	0.00071	0.01881	31.47	0.00074	0.01677	31.31
bior1.1	0.00376	0.05067	24.25	0.00513	0.05670	22.90

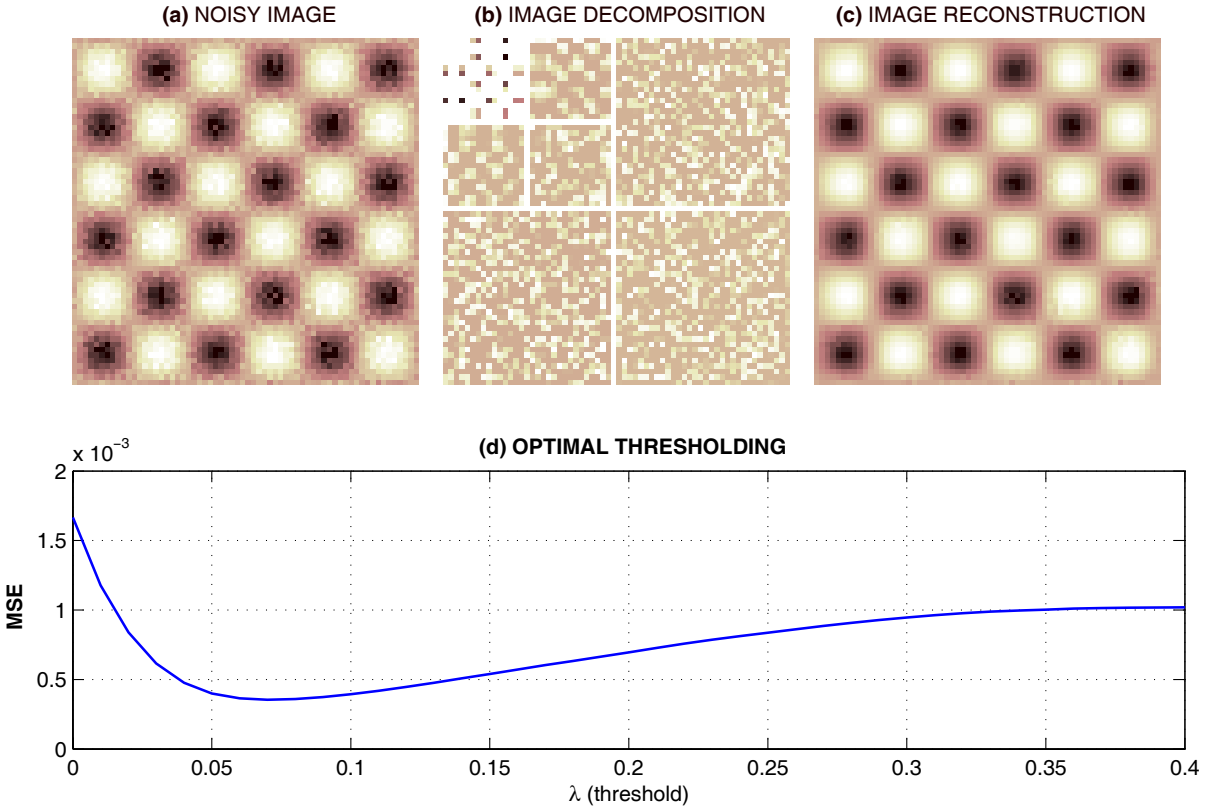


FIGURE 4.9. Discrete wavelet transform of the (a) noisy simulated image using a *db*4 wavelet function, (b) the decomposed image, (d) optimal thresholding showing the optimum threshold which minimizes the MSE, and (c) the de-noised simulated image

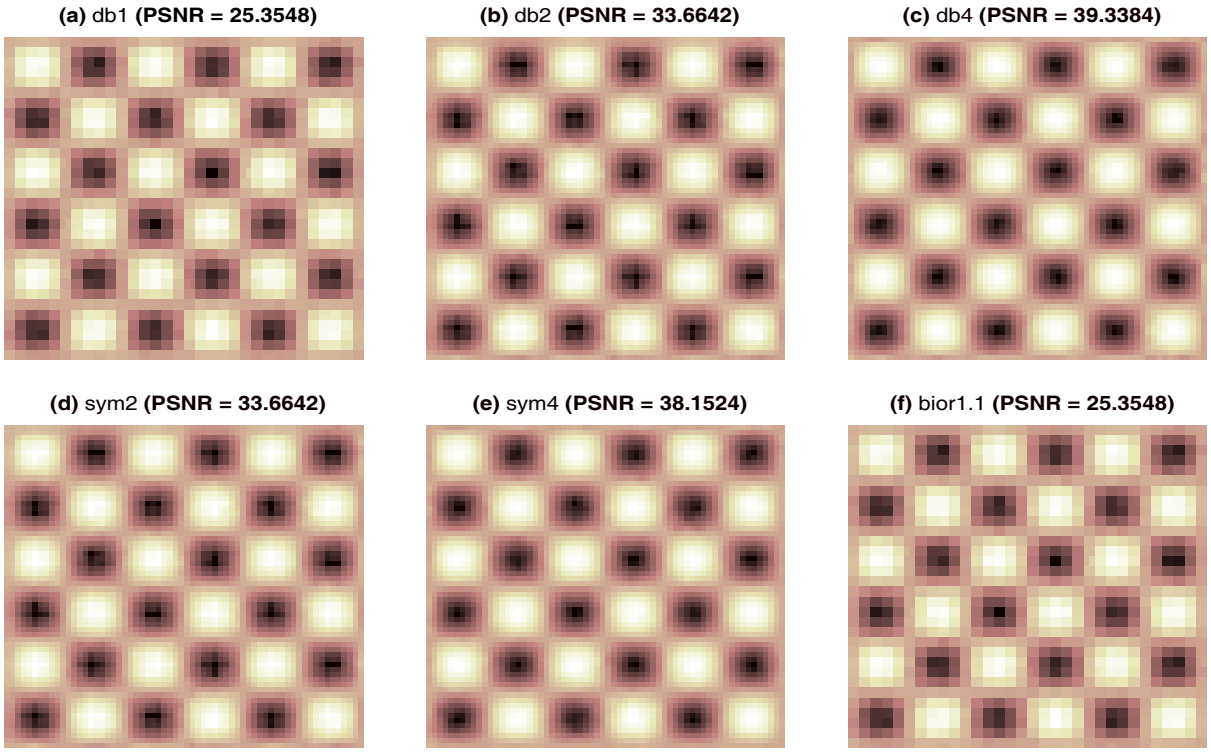


FIGURE 4.10. De-noised simulated images using different types of wavelets - level 1

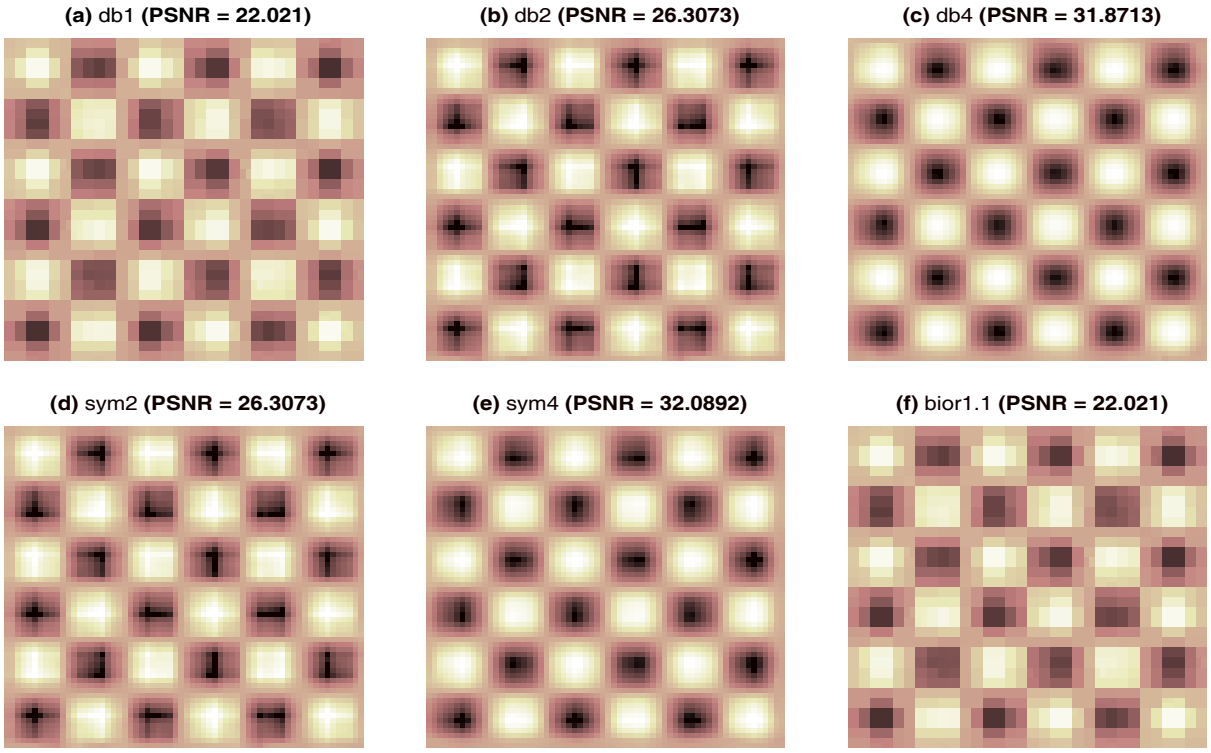


FIGURE 4.11. De-noised images using different types of wavelets - level 2

4.2.2 Results from Real MRI Data - 2-D

We have done simulations with uniform random noise added to the MR image. An example of a noisy magnetic resonance image (MRI) which consists of 128×128 pixels is shown in Fig. 4.12. As can be seen in the background the image has been uniformly corrupted with additive noise. The de-noising techniques discussed in the previous section are applied to the noisy MR image to test the efficiency of the different threshold methods.

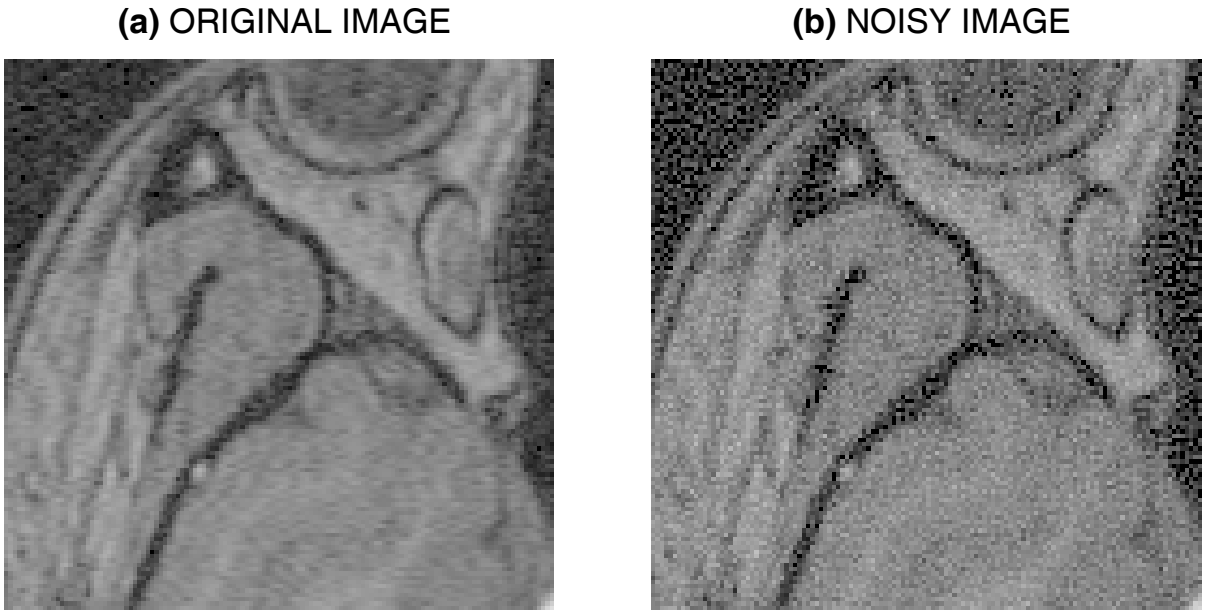


FIGURE 4.12. (a) Original image and (b) noisy image

Comparison of de-noising results for a various set of wavelets for the 128×128 MR image, corrupted with additive uniform random noise are shown in Tabs. 4.4, 4.5 and 4.6. The MSE and PSNR values from the experimental results show that *db*4 wavelet yields significantly improved visual quality as well as lower mean square error (MSE) and higher PSNR value compared to other wavelet functions. The simplest Haar wavelet produces the worst results as evidenced by the higher MSE, lower PSNR values and poor visual quality (Fig. 4.17(a)), thus it's not effective in removing noise.

Figs. 4.16 and 4.17 show the result of performing the de-noising algorithm with different types of wavelets, both for one and two levels of decomposition using the global tresholding method. Note the reduction in noise in the image Fig. 4.16(c) which was obtained by the *db*4 wavelet. It can also be seen that the background noise has been eliminated and the edge details are preserved.

TABLE 4.4. Qualitative analysis (MRI image) - Global thresholding

Type of wavelet	Level 1			Level 2		
	MSE	**MAE**	**PSNR** [dB]	**MSE**	**MAE**	**PSNR** [dB]
Noisy image	0.00472	0.29805	23.26	0.00472	0.29805	23.26
Haar	0.00169	0.03245	27.71	0.00157	0.02975	28.04
db2	0.00126	0.02702	28.99	0.00135	0.02789	28.69
db4	**0.00089**	**0.02353**	**30.52**	**0.00115**	**0.02565**	**29.39**
sym2	0.00126	0.02702	28.99	0.00135	0.02789	28.69
sym4	0.00092	0.02334	30.36	0.00119	0.02600	29.24
bior1.1	0.00169	0.03245	27.71	0.00157	0.02975	28.04

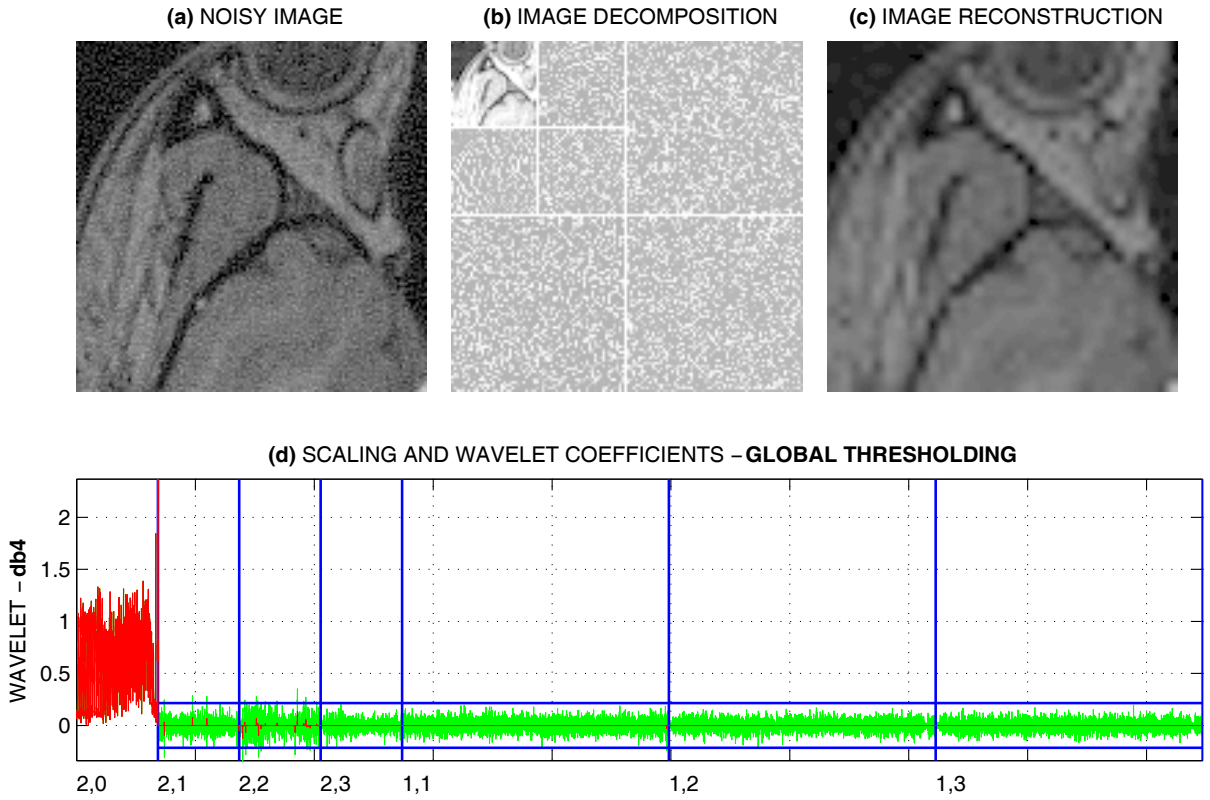


FIGURE 4.13. The 2-D image decomposition of the (a) noisy MR image using a *db*4 wavelet function, (b) the approximation image (low-frequency component) is in the top-left corner of the transform display, the other subimages contain the high frequency details, (d) global thresholding of the subband coefficients, and (c) shows the resulting de-noised MR image.

TABLE 4.5. Qualitative analysis (MRI image) - Level-dependent thresholding

Type of wavelet	Level 1			Level 2		
	MSE	**MAE**	**PSNR** [dB]	**MSE**	**MAE**	**PSNR** [dB]
Noisy image	0.00472	0.29805	23.26	0.00472	0.29805	23.26
Haar	0.00169	0.03244	27.72	0.00158	0.02979	28.02
db2	0.00126	0.02700	29.01	0.00135	0.02785	28.70
db4	**0.00088**	**0.02347**	**30.55**	**0.00113**	**0.02552**	**29.45**
sym2	0.00126	0.02700	29.01	0.00135	0.02785	28.70
sym4	0.00092	0.02329	30.39	0.00119	0.02594	29.26
bior1.1	0.00169	0.03244	27.72	0.00158	0.02979	28.02

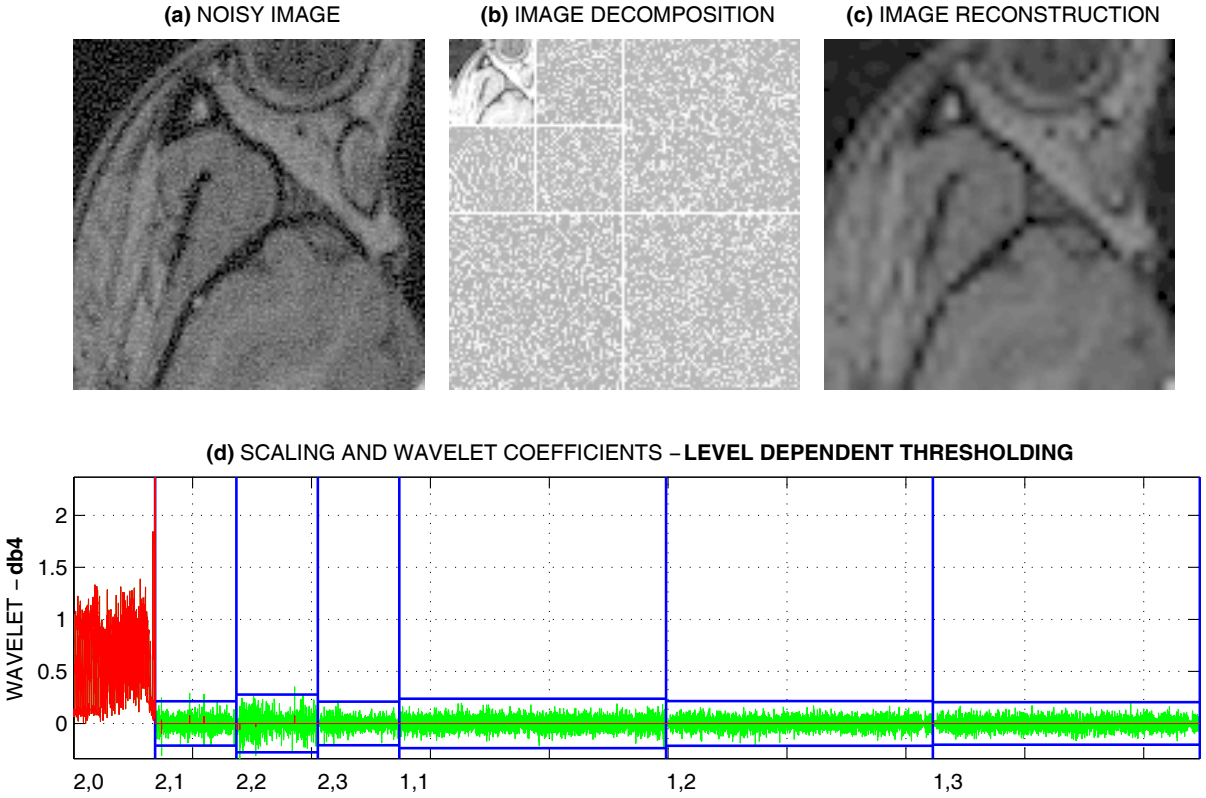


FIGURE 4.14. Discrete wavelet transform of the (a) noisy MR image using a *db*4 wavelet function, (b) two-level image decomposition, (d) level dependent thresholding of the subband coefficients, and (c) the resulting de-noised image, obtained by taking the inverse thresholded coefficients.

TABLE 4.6. Qualitative analysis (MRI image) - Optimal thresholding

Type of wavelet	Level 1			Level 2		
	MSE	**MAE**	**PSNR** [dB]	**MSE**	**MAE**	**PSNR** [dB]
Noisy image	0.00472	0.29805	23.26	0.00472	0.29805	23.26
Haar	0.00169	0.03244	27.72	0.00156	0.02972	28.07
db2	0.00126	0.02700	29.01	0.00135	0.02790	28.69
db4	**0.00088**	**0.02347**	**30.55**	**0.00115**	**0.02568**	**29.39**
sym2	0.00126	0.02700	29.01	0.00135	0.02790	28.69
sym4	0.00092	0.02329	30.39	0.00119	0.02595	29.25
bior1.1	0.00169	0.03244	27.72	0.00156	0.02972	28.07

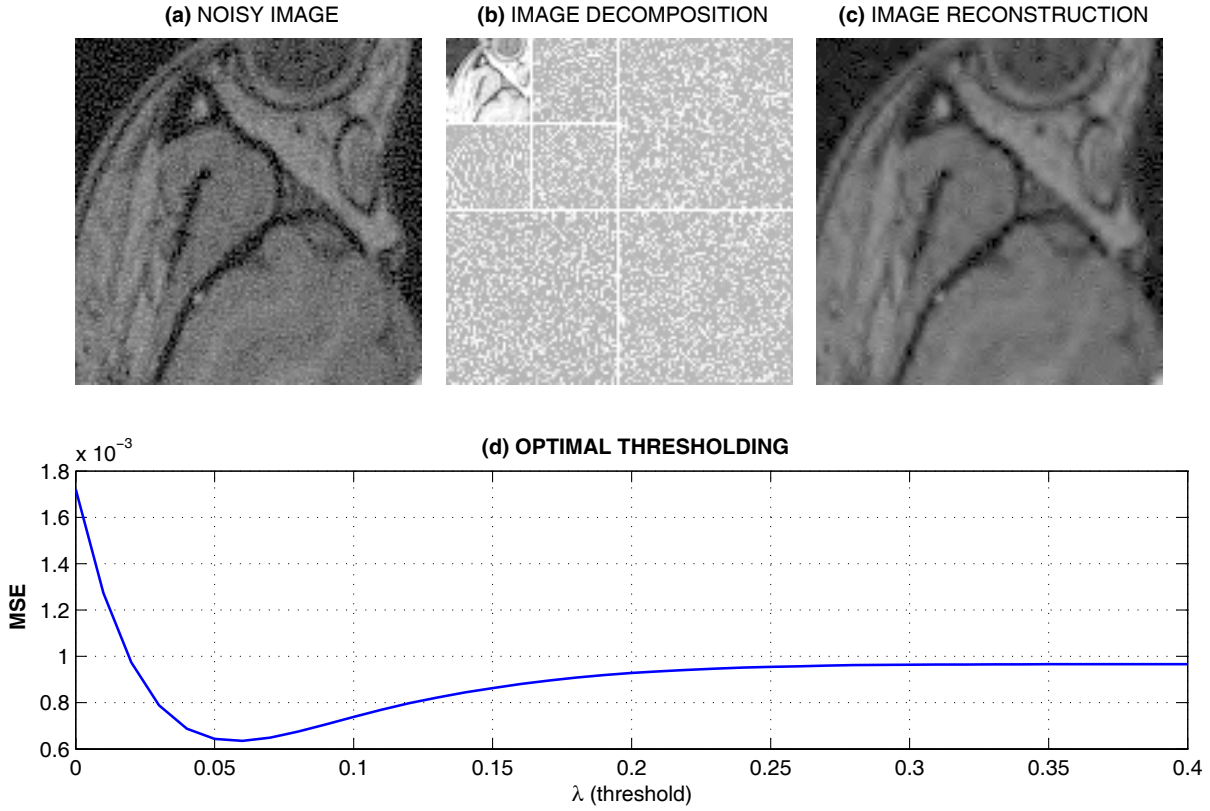


FIGURE 4.15. Discrete wavelet transform of the (a) noisy MR image using a *db*4 wavelet function, (b) the decomposed image showing the approximation image and the detail subband subimages, (d) optimal thresholding showing the optimum threshold which minimizes the MSE, and (c) shows the de-noised MR image.

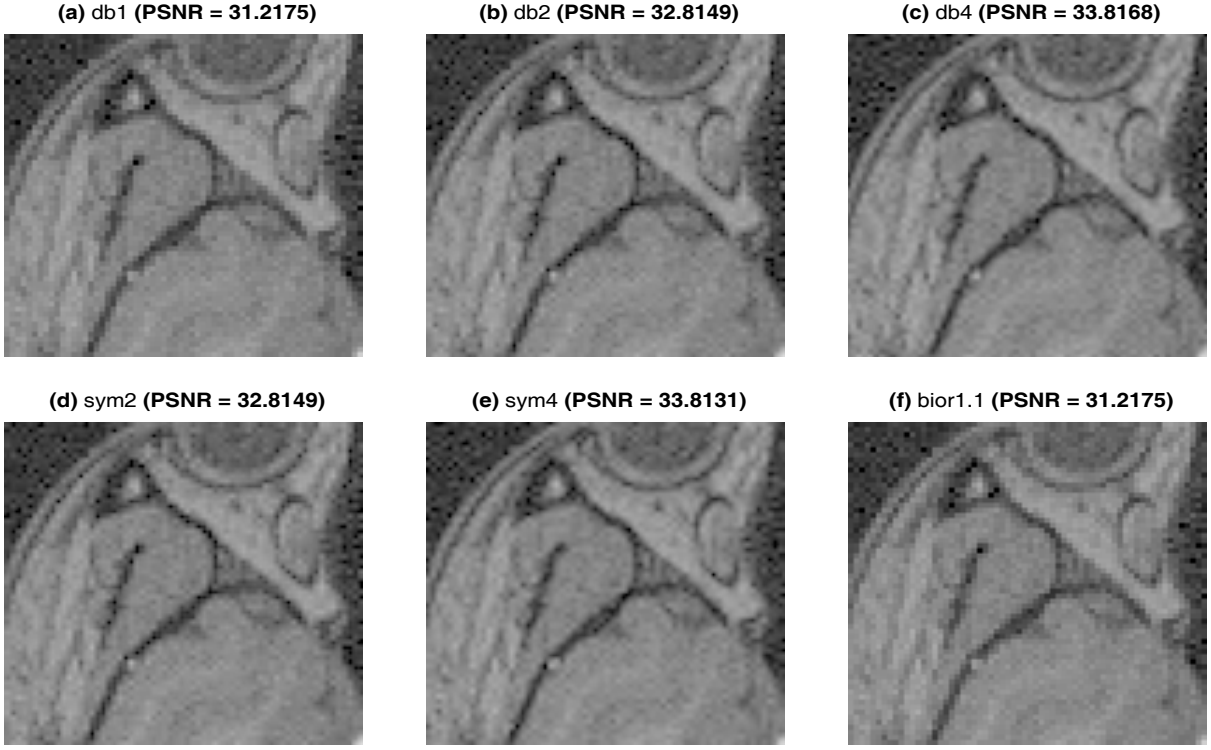


FIGURE 4.16. De-noised MR images using different types of wavelets for a one level decomposition

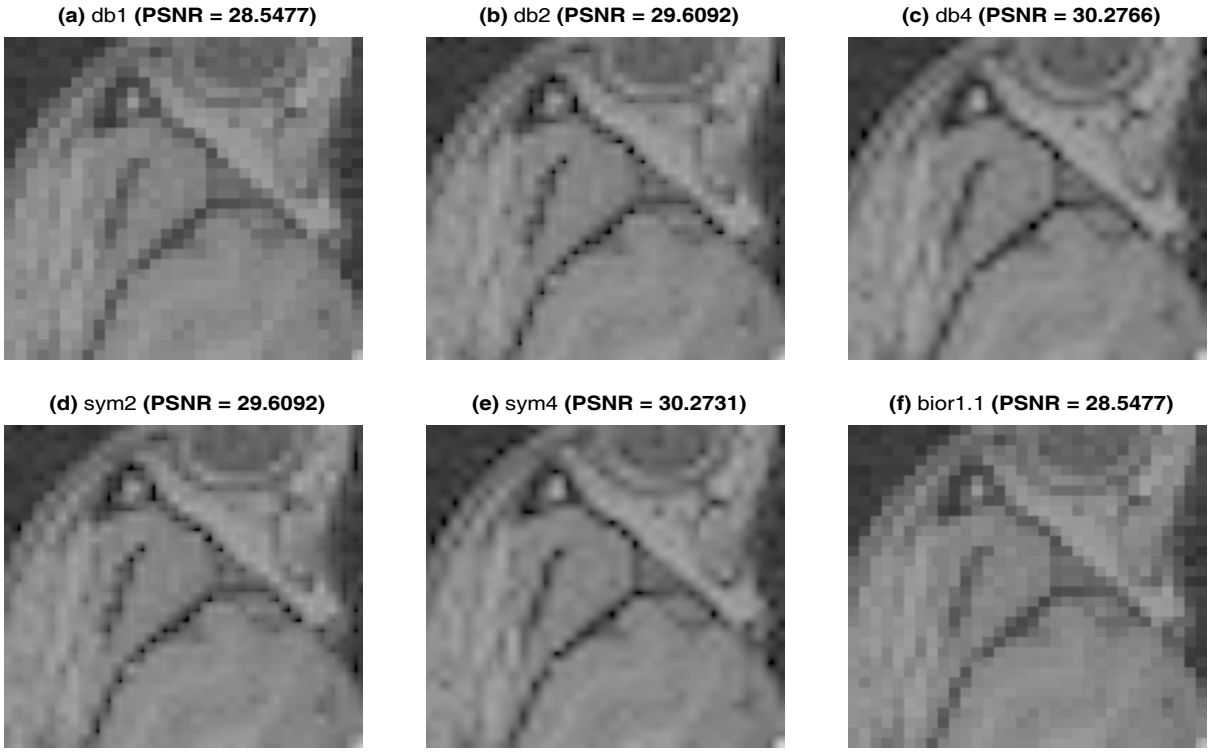


FIGURE 4.17. De-noised MR images using different types of wavelets for a two level decomposition

4.2.3 Results from Simulated Realistic Data - 3-D

The de-noising algorithm was applied to simulated image volume corrupted with randomly distributed noise generated using the MATLAB `rand` function. The effectiveness of different wavelet functions are compared using the mean square error (MSE), signal-noise-ratio (SNR) and visual criteria. The simulated volume consists of a number of $64 \times 64 \times 8$ three-dimensional data.

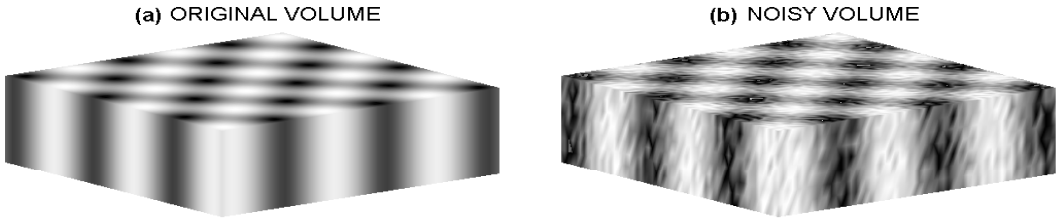


FIGURE 4.18. (a) Original and (b) noisy simulated image volume

The 3-D discrete wavelet transform has been used to de-noise the noisy volume. Fig. 4.19 shows a plot of the coefficients of a one level of the transform. The horizontal lines shown in the graph are the soft thresholding levels of 0.5. The effect of this thresholding will set all the values in the filtered signal that have an absolute value less than 0.5 to zero and rescales other coefficients. Applying the given threshold level and taking the inverse of the result a de-noised volume is obtained as shown in Fig. 4.19(c).

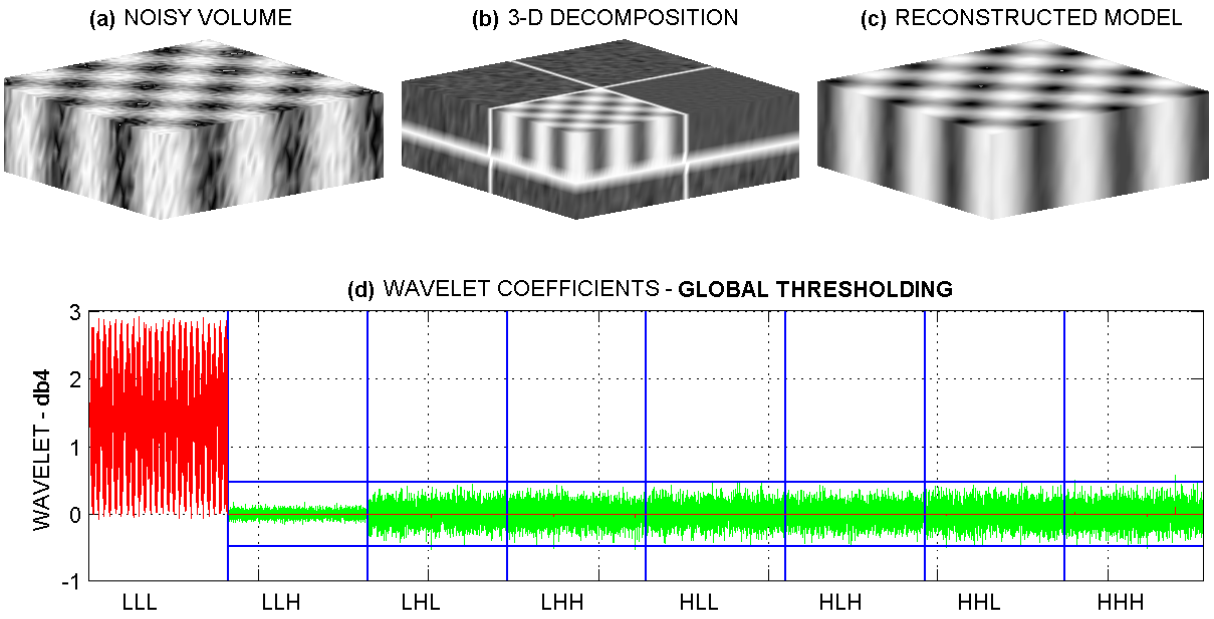


FIGURE 4.19. 3-D DWT of the (a) noisy simulated image volume using a *db*4 wavelet function, (b) the decomposed image subband volumes, (d) global thresholding of the wavelet coefficients, and (c) the resulting de-noised image image volume

4.2.4 Results from Real MRI Data - 3-D

The de-noising algorithm was applied to an MRI sub-volume corrupted with randomly distributed noise. Results from different wavelets are compared using the mean square error (MSE), peak signal-noise-ratio (PSNR) and visual criteria. The MSE is computed relative to the original image volume i.e. it measures the difference between the values of the corresponding pixels from the two volumes. Fig. 4.20 shows the original MR sub-volume and the noisy volume.

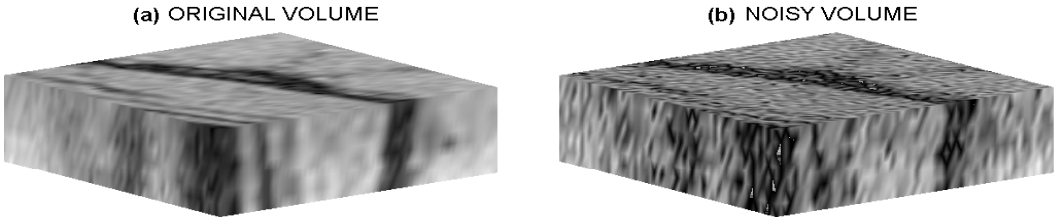


FIGURE 4.20. (a) Original and (b) noisy MR image volume

We applied 3-D wavelet transform algorithm to an MRI scan of a human brain (128 x 128 x 16). Fig. 4.21 shows a one level decomposition of the 3-D volume obtained by using the *db*4 wavelet. The ability of the discrete wavelet transform to reduce distortion in the reconstructed volume while retaining all the significant features present in the image volume is seen in Fig. 4.21(c).

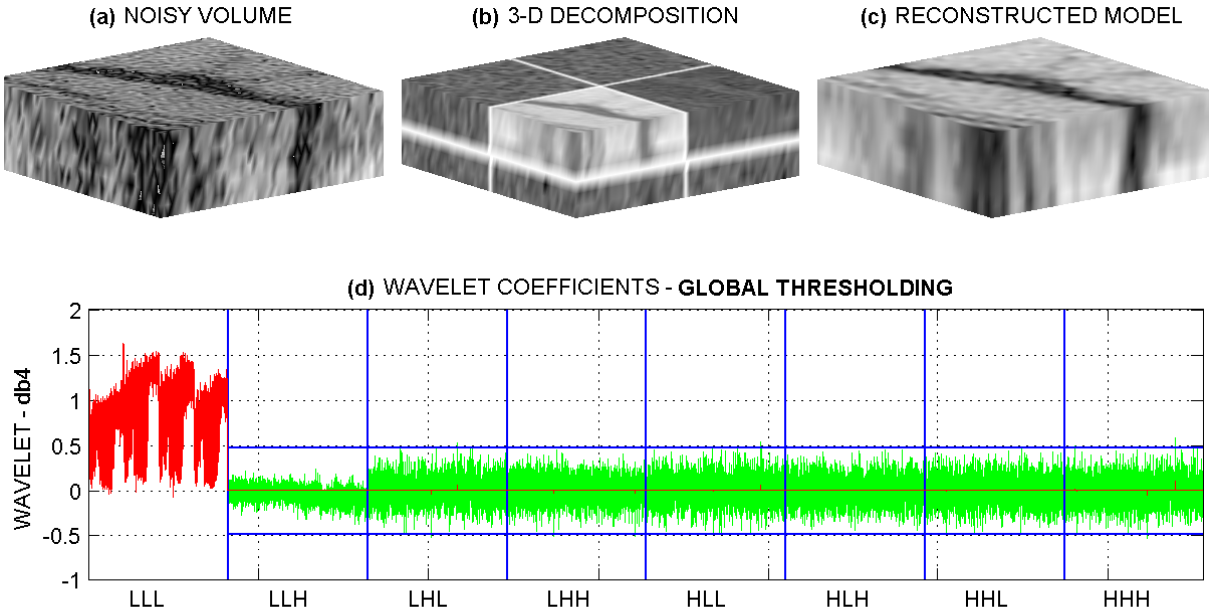


FIGURE 4.21. Three-dimensional decomposition of (a) noisy MR image volume, (b) one-level image volume decomposition, (d) wavelet coefficients of different subbands (the thresholded values are the ones in red), and (c) the reconstructed model

The proposed 3-D discrete wavelet de-noising algorithm has been evaluated on the noisy MR volume, by visual inspection and by computing quantitative measures of the similarity between the reference image and the de-noised image. The performance of different wavelets is compared by computing the error criteria mean square error, mean absolute error and the peak signal-to-noise ratio of the noisy image and the de-noised image. Tables 4.7, 4.8 and 4.9 show the numerical results after the implementation of the global, level-dependent and optimal thresholding respectively. In all the cases the *db*4 wavelet outperforms other wavelets as can be seen from their increase of the PSNR values.

TABLE 4.7. Qualitative analysis (MRI image volume) - Global thresholding

	Level 1		
Type of wavelet	**MSE**	**MAE**	**PSNR** [dB]
Noisy image	0.01479	0.10096	18.30
Haar	0.00278	0.04145	25.56
db2	0.00322	0.04388	24.92
db4	**0.00216**	**0.03654**	**26.66**
sym2	0.00322	0.04388	24.92
sym4	0.00294	0.04144	25.32
bior1.1	0.00278	0.04145	25.56

TABLE 4.8. Qualitative analysis (MRI image volume) - Level-dep. thresholding

	Level 1		
Type of wavelet	**MSE**	**MAE**	**PSNR** [dB]
Noisy image	0.01479	0.10096	18.30
Haar	0.00279	0.04149	25.55
db2	0.00324	0.04397	24.89
db4	**0.00216**	**0.03655**	**26.65**
sym2	0.00324	0.04397	24.89
sym4	0.00294	0.04148	25.31
bior1.1	0.00279	0.04149	25.55

TABLE 4.9. Qualitative analysis (MRI image volume) - Optimal thresholding

	Level 1		
Type of wavelet	**MSE**	**MAE**	**PSNR** [dB]
Noisy image	0.01479	0.10096	18.30
Haar	0.00277	0.04137	25.58
db2	0.00306	0.04329	25.14
db4	**0.00216**	**0.03651**	**26.66**
sym2	0.00306	0.04329	25.14
sym4	0.00280	0.04111	25.53
bior1.1	0.00277	0.04137	25.58

4.3 Discussions and Conclusions

The de-noising process consists of decomposing the image, thresholding the detail coefficients, and reconstructing the image. The decomposition procedure of the de-noising example is accomplished by using the DWT. Wavelet thresholding is an effective way of de-noising as shown by the experimental results obtained with the use of different types of wavelets. Thresholding methods implemented comprised of the universal global thresholding, level (subband) thresholding and optimal thresholding. More levels of decomposition can be performed, the more the levels chosen to decompose an image or volume, the more detail coefficients we get. But for de-noising the noisy MR data sets, a two-level decomposition provided sufficient noise reduction.

In this chapter we have presented the generalization of the DWT method from the 2-D to the 3-D case. The resulting algorithms have been used for the processing of noisy MR image volumes. Experimental results have shown that despite the simplicity of the proposed de-noised algorithm it yields significantly better results both in terms of visual quality and mean square error values. Considering the simplicity of the proposed method, we believe these results are very encouraging for other forms of de-noising. The fourth-order Daubechies wavelet ($db4$) gave the best results compared to other wavelets and the simple Haar wavelet produces the worst results. However, the Haar wavelet is a useful and simple wavelet which is normally used for demonstrating purposes of the discrete wavelet transform. DWT using a $db4$ produces sharper edges and retains more detail, providing a closer resemblance to the original than the other wavelets.

The noise assumption used in Donoho's ([24, 23]) derivation fails when images are not contaminated with additive noise (uniform random noise, gaussian noise). Nonlinear image processing techniques are required to remove multiplicative noise whereas linear spatial filtering methods (DWT) are used to remove additive noise. If for example we don't have a reference image it is possible to take the average of multiple images of the same image at the same noise level and this form is also useful for removing noise.

Finally, a great advantage of the wavelet transform is that often a large number of the detail coefficients turns out to be very small in magnitude, truncating (removing) these small coefficients from the representation introduces only small errors in the reconstructed image, giving an image which closely resembles the original image and also preserving edge features.

5
Dual Tree Complex Wavelet Transform

Different wavelet techniques can be successfully applied in various signal and image processing methods, namely in image de-noising, segmentation, classification and motion estimation. The chapter discusses the application of dual tree complex wavelet transform (DT CWT) which has significant advantages over real wavelet transform for certain signal processing problems. The transform was proposed by Dr. Nick Kingsbury of Cambridge University. He has written various papers in line to this topic providing a solid mathematical background that allows practical use of complex wavelets in image processing [45, 44, 46, 43]. The DT CWT can be used for a variety of applications such as de-noising, edge detection, image restoration, enhancement, and image compression.

DT CWT is a form of discrete wavelet transform, which generates complex coefficients by using a dual tree of wavelet filters to obtain their real and imaginary parts. What makes the complex wavelet basis exceptionally useful for de-noising purposes is that it provides a high degree of shift-invariance and better directionality compared to the real DWT. The main part of the chapter is mainly focussed on the theoretical analysis of complex wavelet transform. The resulting wavelet algorithm is then applied to the analysis and de-noising of magnetic resonance (MR) biomedical images. In this chapter we will also explore the performance of these enhanced transforms in MRI data analysis and show that these algorithms can powerfully enhance the PSNR in noisy MRI data sets.

5.1 Complex Wavelet Transform

Complex wavelets have not been used widely in image processing due to the difficulty in designing *complex* filters which satisfy a perfect reconstruction property. To overcome this Dr. Kingsbury [44] proposed a *dual-tree* implementation of the CWT which uses two trees of real filters to generate the real and imaginary parts of the wavelet coefficients separately. The two trees are shown in Fig. 5.1 for 1-D signals, complex wavelet coefficients are estimated by dual tree algorithm and their magnitude is shift invariant. Even though the outputs of each tree are downsampled by summing the outputs of the two trees during reconstruction we are able to suppress the aliased components of the signal and achieve approximate shift invariance.

5.2 1-D Complex Wavelet Transform

The 1-D dual-tree wavelet transform [39] is implemented using a pair of filter banks operating on the same data simultaneously. The upper iterated filter bank represents the real part of a complex wavelet transform. The lower one represents the imaginary part. The transform is an expansive (or oversampled) transform. The transform is two times expansive because for an N-point signal it gives $2N$ DWT coefficients. When designed in this way the DT CWT is nearly shift invariant, in contrast to the classic DWT.

The structure of a resulting analysis filter bank is sketched in Fig. 5.1, where index a stands for the original filter bank and the index b is for the additional one. The dual-tree complex wavelet transform of a signal $x(n)$ is implemented using two critically-sampled DWT in parallel on the same data.

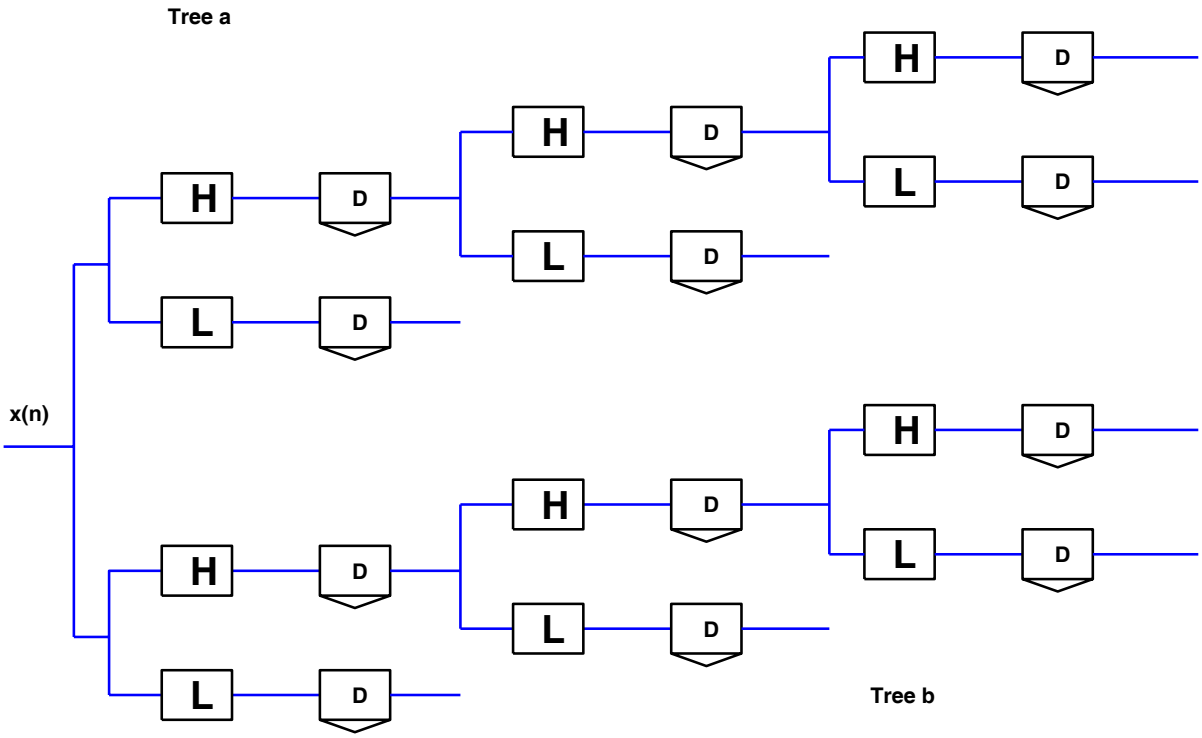


FIGURE 5.1. The 1-D dual-tree wavelet transform is implemented with a pair of filter banks operating on the same data simultaneously.

In the thesis a dual-tree CWT using a length-10 filters [46] is used, the table of coefficients of the analysing filters in the first stage (Tab. 5.1) and the remaining levels (Tab. 5.2) are shown. The coefficients of the synthesis filters are simply the transposes of the analysis filters (orthogonal filters). The complex wavelet associated with the dual tree CWT is shown in Fig. 5.2. The impulse responses of these then look very like the real and imaginary parts of the complex wavelets.

TABLE 5.1. First level coefficients of the analysis filters

Tree a		*Tree b*	
H_{0a}	H_{1a}	H_{0b}	H_{1b}
0	0	0.01122679	0
-0.08838834	-0.01122679	0.01122679	0
0.08838834	0.01122679	-0.08838834	-0.08838834
0.69587998	0.08838834	0.08838834	-0.08838834
0.69587998	0.08838834	0.69587998	0.69587998
0.08838834	-0.69587998	0.69587998	-0.69587998
-0.08838834	0.69587998	0.08838834	0.08838834
0.01122679	-0.08838834	-0.08838834	0.08838834
0.01122679	-0.08838834	0	0.01122679
0	0	0	-0.01122679

TABLE 5.2. Remaining levels coefficients of the analysis filters

Tree a		*Tree b*	
H_{00a}	H_{01a}	H_{00b}	H_{01b}
0.03516384	0	0	-0.03516384
0	0	0	0
-0.08832942	-0.11430184	-0.11430184	0.08832942
0.23389032	0	0	0.23389032
0.76027237	0.58751830	0.58751830	-0.76027237
0.58751830	-0.76027237	0.76027237	0.58751830
0	0.23389032	0.23389032	0
-0.11430184	0.08832942	-0.08832942	-0.11430184
0	0	0	0
0	-0.03516384	0.03516384	0

The approximations and details for the two trees are denoted respectively (a_A, d_A) and (a_B, d_B). The details d_A and d_B can be interpreted as the real and imaginary parts of a complex process $z = d_A + id_B$. The essential property of this transform is that the magnitude of the step response is approximately invariant with the input shift [45]. If we only consider the magnitude $|z|$ for a given scale, it corresponds to an approximately shift invariant transform, and thresholding this magnitude produces less artifacts than thresholding real transforms. DT CWT is not really a complex wavelet transform, since it does not use any complex wavelet instead it is implemented with real wavelet filters. The reconstruction is done in each tree independently, by using the dual filters, the results are averaged to obtain $x(n)$ and to ensure the symmetry between the trees, thus producing the desired shift invariance.

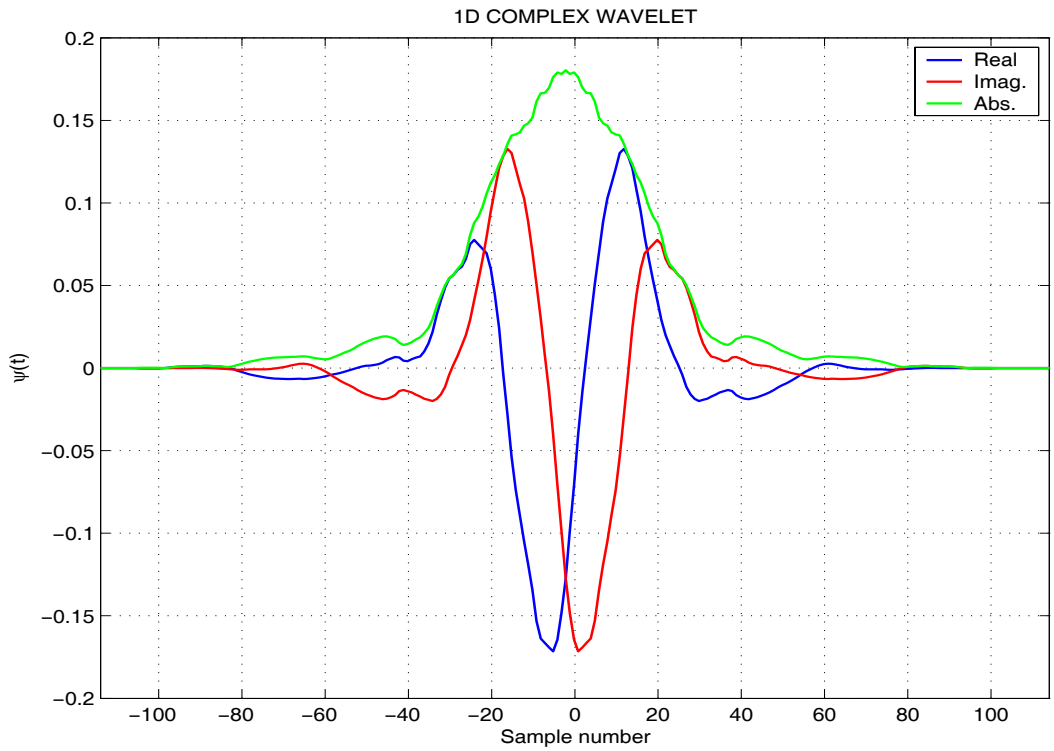


FIGURE 5.2. A complex wavelet associated with the real and imaginary wavelets for 1-D DT DWT

Translation Invariance by Parallel Filter Banks

The orthogonal [63] two-channel filter bank with analysis low-pass filter given by the z-transform $H_0(z) = \sum_{k\in Z} h_0(k)z^{-k}$, analysis highpass filter $H_1(z) = \sum_{k\in Z} h_1(k)z^{-k}$ and with synthesis filters $G_0(z) = H_0(z^{-1})$ and $G_1(z) = H_1(z^{-1})$ is shown by the diagram in Fig. 5.3.

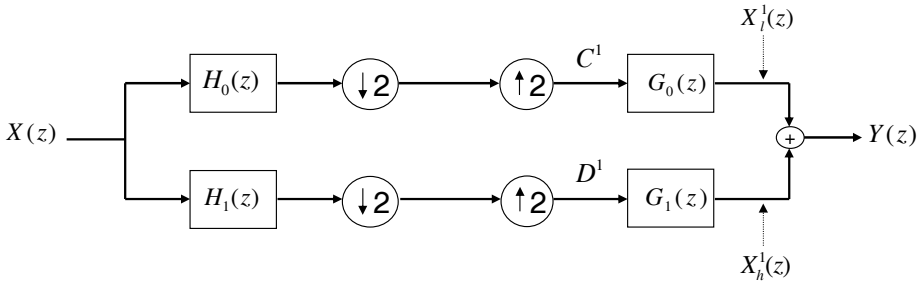


FIGURE 5.3. One level signal wavelet decomposition and reconstruction

For an input signal $X(z)$, the analysis part of the filter bank inclusive subsequent upsampling produces the low-pass and the high-pass coefficients respectively,

$$C^1(z^2) = \frac{1}{2}\{X(z)H_0(z) + X(-z)H_0(-z)\} \tag{5.1}$$

$$D^1(z^2) = \frac{1}{2}\{X(z)H_1(z) + X(-z)H_1(-z)\} \tag{5.2}$$

and decomposes the input signal into a low frequency part $X_l^1(z)$ and a high frequency part $X_h^1(z)$ in the form

$$X(z) = X_l^1(z) + X_h^1(z)$$

where

$$X_l^1(z) = C^1(z^2)G_0(z) = \frac{1}{2}\{X(z)H_0(z)G_0(z) + X(-z)H_0(-z)G_0(z)\} \tag{5.3}$$

$$X_h^1(z) = D^1(z^2)G_1(z) = \frac{1}{2}\{X(z)H_1(z)G_1(z) + X(-z)H_1(-z)G_1(z)\} \tag{5.4}$$

However this decomposition is not shift invariant due to the $X(-z)$ terms in Eq. (5.3) and Eq. (5.4), which are aliasing terms introduced by the downsampling and upsampling operations. For a single shift of the input signal $z^{-1}S(z)$ and the implementation of the filter bank we have

$$z^{-1}X(z) = \tilde{X}_l^1(z) + \tilde{X}_h^1(z)$$

where

$$\tilde{X}_l^1(z) = C^1(z^2)G_0(z) \tag{5.5}$$

and similarly for the high-pass part. For an input signal $z^{-1}X(z)$ we have

$$C^1(z^2) = \frac{1}{2}\{z^{-1}X(z)H_0(z) + (-z)^{-1}X(-z)H_0(-z)\} \tag{5.6}$$

and

$$\tilde{X}_l^1(z) = \frac{1}{2}z^{-1}\{X(z)H_0(z)G_0(z) - X(-z)H_0(-z)G_0(z)\} \neq z^{-1}X_l^1(z) \tag{5.7}$$

which of course is not the same as $z^{-1}X_l^1(z)$ if we substitute for z^{-1} in Eq. (5.3). From this calculation one can see that the shift dependence is caused by the terms containing $X(-z)$, the so called *aliasing terms*.

One possibility to obtain a shift invariant decomposition consists of applying an additional filter bank with shifted analysis filters $z^{-1}H_0(z)$ and $z^{-1}H_1(z)$ that is for synthesis filters ($zG_0(z)$ and $zG_1(z)$) and averaging the low and the highpass channels of both filter banks.

If we label the first filter bank by an index a and the second one by an index b then this procedure implies the decomposition

$$X(z) = X_l^1(z) + X_h^1(z)$$

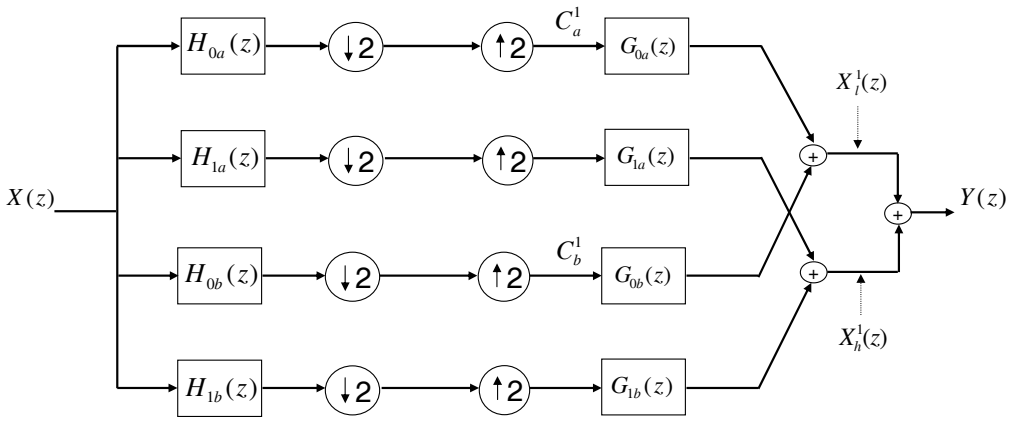


FIGURE 5.4. One level complex dual tree

where

$$\begin{aligned}
X_l^1(z) &= \frac{1}{2}\{C_a^1(z^2)G_{0a}(z^{-1}) + C_b^1(z^2)G_{0b}(z^{-1})\} \\
&= \frac{1}{4}\{[X(z)H_0(z) + X(-z)H_0(-z)]G_0(z) \\
&\quad + [X(z)z^{-1}H_0(z) + X(-z)(-z)^{-1}H_0(-z)]zG_0(z)\} \\
&= \frac{1}{4}\{X(z)H_0(z)G_0(z) + X(-z)H_0(-z)G_0(z) \\
&\quad + z^{-1}.z(X(z)H_0(z)G_0(z) - X(-z)H_0(-z)G_0(z))\} \\
&= \frac{1}{4}\{X(z)[H_0(z)G_0(z) + H_0(z)G_0(z)] + X(-z)[H_0(-z)G_0(z) - H_0(-z)G_0(z)]\} \\
&= \frac{1}{4}\{X(z)(H_0(z)G_0(z) + H_0(z)G_0(z))\} \\
&= \frac{1}{2}X(z)H_0(z)H_0(z^{-1})
\end{aligned} \tag{5.8}$$

and similarly for the high-pass part. The aliasing term containing $X(-z)$ in X_l^1 has disappeared and the decomposition becomes indeed shift invariant. Using the same principle for the design of shift invariant filter decomposition, Dr. Kingsbury suggested in [44] to apply a dual-tree of two parallel filter banks and combine their bandpass outputs. The structure of a resulting analysis filter bank is sketched in Fig. 5.1, where index a stands for the original filter bank and the index b is for the additional one. The dual-tree complex DWT of a signal $x(n)$ is implemented using two critically-sampled DWTs in parallel on the same data.

The DT CWT is less sensitive to signal shift than the DWT [27]. To illustrate this fact Fig. 5.5(a) shows the the original input signal and the subband coefficients at first and second level. On shifting this signal by one sample Fig. 5.5(b) is obtained. From these figures the difference between the subband coefficients before and after shifting is clearly seen in contrast to the DT CWT (Fig. 5.6) where the shape of the subband coefficients remains the same. This shows that the DWT is shift sensitive because DWT coefficients behave unpredictably under signal shifts. Indeed this shift invariance problem is a big disadvantage that makes the DWT not suitable for other several signal processing applications.

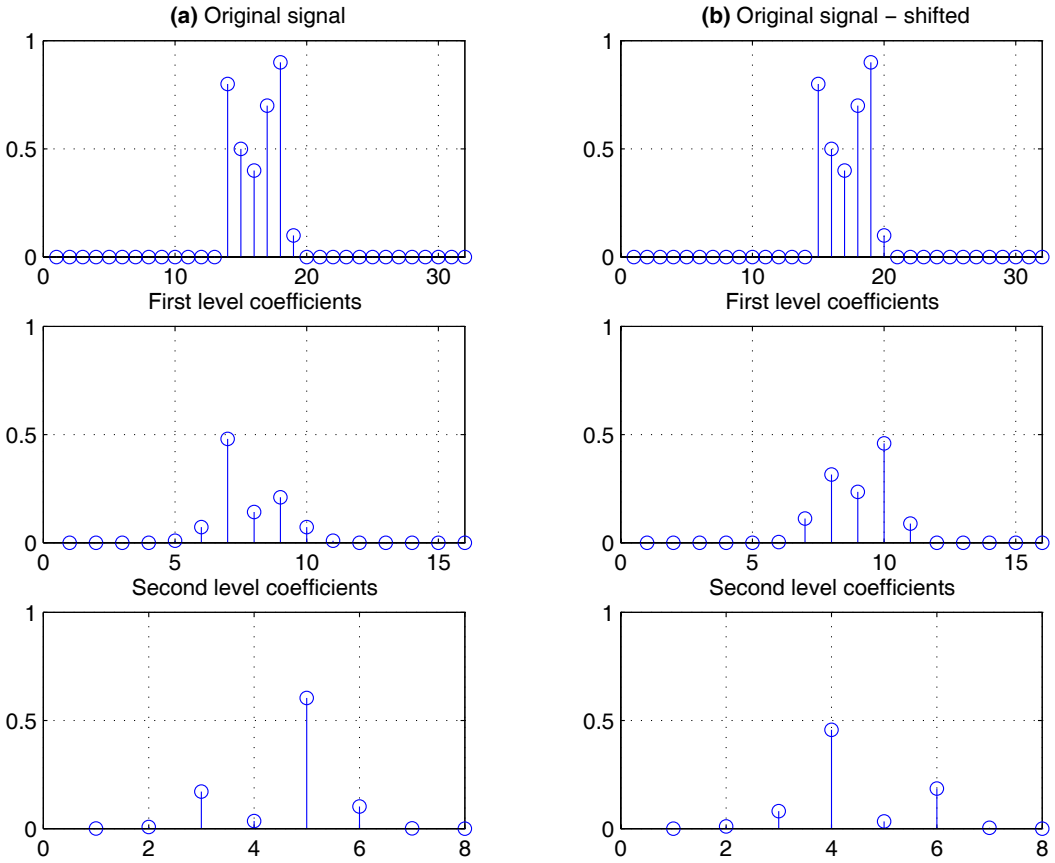


FIGURE 5.5. Shift sensitivity of the discrete wavelet transform - magnitude of the first and second level subband coefficients (a) original signal and (b) shifted one sample to the right

A transform is shiftable if the coefficient energy in each transform subband is conserved under input-signal shifts. Thus shiftability is very important for certain applications because it guarantees that the information in a transform subband is confined within that subband even when the input signal is shifted in position.

The necessary and sufficient condition for shiftability is that each of the subband energy is constant for any signal shifts. To prove that the DT CWT is shift invariance we

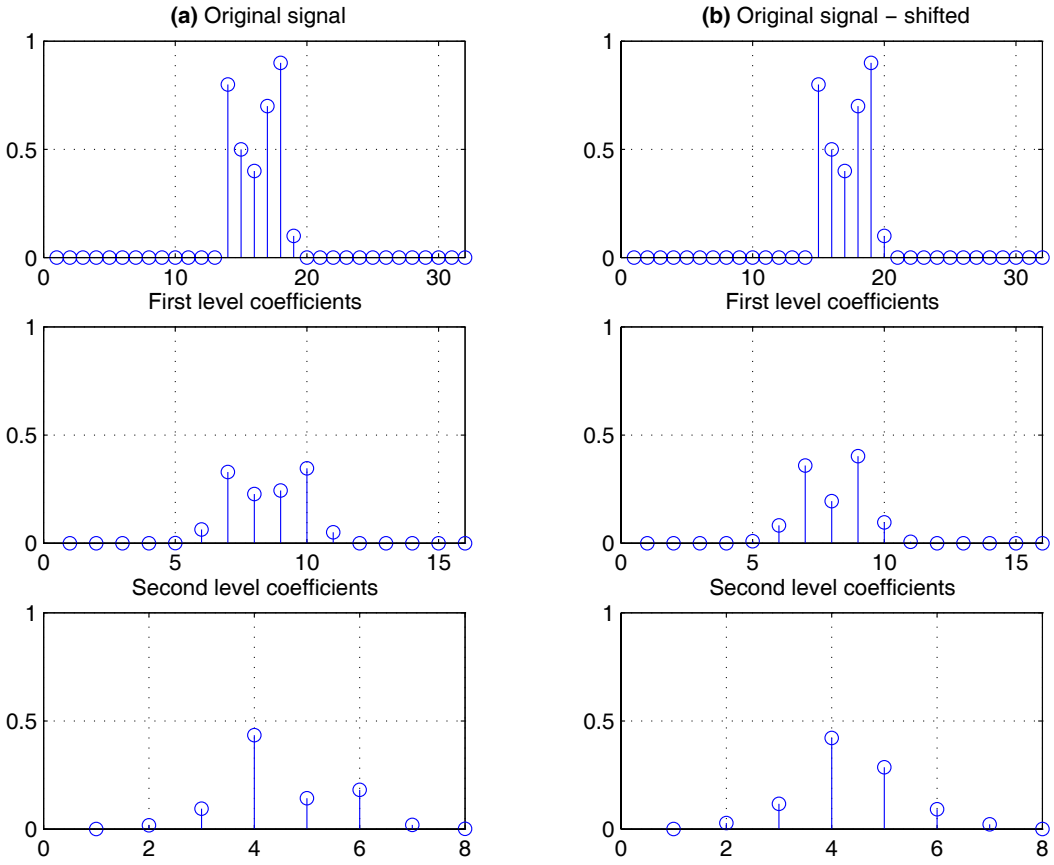


FIGURE 5.6. Shift sensitivity of the dual tree CWT - magnitude of the first and second level subband coefficients (a) original signal and (b) shifted one sample to the right

must show that its subband energy is approximately constant under input signal shifts. Fig. 5.5(a) and Fig. 5.6(a) show the original input signal on which a two level DWT and CWT has been applied. Then for both transforms the subband energy over circular shifts of the input signal is computed. From Fig. 5.7 it can be observed that there are large oscillations in the DWT subband energies in contrast to the DT CWT subband energy that remains approximately constant over input-signal shifts. This also gives the proof of the approximate shift invariance for the DT CWT.

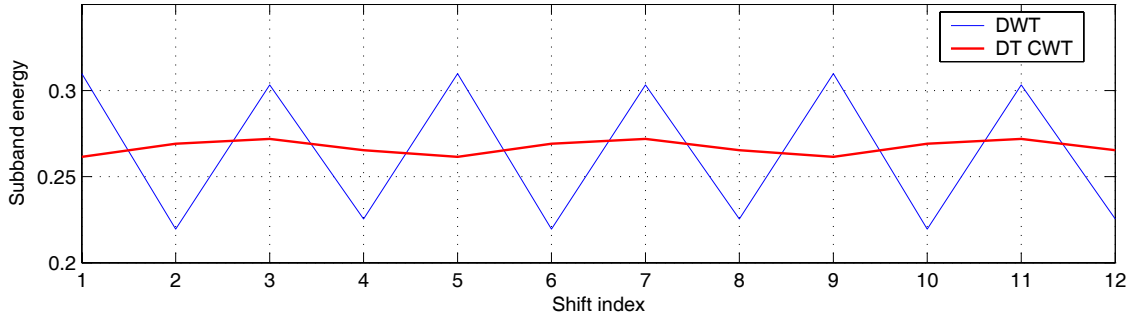


FIGURE 5.7. Subband energy for the DWT and DT CWT methods

5.3 2-D Dual-Tree Wavelet Transform

To extend the transform to higher-dimensional signals, a filter bank is usually applied separably in all dimensions. To compute the 2-D DT CWT of images the pair of trees are applied to the rows and then the columns of the image as in the basic DWT. This operation results in six complex high-pass subbands at each level and two complex low-pass subbands on which subsequent stages iterate in contrast to three real high-pass and one real low-pass subband for the real 2D transform. This shows that the complex transform has a coefficient redundancy of 4:1 or $2^m : 1$ in m dimensions. The price for high redundancy is a reasonable tradeoff for a shift-invariant, multiresolution transform. The overall redundancy of the 2-D DT CWT is 4:1, i.e. 4 real numbers are produced for every input pixel, regardless of how many levels are computed.

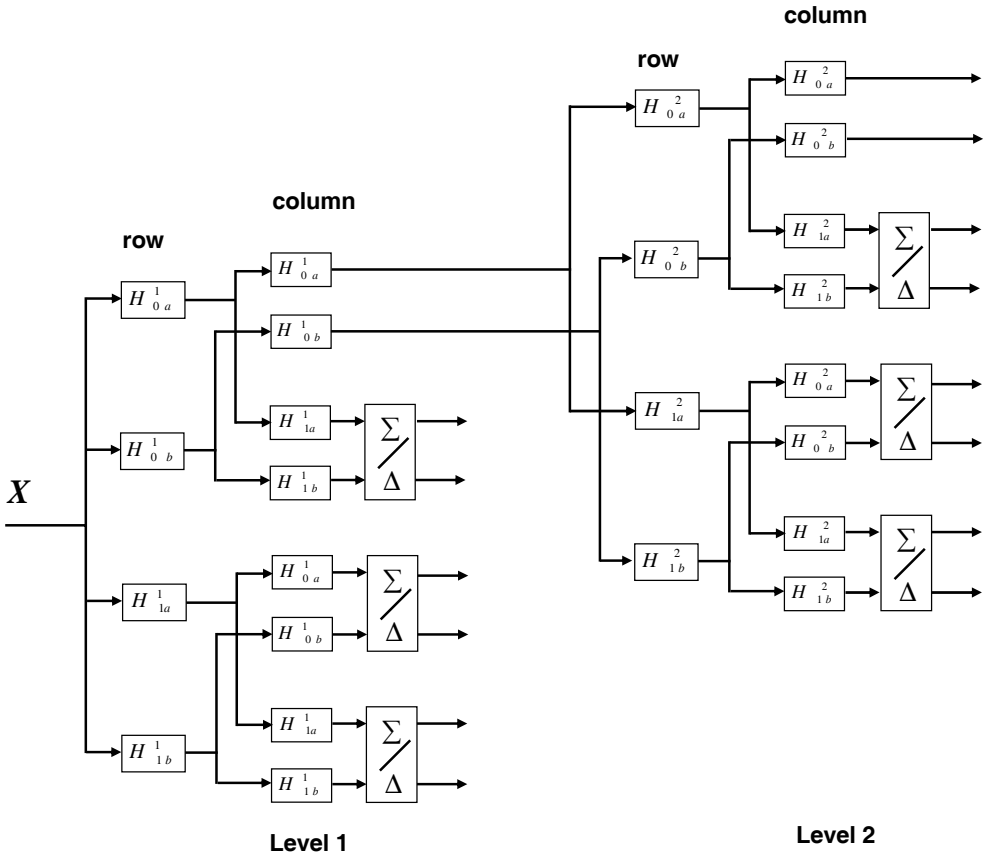


FIGURE 5.8. 2D dual tree CWT, each I-D convolution is followed by a 1-D downsampling of 2.

We implement the 2-D DWT [43] separably so that only 1D convolutions and downsampling are required. This results in three bandpass subimages and one lowpass subimage, on which the basic block is iterated. Fig. 5.8 shows a complete structure over 2 levels. The real 2-D dual-tree CWT of an image $x(n)$ is implemented using two critically-sampled

separable 2-D DWT in parallel. Then for each pair of subbands we take the sum and difference. Each level of the tree produces 6 bandpass subimages as well as two lowpass subimages, on which subsequent stages iterate. In case of real 2-D filter banks the three highpass subbands have orientations of $0^0, 45^0$ and 90^0 compared to the complex filters which have six high-pass subbands at each level which are oriented at $\pm 15^0, \pm 45^0, \pm 75^0$ (Fig. 5.9).

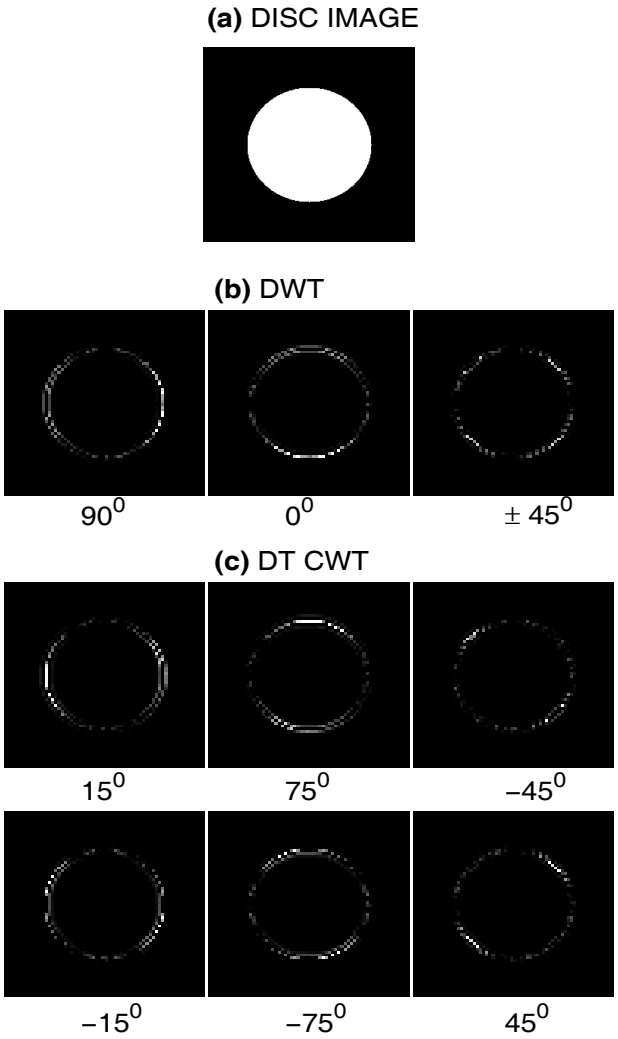


FIGURE 5.9. Complex filter response showing the orientations of the complex wavelets (a) original disc image, (b) DWT has three orientations of $0^0, 45^0, 90^0$, and (c) DT CWT has six directions oriented at $\pm 15^0, \pm 45^0, \pm 75^0$

The shift invariance of 2D DT CWT is illustrated in Fig. 5.10, shown are the contributions of the different levels for a circular disk image. Reconstructed images are obtained from the respective details coefficients of the different levels of the DWT and DT CWT. The classical wavelet transform shows aliasing compared to the DT CWT with images which look better and almost free of the aliasing effect.

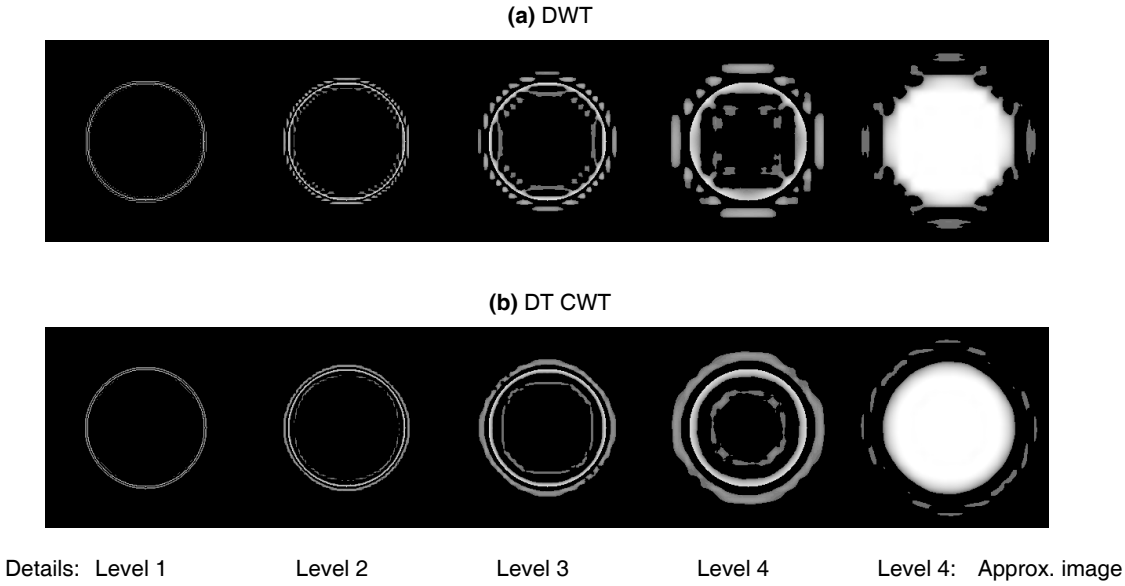


FIGURE 5.10. Comparison of the shift invariance for (a) DWT and (b) DT CWT

Fig. 5.11 illustrates the decomposition of the MR subimage both in the DT CWT and the DWT domain. For each decomposition only two levels are shown and the orientation of the corresponding filter is shown in the corner of each subband. From these figures it is clearly evident that the DT CWT [52] can distinguish the direction in many different orientations compared to the DWT.

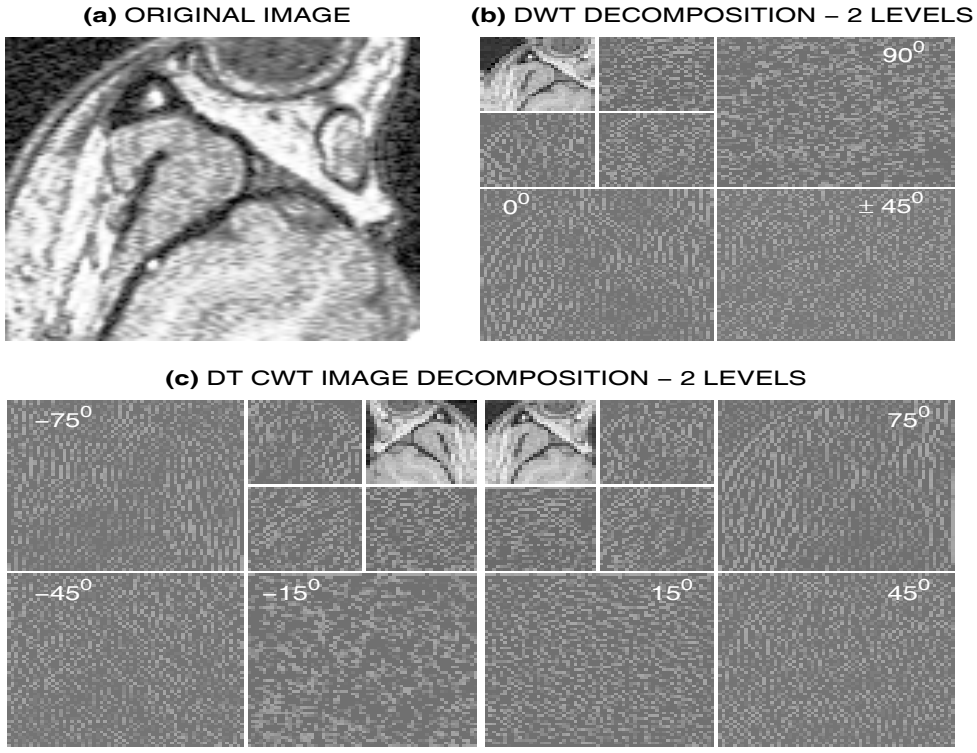


FIGURE 5.11. Two level decomposition of the (a) original MR image in both, (b) the DWT, and (c) DT CWT domain

5.4 Experimental Results

The shift invariance and directionality of the DT CWT may be applied in many areas of image processing like de-noising, feature extraction, object segmentation and image classification. Here we shall consider the de-noising application. For de-noising a soft thresholding and hard thresholding methods are used. The choice of threshold limits λ for each decomposition level and modification of the coefficients is defined by Eq. (5.9).

$$\overline{c}_s(k) = \begin{cases} \text{sign } c(k)(|\,c(k)\,| - \lambda) & \text{if } |\,c(k)\,| > \lambda \\ 0 & \text{if } |\,c(k)\,| \leq \lambda \end{cases} \tag{5.9}$$

To compare the efficiency of the DT CWT with the classical DWT the quantitative MSE, MAE and PSNR are used. In both cases the optimal thresholds points λ were selected to give the minimum square error from the original image.

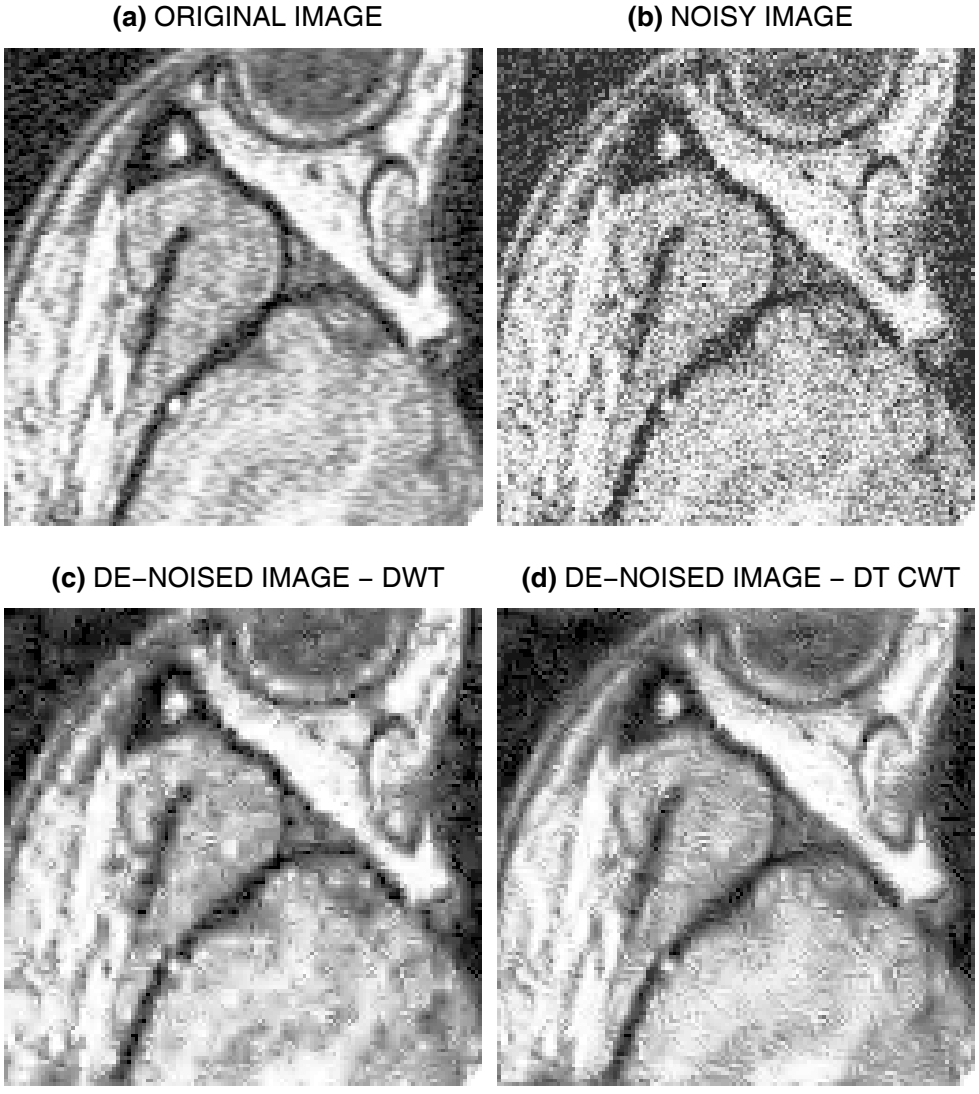


FIGURE 5.12. MR image scan: (a) original image, (b) with random noise added, (c) de-noised with DWT, and (d) de-noised with dual tree CWT

From Fig. 5.12(c) it may be seen that DWT introduces prominent worse artifacts, while the DT CWT provides a qualitatively restoration with a better optimal minimum MSE error (Fig. 5.14).

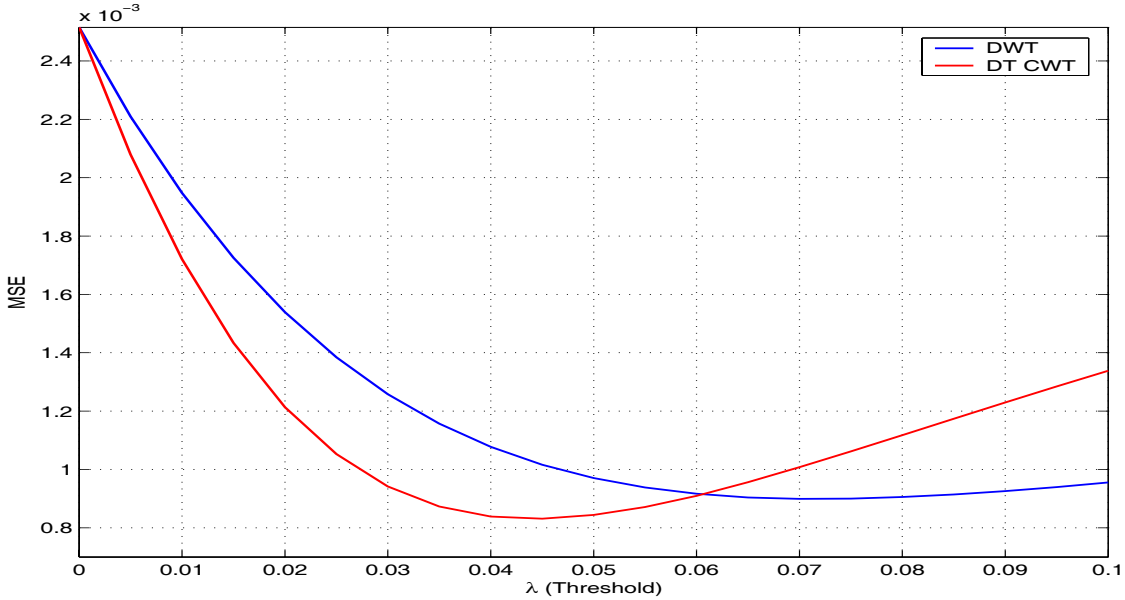


FIGURE 5.13. Optimal threshold values for the DWT and DT CWT methods obtained by using the soft thresholding algorithm

De-noising results obtained after implementing the hard threshold algorithm to both DWT and DT CWT methods are shown in Fig. 5.14. From the de-noised images it can be seen that the complex wavelet transform performs even better than the classical wavelet transform. The optimum MSE curves for the respective methods are shown in Fig. 5.15.

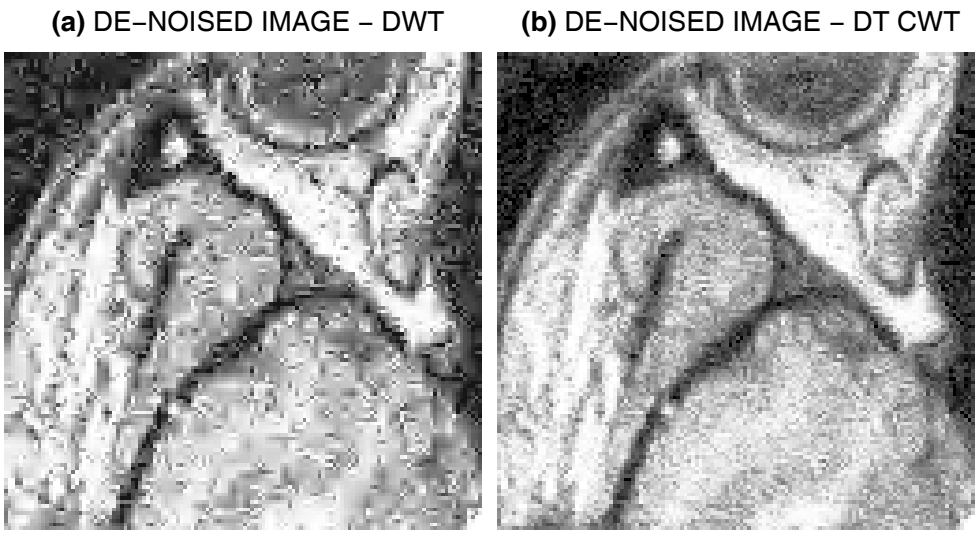


FIGURE 5.14. Noisy MR image (a) de-noised with DWT and (b) de-noised with DT CWT

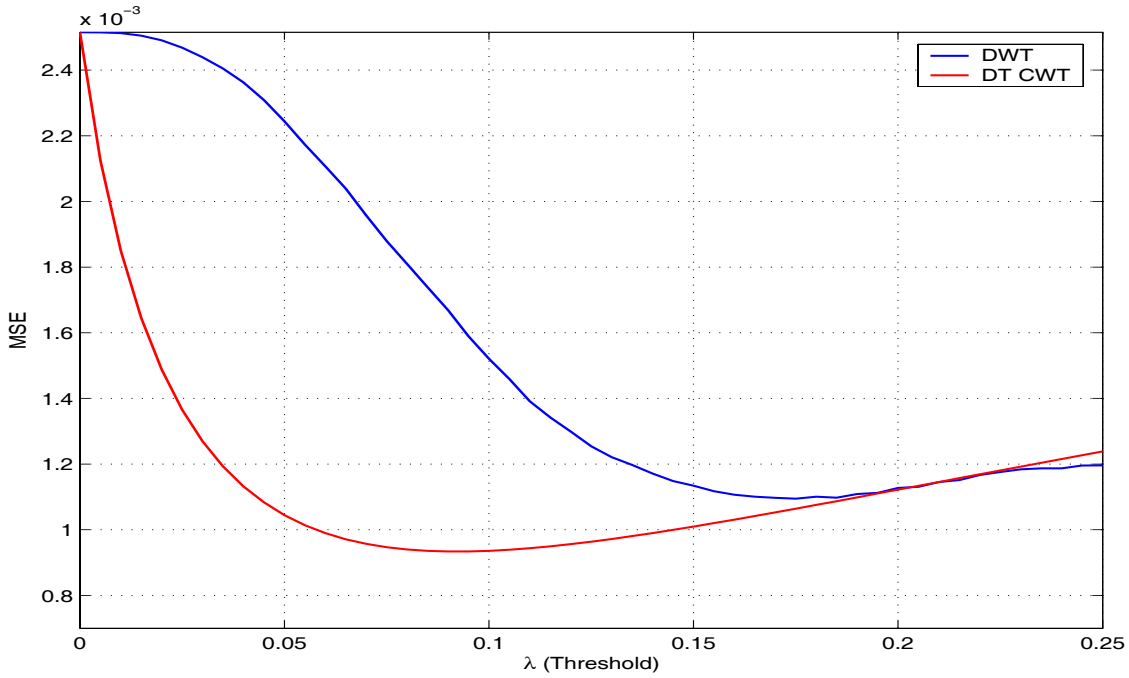


FIGURE 5.15. Optimal MSE curves for the DWT and DT CWT methods

5.5 Discussion

In this chapter, we introduced a dual-tree complex wavelet transform with approximate shift invariance, good directional selectivity, perfect reconstruction, limited redundancy and efficient computation. These properties are important for many applications in image processing including de-noising, deblurring, segmentation and classification. An application method based on DT CWT was carried out for the de-noising of MR images. Experimental results show a great effectiveness of the DT CWT in removing the noise compared to the classical DWT as shown in Tab. 5.3 by the increase of the $PSNR$ value and the reduction of the MSE.

TABLE 5.3. Comparison of the 2-D DWT and the 2-D DT CWT de-noising methods

Type of method	Hard thresholding			Soft thresholding		
	MSE	**MAE**	**PSNR** [dB]	**MSE**	**MAE**	**PSNR** [dB]
Noisy image	0.0025	0.0399	25.99	0.0025	0.0399	25.99
DWT	0.0011	0.0257	29.61	0.0009	0.0235	30.46
DT CWT	**0.0009**	**0.0241**	**30.29**	**0.0008**	**0.0226**	**30.81**

6
Image Segmentation and Feature Extraction

During the last three decades computerized segmentation techniques [36] have been studied extensively among numerous scientific fields including, e.g., pattern recognition, computer vision systems, and medical imaging. Segmentation is a technique that is used to find the objects of interest. Segmentation subdivides an image into its constituent regions or objects and it is an important step toward the analysis phase. It allows quantification and visualization of the objects of interest.

In the analysis of the objects in images it is essential that we can distinguish between the objects of interest and the rest e.g. the background. The techniques that are used to find the objects of interest are usually referred to as segmentation techniques - segmenting the foreground from background. In this chapter we will present the *watershed transform* which will be tested for the segmentation of the simulated image texture and MR image of the knee. We will present techniques for improving the quality of the segmentation result. The proposed segmentation technique uses a Vincent and Soilles immersion-based watershed algorithm which is either applied to the original or gradient image.

6.1 Image Segmentation

Segmentation appears to be a key issue in modern medical image analysis enabling numerous clinical applications. Segmentation is the process of assigning labels to pixels in 2-D images or voxels in 3-D images. The effect is that the image is split up into segments also called as regions or areas. In medical imaging this is essential for quantification of outlined structures and for 3-D visualization of relevant image data. For clinical purposes segmentation techniques using MRI has been widely used in monitoring brain infarctions, brain tumors. For example reliable 3-D images constructed from the segmented images helps understand the relation between the lesions and surrounding normal brain structures [17].

From a medical perspective [72], many researchers have noticed the importance of texture in MR and different research groups have been established. Texture will be more and more important as the resolution of the magnetic resonance (MR) equipment increases. The current resolution already allows the texture of bone and muscle to be easily seen

e.g. the human knee. By further increasing the scanning resolution it will be possible to reveal the texture of soft tissues in a similar way that microscopes reveal the texture of bones, cartilages, tendons or the skin. It is therefore important to develop methods that analyse textural information of MR images. However it must be noted that the textures encountered in MR images are significantly different from synthetic (artificial) textures which we will tend to be focussing on in this chapter.

Image-guided surgery is another important application of segmentation. Recent advances in technology have made it possible to acquire images of the patient while the surgery is taking place. The goal is then to segment relevant regions of interest and overlay them on an image of the patient to help guide the surgeon in his work. Segmentation is therefore a very important task in medical imaging. Different algorithms used include watershed algorithm which is evaluated for tumor segmentation in MR brain images.

6.1.1 Watershed Transform

The concept of watersheds and catchment basins are well known in topography. A *watershed* [31] is a ridge that divides areas drained by different river systems. A *catchment basin* is the area draining into a river or reservoir. The *watershed transform* applies these ideas to gray-scale image processing in a way that can be used to solve a variety of image segmentation problems.

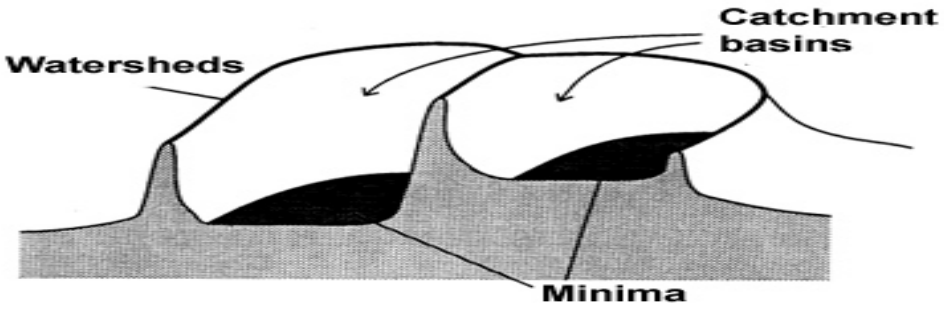


FIGURE 6.1. Watersheds and catchment basins

Using watershed transform a gray-scale image is taken as a topological surface where the values of $f(x, y)$ are interpreted as heights. By visualizing the image in Fig. 6.2 as the three-dimensional surface it is easily seen that water would collect in the two areas labelled as catchment basins. The watershed transform finds the catchment basins and ridge lines in a gray-scale image. For image segmentation problems, the key concept is to change the original image into another image whose catchment basins are the objects or regions we want to identify.

An efficient and accurate watershed algorithm was developed by Vincent and Soille [94] who used an *immersion* based approach to calculate the watersheds. The operation of their technique has been verified, the principle of proposed algorithm is simply described in Algorithm 2:

Algorithm 2: Watershed by immersion

- Visualize the image $f(x, y)$ as a topographic surface, with both valleys and mountains.
- Assume that there is a hole in each minima and the surface is immersed into a lake.
- The water will enter through the holes at the minima and flood the surface.
- To avoid the water coming from two different minima to meet, a dam is build whenever there would be a merge of the water.
- Finally, the only thing visible of the surface would be the dams. These dam walls are called the watershed lines.

The procedure results in a partitioning of the image in many catchment basins of which the borders define the watersheds. Of all the watershed transforms the immersion technique [33] was shown to be the most efficient one in terms of edge detection accuracy and processing time or speed of computation. The algorithm is well explained in Chapter 10 of [30], it can be used directly on the image, on an edge enhanced image or on a distance transformed image as we shall show in this section. So how are watersheds and catchment basins related to analyzing biomedical images and what is the connection to image processing? The connection is through computer analysis of objects in digital images. To understand the watershed transform a grayscale image is represented as a topographic surface (Fig. 6.2). For example, consider the image below:

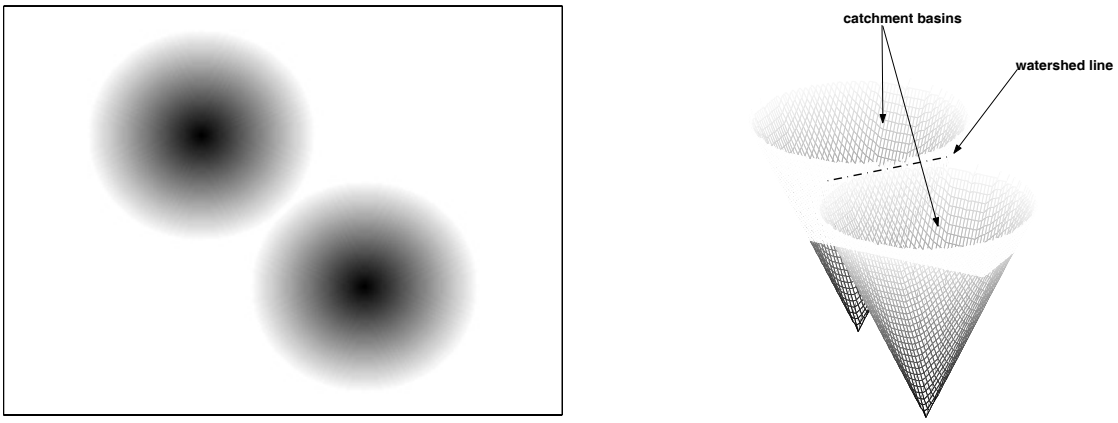


FIGURE 6.2. The blob and basin - a grayscale image and its representation as a topographic surface.

By imagining that bright areas are high and dark areas are low then the image look like the surface in Fig. 6.2, thus the image is viewed in terms of catchment basins and watershed lines. The key behind using the watershed transform for segmentation is to first change the grayscale image to a binary image. Here we will show how watershed segmentation is carried out on a binary image. As we shall see it is necessary to change the image into another image whose catchment basins are the objects we want to identify.

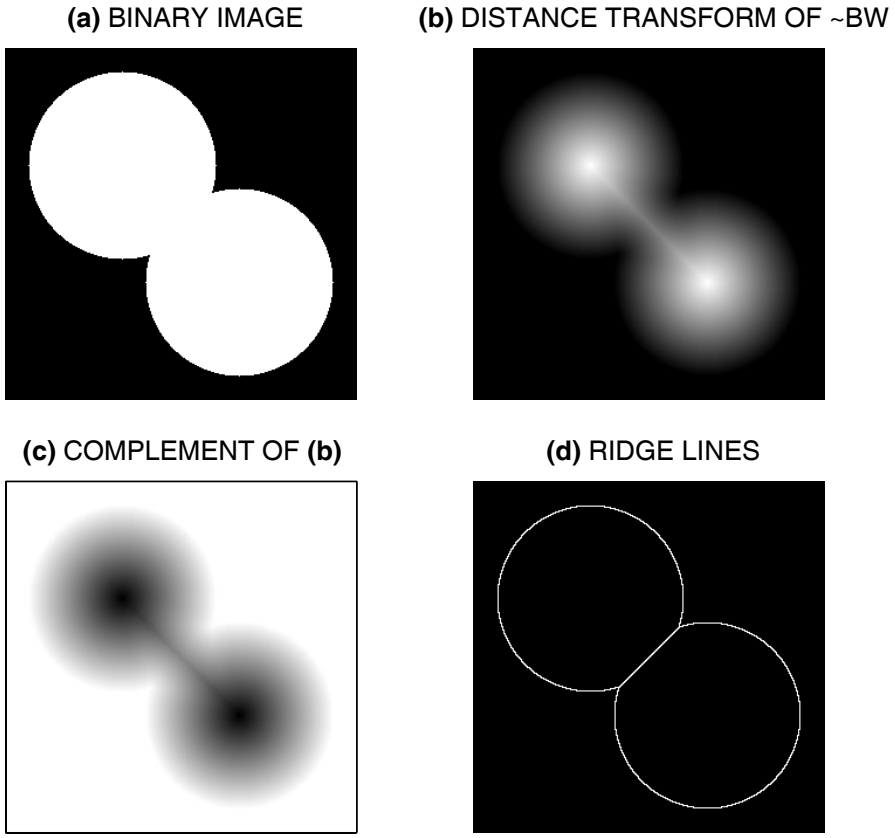


FIGURE 6.3. Segmenting a binary image - (a) Binary image of two touching objects, (b) Distance transform map obtained for the binary image, (c) complement of the distance transformed image, and (d) watershed segmentation performed using the distance map

Consider the task of separating the two touching objects in this binary image (Fig 6.3). How can we modify this image so its catchment basins are two circular objects? To do this we will use the distance transform [30]. The distance transform of a binary image is the distance from every pixel to the nearest nonzero-valued pixel, as this example shows. The distance transform of the complement of the binary image, looks like an image in Fig. 6.3(b). This image however has two bright areas covering the whole images. However we need to negate this image to turn the two bright areas into catchment basins (dark areas). Now there is one catchment basin for each object and the background has also been modified to be a catchment basin by setting its depth to infinite. (Appendix D)

6.1.2 Segmentation Results

Two different textural images are considered for the watershed segmentation. The proposed watershed algorithm is tested both for artificial texture image and real MR knee image. Segmentation of an artificial texture image which contains image objects of different shapes is shown in Fig. 6.4.

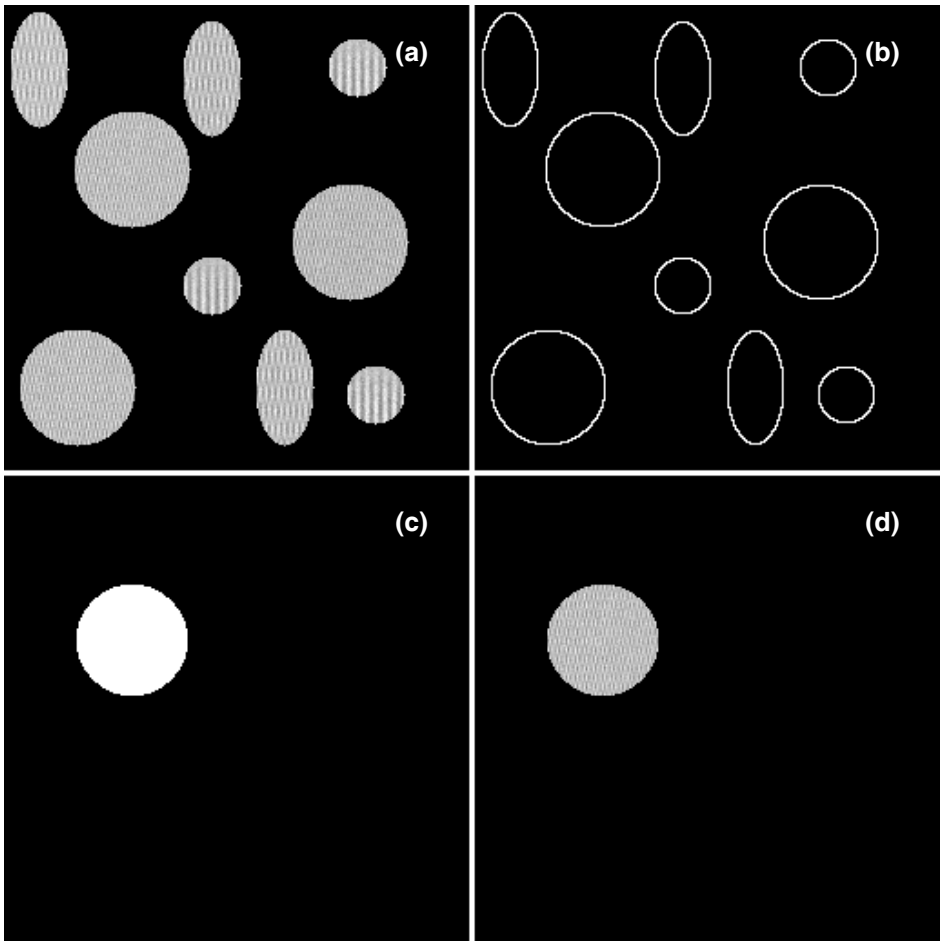


FIGURE 6.4. Segmentation of (a) simulated image texture, (b) watershed lines, (c) binary image of the selected object, and (d) segmented texture

The most interesting image is the human knee MRI in which three anatomical regions of muscle, bone, tissue and the background of the image are to be segmented. Fig. 6.5 shows segmentation of bone from MR knee image.

The bone is distinguished from the tissue, the background and the muscle can also be correctly segmented. For clinical application of image analysis, accurate determination of object boundary is often required and such a task is not trivial due to the complexity of biological objects. The watershed-based segmentation of the MR knee image presented here might be useful for expert users to extract object boundary from medical images reproducibly and accurately.

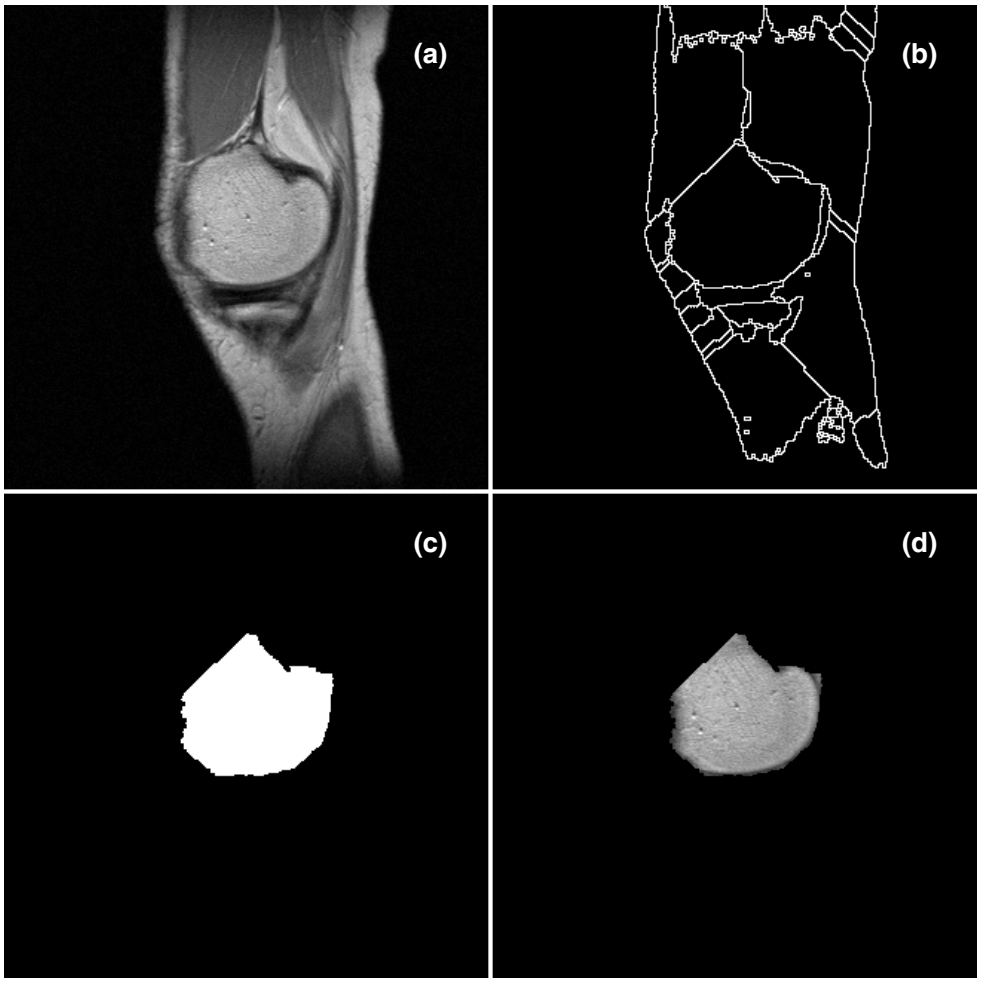


FIGURE 6.5. Segmentation of the femur in an MR image of the knee. (a) MRI of the knee, (b) watershed lines, (c) binary image of the selected object, and (d) segmented bone

Discussions

Watershed segmentation is sensitive to noise and it results in over-segmentation when noise is present due to an increased number of local minima, such that many catchment basins are subdivided. This is a limitation drawback of watershed segmentation. The first pre-processing step concerns the reduction of random noise as segmentation results are highly dependent on image noise. This is because noise tends to dislocate edges and hampers the detection of fine image detail. The goal in the de-noising is to smooth out the background regions in images while preserving features that represent object boundaries by using the discrete wavelet transform. Once the data has been smoothed, we can compute the watershed transform. It is important to note that there is no universally applicable segmentation technique that will work for all images and no segmentation technique is perfect. In this section we have presented a watershed method for segmentation of simulated image texture and MR image of the knee. The results indicate that the process of segmenting bone, tissue and muscles has been significantly improved.

6.2 Wavelet-based Feature Extraction

The main aim of this section is to study the major problems of texture analysis, including the classification of textures and propose solutions based on wavelet transform. Texture analysis is used in a variety of applications, including remote sensing, automated inspection, and medical image processing. Šonka [3] stresses that texture is scale dependent, therefore a multi-scale or multiresolution analysis of an image is required if texture is going to be analysed. For texture analysis and characterization a multichannel scale/orientation decomposition is performed using wavelet frame analysis. The main advantages of the wavelet transform, as a tool for analyzing signals, are: orthogonality, good spatial and frequency localization, and the ability to perform multiresolution decomposition.

To describe image features various methods are used however in this chapter we have proposed the use of image wavelet decomposition [21, 73], using wavelet coefficients at selected levels. For a classical wavelet transform method with a one level decomposition, the given image texture is decomposed into four subimages of low-low, low-high, high-low and high-high subbands. The energies of each subimage at selected decomposition levels are used as image features for the classification of the images. The energy of the wavelet coefficients at the kth scale is defined as

$$E_{sub,k} = \sum_{i=0}^{M_k-1} \sum_{j=0}^{N_k-1} (w_{sub,k}(i,j))^2 \tag{6.1}$$

where $k = 1, \ldots, J$, $sub = LH, HL, HH$ and M_k, N_k are width and height of the subimage at scale k. The subband energy shows the distribution of energy and has proven very efficient for texture characterization. The steps involved in extracting the texture feature vector from a grayscale image, are outlined in the following procedure (Algorithm 3):

Algorithm 3: Feature extraction

- Subject the grayscale image to a chosen level discrete wavelet decomposition using the chosen wavelet e.g. the Daubechies wavelet
- Calculate the energies (E_k) of the image detail subands.
- Select any two energies from the detail coefficients to form the feature vector

Summed squared coefficients at selected decomposition levels can then be used as definition of image features for their classification. This process can be performed in various ways owing to the wide range of wavelet functions which can be used for image decompo-

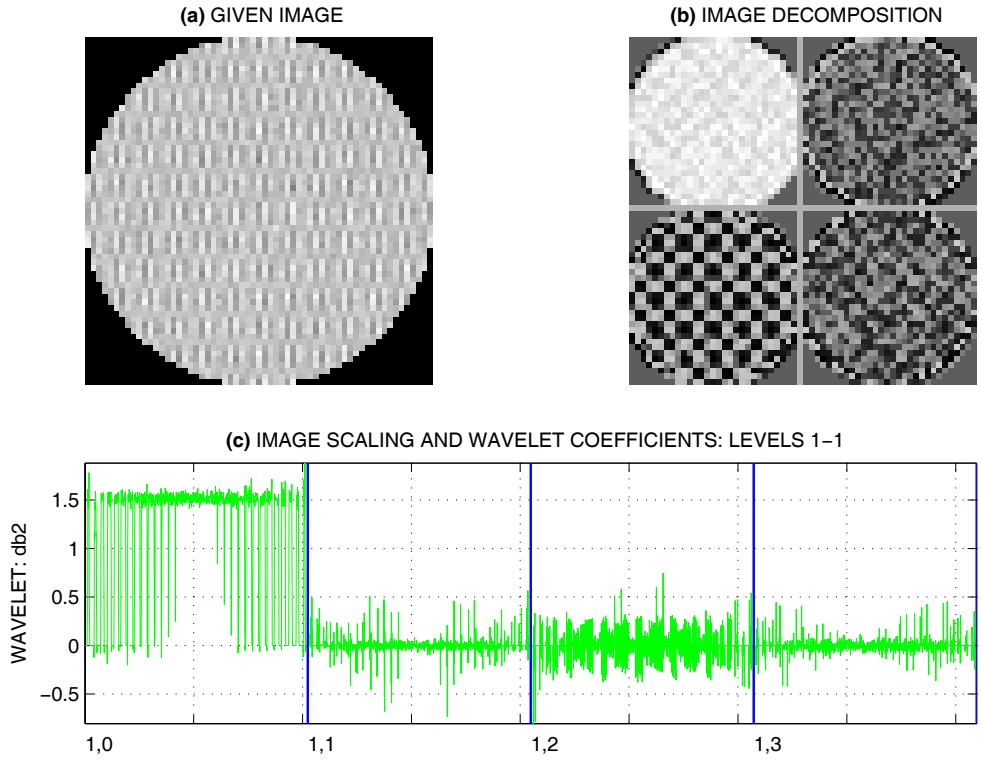


FIGURE 6.6. One level decomposition of (a) selected image texture, (b) its wavelet decomposition, and (c) the wavelet coefficients

sition. The choice of the level of image decomposition and selection of wavelet coefficients (horizontal, vertical and diagonal) determine the estimation of image features.

In some cases the image texture can be degraded by image artifacts and noise components so there is a need for image de-noising before the classification. This process can affect the results of image classification owing to the choice of the subimage subbands and processing of their features. The scheme of subband decomposition adopted for the purpose is shown in Fig. 6.6. For feature extraction only the HH, HL and LH subbands of each stage are considered.

Another method of feature extraction is the computation of (i) the mean of the absolute value of the coefficients in each subband - these features provide information about the frequency distribution of the image signal and (ii) the standard deviation of the coefficients in each subband - these features provide information about the frequency distribution of the image signal. The mean and standard deviation are calculated using Eq. (6.2) and Eq. (6.3) respectively.

$$c_{mean} = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} |w(i,j)| \tag{6.2}$$

$$c_{std} = \frac{1}{MN} \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} (|w(i,j)| - c_{mean})^2} \tag{6.3}$$

Eqs. (6.2) and (6.3) describe the process of getting the features where $w(i, j)$ is the wavelet coefficient for any subband of size $M \times N$. The wavelet features are obtained by using the above two equations which implement the mean and standard deviations of the magnitude of the wavelet coefficients. It is assumed that the local texture regions are spatially homogeneous, and the mean and the standard deviation of the magnitude of the transform coefficients are used to represent the region for classification and retrieval purposes. These feature vectors will be input to the neural network.

6.3 Experimental Results

The suggested subband energy feature extraction algorithm is implemented using an appropriately chosen wavelet transform. In this algorithm, we use the two-dimensional discrete wavelet transform to decompose each image texture object in Fig. 6.4(a) into four frequency bands of LL, LH, HL, and HH. Feature vectors are either formed from HL, HH or LH subbands because these high-frequency subbands tend to amplify the corresponding horizontal, vertical, and diagonal edge details. Selecting a proper set of features cannot only reduce the dimension of feature vectors, but also speed up and improve the classification. For our simulated image texture a set of 8 feature vectors are formed. The image texture features from different subbands are presented in Figs. 6.7 and 6.8. Each texture is represented by features in separate columns of pattern matrix `P`.

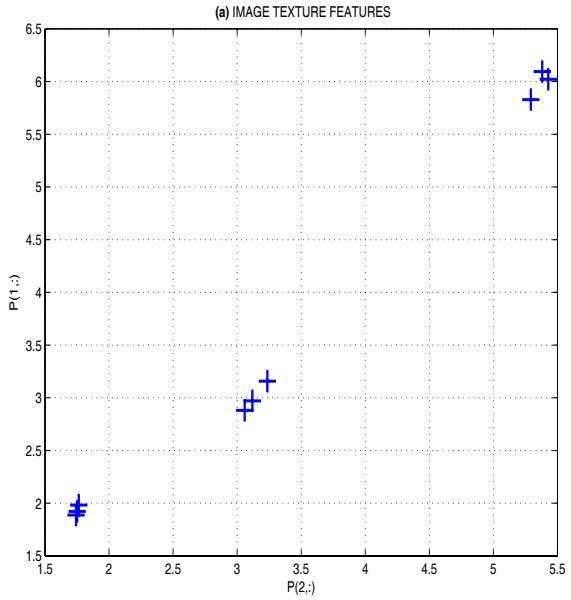


FIGURE 6.7. Image texture features from the horizontal (H) and diagonal (D) detail coefficients

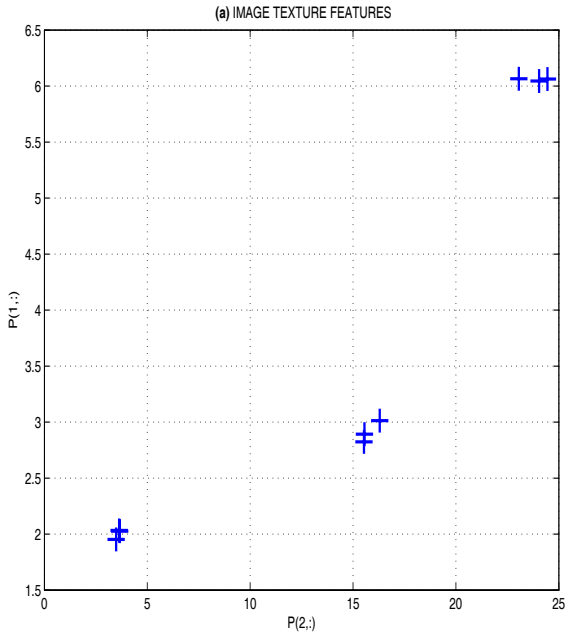


FIGURE 6.8. Image texture features from the horizontal (H) and vertical (V) detail coefficients

The pattern matrix P of size 2×8 for the feature vectors from the energies of the horizontal (H) and diagonal (D) detail coefficients at the first level of wavelet decomposition is

$$\mathtt{P} = \begin{bmatrix} 0.7833 & 0.5980 & 0.6853 & 0.2725 & 0.2518 & 0.3926 & 0.2780 & 0.4277 & 0.4051 \\ 0.9342 & 0.8325 & 0.8345 & 0.2930 & 0.3219 & 0.4389 & 0.3258 & 0.4415 & 0.3939 \end{bmatrix}$$

The feature vectors obtained by taking the the mean and standard deviation of the subband detail coefficients of each image texture object at different levels of decomposition are shown in Fig. 6.9 and Fig. 6.10. A feature vector of size 2×8 elements is formed. Two features are used here to enable a simple visualization even though more features can enable better classification results in many cases.

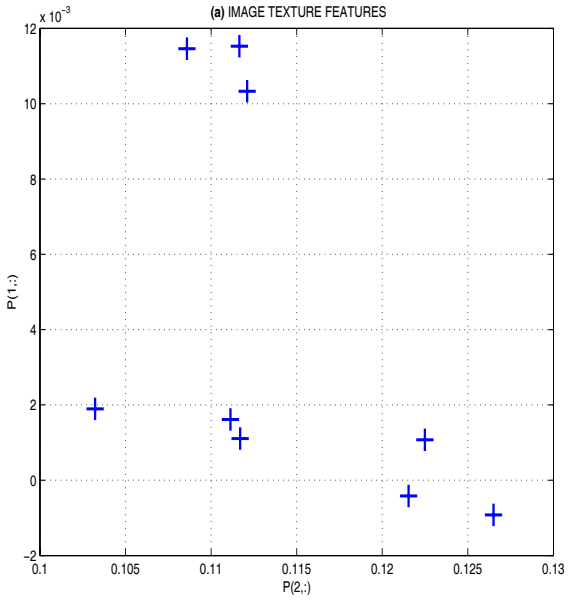


FIGURE 6.9. Image texture features from the mean and standard deviation of the first level detail coefficients

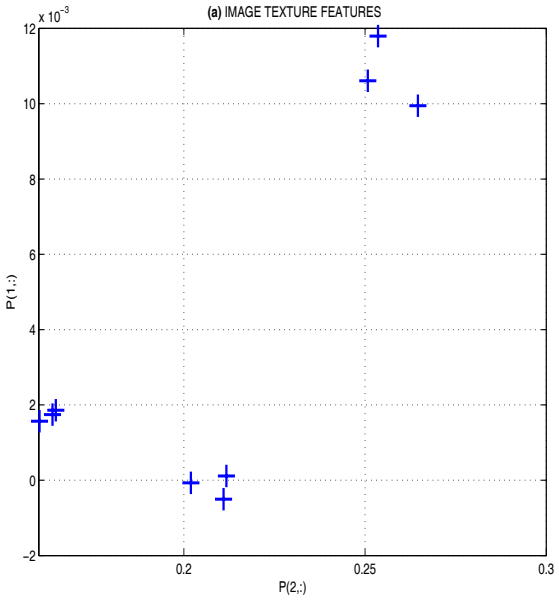


FIGURE 6.10. Image texture features from the mean of level 1 detail coefficients and standard deviation of level 2 detail coeff.

6.4 Discussion

We have proposed and investigated a texture feature extraction technique using the discrete wavelet transform. The experiments were conducted using an appropriate simulated image texture. The results obtained are very promising and showed that the proposed DWT based feature extractor was computationally efficient. It was observed that first level wavelet decomposition is sufficient enough to get the necessary input feature vector essential for classification of the artificial texture. To reduce the size of the input vector provided to the neural network, the mean and standard deviation was calculated for each subimage.

7

Image Texture Classification

This chapter presents the classification of image texture features using a competitive neural network. The classification is done by assigning data to one of the fixed number of possible classes. Optimization results in values of weights of output neurons pointing to typical class features. Classification generally comprises four steps shown in Algorithm 4:

```
Algorithm 4: Classification procedure

  • Pre-processing - feature extraction
  • Training - selection of the particular feature which best
    describes the pattern.
  • Decision - choice of suitable method for comparing the image
    patterns with the target patterns.
  • Assessing the accuracy of the classification.
```

The chapter proposes a method for classification of textures based on the (i) energies of image subbands and (ii) mean and standard deviations of image detail coefficients. We show that even with this relatively simple feature set, effective texture classification can be achieved.

Pattern recognition is the process of assigning patterns to one of a number of classes. Unsupervised classification (e.g.. clustering) in which the pattern is assigned to a hitherto unknown class. pattern recognition algorithms. Common operations include segmenting images to identify and separate the various objects present, measuring and classifying those objects. Pattern recognition continues by classifying the objects on the basis of their characteristics, applying statistical analysis to the results. Patterns of a class should be grouped or clustered together in pattern or feature space if decision space is to be partitioned objects near together must be similar objects far apart must be dissimilar.

An artificial neural network is simply defined as a network with interactions, in attempt to mimicke the brain. A neuron is a basic building-block of the brain. The artificial neuron model has a set of inputs, each of which has a weight and bias assigned to them. The net value of the neuron is calculated as weighted sum which is the sum of all the inputs multiplied by their specific weight. Each neuron has its own unique threshold value which is obtained from an activation (transfer) function and from this we get the neuron output.

7.1 Competitive Neural Network

Classification of image segments into a given number of classes using segments features is done by using a Kohonen competitive neural network, a schematic architecture of the network is shown in Fig. 7.1. The Kohonen network is an N-dimensional network, where N is the number of inputs. For simplicity, we will look at 2-dimensional networks. Each input node is connected to each output node. The dimension of the training patterns determines the number of input nodes. The networks are feed-forward networks that use an unsupervised training algorithm, and through a process called self-organization, configure the output units into a spatial map.

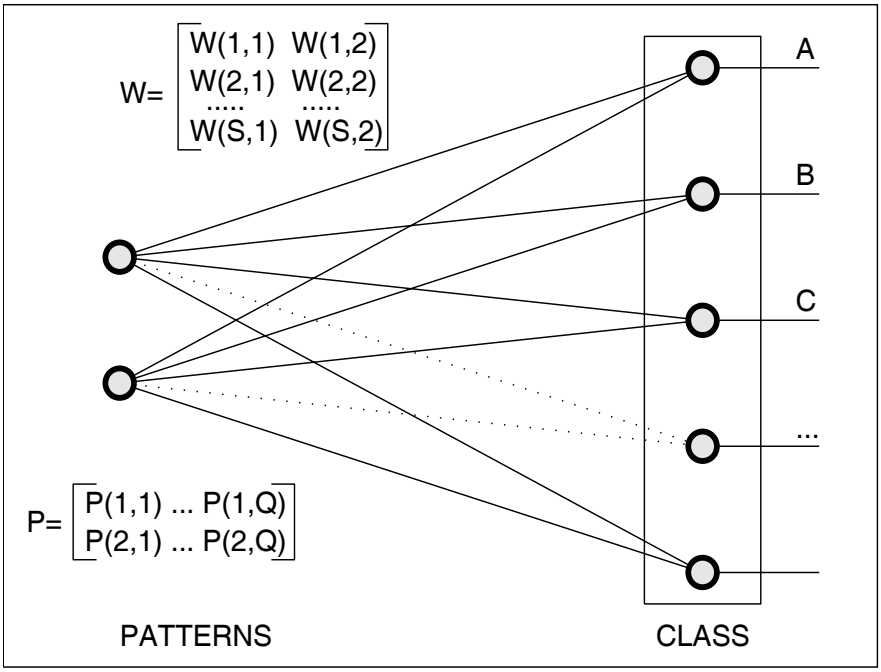


FIGURE 7.1. Competitive neural network

The network contains two layers of nodes - an input layer and a mapping (output) layer which are fully connected. Each image texture is represented by two features in separate columns of the pattern matrix P. The weight matrix W is the connection matrix for the input layer to the output layer. The number of nodes in the input layer is equal to the number of features associated with the input. A Kohonen's competitive learning algorithm that has been verified for our classification purposes can be summarized as follows (Algorithm 5):

Algorithm 5: Kohonen's competitive learning algorithm

- Let the pattern matrix P be denoted by

$$P = \begin{bmatrix} p_{11} & \cdots & p_{1k} & \cdots & p_{1Q} \\ p_{21} & \cdots & p_{2k} & \cdots & p_{1Q} \end{bmatrix}$$

  and the weight matrix W represented by

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ \cdots & \cdots \\ w_{S1} & w_{S2} \end{bmatrix} \tag{7.1}$$

- Step 1. Initialize all weights to random values. Each neuron in the Kohonen layer receives a complete copy of an input pattern.

- Step 2. Find the wining neuron. This neuron is the one with the smallest Euclidean distance d_i from the point represented by the input vector to the point represented by the weights, given by:

$$d_i = \sum_{j=1}^{R} \sqrt{(p_{jk} - w_{ij})} \tag{7.2}$$

  w_{ij} is the weight that connects the input node i to neuron j, where $i = 1, 2, \ldots, S$, $j = 1, 2, \ldots, R$, $k = 1, 2, \ldots, Q$. S is the number of output neurons (classes), R is the number of input nodes and Q is the number of columns of the input vector

- Step 3. For the winning neuron, the following learning formula is used to modify the weights:

$$\Delta w_{ij} = \alpha.(p_{jk} - w_{ij}) \tag{7.3}$$

  In other words, the change in the weights of the winner is just proportional to the difference between the input vector and the weight vector for the winning node. The constant of proportionality α is the learning rate. Update the weight vector of winning unit only with

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{7.4}$$

  The network learns by moving the winning weight vector towards the input vector.

- Step 4. Repeat Steps 1-3 using the entire input patterns. The weights are adjusted each time.

- Step 5. Repeat Step 4 for a specified number of times (epochs). Eventually each of the weight vectors would converge to the centroid of one cluster. At this point, the training is complete.

The following diagram (Fig. 7.2) illustrates what happens. Competitive layers can be understood better when their weight vectors and input vectors are shown graphically. The diagram shows 9 two-element input vectors represented as with '+' markers. The input vectors appear to fall into clusters. There are 3 output nodes and therefore, 3 weight vectors (the small circles). The weights are initially random, as in Fig. 7.2(a). A competitive network of eight neurons is used to classify the vectors into such clusters.

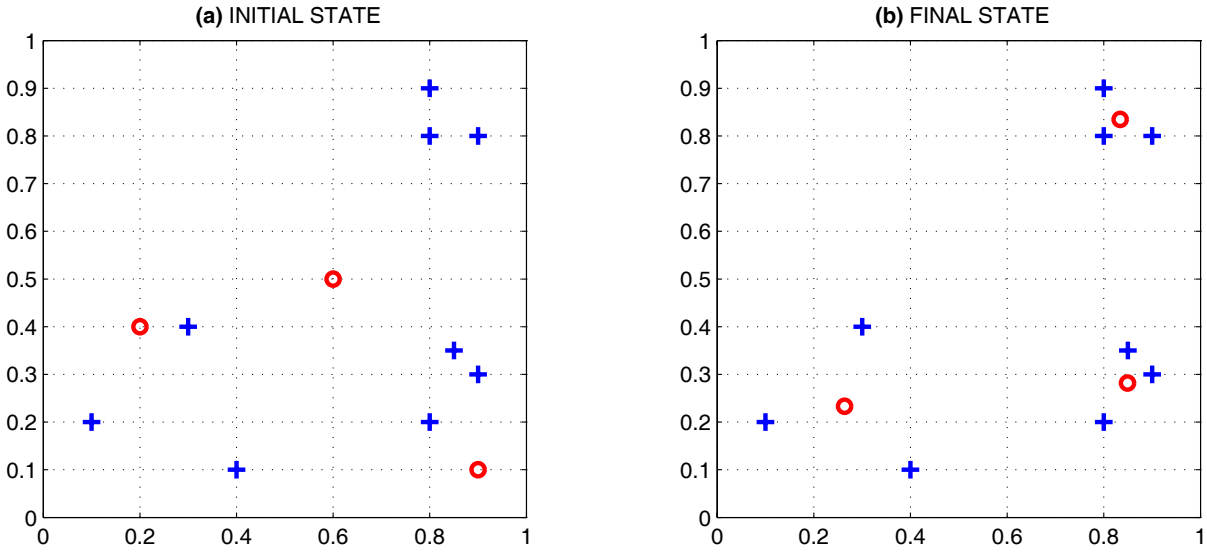


FIGURE 7.2. Competitive learning process. The dots represent the input vectors and the crosses represent the weights of the three output neurons (a) before learning and (b) after learning

After all of the inputs of the pattern matrix P are processed (usually after hundreds of repeated iterations), the result should be a 2D spatial organization of the input data organized into clusters of similar (neighbouring) regions. The topology of the Kohonen self-organizing feature map is represented as a two dimensional grid.

During training [35], the weight vectors 'move around'. After the learning process we have a final state of the network shown in Fig. 7.2(b). Each weight vector is at the 'center of gravity' of each input vector group. The weight vectors represent a kind of average of the input vectors in each group. When new samples are presented to a trained net, it is compared to the weight vectors which are the centroids of each cluster. By measuring the distance from the weight vectors using the neural net, the sample would be correctly grouped into the cluster to which it is closest. It must be noted some neurons never or rarely participate in the competition and don't become winners at all, their weight vectors are not be updated such type of neurons are called *dead neurons*.

MATLAB Implementation

Competitive learning algorithm for the neural network design is implemented by using the MATLAB Neural Network toolbox [22]. The MATLAB functions used for the classification are shown in Fig. 7.3.

```
% Neural Network Pattern Classification
% PAT -- Pattern matrix of size 2xN

% initializing the network
net = newc(minmax(PAT),S,klr,0);

%training the network
net = train(net,PAT);

% simulation
A = sim(net,PAT);

% class allocation
Ac = vec2ind(A);

% class analysis
ClassAnal=[[1:S]; ClassLength; ClassSTD; ClassTypical];
```

FIGURE 7.3. Neural network function algorithm

The important functions used include:

- `newc` - create a competitive layer it takes these inputs, `PAT` - matrix of min and max values for PAT input elements. `S` - Number of classes and `klr` - is the Kohonen learning rate. The layer has a weight from the input, and a bias b. If the samples are in clusters, then every time the winning weight vector moves towards a particular sample in one of the clusters. Eventually each of the weight vectors would converge to the centroid of one cluster. At this point, the training is complete.

- `train` - train a neural network by choosing the number of epochs, which represents the total number of times the entire set of training data will pass through the network structure.

- `sim` - simulates the neural network by taking the initialized net and network input matrix `PAT`

- `vec2ind` - convert vectors to class indices

Finally the `ClassAnal` matrix provides the full information concerning the classification of the images

$$\texttt{ClassAnal} = \begin{bmatrix} 1.0000 & 2.0000 & 3.0000 \\ 3.0000 & 3.0000 & 3.0000 \\ 0.0214 & 0.0460 & 0.0789 \\ 8.0000 & 1.0000 & 5.0000 \end{bmatrix}$$

The first row of the `ClassAnal` matrix gives the class indices, the second row is the number of image textures in each class, the third row is the standard deviation of the image textures and the last row gives the texture which is typical of the respective class e.g. for class index 2, texture image number 1 is the typical texture of the class.

7.2 Determination of Classification Boundaries

During the learning phase, classification boundaries are constructed that may depend on the distribution of samples in the learning set. A neural network has to assign every input feature vector to a class and group the input vectors into clusters. In the case that the learning process is successfully completed network weights belonging to separate output elements represent typical class features. The image corresponding to image features closest to separate pair of weights in this case represent a typical image of a selected class.

7.2.1 Empirical Computation of Class Boundaries

An empirical method which finds the class boundaries even for cases when the bias is non-zero has been proposed. The algorithm is described as follows (Algorithm 6):

Algorithm 6: Empirical computation

- Find the minimum and maximum values of the pattern matrix P
- Divide the region into a space of small squares
- Find the co-ordinates of the squares
- Determine the class of each square

The whole procedure described above is done for all the squares and each square is assigned its class and coloured according to which class it belongs. If the dimensions of the squares are very small the boundaries are very smooth and for example when $b = 0$ it can be shown that the boundary lines of Algorithm 6 and 7 closely relate to each other. By using the algorithm described above the class boundaries for the cases when $b \neq 0$ are easily found even though they would not be so smooth.

7.2.2 Mathematical Computation of Class Boundaries

The decision boundaries are determined completely by the weights w_{ij} and the biases b_i. Algorithm 7 below shows how the class boundaries are computed with respect to the Euclidean distance of the the feature vectors and the weights.

Algorithm 7: Mathematical computation

- Find the Euclidian distance between the elements of the pattern matrix and the centre weights

$$d_i = \sqrt{(p_{1k} - w_{i1})^2 + (p_{2k} - w_{i2})^2} + b_i$$

where $i = 1, 2, \ldots, S$ and $k = 1, 2, \ldots, Q$. S is the number of classes and Q is the number of columns in pattern matrix P.

- Equate each distance to each and every other distances and setting the biases to zero we will get equations of the boundary lines e.g. if $S = 3$ that means $d_1 = d_2$ and $d_1 = d_3$. By equating d_1 to d_2 and expressing p_{1k} as the dependent variable and p_{2k} as the independent variable we will have an equation in the form $p_{1k} = \mathrm{f}(p_{2k})$

$$p_{1k} = \frac{w_{11}^2 + w_{12}^2 - (w_{21}^2 + w_{22})^2 - 2p_{2k}(w_{12} - w_{22})}{2(w_{11} - w_{21})}$$

This is a linear equation since all the weights are constant except for the variables p_{1k} and p_{2k}. For d_1 and d_3 we have

$$p_{1k} = \frac{w_{11}^2 + w_{12}^2 - (w_{31}^2 + w_{32})^2 - 2p_{2k}(w_{12} - w_{32})}{2(w_{11} - w_{31})}$$

and finally for $d_2 = d_3$

$$p_{1k} = \frac{w_{21}^2 + w_{22}^2 - (w_{31}^2 + w_{32})^2 - 2p_{2k}(w_{22} - w_{32})}{2(w_{21} - w_{31})}$$

- Calculate points of intersection for each line with other lines
- Examine the given line segments between all points of intersection - if the middle point of the current segment forms a part of a boundary, the entire segment is a boundary line
- Class boundaries are found in such a way that the two-dimensional plane is divided into the same number of region boundaries as the number of classes

7.3 Classification Results of Artificial Image Texture

A competitive neural network is applied for the classification of the simulated image textures (Fig. 7.4). In texture classification the goal is to assign an object into one of a set of predefined set of texture classes. The classification is performed by using a subset of the subband energies that are measured to produce a feature vector that describes the texture. In these experiments, both mean absolute value and variance of frequency bands were used as energy measures in the feature sets. The algorithms described in the previous sections of the chapter have been verified for the classification of a simulated image texture presented in Fig. 7.4.

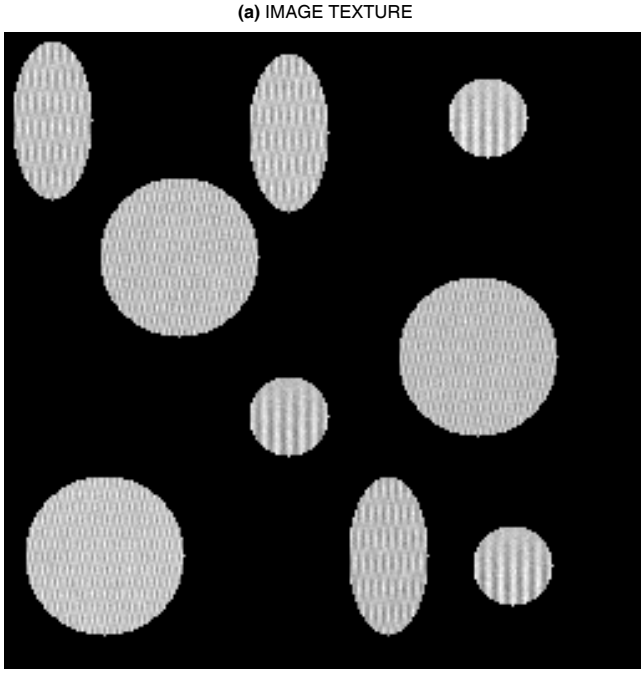


FIGURE 7.4. Simulated image texture

Results of the classification into three classes by a self-organizing neural network are shown in Fig. 7.5 and Fig. 7.6. The feature vectors are obtained from energies of the image detail subbands (horizontal (H), vertical (V) and diagonal (D) detail coefficients) for a chosen decomposition level. The classification results performed by using the mean and standard deviations (STD) of the image detail coefficients at each decomposition level are represented in Fig. 7.7 and Fig. 7.8. Features in wavelet domain have shown to be effective in texture representation for the classification purposes. The class boundaries for all the cases are found by setting the biases to zero. Euclidian distances between the elements of the pattern matrix and the centre weights have been used for the determination of the class boundaries (Algorithm 7).

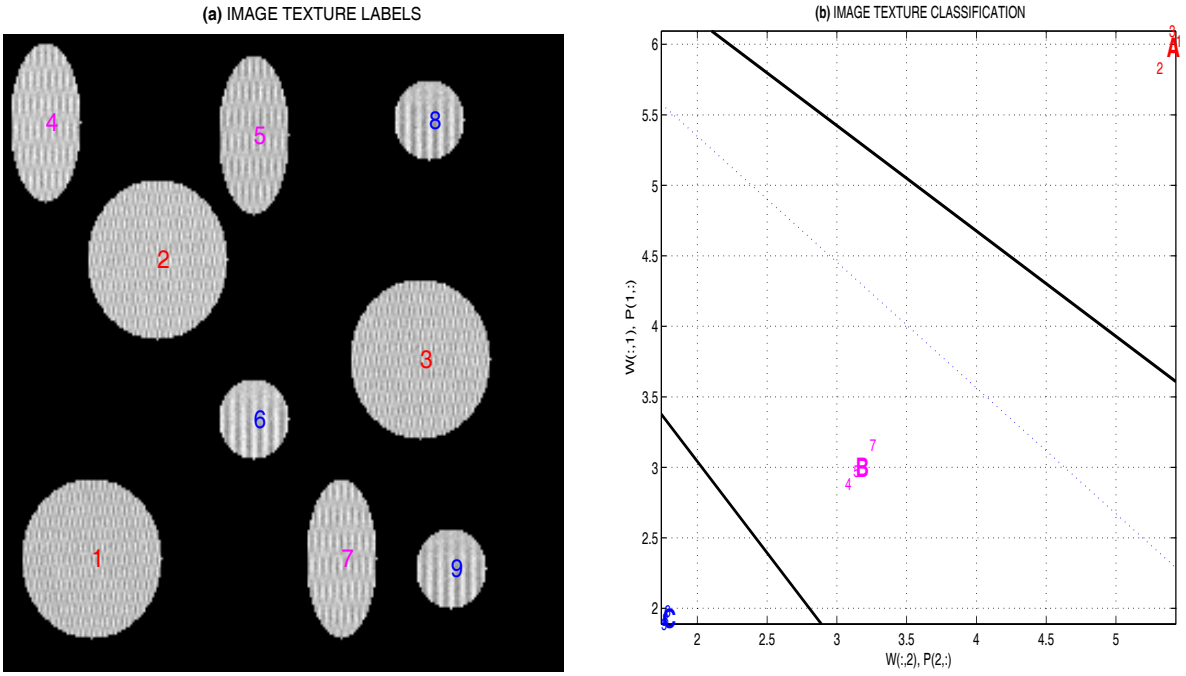


FIGURE 7.5. Class boundaries and classification of the simulated texture image into three classes. Horizontal (H) and diagonal (D) features from the first level decomposition.

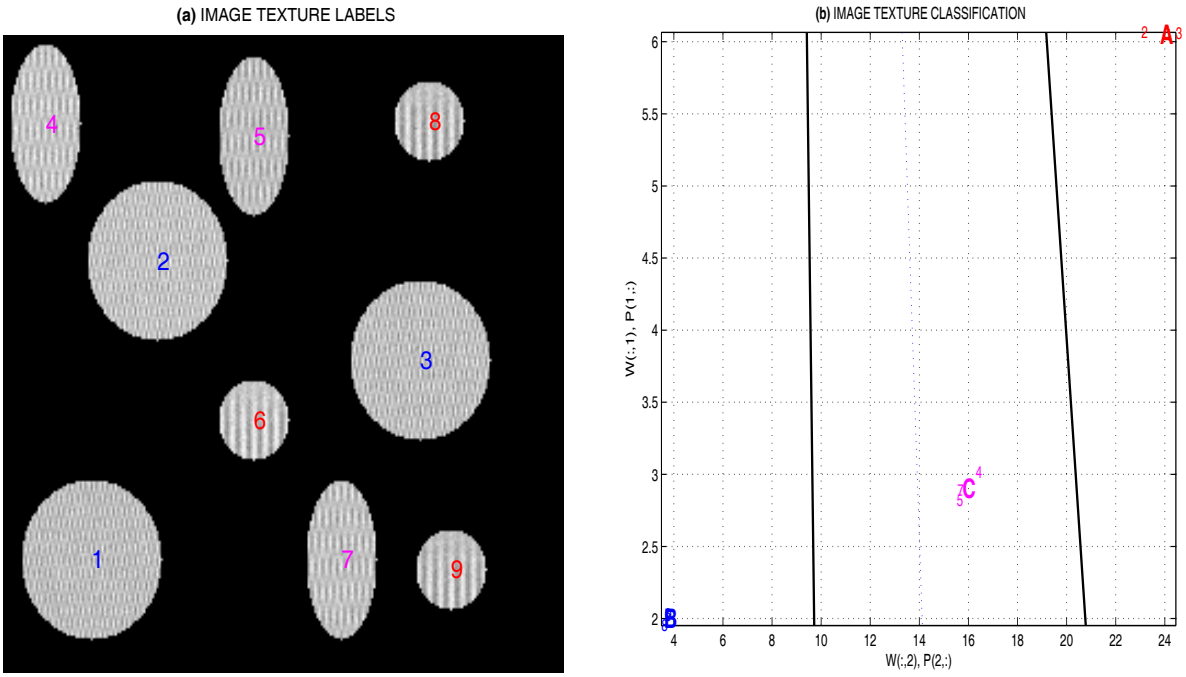


FIGURE 7.6. Class boundaries and classification of the simulated texture image into three classes. Horizontal (H) and vertical (V) features from the first level decomposition.

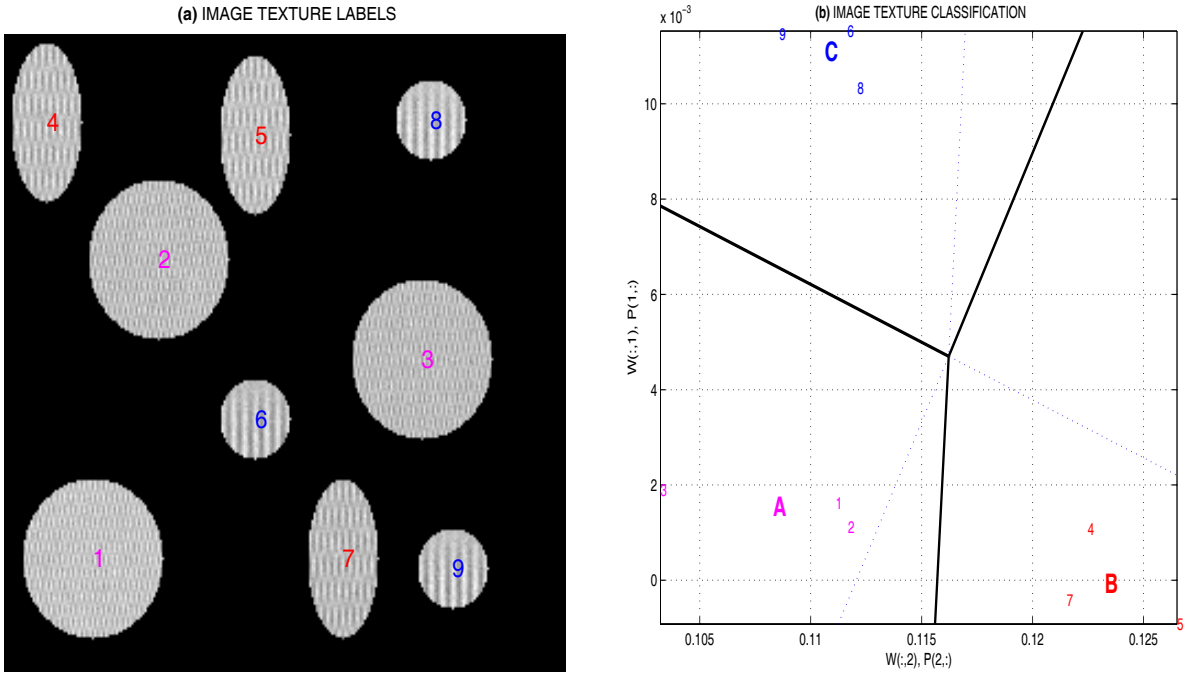


FIGURE 7.7. Class boundaries and classification of the simulated texture image into three classes. Features from the mean and the standard deviation of the first level detail coefficients.

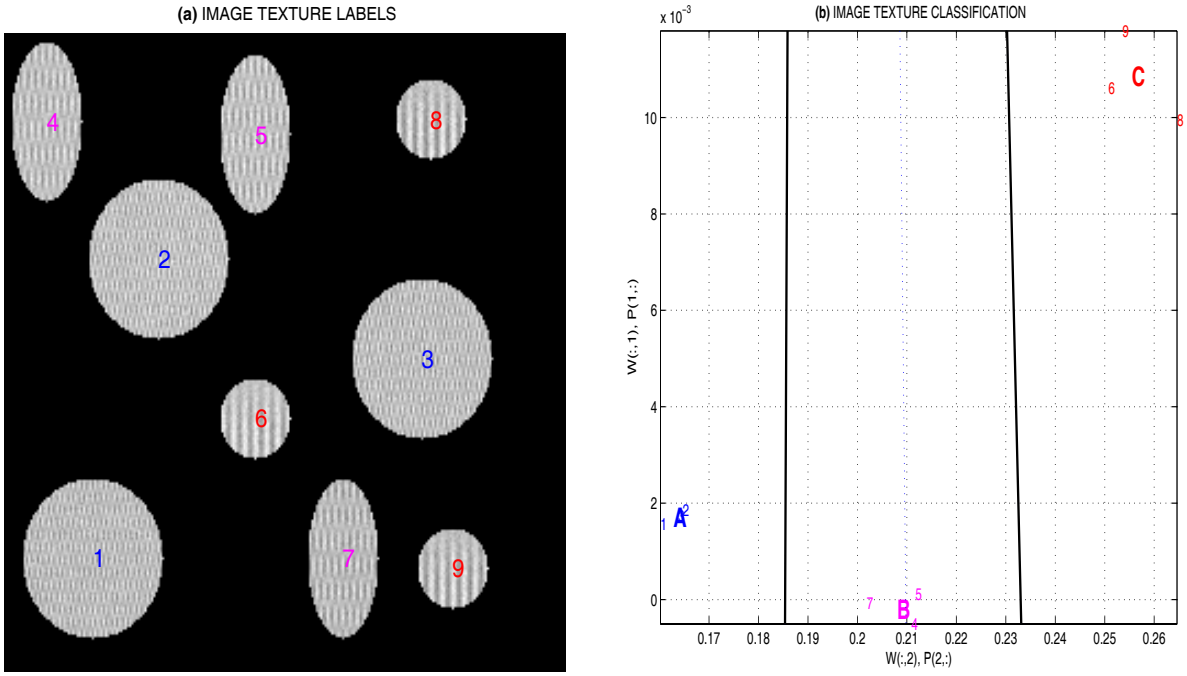


FIGURE 7.8. Class boundaries and classification of the simulated texture image into three classes. Features from the mean of the first level detail coefficients and the standard deviation of the second level image detail coefficients.

Clustering of the image features shows that image textures of similar structure belong to the same class. Fig. 7.9 presents the texture images typical for individual classes.

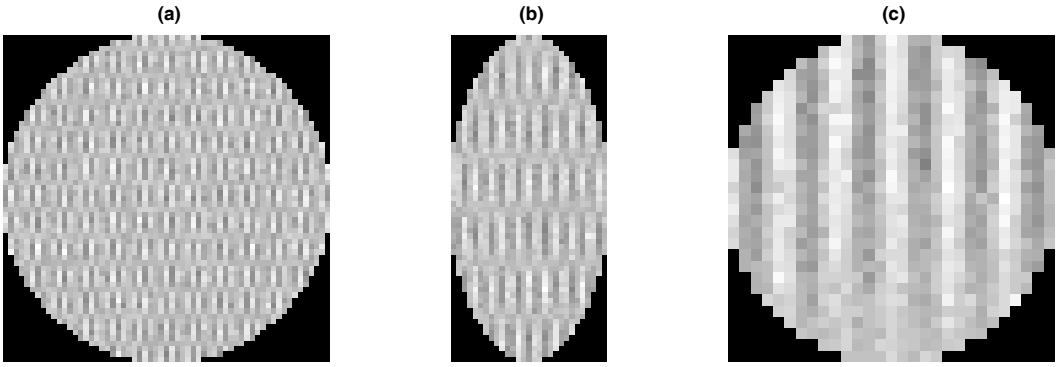


FIGURE 7.9. Typical image texture of separate classes

Results of texture classification into three classes for different features are compared in Tabs. 7.1 and 7.2. Each class is characterized by the variation of signal features distances from the typical class element which resulted from the classification process. It can be observed that features obtained by the discrete wavelet transform based upon different wavelet functions and different number of decomposition levels provide similar results.

TABLE 7.1. Comparison of image segments classification into three classes using a one level discrete wavelet decomposition (daubechies2 (db2) wavelet function) and taking the horizontal (H), vertical (V) or diagonal (D) coefficients feature source

	Typical Class Image / Number of Images		
	Class Standard Deviation		
Feature	A	B	C
H, D	8/3 0.0214	1/3 0.0460	5/3 0.0789
H, V	6/3 0.0395	1/3 0.2802	7/3 0.1831

TABLE 7.2. Comparison of image segments classification into three classes using wavelet decomposition into given number of levels (1, 2) by applying the daubechies2 (db2) wavelet function and using the mean (M) and standard deviation (STD) feature source

	Typical Class Image / Number of Images		
	Class Standard Deviation		
Feature	A	B	C
1M, 1STD	6/3 0.0005	4/3 0.0010	1/3 0.0012
1M, 2STD	3/3 0.0008	9/3 0.0036	4/3 0.0016

7.4 Discussion

Competitive learning neural networks have been successfully used as unsupervised training methods for simulated image texture classification. A correct classification was attained by using subband energy-based feature sets from image decompositions. The contribution in the use of wavelet transform for image classification was discussed. Mathematical basis of the discrete wavelet transform and the numerical experiments proved that image features based on wavelet transform coefficients can be used very efficiently for image texture classification.

However there are problems which are associated with the choice of wavelets, suitable level of decomposition and which subband details to consider for the feature extraction whether the horizontal, vertical or diagonal details. Using other transform methods such as the complex wavelet transform [45, 34], the classification of image textures can also be greatly enhanced. Another major problem of neural networks when used as classifiers lies in their lack of good rejection capabilities: a neural network has to assign every input feature vector to a class even though some vectors may not belong to any of the learnt classes.

It is assumed that further research will be devoted to more sophisticated methods of image feature classification of real MR images. Brain tissue classification from MR images help MR image feature analysis for automatic disease classification. The key element is to choose those MR image features that best discriminate disease classes.

8
Conclusions

The thesis has been devoted to the de-noising algorithms based upon the discrete wavelet transform that can be applied to enhance noisy multidimensional MR data sets i.e. 2-D image slices and 3-D image volumes. The trade-off between noise elimination and detail preservation was analyzed using the MSE, MAE, PSNR and visual criteria. Thus a comparison between the qualities and performance of various wavelet functions were deduced using these criteria. Effectiveness of each filter is dependent on the type of image, the error criterion used, the nature and amount of contaminating noise. It was seen that the fourth-order Daubechies ($db4$) wavelet function performed well for the de-noising of the random noise both in the cases of simulated and real MR data sets, this can be clearly seen with its considerable improvement in PSNR and producing visually more pleasing images. The advantage of the DWT is in its flexibility caused by the choice of various wavelet functions. Since normal random noise constitutes most of the high frequency components by designing a suitable filter the noise was suppressed to a minimum.

For representing the image data sets, we have used a wavelet transform that is, a multiscale, multiorientation invertible subband representation. The wavelet transform is used to decompose an image into a low-frequency component and a set of higher-frequency details at varying scales of resolution. By analyzing the wavelet transform coefficients, high frequency details which correspond to image noise can be eliminated by using different threshold methods. This way noise is reduced from an image without losing the anatomical information which is of interest to medical doctors. Wavelets have demonstrated to be a very powerful tool for analysis, processing and synthesis of relevant image features.

The implementation of the DT CWT in the de-noising of MR images has been also examined. As shown by the experimental results for most of the de-noising applications the DT CWT gives better results than the classical wavelet transform. Complex wavelets have also proved that they are important tools which can used for other implications of the research such as segmentation and classification. It can be concluded that the DWT and DT CWT are important mathematical tools in biomedical image processing. Finally it is assumed that processing of MR images will result in further methods of image de-noising, edge detection and their enhancement using methods like spline interpolation, resolution enhancement, image reconstruction and feature extraction.

A watershed algorithm for segmentation of MR images was proposed. The principle of the method was described and tested, first, with artificial textures yielding fairly good results. The algorithm was then used to segment a human knee MR image. The anatomical regions: muscle, bone and tissue, and the background could be distinguished. The results show that this method can be comparable to the widely available commercial software, and it performs reasonably well with respect to manual segmentation.

The feature extraction method was computationally efficient in the determination of the feature vectors using coefficients of different subbands. The classification methods studied were neural network methods based on self-organizing network. The methods can be applied for the classification of MR images especially for diagnosis of diseases.

The work presented in this thesis can be extended in several directions. Here we present several of the research directions which might be followed for the further applications in the segmentation and classification of real MR image features rather than the simulated image textures we have used. We review the possible improvements that can be brought to the segmentation algorithms and classification methods proposed in this work.

Efficient texture representation is important for the retrieval of image data. The principle lies in the computation of a small set of of texture describing features for each image in a database so that it is possible to search in the database for images containing a certain feature. DT CWT has been found to be useful for classification as documented in [34].

For clinical application of image analysis, accurate determination of object boundary is often required and such a task is not trivial due to the complexity of biological objects. This thesis presented a watershed-based segmentation algorithm which might be useful for expert users to extract object boundary from medical images reproducibly and accurately. Our on-going research effort is to extend the present algorithm such that it can also distinguish the texture and ultimately incorporate such textural descriptors to real MR brain images for a cooperative boundary or region segmentation framework.

In the thesis we have shown that watershed method provides promising segmentation results which are useful for the segmentation of MR image texture. Better results can be achieved in the case of more precise image segmentation and image pre-processing for noise rejection and artifacts removal. These possibilities can be further extended by the use of complex wavelet functions [34] applied for image decomposition.

Selected results are available from the author's web page: $http://dsp.vscht.cz$

9

References

[1] A. N. Akansu and R. A. Haddad. *Multiresolution Signal Decomposition: Transforms, Subbands, Wavelets*. Academic Press, 2000.

[2] A. Bilgin and M.W. Marcellin. Three dimensional Image Compression with Integer Wavelet Transforms. *Applied Optics*, 39, April 2000.

[3] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, Engelwood Cliffs, U.S.A., third edition, 1994.

[4] Michael Breakspear, Michael Brammer, Edward Bullmore, Pritha Das, and Leanne Williams. Spatiotemporal Wavelet Resampling for Functional Neuroimaging Data. *Human Brain Mapping*, 23(1):1–25, 2004.

[5] A. Bruce, D. Donoho, and Hong-Ye Gao. Wavelet Analysis. *IEEE Spectrum*, 33(10):26–35, October 1996.

[6] E. N. Bruce. *Biomedical Signal Processing and Signal Modeling*. John Wiley & Sons, 2000.

[7] Edward Bullmore, Jalal Fadili, Michael Breakspear, Raymond Salvador, John Suckling, and Mick Brammer. Wavelets and Statistical Analysis of Functional Magnetic Resonance Images of the Human Brain. *Statistical Methods in Medical Research*, 12(5):375 – 399, 2003.

[8] Edward Bullmore, Jalal Fadili, Voichita Maxim, Levent Sendur, John Suckling Brandon Whitcher, Mick Brammer, and Michael Breakspear. Wavelets and Functional Magnetic Resonance Imaging of the Human Brain. *NeuroImage*, 23(Sup 1):234–249, 2004.

[9] R. L. Burden and J. D. Faries. *Numerical Analysis*. Brooks/Cole, seventh edition, 2001.

[10] C. S. Burrus, R.A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms*. Prentice Hall, New Jersey, 1998.

[11] K. R. Castleman. *Digital Image Processing*. Prentice Hall Press, 1996.

[12] Y. T. Chan. *Wavelet Basics*. Kluwer Academic Publishers, Boston, 1995.

[13] S. Chang, B. Yu, and M. Vetterli. Spatially adaptive wavelet thresholding with context modeling for image de-noising. *ICIP*, 1:535–539, 1998.

[14] R. Chellappa. *Digital Image Processing*. IEEE Computer Society Press, Los Alamitos, USA, 1992.

[15] L. Cohen. *Time–frequency Analysis*. Prentice Hall, Englewood Cliffs (NJ), 1995.

[16] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[17] P. Dastidar. Overview of Neuroradiological MRI . *International Journal of Bioelectromagnetism*, 1(1), 1999.

[18] I. Daubechies. Orthonormal Bases of Compactly Supported Wavelets. *Communications on Pure Applied Mathematics*, 41:909–996, 1988.

[19] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Trans. Info. Theory*, 36:961–1005, 1990.

[20] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1992. 357p.

[21] L. Debnath. *Wavelets and Signal Processing*. Birkhauser Boston, 2003.

[22] H. Demuth and M. Beale. *Neural Network Toolbox*. The MathWorks, Inc., Natick, Massachusetts 01760, 1998.

[23] D.L. Donoho. De-noising by Soft Thresholding. *IEEE Trans. on Information Theory*, 38(2):613–627, 1995.

[24] D.L. Donoho and I.M. Johnstone. Ideal Spatial Adaption via Wavelet Shrinkage. *Biometrika*, 81(3):425–455, 1994.

[25] Jalal Fadili and Edward Bullmore. Wavelet-Generalized Least Squares: A New Blu Estimator of Linear Regression Models with 1/f Errors. *Neuroimage*, 15(1):217 – 232, 2002.

[26] Jalal Fadili and Edward Bullmore. A Comparative Evaluation of Wavelet-Based Methods for Hypothesis Testing of Brain Activation Maps. *NeuroImage*, 23(3):1112 – 1128, 2004.

[27] F. Fernandes. *Directional, Shift-Insensitive, Complex wavelet Transforms with Controllable Redundancy*. PhD thesis, Rice University, August 2001.

[28] F. Fernandes, I. Selesnick, R. van Spaendonck, and C. Burrus. Complex Wavelet Transforms with Allpass Filters. *Signal Processing*, 40(9):1689–1706, August 2003.

[29] H. Frey, A. Lahtinen, T. Heinonen, and P. Dastidar. Clinical Application of MRI Image Processing in Neurology. *International Journal of Bioelectromagnetism*, 1(1), 1999.

[30] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (2nd Edition)*. Addison-Wesley, 2002.

[31] R. C. Gonzalez, R. E. Woods, and S. L. Eddins. *Digital Image Processing Using MATLAB*. Prentice Hall, Englewood Cliffs, NJ, USA, 2004.

[32] H. Guo, G.A. Sitton, and C.S. Burrus. The Quick Fourier Transform: An FFT Based on Symmetries. *IEEE Trans. on Signal Processing*, 46(2):335–341, February 1998.

[33] D. Hagyard, M. Razaz, and P. Atkin. Analysis of watershed algorithms for greyscale images. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3, pages 41–44, 1996.

[34] S. Hatipoglu, S. K. Mitra, and N. G. Kingsbury. Texture classification using dual–tree complex wavelet transform. *Image Processing And Its Applications*, (465):344–347, 1999.

[35] S. Haykin. *Neural Networks*. IEEE Press, New York, 1994.

[36] T. Heinonen, P. Dastidar, H. Frey, and H. Eskola. Applications of MR Image Segmentation. *International Journal of Bioelectromagnetism*, 1(1), 1999.

[37] V. K. Ingle and J. G. Proakis. *Digital Signal Processing Using MATLAB*. Brooks/Cole, 2000.

[38] K. Jafari-Khouzani and H. Soltanian Zadeh. Rotation-Invariant Multiresolution Texture Analysis Using Radon and Wavelet Transforms. *IEEE Transactions on Image Processing*, 14(6):783 – 795, 2005.

[39] A. Jalobeanu, L. Blanc-Feraud, and J. Zerubia. Satellite Image Deconvolution Using Complex Wavelet Packets. Technical report, Institut National de Recherche en Informatique et en Automatique, France, 2000.

[40] N. Kalouptsidis. *Signal Processing Systems: Theory and Design*. John Wiley and Sons, Inc., UK, 1997.

[41] C. Kamath and I.K. Fodor. Undecimated Wavelet Transforms for Image De-noising. *Applied Scientific Computing, Lawrence Livermore National Laboratory*, November 2002.

[42] H. Khalil and S. Shaheen. Three Dimensional Video Compression. *IEEE Transactions on Image Processing*, 8:762–773, June 1999.

[43] N. Kingsbury and J. Magarey. Motion Estimation Using Complex Wavelets. Technical report, Cambridge University Engineering Department, UK, 1995.

[44] N. G. Kingsbury. The Dual–Tree Complex Wavelet Transform: A New Technique for Shift Invariance and Directional Filters . In *Proceedings of the IEEE Digital Signal Processing Workshop*, 1998.

[45] N. G. Kingsbury. Image Processing with Complex Wavelets. *Phil. Trans. Royal Society London*, 1999.

[46] N. G. Kingsbury and J. F. A. Magarey. Wavelet Transforms in Image Processing. *Proc. First European Conference on Signal Analysis and Prediction*, pages 23–24, June 1997.

[47] R. Klette and P. Zamperoni. *Handbook of Image Processing Operators*. John Wiley & Sons, New York, 1994.

[48] R. Kuc. *Introduction to Digital Signal Processing*. McGraw-Hill, New York, 1982.

[49] J. Kukal, D. Majerová, V. Musoko, A. Procházka, and A. Pavelka. Nonlinear Filtering of 3D MRI in MATLAB. In *The 11th Scientific Conference MATLAB2003*. Humusoft, 2002.

[50] Sheng-Tun Li, Shih-Wei Chou, and Jeng-Jong Pan. Multi-resolution spatio-temporal data mining for the study of air pollutant regionalization. In *33rd Annual Hawaii International Conference on System Sciences(HICSS)*, 2000.

[51] J.S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice Hall, 1990.

[52] P. Loo. *Digital Watermarking Using Complex Wavelets*. PhD thesis, Signal Processing and Communications Laboratory, University of Cambridge, 2002.

[53] A.K. Louis, P. Mass, and A. Rieder. *Wavelets: Theory and Applications*. John Wiley & Sons, Chichester, U.K., 1997.

[54] S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.

[55] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 2nd edition, 1999.

[56] M. Misiti, Y. Misiti, G. Oppenheim, and J.-M. Poggi. *Wavelet Toolbox*. The MathWorks, Inc., Natick, Massachusetts 01760, March 1996.

[57] V. Musoko, M. Kolínová, and A. Procházka. Image Classification Using Competitive Neural Networks. In *The 12th Scientific Conference MATLAB2004*. Humusoft, 2002.

[58] V. Musoko and A. Procházka. Non-Linear Median Filtering Of Biomedical Images. In *The 10th Scientific Conference MATLAB2002*. Humusoft, 2002.

[59] V. Musoko and A. Procházka. Three-Dimensional Biomedical Image De-Noising. In *The 15th International Conference on Process Control*. STU, Slovakia, 2003.

[60] V. Musoko and A. Procházka. Complex Wavelet Transform in Signal and Image Analysis. In *The 14th International Conference on Process Control*. VŠCHT Pardubice, 2004.

[61] V. Musoko, A. Procházka, and A. Pavelka. Implementation of a Matlab Web Server for Internet-based Processing of Biomedical Images. In *The 11th Scientific Conference MATLAB2003*. Humusoft, 2002.

[62] L. Najman and M. Couprie. Watershed algorithms and contrast preservation. In *Discrete geometry for computer imagery*, volume 2886 of *Lecture Notes in Computer Science*, pages 62–71. Springer, 2003.

[63] J. Neumann and G. Steidl. Dual–Tree Complex Wavelet Transform in the Frequency Domain and an Application to Signal Classification. *International Journal of Wavelets, Multiresolution and Information Processing IJWMIP*, 2004.

[64] D. E. Newland. *An Introduction to Random Vibrations, Spectral and Wavelet Analysis*. Longman Scientific & Technical, Essex, U.K., third edition, 1994.

[65] M. Petrou and P. Bosdogianni. *Image Processing: The Fundamentals*. Wiley, 1999.

[66] P. Pinnamaneni and J. Meyer. Three-dimensional Wavelet Compression . *2nd Annual Tri-State Engineering Society Meeting*, June 2001.

[67] R. Porter and N. Canagarajah. A robust automatic clustering scheme for image segmentation using wavelets. *IEEE Trans. on Image Processing*, 5:662 – 665, 1996.

[68] J. Portilla and E. P. Simoncelli. Image De-noising via Adjustment of Wavelet Coefficient Magnitude Correlation. In *Proceedings of the 7th International Conference on Image Processing*, Vancouver, BC, Canada, September. IEEE Computer Society.

[69] W. H. Press et al. *Numerical Recipes in C: The Art of Scientific Computing*, chapter 13.10, pages 591–606. Cambridge University Press, Cambridge, 2 edition, 1992.

[70] J.G. Proakis and D.G. Manolakis. *Digital Signal Processing*. Prentice Hall, 1996.

[71] A. Procházka, M. Pánek, V. Musoko, M. Mudrová, and J. Kukal. Biomedical Image De-noising Using Wavelet Transform. *16th Biennal International EURASIP Conference*, pages 350–352, June 2002.

[72] C. C. Reyes-Aldasoro and A Bhalerao. Sub-band filtering for MR texture segmentation. In *Proceedings of Medical Image Understanding and Analysis*, pages 185–188, Portsmouth, UK, July 2002.

[73] O. Rioul and M. Vetterli. Wavelets and signal processing. *IEEE Signal Processing Magazine*, pages 14–38, October 1991.

[74] J. C. Russ. *The Image Processing Handbook (3rd ed.)*. CRC Press, Inc., 1999.

[75] I. W. Selesnick and K. Y. Li. Video de-noising using 2-d and 3-d dual–tree complex wavelet transforms. *Wavelets: Applications in Signal and Image Proc.*, 5207(10):607–618, August 2003.

[76] P. Shukla. Complex wavelet transforms and their applications. Master's thesis, University of Strathclyde, October 2003.

[77] J. Sijbers, M. Verhoye, P. Scheunders, A. Van der Linden, D. Van Dyck, and E. Raman. Watershed-based segmentation of 3d mr data for volume quantization. *Magnetic Resonance Imaging*, 15(6):679–688, 1997.

[78] E. Simoncelli, W. Freeman, E. Adelson, and D. Heeger. Shiftable multi–scale transforms. *IEEE Trans. Info. Theory*, 38:587–607, 1992.

[79] P. Soille. *Morphological Image Analysis: Principles and Applications*. Springer, 2003.

[80] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. Wavelets for computer graphics: A primer, part 1. *IEEE Computer Graphics and Applications*, 15(3):76–84, May 1995.

[81] G. Strang. Wavelets and dilation equations: A brief introduction. *SIAM Review*, 31, 1989.

[82] G. Strang. Wavelet transforms versus fourier transforms. *American Mathematical Society*, 28(2):228–305, April 1993.

[83] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.

[84] Inc. The MathWorks. *Matlab Image Processing Toolbox*. The MathWorks, Inc., Natick, Massachusetts 01760, May 1997.

[85] Inc. The MathWorks. *Image Processing Toolbox User's Guide*. The MathWorks, Inc., Natick, Massachusetts 01760, 1998.

[86] Inc. The MathWorks. *MATLAB Web Server*. The MathWorks, Inc., Natick, Massachusetts 01760, 1999.

[87] J. Uhlíř and P. Sovka. *Číslicové zpracování signálů*. Vydavatelství ČVUT, Praha, 1995.

[88] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560, November 1995.

[89] M. Unser. Splines; A Perfect Fit for Signal and Image Processing. *IEEE Signal Processing Magazine*, 16(6):22–38, 1999.

[90] M. Unser and A. Aldroubi. A review of wavelets in biomedical applications. *Proceedings of the IEEE*, 84(4):626–638, April 1996.

[91] J. Van De Vegte. *Fundamentals of Digital Signal Processing*. Prentice Hall, 2001.

[92] M. Vetterli. Wavelets and filter banks theory and design. *IEEE Trans. Signal Processing*, 40(9):2207–2232, September 1992.

[93] B. Vidaković and P. Müller. Wavelets for kids: A tutorial introduction. Technical report, Duke University, 1991.

[94] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transcations Pattern Analysis and Machine Intelligence*, 13(6):583–593, 1991.

[95] M. Šonka, V. Hlaváč, and R. Boyle. *Image Processing: Analysis and Machine Vision*. O'Reilly, 1999.

[96] T. C. Wang and N. B. Karayiannis. Detection of microcalcifications in digital mammograms using wavelets. *IEEE Trans. on Medical Imaging*, 17(4), August 1998.

[97] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Trans Image Processing*, 13(4):600–612, April 2004.

[98] C.D. Watkins, A. Sadun, and S. Marenka. *Modern Image Processing: Warping, Morphing, and Classical Techniques*. Academic Press Professional, 1993.

[99] A. B. Watson, R. Borthwick, and M. Taylor. Image quality and entropy masking. In *Human Vision, Visual Processing, and Digital Display VIII*, volume 30(16) of *SPIE Proceedings*, San Jose, CA, USA, 1997.

[100] A. B. Watson and J. A. Solomon. A model of visual contrast gain control and pattern masking. *Journal of the Optical Society of America*, 14, 1997.

[101] S. Zhong and V. Cherkassky. Image de-noising using wavelet thresholding and model selection. In *Proc. IEEE Int. Conf. on Image processing*, 2000.

10
List of the Author's Publications

1. A. Procházka, M. Pánek, V. Musoko, M. Mudrová and J. Kukal. Biomedical Image De-noising Using Wavelet Transform. *16th Biennal International EURASIP Conference*, 2002.

2. V. Musoko and A. Procházka. Non-Linear Median Filtering of Biomedical Images. *10th Scientific Conference MATLAB2002*, Humusoft, 2002.

3. V. Musoko, M. Pánek and M. Mudrová. Detection of information carrier signals using band-pass filters. *10th Scientific Conference MATLAB2002*, Humusoft, 2002.

4. V. Musoko and A. Procházka. Three-Dimensional Biomedical Image De-noising. *15th International Conference on Process Control*. STU, Slovakia, 2003.

5. V. Musoko, A. Procházka and A. Pavelka. Implementation of a Matlab Web Server for Internet-based Processing of Biomedical Images. *11th Scientific Conference MATLAB2003*, Humusoft, 2003.

6. J. Kukal, D. Majerová, V. Musoko, A. Procházka and A. Pavelka. Nonlinear Filtering of 3D MRI in MATLAB. *11th Scientific Conference MATLAB2003*, Humusoft, 2003.

7. V. Musoko. Application of Wavelet Transform in Biomedical Image De-noising. *Conference of PhD students*, Faculty of Chemical Engineering, ICT, Prague, 2003.

8. V. Musoko and A. Procházka. Complex Wavelet Transform in Signal and Image Analysis. *14th International Conference on Process Control*. VŠCHT Pardubice, 2004.

9. V. Musoko, M. Kolínová and A. Procházka. Image Classification Using Competitive Neural Networks. *12th Scientific Conference MATLAB2004*, Humusoft, 2004.

10. V. Musoko and A. Procházka. Complex Wavelet Transform in Pattern Recognition. *17th International Conference on Process Control*. STU, Slovakia, 2005.

11
APPENDICES

Appendix A
Mathematics

A.1 Linear Algebra

The mathematics of wavelets [80] rely heavily on fundamental ideas from linear algebra. This appendix reviews a few important ideas.

Vector Spaces

Definition A vector space (over the reals) can be loosely defined as a collection V of elements where

- For all $a, b \in R$ and for all $u, v \in V$, $au + bv \in V$.
- There exists a unique element $\mathbf{0} \in V$ such that
  - for all $u \in V$, $\mathbf{0}u = \mathbf{0}$, and
  - for all $u \in V$, $0 + u = u$.
- Other axioms [80] hold true, most of which are necessary to guarantee that multiplication and addition behave as expected.

The elements of a vector space V are called vectors, and the element 0 is called the zero vector. The vectors may be geometric vectors, or they may be functions, as is the case when discussing wavelets and multiresolution analysis.

Bases and Dimension

Definition A collection of vectors $u_1, u_2, \cdots$ in a vector space V are said to be linearly independent if

$$c_1u_1 + c_2u_2 + \cdots = \mathbf{0} \text{ if and only if } c_1 = c_2 = \cdots = 0.$$

A collection $u_1, u_2, \cdots \in V$ of linearly independent vectors is a basis for V if every $v \in V$ can be written as

$$v = \sum_i c_i v_i$$

for some real numbers $c_1, c_2, \cdots$. The vectors in a basis for V are said to span V. Intuitively speaking, linear independence means that the vectors are not redundant, and a basis consists of a minimal complete set of vectors.

If a basis for V has a finite number of elements $u_1, u_2, \cdots, u_m$, then V is *finite-dimensional* and its dimension is m. Otherwise, V is said to be *infinite-dimensional*.

Example: R^3 is a three-dimensional space, and $e_1 = (1, 0, 0)$, $e_2 = (0, 1, 0)$, $e_3 = (0, 0, 1)$ is a basis for it.

Example: The set of all functions continuous on $[0, 1]$ is an infinite-dimensional vector space. Well call this space $C[0, 1]$.

Inner Products and Orthogonality

Definition The inner (dot) product of two vectors is the sum of the point-wise multiplication of each component:

$$u.v = \sum_i u_i v_i$$

The generalization of the dot product to arbitrary vector spaces is called an *inner product*. Formally, an inner product $\langle \cdot | \cdot \rangle$ on a vector space V is any map from $V \times V$ to R

Example: It is straightforward to show that the dot product on R^3 defined by

$$\langle (a_1, a_2, a_3) | (b_1, b_2, b_3) \rangle = a_1 b_1 + a_2 b_2 + a_3 b_3$$

For functions?

$$f.g = \int_{-\infty}^{\infty} f(x) g(x) dx$$

Example: The following "standard" inner product on $C[0, 1]$ plays a central role in most formulations of multiresolution analysis:

$$\langle f | g \rangle = \int_0^1 f(x) g(x) dx$$

One of the most important uses of the inner product is to formalize the idea of orthogonality. Two vectors u, v in an inner product space are said to be *orthogonal* if $\langle u | v \rangle = 0$. It is not difficult to show that a collection $u_1, u_2, \cdots$ of mutually orthogonal vectors must be linearly independent, suggesting that orthogonality is a strong form of linear independence. An *orthogonal basis* is one consisting of mutually orthogonal vectors.

Norms and Normalization

Definition A *norm* is a function that measures the length of vectors. In a finite dimensional vector space, we typically use the norm $\|u\| = \langle u | u \rangle^{1/2}$. If we are working with a function space such as $C[0, 1]$, we ordinarily use one of the L^p norms, defined as

$$\|u\| = \left(\int_0^1 |u(x)|^p dx \right)^{1/p}$$

In the limit as p tends to infinity, we get what is known as the *max-norm*:

$$\|u\|_\infty = \max_{x\in[0,1]}|u(x)|$$

Even more frequently used is the L^2 norm, which can also be written as $\|u\|_2 = \langle u|u\rangle^{1/2}$ if we are using the standard inner product. A vector u with $\|u\| = 1$ is said to be *normalized*. If we have an orthogonal basis composed of vectors that are normalized in the L^2 norm, the basis is called *orthonormal*. Stated concisely, a basis $u_1, u_2, \ldots$ is orthonormal if

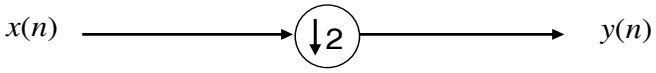
$$\langle u_i|u_j\rangle = \delta_{ij}$$

where δ_{ij} is called the Kronecker delta and is defined to be 1 if $i = j$, and 0 otherwise.

Example: The vectors $e_1 = (1, 0, 0),\ e_2 = (0, 1, 0),\ e_3 = (0, 0, 1)$ form an orthonormal basis for the inner product space R^3 endowed with the dot product.

A.2 Downsampling - Decimation

The down-sampler [1], represented by the following diagram,



is defined as

$$y(n) = x(2n) \tag{A.1}$$

The usual notation is

$$y(n) = [\downarrow 2]x(n) \tag{A.2}$$

The down-sampler simply keeps every second sample, and discards the others (this has the effect of compressing the sequence in time). For example, if $x(n)$ is the sequence

$$x(n) = \{\ldots, 7, 3, \underline{5}, 2, 9, 6, 4, \ldots\}$$

where the underlined number represents $x(0)$, then $y(n)$ is given by

$$y(n) = [\downarrow 2]x(n) = \{\ldots, 7, \underline{5}, 9, 4, \ldots\}$$

Given $X(z)$, what is $Y(z)$? Using the example sequence above we directly write

$$X(z) = \ldots + 7z^2 + 3z + 5 + 2z^{-1} + 9z^{-2} + 6z^{-3} + 4z^{-4} + \ldots \tag{A.3}$$

and

$$Y(z) = \ldots + 7z + 5 + 9z^{-1} + 4z^{-2} + \ldots$$

To express $Y(z)$ in terms of $X(z)$, consider the sum of $X(z)$ and $X(-z)$. Note that $X(-z)$ is given by

$$X(-z) = \ldots + 7z^2 - 3z + 5 - 2z^{-1} + 9z^{-2} - 6z^{-3} + 4z^{-4} + \ldots \tag{A.4}$$

The odd terms are negated. Then

$$X(z) + X(-z) = 2.(\ldots + 7z^2 + 5 + 9z^{-2} + 4z^{-4} + \ldots) \tag{A.5}$$

$$X(z) + X(-z) = 2.Y(z^2) \tag{A.6}$$

Using the Z transform definition we have after multiplying both sides of Eq. (A.6) by $z^{\frac{1}{2}}$

$$\sum_n x(n)z^{-n}z^{\frac{1}{2}} + \sum_n x(n)(-z)^{-n}z^{\frac{1}{2}} = 2.\sum_n y(n)z^{-2n}z^{\frac{1}{2}}$$

$$\sum_n x(n)z^{-\frac{n}{2}} + \sum_n x(n)(-z)^{-\frac{n}{2}} = 2.\sum_n y(n)z^{-n}$$

$$X(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}}) = 2.Y(z)$$

Thus

$$Y(z) = Z\{[\downarrow 2]x(n)\} = \frac{X(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}})}{2} \tag{A.7}$$

The effect of down-sampling on the Fourier transform of a signal?

The discrete-time Fourier transform of $y(n)$ is given by

$$\begin{aligned} Y(e^{j\omega}) &= \frac{X(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}})}{2}\Big|_{z=e^{j\omega}} \\ &= \frac{1}{2}\big(X(e^{j\frac{\omega}{2}}) + X(-e^{j\frac{\omega}{2}})\big) \\ &= \frac{1}{2}\big(X(e^{j\frac{\omega}{2}}) + X(e^{-j\pi}e^{j\frac{\omega}{2}})\big) \\ &= \frac{1}{2}\big(X(e^{j\frac{\omega}{2}}) + X(e^{j(\frac{\omega}{2}-\pi)}\big) \\ &= \frac{1}{2}\big(X^f(\frac{\omega}{2}) + X^f(\frac{\omega}{2}-\pi)\big) \end{aligned}$$

$$Y(z) = DTFT\{[\downarrow 2]x(n)\} = \frac{1}{2}\big(X^f(\frac{\omega}{2}) + X^f(\frac{\omega}{2}-\pi)\big) \tag{A.8}$$

where we have used the notation $Y^f(\omega) = Y(e^{j\omega})$ and $X^f(\omega) = X(e^{j\omega})$.

Note that because $X^f(\omega)$ is periodic with a period of 2π the functions $X^f(\frac{\omega}{2})$ and $X^f(\frac{\omega-2\pi}{2})$ are each periodic with a period of 4π. But as $Y^f(\omega)$ is the Fourier transform of a signal, it must be 2π periodic. What does $Y^f(\omega)$ look like. It is best illustrated with an example (Fig. A.1).

Notice that while the two terms $X^f(\frac{\omega}{2})$ and $X^f(\frac{\omega-2\pi}{2})$ are 4π-periodic, because one is shifted by 2π, their sum is 2π-periodic, as a Fourier transform must be.

Notice that when a signal $x(n)$ is down-sampled, the spectrum $X^f(\omega)$ may overlap with adjacent copies, depending on the specific shape of $X^f(\omega)$. This overlapping is called *aliasing*. When aliasing occurs, the signal $x(n)$ can not in general be recovered after it is down-sampled. In this case, information is lost by the downsampling. If the spectrum $X^f(\omega)$ were zero for $\frac{\pi}{2} \leq |\omega| \leq \pi$, then no overlapping would occur, and it would be possible to recover $x(n)$ after it is down-sampled.

General case: An M-fold down-sampler, represented by the diagram,

is defined as

$$y(n) = x(Mn) \tag{A.9}$$

The usual notation is

$$y(n) = [\downarrow M]x(n) \tag{A.10}$$

The M-fold down-sampler keeps only every M^{th} sample. For example, if the sequence $x(n)$

$$x(n) = \{\ldots, 8, 7, 3, \underline{5}, 2, 9, 6, 4, 2, 1, \ldots\}$$

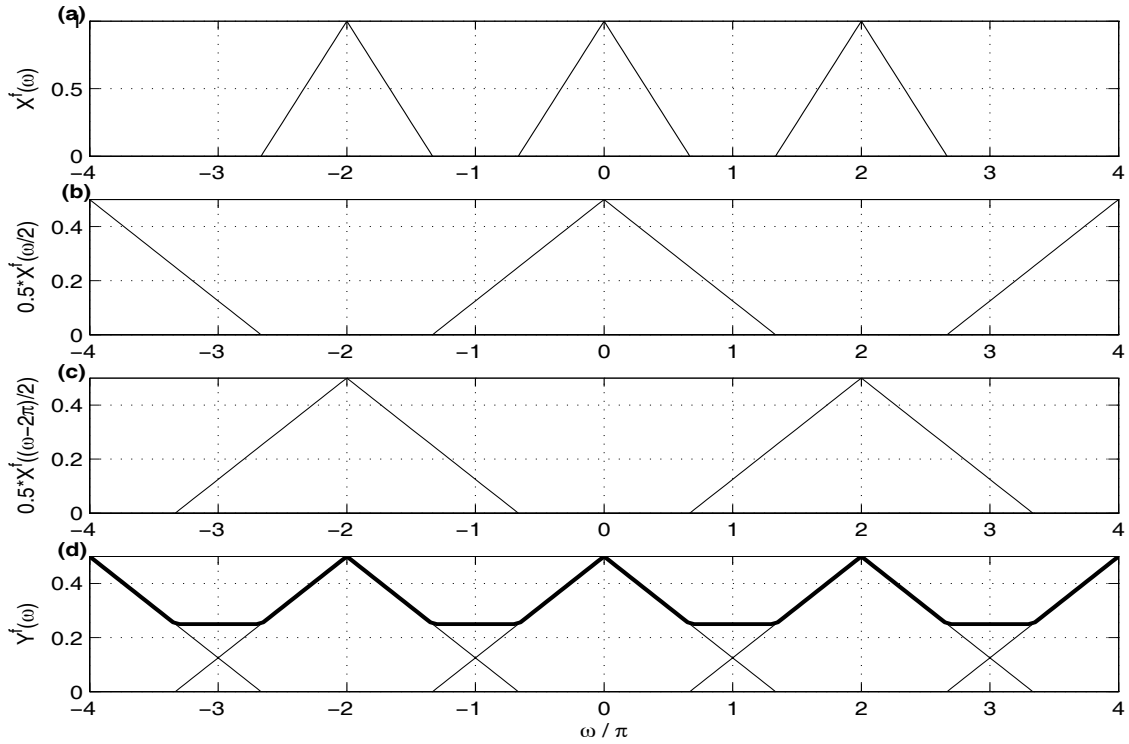
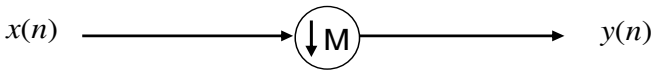


FIGURE A.1. Spectrum of the original signal $x(n)$.



is down-sampled by a factor M = 3, the result is the following sequence

$$y(n) = [\downarrow 2]x(n) = \{\ldots, 8, \underline{5}, 6, 1, \ldots\}$$

Similarly, we have

$$Y(z) = Z\{[\downarrow M]x(n)\} = \frac{1}{M}\sum_{k=0}^{M-1} X(W^k z^{\frac{1}{M}}) \tag{A.11}$$

where $W = e^{j\frac{2\pi}{M}}$ and $Y^f(\omega)$ is defined by

$$Y^f(\omega) = DTFT\{[\downarrow M]x(n)\} = \frac{1}{M}\sum_{k=0}^{M-1} X^f(\frac{\omega + 2\pi k}{M}) \tag{A.12}$$

Remarks

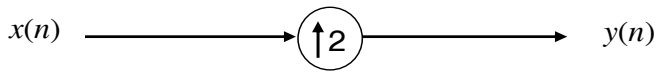
1. In general, information is lost when a signal is down-sampled.

2. The down-sampler is a linear but not a time-invariant system.

3. In general, the down-sampler causes *aliasing.*

A.3 Upsampling

The up-sampler, represented by the diagram,



is defined by the relation

$$y(n) = \begin{cases} x(\frac{n}{2}) & \text{for n even} \\ 0 & \text{for n odd} \end{cases} \tag{A.13}$$

The up-sampler simply inserts zeros between samples. For example, if $x(n)$ is the sequence

$$x(n) = \{\ldots, 3, \underline{5}, 2, 9, 6, \ldots\}$$

where the underlined number represents $x(0)$, then $y(n)$ is given by

$$y(n) = [\uparrow 2]x(n) = \{\ldots, 0, 3, 0, \underline{5}, 0, 9, 0, 6, 0 \ldots\}$$

Given $X(z)$, what is $Y(z)$? Using the example sequence above we directly write

$$X(z) = \ldots + 3z + 5 + 2z^{-1} + 9z^{-2} + 6z^{-3} + \ldots \tag{A.14}$$

and

$$Y(z) = \ldots + 3z^{2} + 5 + 2z^{-2} + 9z^{-4} + 6z^{-6} + \ldots \tag{A.15}$$

It is clear that

$$Y(z) = Z\{[\uparrow 2]x(n)\} = X(z^2) \tag{A.16}$$

We can also derive this using the definition:

$$\begin{aligned} Y(z) &= \sum_{n} y(n) z^{-n} \\ &= \sum_{n \text{ even}} x(\frac{n}{2}) z^{-n} \\ &= \sum_{n} x(n) z^{-2n} \\ &= X(z^2) \end{aligned} \tag{A.17}$$

How does up-sampling affect the Fourier transform of a signal?

The discrete-time Fourier transform of $y(n)$ is given by

$$\begin{aligned} Y(e^{j\omega}) &= X(z^2)\Big|_{z=e^{j\omega}} \\ &= X((e^{j\omega})^2) \end{aligned} \tag{A.18}$$

so we have

$$Y(e^{j\omega}) = X(e^{j2\omega}) \tag{A.19}$$

Or using the notation $Y^f(\omega) = Y(e^{j\omega})$, $X^f(\omega) = X(e^{j\omega})$, we have

$$Y^f(\omega) = DTFT\{[\uparrow L]x(n)\} = X^f(2\omega) \tag{A.20}$$

When sketching the Fourier transform of an up-sampled signal, it is easy to make a mistake. When the Fourier transform is as shown in Fig. A.2, it is easy to incorrectly think that the Fourier transform of $y(n)$ is given by the second figure. This is not correct, because the Fourier transform is 2π-periodic. Even though it is usually graphed in the range $-\pi \leq \omega \leq \pi$ or $0 \leq \omega \leq \pi$ outside this range it is periodic.*Because $X^f(\omega)$ is a 2π-periodic function of ω, $Y^f(\omega)$ is a π-periodic function of ω.*
The correct graph of $Y^f(\omega)$ is the second subplot in Fig. A.2. Note that the spectrum of

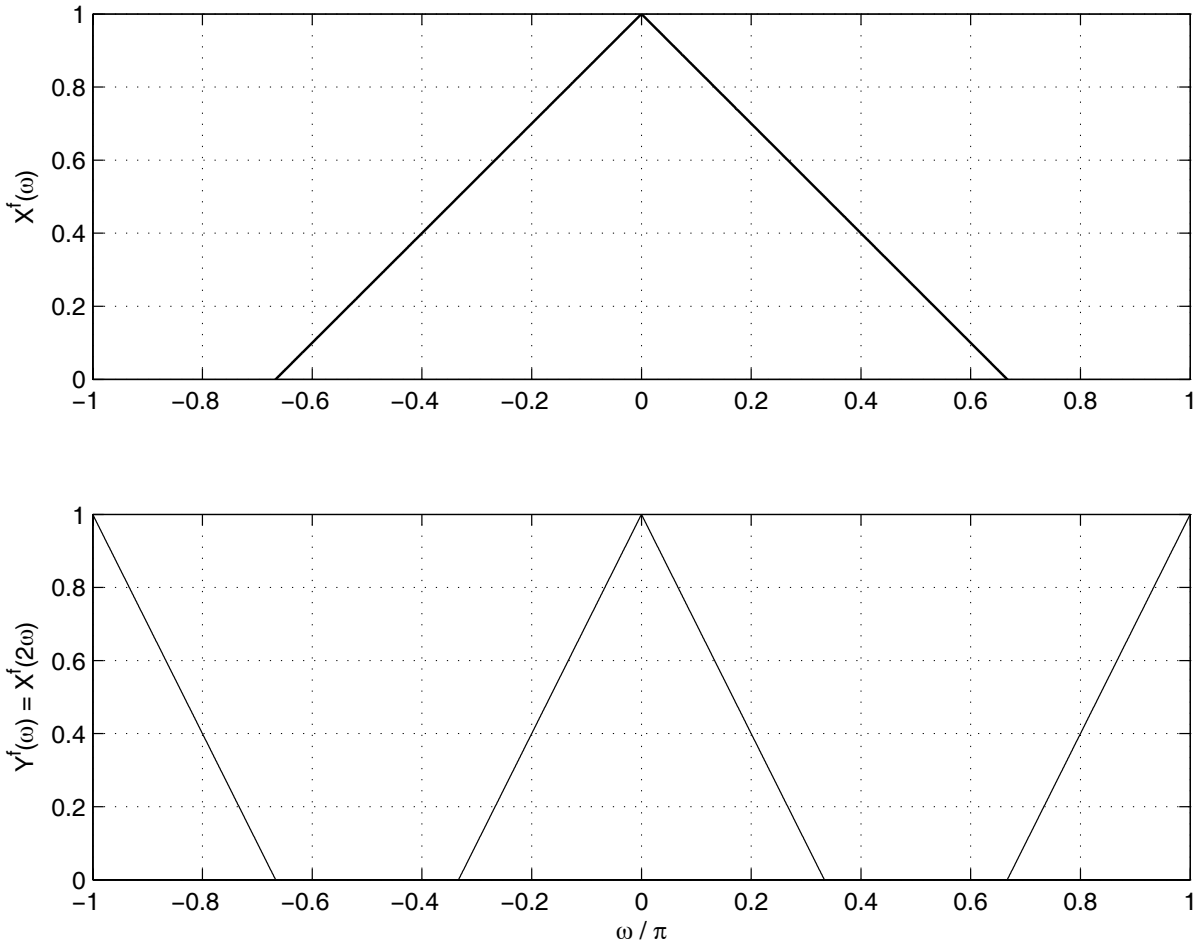
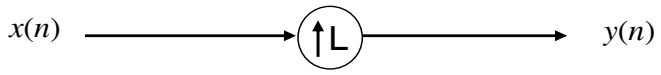


FIGURE A.2. Upsampling by a factor of 2: Spectrum of the original signal $x(n)$ and its compressed spectral replicas.

$X^f(\omega)$ is repeated there is an 'extra' copy of the spectrum. This part of the spectrum is called the *spectral image*.

General case: An L-fold up-sampler, represented by the diagram,



is defined as

$$y(n) = [\uparrow L]x(n) = \begin{cases} x(\frac{n}{L}) & \text{when } n \text{ is a multiple of } L \\ 0, & \text{otherwise} \end{cases} \tag{A.21}$$

The L-fold up-sampler simply inserts $L-1$ zeros between samples. For example, if the sequence $x(n)$

$$x(n) = \{\ldots, 3, \underline{5}, 2, 9, 6, \ldots\}$$

is up-sampled by a factor $L = 4$, the result is the following sequence

$$\begin{aligned} y(n) &= [\uparrow 4]x(n) \\ &= \{\ldots, 0, 3, 0, 0, 0, \underline{5}, 0, 0, 0, 9, 0, 0, 0, 6, 0 \ldots\} \end{aligned}$$

Similarly, we have

$$Y(z) = Z\{[\uparrow L]x(n)\} = X(z^L) \tag{A.22}$$

$$Y(e^{j\omega}) = X(e^{jL\omega}) \tag{A.23}$$

$$Y^f(\omega) = DTFT\{[\uparrow L]x(n)\} = X^f(L\omega) \tag{A.24}$$

The L-fold up-sampler will create $L-1$ spectral images. For example, when a signal is up-sampled by 4, there are 3 spectral images as shown in the following figure (Fig. A.3).

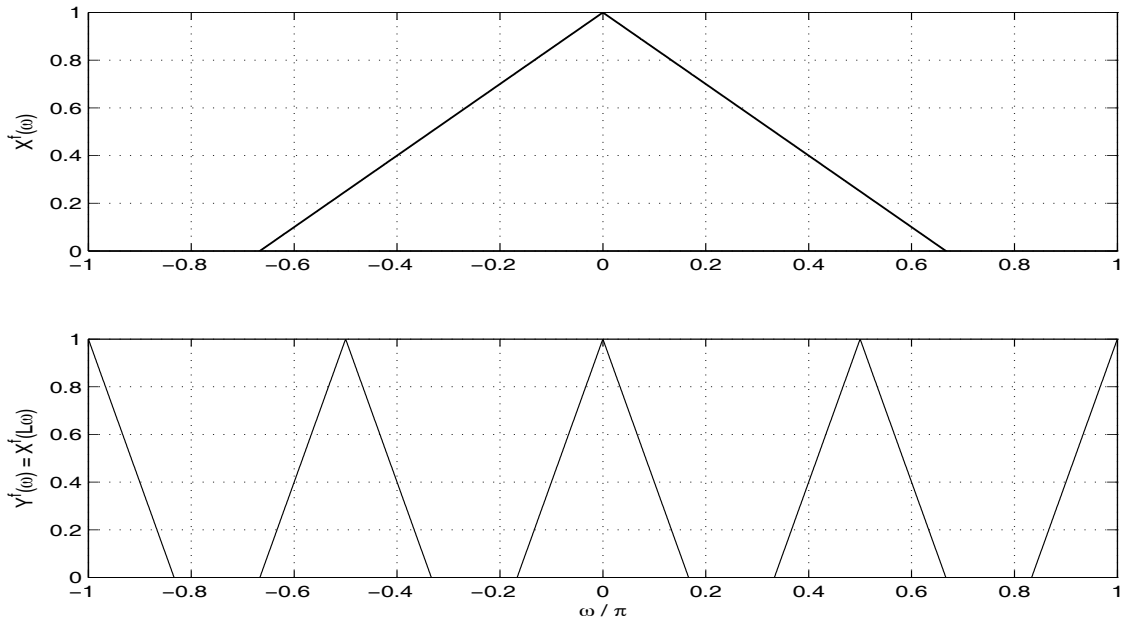


FIGURE A.3. Upsampling by a factor of 4: Spectrum of the original signal $x(n)$ and the resulting 3 compressed spectral replicas.

Remarks

1. No information is lost when a signal is up-sampled.

2. The up-sampler is a linear but not a time-invariant system.

3. The up-sampler introduces *spectral images*.

A.4 Cubic Spline Interpolation

The simplest piecewise polynomial approximation is piecewise linear interpolation which consists of joining a set of data points

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))\} \tag{A.25}$$

by a series of straight lines. Using linear function approximation is sometimes not good enough as there is no differentiability at the endpoints of the subintervals and that also means the interpolating function is not *smooth*. Usually smoothness is required so the approximating function must be continuously differentiable.

The simplest type of differentiable piecewise polynomial function on an entire interval $[x_0, x_n]$ is the function obtained by fitting one quadratic polynomial between each successive pair of nodes. The most common piecewise polynomial approximation uses cubic polynomials between each successive pair of nodes and is called *cubic spline interpolation*. A general cubic polynomial involves four constants so there is sufficient flexibility to ensure that the interpolant is not only continuously differentiable on the interval but also has a continuous second derivative. The cubic spline does not assume that the derivatives of the interpolant agree with those of the function it is approximating, even at the nodes.

Definition Given a function f defined on $[a, b]$ and a set of nodes $a = x_0 < x_1 \ldots < x_n = b$, a *cubic spline interpolant* S for f is a function that satisfies the following conditions:

(a) $S(x)$ is a cubic polynomial denoted by $S_j(x)$ on the subinterval $[x_j, x_{j+1}]$ for each $j = 0, 1, \ldots, n-1$;

(b) $S(x_j) = f(x_j)$ for each $j = 0, 1, \ldots, n-1$;

(c) $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;

(d) $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;

(e) $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ for each $j = 0, 1, \ldots, n-2$;

(f) One of the following sets of boundary conditions is satisfied

  i. $S''(x_0) = S''(x_n) = 0$ (free or natural boundary)

  ii. $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ (clamped boundary)

There are other forms of cubic splines defined with other boundary conditions. When the free boundary conditions occur the spline is called a *natural spline* and its graph approximates the shape that a long flexible rod would bend to if forced to go through

the data points $\{(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))\}$. In general clamped boundary conditions lead to more accurate approximations since they include more information about the function. For this type of boundary condition it is necessary to have either the values of the derivatives at the endpoints or an accurate approximation to those values.

For the construction of the cubic spline interpolant for a given function f the conditions in the definition are applied to the cubic polynomials of the form

$$S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \tag{A.26}$$

for each $j = 0, 1, \ldots, n - 1$.

Since $S_j(x_j) = a_j = f(x_j)$ condition (c) can be applied to obtain

$$a_{j+1} = S_{j+1}(x_{j+1}) = S_j(x_{j+1}) = a_j + b_j(x_{j+1} - x_j) + c_j(x_{j+1} - x_j)^2 + d_j(x_{j+1} - x_j)^3,$$

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

Since the terms $x_{j+1} - x_j$ are used repeatedly it is convenient to write it as

$$h_j = x_{j+1} - x_j,$$

for each $j = 0, 1, \ldots, n - 1$. If we define $a_n = f(x_n)$, then the equation

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 \tag{A.27}$$

holds for each $j = 0, 1, \ldots, n - 1$.

In a similar manner define $b_n = S'(x_n)$ and observe that

$$S_j'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2 \tag{A.28}$$

implies $S_j'(x_j) = b_j$, for each $j = 0, 1, \ldots, n - 1$. Applying condition (d) gives

$$b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2, \tag{A.29}$$

for each $j = 0, 1, \ldots, n - 1$.

Another relationship between the coefficients of S_j is obtained by defining $c_n = S''(x_n)/2$ and applying condition (e). Then for each $j = 0, 1, \ldots, n - 1$,

$$c_{j+1} = c_j + 3d_j h_j \tag{A.30}$$

Solving for d_j in Eq. (A.30) and substituting this value into Eq. (A.27) and Eq. (A.29) gives for each $j = 0, 1, \ldots, n - 1$, the new equations

$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}(2c_j + c_{j+1}) \tag{A.31}$$

and

$$b_{j+1} = b_j + h_j(c_j + c_{j+1}) \tag{A.32}$$

The final relationship involving the coefficients is obtained by solving the appropriate equation in the form of Eq. (A.31), first for b_j,

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1}) \tag{A.33}$$

and then with a reduction of the index for b_{j-1} we get

$$b_{j-1} = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j)$$

Substituting these values into Eq. (A.32), with the index reduced by one gives the linear system of equations

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1}) \tag{A.34}$$

for each $j = 1, 2, \ldots, n-1$. This system involves only the $\{c_j\}_{j=0}^n$ as unknowns since the values of $\{h_j\}_{j=0}^{n-1}$ and $\{a_j\}_{j=0}^n$ are given, respectively by the spacing of the nodes $\{x_j\}_{j=0}^n$ and the values of f at the nodes.

When the values of $\{c_j\}_{j=0}^n$ are found it is a simple matter to find the remainder of the constants $\{b_j\}_{j=0}^{n-1}$ from Eq. (A.33) and $\{d_j\}_{j=0}^{n-1}$ from Eq. (A.30) and to construct the cubic polynomials $\{S_j(x)\}_{j=0}^{n-1}$.

A big question that arises with this construction is whether the values of $\{c_j\}_{j=0}^n$ can be found using the system of equations given in Eq. (A.34) and if so whether these values are unique. The following theorems indicate that this is the case when either of the boundary conditions given in part (f) of the definition are imposed. The proofs of these theorems can be found in Chapter 6 of [9].

Theorem A.4.1 *If f is defined at $a = x_0 < x_1 \ldots < x_n = b$, then f has a unique natural spline interpolant S on the nodes $x_0, x_1, \ldots, x_n$; that is a spline interpolant that satisfies the boundary conditions $S''(a) = 0$ and $S''(b) = 0$.*

Proof The boundary conditions in this case imply that $c_n = S''(x_n)/2 = 0$ and that

$$S''(x_0) = 2c_0 + 6d_0(x_0 - x_0) = 0$$

so $c_0 = 0$.

The two equations $c_0 = 0$ and $c_n = 0$ together with the equations in Eq. (A.34) produce a linear system described by the vector equation $A\mathbf{x} = \mathbf{b}$, where A is an $(n+1)$ by $n+1$ matrix.

$$A = \begin{bmatrix} 1 & 0 & 0 & \dots & \dots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \ddots & & \vdots \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \dots & \dots & 0 & 0 & 1 \end{bmatrix}$$

and b and x are the vectors

$$\mathbf{b} = \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}$$

Matrix A is strictly diagonally dominant and it satisfies the hypotheses of **Theorem 6.19** in Chapter 6 of [9]. Therefore the linear system has a unique solution for $c_0, c_1, \dots, c_n$

The solution to the cubic spline problem with the boundary conditions $S''(x_0) = S''(x_n) = 0$ can be obtained by applying **Algorithm 3.4** (Chapter 3 of [9]).

Appendix B
Image Coding

Images have their information encoded in the *spatial domain*, the image equivalent of the time domain. A digital image is a set of $N \times M$ values each representing a *picture element* (pixel). The value of each pixel is a sampled representation of an $\frac{1}{N} \times \frac{1}{M}$. The pixel value is represented in a bit depth integer value corresponding to the luminance of the physical area of the pixel for black-white images, or a set of integer values for colour components such as the *(R,G,B)* space for the same pixel area.

The limiting factor for displaying a digital image is the number of integers needed to describe an image. Suppose for a 1000×1000 pixels *RGB* colour image coded on 8 bits per component,we need almost 8 megabytes of memory. Thus it's necessary to compress the image so as to represent the same sequence of pixels with the least number of bits possible. Image coding techniques can be classified into two classes namely *lossless* and *lossy* techniques. Lossless techniques (e.g. *TIFF* and *GIF* format), represent the exact set of pixels in the minimum number of bits. Lossy techniques (e.g. *JPEG* format) do not represent the exact pixel values but only give a good approximation of them.

Image Types

This section describes the basic types of images which can be represented in **MATLAB** [85]. The image is often presented as a two-dimensional function *f(x,y)*, where *x,y* - are spatial coordinates.

Indexed Images

An indexed image (Fig. B.1) consists of a data matrix, X and a colormap matrix, map. `map` is an m-by-3 array of class double containing floating-point values in the range (0,1). To display an indexed image with `imshow` function, you specify both the image matrix and the `colormap`.

$$\texttt{imshow}(\texttt{X}, \texttt{map})$$

For each pixel in X, `imshow` displays the color stored in the corresponding row of map. The value 1 points to the first row in the colormap, the value 2 points to the second row, and so on.

```
X=[1 4 5;
   3 2 6;
   7 8 9];
%colormap
M=[0 0 0; %black
   1 1 1; %white
   1 0 0; %pure red
   0 0 1; %blue
   0 1 0; %green
   0 0 0; %black
   1 1 1; %white
   1 0 0; %pure red
   0 0 1];%blue
imshow(X,map,'notruesize');
```

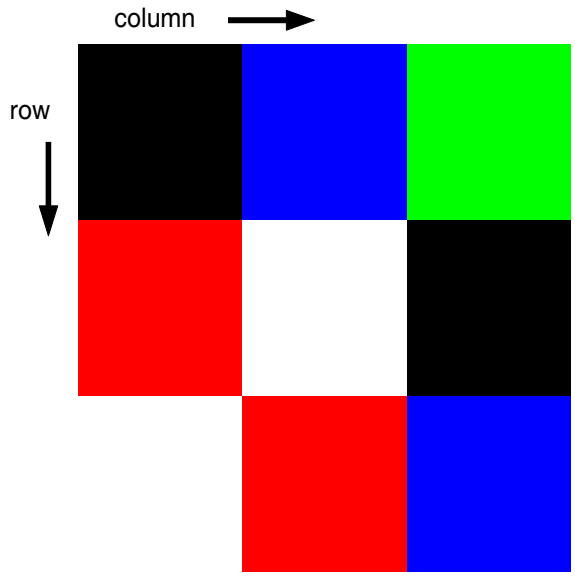


FIGURE B.1. Indexed image

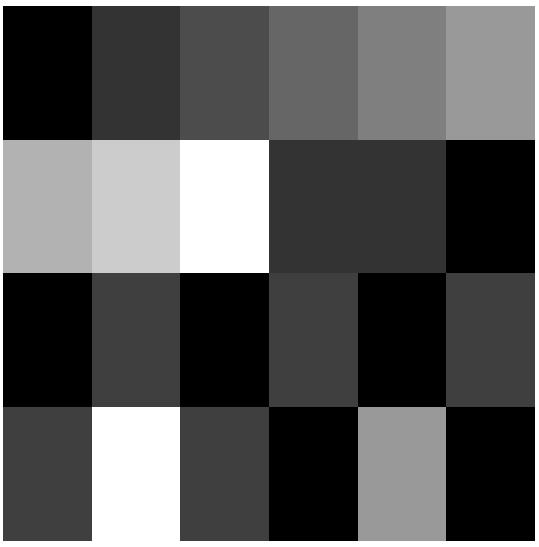


FIGURE B.2. Intensity image

Intensity Images

An intensity image (Fig. B.2) is a data matrix, `I`, whose values represent intensities within some range. To display a grayscale intensity image, the syntax is:

```
imshow(I)
```

`imshow` displays the image by scaling the intensity values to serve as indices into a grayscale colormap whose values range from black to white. You can explicitly specify the number of gray levels with `imshow`, normally the number of levels of gray in the colormap is 256 on systems with 24-bit color, and 64 on other systems.

```
%Intensity image-grayscale
I=[0      0.20   0.30   0.40   0.50   0.60 ;
   0.70   0.80   1      0.20   0.20   0    ;
   0.     0.25   0      0.25   0      0.25 ;
   0.25   1.00   0.25   0      0.60   0   ];

figure,
%256 gray levels
imshow(I,256,'notruesize');
```

Binary Images

A binary image (Fig. B.3) is stored as a two-dimensional matrix of 0's and 1's. To display a binary image, the syntax is:

```
imshow(BW)
```

The 0 values in the image matrix BW display as black, and the 1 values display as white.

```
%Binary image
BW=[1 0 1 0 1;
    0 1 0 1 0;
    1 0 1 0 1];
figure, imshow(BW,'notruesize');
```

RGB Images

An RGB image (Fig. B.4), sometimes referred to as a '`truecolor`' image, is stored in `MATLAB` as an m-by-n-by-3 data array that defines red, green, and blue color components for each individual pixel. RGB images do not use a palette (colormap). The color of each pixel is determined by the combination of the red, green, and blue intensities stored in each color plane at the pixels location. To display an RGB image, the syntax is:

```
imshow(RGB)
```

For each pixel `(r,c)` in RGB, `imshow` displays the color represented by the triplet `(r,c,1:3)`. An RGB image array can be of class double, uint8, or uint16:

- If the array is of class double, the data values are in the range [0, 1].
- If the array is of class uint8, the data range is [0, 255].
- If the array is of class uint16, the data range is [0, 65535].

```
%RGB image
RGB(:,:,1)=[0.44  0.75  0.25  0.77 ;
            0.78  0.32  0.13  0.11 ;
            0.61  0.23  0.19  0.87];
RGB(:,:,2)=[0.59  0.44  0.92  0.45 ;
            0.98  0.41  0.35  0.32 ;
            0.71  0.31  0.67  0.78];
RGB(:,:,3)=[0.66  0.22  0.33  0.22 ;
            0.13  0.81  0.73  0.24 ;
            0.54  0.64  0.66  0.52];
figure, image(RGB);
```

Multiframe Image Arrays

These are used for applications,where one may need to work with collections of images related by time or view, such as magnetic resonance imaging (MRI) slices or movie frames. The MRI data typically contains a number of slice planes taken through a volume, such as the human body. To display a multiframe indexed image, the syntax is:

```
%Displaying 20 slices
```

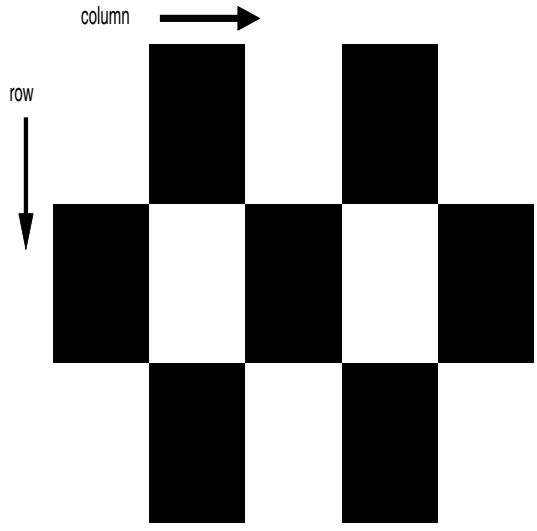


FIGURE B.3. Binary image

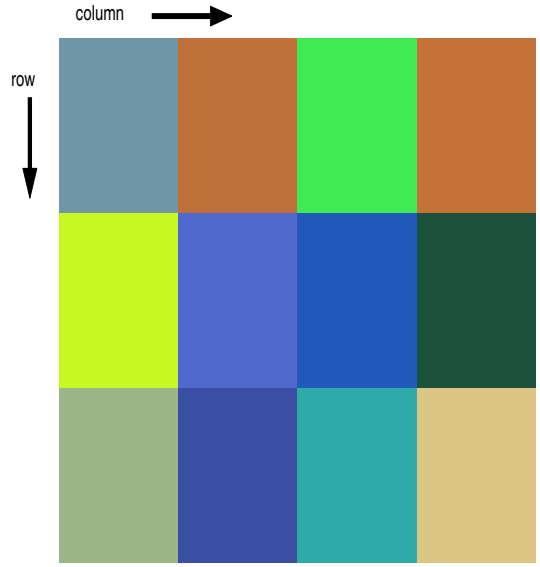


FIGURE B.4. RGB image

```
for i=1:20
   mri=['p_0',num2str(i),'.mat'];
   load(mri)
   A1(:,:,1,i)=A;
end

montage(A1) brighten(0.6)
```

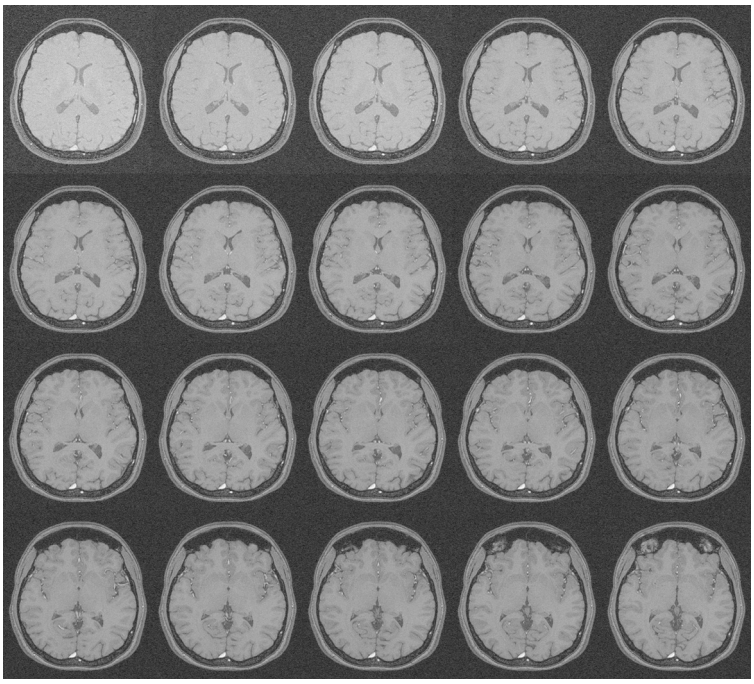


FIGURE B.5. Multi-frame indexed image - MRI slices

Appendix C

Remote Image Processing Using MATLAB Web Server

The implementation of the MWS with the goal of its use for biomedical image processing - remote image processing is to be discussed. MWS integrates the graphical and computational capabilities of MATLAB with a remote access through the Internet. MATLAB toolboxes provide a range of algorithms that enable image processing by different mathematical methods including the discrete wavelet transform. The design of a graphical user's interface enables the choice of input parameters and presentation of the results.

A key application of modern educational technology is e-learning with a web-based approach using for example a MATLAB Web Server (MWS) application. MWS [86] allows clients (users) to run MATLAB applications remotely over the internet. Clients interact with MATLAB over a network with a TCP/IP protocol. This interaction takes place through HTML forms which serve as graphical interfaces for the application. This not only allows users to use MATLAB based tools without any prior MATLAB programming knowledge but it also prevents unauthorized user access to source code. Fig. C.1 shows the interaction between clients and MWS.

Implementation of the Matlab Web Server

The MATLAB application resides on the server machine and the components of the MWS [86] include:

- `matlabserver` TCP/IP server running MATLAB continuously.
- `matweb.exe` common gateway interface (CGI) program that resides on the HTTP server and communicates with `matlabserver`. `matweb` requires information found in `matweb.conf` to locate `matlabserver`. An instance of `matweb.conf` looks like

```
[mri2D]
mlserver=atlas
mldir=H:/PHOBOS/MUSOKOV/m
```

  where `mlserver` is the name of host running `matlabserver` and `mldir` is the working directory for reading m-files or save any generate any graphics writing files.

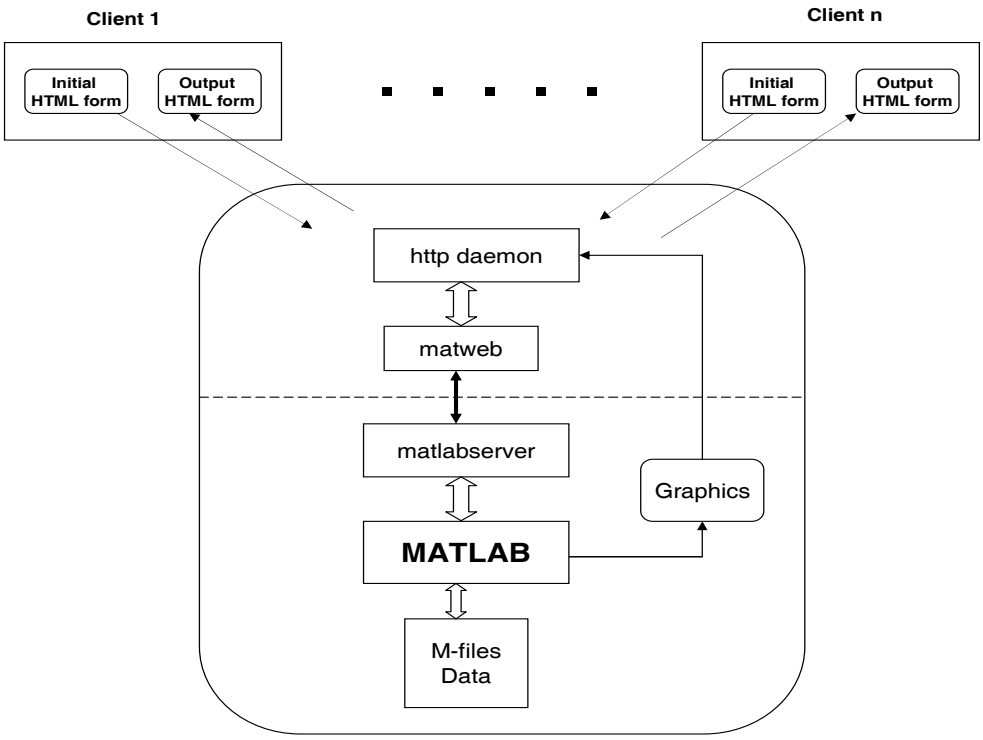


FIGURE C.1. How MATLAB operates on the web

- Input HTML form data functions MATLAB utility functions that retrieve data from input HTML documents and then provide that data in a form convenient to MATLAB programmers.
- Output HTML document functions MATLAB functions for inserting MATLAB data into template HTML forms for return transmission to the end users Web browser.

There are three files associated with an `mri2D` function *http* : *//dsp.vscht.cz/musoko*:

- `mri2D_input.html`: the `mri2D` input document
- `mri2D_output.html`: the `mri2D` output document
- `mri2D.m`: the `mri2D` MATLAB m-file

Looking at the source from the `mri2D_plot.html` file there is a line

```
<input type="hidden" name="mlmfile" value="mri2D">
```

that sets the argument `mlmfile` to the value `mri2D`. The `mlmfile` argument contains the name of the MATLAB m-file to run. `matlabserver` uses the value of `mlmfile` obtained from the `matweb`, to run the MATLAB application. mri2D takes the input data from

mri2D_input.html, carries out the de-noising application and outputs the results using mri2D_outpt.html as a template.
The mri2D function uses the htmlrep command to place the computed graphics into the mri2D_output.html output template using the code

```
templatefile = which('mri2D_output.html');
rs = htmlrep(s, templatefile);
```

where s is a MATLAB structure containing the results of the mri2D computation. htmlrep extracts data from s and replaces variable fields in mri2D_output.html with the results of the MATLAB computation. The completed mri2D_output.html form is transmitted to the user's browser. In examining some parts of the mri2D and mri3D code, we will see how to include MATLAB graphics as part of a MATLAB Web Server application. The input document allows us to set the characteristics of the MRI plot we want to generate. The code in the source file

```
<form action="/cgibin/matweb.exe" method="POST"
target="outputwindow">
<input type="hidden" name="mlmfile" value="mri2D">
```

calls the mri2D function and targets the output to a frame on the lower portion of the input document itself. In mri2D.m the code

```
mlid = getfield(h,'mlid')
```

extracts mlid from the structure h. mlid is a unique identifier that matlabserver provides. Using the value of mlid to construct filenames ensures that filenames are unique. The code

```
s.graf1 = sprintf('../icons/%smri2D.jpeg', mlid);
```

creates a name for a graphic (jpeg) file and save the files in the directory /icons.
The function htmlrep replaces MATLAB variable names it finds in the HTML output template file mri2D_output.html with the values in the input structure s in our case graph1.

```
templatefile = which('mri2D_output.html');
rs = htmlrep(s, templatefile);
```

From the mri2D_output.html, $graph1$, the variable that represents the graphic output, is found in the line

```
<img border=0 src="$graph1$">
```

An example of how of one level image processing of a noisy MR image slice is shown in Fig. C.2 on an internet browser. In our demonstration the client has to select the type of data to be loaded, the number of levels and threshold value for de-noising of a noisy MRI slice.

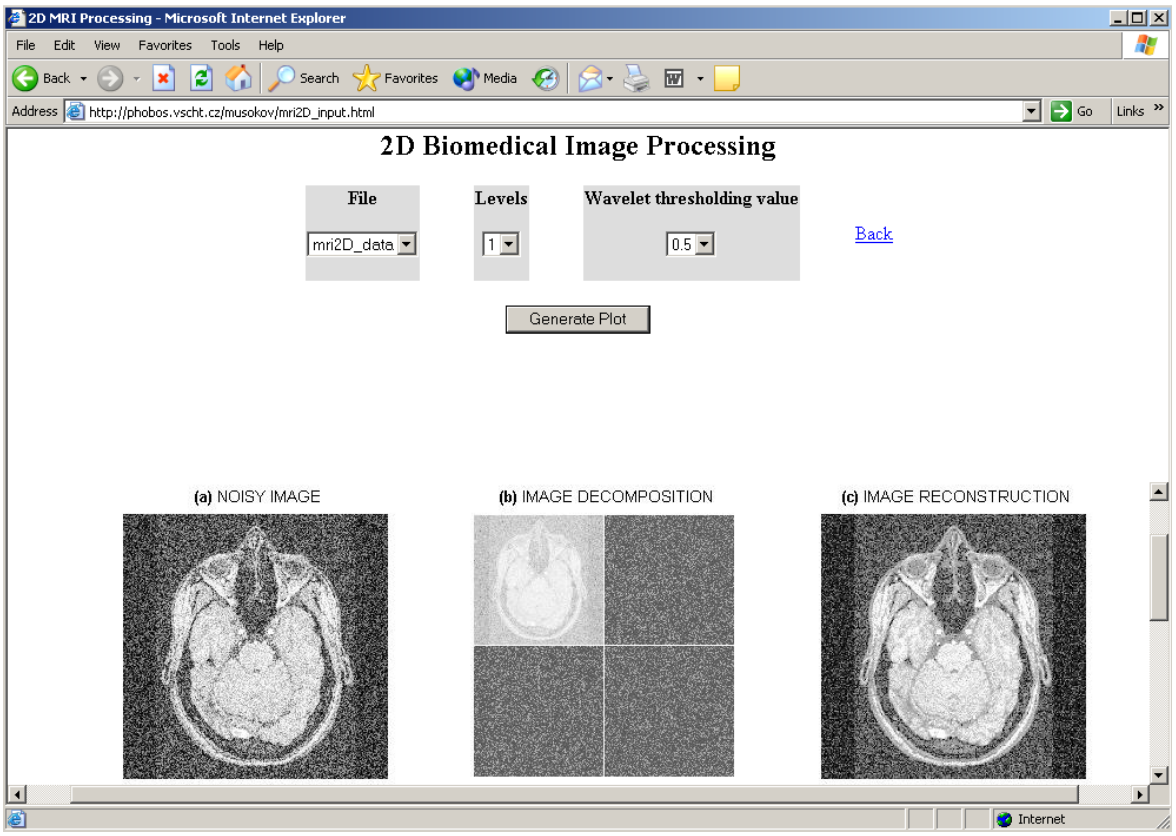


FIGURE C.2. 2-D Biomedical image processing.

Appendix D
Selected Algorithms in MATLAB

In this section, I provide the implementation and examples of the algorithms I have used in the thesis. The algorithms were implemented numerically as MATLAB functions (scripts) and these programs were used to create the various examples and figures in this thesis. This form of implementation is computationally more efficient and enables experimentation on the multi-dimensional signal data. The software, along with supporting documents, is made available on the following web address: $http://dsp.vscht.cz$.

D.1 Volume Visualization

```
%-----------------------------------
% 3-D BIOMEDICAL VISUALIZATION
%-----------------------------------
clear,close all,clc

% loading image slice data into a 3D array
for i=1:70
    load(['../data/p_0',num2str(i),'.mat']);
    A=A(:,65:448);
    A1(:,:,i)=A;
end

% 4-D array squeezed to 3D
A2 = squeeze(A1);

% Extracting subset of volume data set
[x,y,z,A2] = subvolume(A2,[200,300,100,nan,nan,nan]);

p1 = patch(isosurface(x,y,z,A2, 5),...
    'FaceColor','none','EdgeColor','none');
isonormals(x,y,z,A2,p1); p2 = patch(isocaps(x,y,z,A2, 5),...
    'FaceColor','interp','EdgeColor','none');
view(3); axis tight; axis on; daspect([1,1,.4]);

xlabel('x-pixel co-ordinate')
ylabel('y-pixel co-ordinate')
zlabel('slice no.')
colormap(gray(100)) brighten(0.7)
```

D.2 2-D MR Image Visualization

```
% ------------------------------------------
% 2-D MR IMAGE VISUALIZATION
% ------------------------------------------
clear, close all, clc

% Load MRI datta
for i=1:70
    mri=['../data/p_0',num2str(i),'.mat'];
    load(mri)
    D(:,:,1,i)=A;
end


% Display all the slices in one figure
figure(1)
montage(D(:,:,1,10:29));
% title('70 MRI Slices')
brighten(0.6)

% Display image slice no. 40
slice_40=D(:,:,:,40);

figure(2)
imshow(slice_40)
title('MR IMAGE SLICE')
brighten(0.7)

% Show all frames of MRI data as a movie
% figure
% immovie(D)

% Sagittal slice image of the MR brain image
D=squeeze(D);  % 4-D to 3-D
slice=D(:,200,:);
sagittal_slice_2D = squeeze(slice);
sagittal_slice = rot90(sagittal_slice_2D);

figure(3);
h1=axes('Position',[0.03 0.4 0.3 0.4]);
imagesc(D(:,:,40))
title('{\bf (a)} AXIAL SLICE')
axis off

h2=axes('Position',[0.35 0.4 0.3 0.4]);
imagesc(sagittal_slice);
title('{\bf (b)} SAGITTAL SLICE')
axis off
```

```
% Coronal slice image of the MR brain image
slice=D(300,:,:);
coronal_slice_2D = squeeze(slice);
coronal_slice = rot90(coronal_slice_2D);

h3=axes('Position',[0.67 0.4 0.3 0.4]);
imagesc(coronal_slice);
title('{\bf (c)} CORONAL SLICE')
colormap gray
axis off
brighten(0.7)

% Interpolation of the coronal slice

a=double(coronal_slice);
b=imrescale(a,4,'linear');  % linear interpolation
c=imrescale(a,4,'spline');  % spline interpolation
d=imrescale(a,4,'cubic');   % cubic intepolation


function b=imrescale(X,factor,method)

% IMGRESCALE rescales raw image data using different types of interpolation.
%    G = IMGRESCALE(X, FACTOR, METHOD) Rescales image data by a scale
%    factor and interpolation double arrays are supported
%    INPUTS:
%    X - original image
%    factor - the scale factor by which the image is to be rescaled method
%    'linear' - Bilinear interpolation
%    'spline' - Cubic spline interpolation
%    'cubic'  - Bicubic interpolation
%    OUTPUT:
%    b - rescaled image

if ~isa(X, 'double')
    X=double(X);
end
% size of the original image
[m,n]=size(X);
% scaling factor
factor=1/factor;
% sampling grid coordinates
[xi,yi]=meshgrid(1:factor:n,1:factor:m);
% 2-D Interpolation
b=interp2(X,xi,yi,method);
```

D.3 Wavelet Decomposition

```
% ------------------------------------
% 1-D SIGNAL DECOMPOSITION
% ------------------------------------
clear, close all

% simulated signal
N=256; n=0:256;
s=sin(2*pi*0.02*n)+rand(1,N+1);

figure(1);
h1=axes('Position',[0.07 0.72 0.9 0.23]);
plot(s,'r');
grid on; axis tight;
title('{\bf (a)} GIVEN SIGNAL')

% Decomposition parameters definition
level=3;
wavelet='db2';

% 1-D DWT decomposition
[c,l]=wavedec(s,level,wavelet);

h3=axes('Position',[0.07 0.39 0.9 0.23]);
stem(c,'g');
grid on; axis tight; v=axis;
title(['{\bf (b)} WAVELET COEFFICIENTS: LEVELS 1-',num2str(level)])
ylabel(['WAVELET: ',wavelet])

% decomposition subands
ind(1)=0;
for i=1:level+1
    ind(i+1)=sum(l(1:i));
    line([ind(i+1); ind(i+1)],[v(3);v(4)],'Linewidth',2)
end

% scalogram
N=length(s);
S=zeros(level,N);
for k=1:level
    d=detcoef(c,l,k); d=d(ones(1,2^k),:);
    S(k,:)=wkeep(d(:)',N);
end

h4=axes('Position',[0.07 0.06 0.9 0.23]);
colormap(pink(64));
img=image(flipud(wcodemat(S,64,'row')));
title('{\bf (c)} SIGNAL SCALOGRAM'); ylabel('Level')
```

D.4 Discrete Fourier Transform De-noising

```
% ----------------------------------------
% FILTERING IN THE FREQUENCY DOMAIN
% ----------------------------------------

N=64; M=64;
% 2D signal
for m=1:M
    for n=1:N
        x(n,m)=sin(0.2*n)+sin(1.6*n);
    end
end

figure(1)
subplot(221)
imshow(x);
title('{\bf (a)} SIMULATED IMAGE')
axis tight
subplot(222)
mesh(x);
xlabel('m');ylabel('n');axis tight
title('{\bf (b)} 3-D PLOT OF THE SIMULATED IMAGE DATA')

% 2D FFT
% x=x-mean(x(:)); % get rid of the DC component
X=fft2(x);
X=fftshift(X);
subplot(223);
mesh([0:n-1]/n-0.5,[0:m-1]/m-0.5,abs(X));
xlabel('F1');ylabel('F2')
title('{\bf (c)} 2-D FOURIER SPECTRUM');hold on

% window function
W=zeros(N,N);
W(14*2:18*2,14*2:18*2)=1;
% Spectrum modification by windowing method
Y=X.*W;
mesh([0:n-1]/n-0.5,[0:m-1]/m-0.5,W*600+1400);
set(gca,'XLim',[-0.5 0.5],'YLim',[-0.5 0.5],'ZLim',[0 2000])
hold off

% 2D inverse FFT
y=ifft2(ifftshift(Y));
subplot(224);
mesh(real(y));
xlabel('m');ylabel('n');
axis tight
title('{\bf (d)} FILTERED IMAGE DATA')
```

D.5 Short Time Fourier Transform

```
% ------------------------------------
% SHORT TIME FOURIER TRANSFORM
% ------------------------------------
clear,close all,clc

T=1;        % sampling time
fs= 1/T;    % sampling freq.
N=512;      % no. of samples
t=[0:N-1]'; % N secs at 1Hz sample rate

% sinusoidal signal x(t) with f1 at time interval t1
% and f2 at time interval t2
f1=0.1;  % freq. 1
f2=0.35; % freq. 2

t1=[1:floor(N/2)]';
t2=[floor(N/2):N]';
x=zeros(N,1);
x(t1)= sin(2*pi*f1*t1);
x(t2)= sin(2*pi*f2*t2);

% FFT magnitude of the signal
Y=abs(fft(x));

n= 0:N-1;
% time interval in sec.
t = n*T;

fnorm= (n/N)*fs; % normalized freq
f = fnorm * fs;  % frequencies
half = 1:(N/2);  % half the number of samples

figure(1)
h=axes('Position',[0.07 0.6 0.4 0.35]);
plot(t,x);
xlabel('Time [s]');
grid on
title('{\bf (a)} ANALYSED SIGNAL')
axis tight

h=axes('Position',[0.57 0.6 0.4 0.35]);
% amplitude spectrum
plot(fnorm(half), Y(half),'b')
xlabel('Frequency [Hz]')
ylabel('Magnitude')
title('{\bf (b)} SPECTRUM OF THE ANALYSED SIGNAL')
grid on
```

```
% SHORT TIME FOURIER TRANSFORM

M=32; % length of window
for j=1:floor(N/M)
    X(:,j)=x((j-1)*M+1:j*M);
end
X=abs(fft(X));
X=X([1:M/2],:);
[m1,n1]=size(X);

% 2-D Spectrogram
h4=axes('Position',[0.57 0.1 0.4 0.35]);
imagesc([1:n1]*fs*M,[0:m1-1]*fs/M,X)
axis xy
xlabel('Time [s]');
ylabel('Frequency [Hz]')
title(['{\bf (d)} 2D SPECTROGRAM']);
axis tight
brighten(0.99)
colormap(pink(64));

% 3-D Spectrogram
axes('Position',[0.07 0.1 0.4 0.35]);
mesh([1:n1]*fs*M,[0:m1-1]*fs/M,X)
xlabel('Time [s]');
ylabel('Frequency [Hz]')
title(['{\bf (c)} 3D SPECTROGRAM']);
axis tight
colormap(pink(64));
```

D.6 2-D Wavelet De-noising

```
% ----------------------------------------
% 2-D DWT DE-NOISING
% ----------------------------------------
clear,clc,close all

i=menu('IMAGE','SIMULATED 2-D IMAGE','2-D MRI IMAGE','Exit');

if i==1
    n=[0:0.5:63.5]';
    % simulated image data
    x=sin(0.6*n)*sin(0.6*n');
    x=(x-min(min(x)))/(max(max(x))-min(min(x)));
    A2=x;
    A2=(A2-min(A2(:)))/(max(A2(:))-min(A2(:)));
    % noisy image
    A2n=rnoise2D(A2);
    A2n=A2n(1:64,1:64);
    A2=A2(1:64,1:64);

elseif i==2
    i=50;
    load(['../../data/p_0',num2str(i),'.mat']);
    % pixel values
    x=100;dx=228;
    y=100;dy=228;
    % uint16--->double
    A=im2double(A);
    %subimage
    A2=A(x:dx-1,y:dy-1);
    A2n=rnoise2D(A2);
else
    break
end

% noisy image data
x=A2n;

% Array dimensions of the original image
[m n]=size(A2);

% wavelets
i=menu('WAVELET','db1(Haar)','db2','db4','sym2','sym4',...
    'sym8','bior1.1','bior1.3','bior1.5','Exit');

% Daubechies
if i==1
    wavelet='db1';
```

```
elseif i==2
    wavelet='db2';
elseif i==3
    wavelet='db4';
    %  Symlets
elseif i==4
    wavelet='sym2';
elseif i==5
    wavelet='sym4';
elseif i==6
    wavelet='sym8';
    %  Biorthogonal
elseif i==6
    wavelet='bior1.1';
elseif i==7
    wavelet='bior1.3';
elseif i==8
    wavelet='bior1.5';
else
    break
end

% no. of decomposition levels
level=2;

% 2D DWT DECOMPOSITION
[W,c,l,ind]=IM_dec(x,level,wavelet);

% DETERMINATION OF THRESHOLDING VALUES FOR DIFF. TYPES OF THRESH. METHODS

j=menu('THRESHOLD','GLOBAL','LEVEL DEPENDENT','OPTIMAL');

if j==1
    % global treshold
    % noise standard deviation of the detail coefficients HH
    x8=W(1).D(:);
    stand=std( x8);
    for i=2:3*level+1
        lambda(i)=stand*sqrt(2*log(m*n));
    end
    ss='GLOBAL THRESHOLDING';
elseif j==2
    % Level dependent treshold
    % noise standard deviation using Donoho estimate
    % sigmak = MAD/0.6745  Mean Absolute Deviation
    w=[];
    for d=level:-1:1;
        w{d}{1}=W(d).H;
        w{d}{2}=W(d).V;
```

```
        w{d}{3}=W(d).D;
    end

    for k=level:-1:1
        for j=1:3
            sk{k}{j}=(median(abs(w{k}{j}(:)))/0.6745)*sqrt(2*log(m*n));
        end
    end
    % threshold values
    lambda=[];
    for k=level:-1:1
        for j=1:3
            lambda=[lambda sk{k}{j}];
        end
    end
    lambda=[0 lambda];
    ss='LEVEL DEPENDENT THRESHOLDING';
elseif j==3
    % optimal thresholding
    % threshold range
    t = 0:0.01:0.4;
    % MSE error values
    e = den2DWT(A2,x,wavelet,level,t);
    % minimum optimal threshold value
    [emin,kmin] = min(e);
    Tt = t(kmin);
    ss='OPTIMAL THRESHOLDING';
    % MSE curve
    figure(2)
    % h=axes('Position',[0.1 0.5 0.8 0.3]);
    h4=axes('Position',[0.07 0.12 0.9 0.3]);
    plot(t,e,'b','LineWidth',1.2);grid on
    title(['{\bf (d)} {\bf',num2str(ss),'}'])
    xlabel('\lambda (threshold)');
    ylabel('{\bf MSE}');
    for i=2:3*level+1
        lambda(i)=Tt;
    end
else
    break
end

figure(1)
h4=axes('Position',[0.07 0.12 0.9 0.3]);
plot(c,'g');  grid on; hold on
set(gca,'XtickLabel',[]); axis tight;
v=axis; dv=v(3)-0.08*(v(4)-v(3));hold on
title(['{\bf (d)} SCALING AND WAVELET COEFFICIENTS - {\bf',num2str(ss),'}'])
ylabel(['WAVELET - {\bf',num2str(wavelet),'}'])
```

```
%subband lines
for i=1:3*level+1
    line([ind(i+1); ind(i+1)],[v(3);v(4)],'LineWidth',1.2);
end
text(v(1), dv, [num2str(level),',','0']);
for i=1:level
    for j=1:3
        text(ind(3*(i-1)+j+1), dv, [num2str(level-i+1),',',num2str(j)]);
    end
end

% threshold lines
for i=2:3*level+1
    line([ind(i);ind(i+1)],[lambda(i);lambda(i)],'LineWidth',1.2);
    line([ind(i);ind(i+1)],[-lambda(i);-lambda(i)],'LineWidth',1.2);
end

% Soft Thresholding using the calculated threshold value T

% modifying coefficients
for i=1:3*level+1
    k=find(abs(c(ind(i)+1:ind(i+1)))<=lambda(i));
    k=k+ind(i); cd(k)=0;
    k=find(abs(c(ind(i)+1:ind(i+1)))>lambda(i));
    k=k+ind(i); cd(k)=sign(c(k)).*(abs(c(k))-lambda(i));
end
plot(cd,'r');  grid on; axis tight;
hold off

% reconstructed image
y=waverec2(cd,l,wavelet);

% figure(2)
h1=axes('Position',[0.07 0.5 0.28 0.4]);
imshow(x); axis tight; v=axis; title('{\bf (a)} NOISY IMAGE')

h2=axes('Position',[0.37 0.5 0.28 0.4]);
A1=IM_decshow05(W);
imshow(A1)
colormap(gray)
axis off
v=axis; title('{\bf (b)} IMAGE DECOMPOSITION')

h3=axes('Position',[0.67 0.5 0.28 0.4]);
imshow(y);
axis tight; v=axis; title('{\bf (c)} IMAGE RECONSTRUCTION')
brighten(0.5)
```

```
function [W,c,l,ind]=IM_dec(S,level,wavelet);
% Image wavelet decomposition to a given level
% S ........... image matrix
% level ....... decomposition level
% wavelet ..... wavelet decomposition function
% W ........... structured variable of decomposition images belonging to
%               decomposition levels
% c ........... vector of decomposition coefficients
% ind ......... vector pointing to indices of vector c defining individual
%               decomposition levels

[c,l]=wavedec2(S,level,wavelet); lfull=[l(1,:)];
% Evaluation of indices dividing vector of decomposition coefficients into
% levels
lfull=[l(1,:)];
for d=level:-1:1
    lfull=[lfull; ones(3,1)*l(level-d+2,:)];
end
ind=cumsum([0 prod(lfull')]);

% Resizing of decomposition coefficients into decomposition images using
% WKEEP function to extract the middle part of convolution matrices
[m,n]=size(S); ls(level)=floor(m/2^level);
for d=level-1:-1:1;
    ls(d)=ls(d+1)*2;
end
W(level).A=wkeep(reshape(c(ind(1)+1:ind(2)),lfull(1,:)),[ls(level) ls(level)]);
for d=level:-1:1; r=3*(level-d)+1;
    W(d).H=wkeep(reshape(c(ind(r+1)+1:ind(r+2)),lfull(r+1,:)),[ls(d) ls(d)]);
    W(d).V=wkeep(reshape(c(ind(r+2)+1:ind(r+3)),lfull(r+2,:)),[ls(d) ls(d)]);
    W(d).D=wkeep(reshape(c(ind(r+3)+1:ind(r+4)),lfull(r+3,:)),[ls(d) ls(d)]);
end
```

D.7 3-D Wavelet De-noising

```
% ----------------------------------------
% 3-D DWT DE-NOISING
% ----------------------------------------

clear,clc,close all

i=menu('VOLUME','SIMULATED 3-D VOLUME','3-D MRI VOLUME','Exit');

if i==1
    n=[0:0.5:31.5]';
    % simulated data set
    x=sin(0.6*n)*sin(0.6*n');
    x=(x-min(min(x)))/(max(max(x))-min(min(x)));
    for i=1:8
        A2(:,:,i)=x;
    end
    A2=(A2-min(A2(:)))/(max(A2(:))-min(A2(:)));
    % random noise
    Rn=rnoise_sim(A2);
    A3=A2+Rn;

elseif i==2
    % pixel values
    x=150;dx=214;
    y=100;dy=164;
    % 3-D MR data set
    for i = 1:8
        %loading the data
        load(['../../data/p_0',num2str(i),'.mat']);
        %uint16--->double
        A=im2double(A);
        %saving each slice in A2
        A2(:,:,i)=A(x:dx-1,y:dy-1);
    end
    % random noise
    Rn=rnoise_sim(A2);
    A3=A2+Rn;
else
    break
end

% Noisy data set
x=A3;
% Array dimensions of x
[m n k]=size(x);

% wavelets
```

```
i=menu('WAVELET','db1(Haar)','db2','db4','sym2','sym4',...
    'sym8','bior1.1','bior1.3','bior1.5','Exit');

% Daubechies
if i==1
    wavelet='db1';
elseif i==2
    wavelet='db2';
elseif i==3
    wavelet='db4';
    %    Symlets
elseif i==4
    wavelet='sym2';
elseif i==5
    wavelet='sym4';
elseif i==6
    wavelet='sym8';
    %  Biorthogonal
elseif i==6
    wavelet='bior1.1';
elseif i==7
    wavelet='bior1.3';
elseif i==8
    wavelet='bior1.5';
else
    break
end

% no. of decomposition levels
level=1;

% analysis and synthesis filter coefficients
[af,sf]=waveletfn(wavelet);


% 3D DWT
w = dwt3D(x, level, af);

% wavelet coefficients
c=[w{level+1}(:)];
for i=level:-1:1
    for j=1:7
        % [LLL LLH LHL LHH HLL HLH HHL HHH]
        c=[c;w{i}{j}(:)];
    end
end

% LLL LLH; LHL LHH
L1=[w{2} w{1}{1};w{1}{2} w{1}{3}];
```

```
% HLL HLH; HHL HHH
H2=[w{1}{4} w{1}{5};w{1}{6} w{1}{7}];

% 3-D decomposed volume
yd=cat(3,H2,L1);
yd=(yd-min(yd(:)))/(max(yd(:))-min(yd(:)));
% partition lines
yd(m/2,:,:)=100;
yd(:,n/2,:)=100;
yd(:,:,k/2)=100;


% DETERMINATION OF THRESHOLDING VALUES FOR DIFF. TYPES OF THRESH. METHODS

j=menu('THRESHOLD','GLOBAL','LEVEL DEPENDENT','OPTIMAL');

if j==1  % GLOBAL THRESHOLDING
    % global treshold values
    % noise standard deviation of the detail coefficients HHH
    x8=w{1}{7}(:);
    stand=std( x8);
    Tn=stand*sqrt(2*log(m*n*k))*2^(-0.5);

    % indexed cell array
    Tind= cell(1,7);
    for i=1:7
        Tt{i}=Tn;
    end

    for i=level:-1:1
        T{i}=Tt;
    end
elseif j==2  % LEVEL DEPENDENT THRESHOLDING
    % Level dependent treshold values
    % noise standard deviation using Donoho estimate
    % sigma = MAD/0.6745  Mean Absolute Deviation
    for i=level:-1:1
        for j=1:7
            T{i}{j}=(median(abs(w{i}{j}(:)))/0.6745)*sqrt(2*log(m*n*k));
        end
    end
elseif j==3 % OPTIMAL THRESHOLDING
    % optimal thresholding values
    % threshold range
    t = 0:0.05:0.7;
    % MSE error values
    e = den3DWT(A2,x,wavelet,level,t);
    [emin,kmin] = min(e);
    % minimum optimal threshold value
```

```
    Tt = t(kmin);
    % MSE curve
    figure(2)
    h=axes('Position',[0.1 0.5 0.8 0.3]);
    plot(t,e,'b','LineWidth',1.2);grid on
    xlabel('\lambda (threshold)');
    ylabel('{\bf MSE}');

    % indexed cell array
    Tind= cell(1,7);
    for i=1:7
        Tt{i}=Topt;
    end

    for i=level:-1:1
        T{i}=Tt;
    end
else
    break
end

figure(1)
h4=axes('Position',[0.07 0.2 0.925 0.3]);
plot(c,'g');  grid on; hold on; %axis tight
set(gca,'XtickLabel',[]);
v=axis;
ylabel(['WAVELET - {\bf',num2str(wavelet),'}'])
title(['{\bf (d)} WAVELET COEFFICIENTS - {\bf',num2str(ss),'}']);

% representation of wavelet coeff. 8 sub-divisions
ll=[];
for j=1:level
    ll=[(m/(2^j))*(n/(2^j))*(k/(2^j)) ll];
end

% no. of coefficients in each subband
Nc=[];
for j=1:level
    Nc=[Nc ll(j)*ones(1,8)];
end

ll=Nc;

ind(1)=0;
block={'LLL' 'LLH' 'LHL' 'LHH' 'HLL' 'HLH' 'HHL' 'HHH'};
for i=1:7*level+1
    ind(i+1)=sum(ll(1:i));
    line([ind(i+1); ind(i+1)],[v(3);v(4)],'LineWidth',1.2);
    %      text(ind(i)+1000,-1.30,block{i}); %sim. model
```

```
    text(ind(i)+1200,-1.20,block{i});
end


% Soft Thresholding using the calculated threshold value T
V=[];
% looping the scales
for j = level:-1:1
    % looping the subbands
    for s = 1:7
        w{j}{s} = soft(w{j}{s},T{j}{s});
        V=[V T{j}{s}];
    end
end


lambda=[0 V]; % lambda(1)=0 approximation coeffiecients not thresholded

% lambda(1)=0;
for i=2:7*level+1
    lambda(i)=abs(lambda(i));
    line([ind(i);ind(i+1)],[lambda(i);lambda(i)],'LineWidth',1.2)
    line([ind(i);ind(i+1)],[-lambda(i);-lambda(i)],'LineWidth',1.2)
end


% modified coefficients
cnew=[w{level+1}(:)];
for i=level:-1:1
    for j=1:7
        cnew=[cnew;w{i}{j}(:) ];
    end
end
% axes(h4)
plot(cnew,'r');  grid on; axis tight;
hold off

% reconstructed image volume
y = idwt3D(w, level, sf);

% 3D view
az=-38;
el=14;

h1=axes('Position',[-0.05 0.6 0.4 0.2]);
model3_sim(x); axis tight; v=axis; title('{\bf (a)} NOISY VOLUME')
view(az,el)

h2=axes('Position',[0.295 0.6 0.4 0.2]);
```

```
model3_sim(yd); axis tight; v=axis;
title('{\bf (b)} 3-D DECOMPOSITION')
view(az,el)

h3=axes('Position',[0.64 0.6 0.4 0.2]);
model3_sim(y); axis tight; v=axis;
title('{\bf (c)} RECONSTRUCTED MODEL')
view(az,el)


function y = denDWT3D(x,wavelet,level,T)

% denDWT3D - thresholding denoising using the 3-D DWT
% x ......... noisy image volume
% wavelet ... type of wavelet ('haar','db2','sym4',..)
% level ..... no. of levels
% T ......... threshold value
% y ......... de-noised image volume


% analysis and synthesis filter coefficients
[af,sf]=waveletfn(wavelet);
% 3-D DWT
w = dwt3D(x,level,af);
% looping  decomposition levels
for j = 1:level
    % looping subbands
    for s = 1:7
        w{j}{s} = soft(w{j}{s},T);
    end
end

% reconstruction
y = idwt3D(w,level,sf);
```

D.8 2-D Complex Wavelet Transform Denoising

Filters used to implement DT-DWT uses Selesnick's Dual-Tree DWT available from: *http* : *//taco.poly.edu/selesi/*

```
% ------------------------------
% 2-D DT CWT - DE-NOISING
% ------------------------------
clear,close all,clc

% loading the image data
load p_050.mat

randn('state',0)

% subimage
s=(A(100:227,100:227));
% converting the uint8 to double
s=im2double(s);

% random noise addition
x=s+0.05*randn(size(s));

% Threshold range
t = 0:0.005:0.1;

%2D DWT method
wavelet='db2';
level=2;
e = den2DWT(s,x,wavelet,level,t);

% 2-D DT CWT method
J=4;
re = den2DTCWT(s,x,J,t);

figure(1)
plot(t,e,'b','LineWidth',1.2)
hold on
plot(t,re,'r','LineWidth',1.2)
axis([0 0.1 0.7e-3 e(1)])
grid on
% axis tight
xlabel('\lambda (Threshold)');
ylabel('MSE');
legend('DWT','DT CWT',0);
hold off

% Finding the optimal minimum  threshold value

% DWT
```

```
[emin,k] = min(e);
T = t(k);
% DT CWT
[emin1,k1] = min(re);
T1 = t(k1);

% De-noising - DWT
y = denDWT(x,wavelet,level,T);

% Denoising - DT CWT
y1 = denDTCWT(x,J,T1);

% MSE and MAE
[MSEn,MAEn]=imgdnoise(s,x)
[MSE1,MAE1]=imgdnoise(s,y)
[MSE2,MAE2]=imgdnoise(s,y1)

% PSNR
PSNRn=psnr(s,x)
PSNR1=psnr(s,y)
PSNR2=psnr(s,y1)

figure(2)

% Original image
s = histeq(s);

% Noisy image
x = histeq(x);

y = histeq(y);
y1 = histeq(y1);

axes('Position',[0.1 0.55 0.45 0.4]);
imshow(s)
title('{\bf (a)} ORIGINAL IMAGE')

axes('Position',[0.41 0.55 0.45 0.4]);
imshow(x)
title('{\bf (b)} NOISY IMAGE')

axes('Position',[0.1 0.08 0.45 0.4]);
imshow(y)
title('{\bf (c)} DE-NOISED IMAGE - DWT')

axes('Position',[0.41 0.08 0.45 0.4]);
imshow(y1)
title('{\bf (d)} DE-NOISED IMAGE - DT CWT')
```

```
function y = denDTCWT(x,level,lambda)

% denDTCWT - thresholding denoising using the 2-D DTCWT
% x ......... noisy image
% level ..... no. of decomposition levels
% lambda .... threshold value
% y ......... de-noised image

% First stage filter coefficients
[Faf, Fsf] = FSfilt;
% Remaining stage filter coefficients
[af, sf] = dualfilt1;

% 2-D DT CWT
w = dualtree2D(x,level,Faf,af);

% Soft thresholding

for j = 1:level
    % looping the subbands
    for s1 = 1:2
        for s2 = 1:3
            w{j}{s1}{s2} = soft(w{j}{s1}{s2},lambda);
%             w{j}{s1}{s2} = hard(w{j}{s1}{s2},lambda);
        end
    end
end

% 2D inverse DT CWT
y = idualtree2D(w,level,Fsf,sf);
```

D.9 Image Segmentation

```
% ------------------------------------------
% WATERSHED IMAGE SEGMENTATION
% ------------------------------------------
delete(get(0,'children'));

% IMAGE DEFINITION
while 1==1
    k=menu('IMAGE','Test1 - Simulated image',...
        'Test2 - MR Knee image',...
        'Exit');
    if k==1
        G=SimImage; bw=G>=0.4;
        D = bwdist(~bw);
        % Complement the distance transform, and force pixels
        % that don't belong to the objects to be at -Inf
        D = -D;
        D(~bw) = -Inf; %deep catchment basin
        ss='simimage';
    elseif k==2
        [X] = dicomread('mr_knee.dcm');
        G=double(X);
        G=(G-min(G(:)))/(max(G(:))-min(G(:)));
        % binary image matrix
        bw=G>=0.2;
        % ~bw complement the BW image
        D = bwdist(~bw,'chessboard');
        D = -D;
        D(~bw) = -Inf;
        ss='knee';
    else
        break
    end

    % IMAGE VISUALIZATION
    hf1=figure('Name','Object','Units','Normal',...
        'Position',[0.0 0.6 0.25 0.25]);
    imshow(G,'n'), title('TEXTURE')

    %% IMAGE WATERSHED TRANSFORM
    LL = watershed(D);
    % w=LL==0; % all zeros in LL are assigned to logical 1
    % complement of LL
    w=(~LL);
    hf2=figure('Name','Ridge','Units','Normal',...
        'Position',[0.25 0.6 0.25 0.25]);
    imshow(w,'n'), title('RIDGE LINES');
```

```
    %% SELECTED REGION DISPLAY
    CONT(1)='y'; col=0; P1=[]; P2=[];
    while CONT(1)=='y'
        col=col+1;
        [j,i]=ginput(1); %one mouse click selection a time
        reg=LL(round(i),round(j))
        % all pixels in the selected region are set to regg
        % resulting in a binary matrix R
        R=LL==reg;
        hf3=figure('Name','Region','Units','Normal',...
            'Position',[0.5 0.6 0.25 0.25]);
        imshow(R,'n'), title(['REGION: ',num2str(reg)]);
        [B,L] = bwboundaries(R,'noholes');
        hold on
        for k = 1:length(B)
            boundary = B{k};
            plot(boundary(:,2), boundary(:,1), 'b', 'LineWidth', 2)
        end

        hf4=figure('Name','Region','Units','Normal',...
            'Position',[0.75 0.6 0.25 0.25]);
        R1=G.*R; imshow(R1,'n'), title('SELECTED OBJECT');
        % CONT=input('Continue (''yes'' or ''no''): ')
        CONT='n';
    end  %end while loop for region extraction


    hf11=figure('Name','Texture Segmentation','Units','Normal',...
        'Position',[0.0 0.3 0.25 0.25]);
    [r,c]=size(G); rd=round(0.1*r); cd=round(0.17*c);
    imshow([G ones(r,3) w;ones(3,2*c+3); R ones(r,3) R1],'n')
    w=[0.999 1 1];
    ta=text(c-cd,rd,'\bf (a)','Color',w)
    tb=text(2*c+3-cd,rd,'\bf (b)','Color',w)
    tc=text(c-cd,r+3+rd,'\bf (c)','Color',w)
    td=text(2*c+3-cd,r+3+rd,'\bf (d)','Color',w)

end  % end while loop - the whole menu
```

D.10 Texture Classification

```
% ---------------------------------------------------------
% IMAGE FEATURE EXTRACTION - DWT DECOMPOSITION
% IMAGE TEXTURE CLASSIFICATION - COMPETITIVE NEURAL NETWORK
% ---------------------------------------------------------
clear,close all,clc

% IM04_VISUALIZATION for visualiation of the set of images
% IM04_FEATURE_SSQ /_MSTD for image feature extraction
% IM04_CLASSIFICATION for image neural network classification
% IM04_CLASS for classification boundaries plot
% IM04_DECOMPOSITION(wavelet,level,GN) for image wavelet analysis
% Also uses  IM_dec.M and IM_decshow.M


% SIMULATED IMAGE TEXTURES
G=SimImage;
% Binary image
bw=G>=0.4;
D = bwdist(~bw);
% Complement the distance transform, and force pixels
% that don't belong to the objects to be at -Inf
D = -D;
% infinite deep catchment basin
D(~bw) = -Inf;


%% IMAGE VISUALIZATION
f1=figure(1);
set(f1,'Units','Normalized','Position',[0.395 0.22 0.39 0.39]);
imshow(G,'n');
title('{\bf (a)} IMAGE TEXTURE');
% hold on

%% IMAGE WATERSHED TRANSFORM
LL = watershed(D);
%   w=LL==0; % all zeros in LL are assigned to logical 1
%   all ridge pixel values are replaced with a logical 1 and the rest logical 0
w=(~LL);

%% SELECTED REGION DISPLAY
for i=2:10
    R{i}=LL==i; % all pixels in the selected region are set to reg
    [B{i},L] = bwboundaries(R{i},'noholes');

    for k = 1:length(B{i})
        boundary{i} = B{i}{k};
    end
```

```
    R1{i}=G.*R{i};
    % region extraction
    e= min(boundary{i});
    f= max(boundary{i});
    x1= floor((e(:,1)+f(:,1))/2);
    y1= floor((e(:,2)+f(:,2))/2);
    R1{i}= R1{i}(e(:,1):f(:,1), e(:,2):f(:,2));
end

R1=R1(2:10); % R1{1} is the background cell matrix

% DWT Decomposition
level=1;
wavelet='db2';
[W,c,ind]=IM_dec(R1{2},level,wavelet);

% Feature Extraction - pattern matrix PAT
% sumsqr evaluation
[W,PAT,f2]=IM04_FEATURE_SSQ(wavelet,level,R1);

% mean std evaluation
% [W,PAT,f2]=IM04_FEATURE_MSTD(wavelet,level,R1);

% Classification, plot of image texture features
S=3;  % Number of classes
f3=1;
color='y'; % input('Color output (''yes'' | ''no''): ');
[W1,b1,ClassPattern,ClassAnal,net,Ac,f3]=...
IM04_CLASSIFICATION(PAT,S,color,f3);
IM04_CLASS(W1,b1,f3);

% Typical images
[f4]=IM04_TYPICAL(R1,ClassAnal);

% Visualization of selected image texture and its wavelet decomposition
% coefficients
[W1,f5]=IM04_DECOMPOSITION(wavelet,level,R1{2});


figure(6)
clf
plot(PAT(2,:),PAT(1,:),'+b','Linewidth',2,'MarkerSize',15);
title('{\bf (a)} IMAGE TEXTURE FEATURES')
grid on;
%axis([min(PAT(2,:)) max(PAT(2,:)) min(PAT(1,:)) max(PAT(1,:))]);
xlabel('P(2,:)'); ylabel('P(1,:)');

f7=figure(7);
set(f7,'Units','Normalized','Position',[0.395 0.22 0.39 0.39]);
```

```
imshow(G,'n'), title('{\bf (a)} IMAGE TEXTURE LABELS');hold on

pal=['b' 'r' 'm' 'k' 'y' 'g' 'c' ];

for i=2:10
    R{i}=LL==i; %all pixels in the selected region are set to reg
    [B{i},L] = bwboundaries(R{i},'noholes');

    for k = 1:length(B{i})
        boundary{i} = B{i}{k};
    end
    R1{i}=G.*R{i};
    e= min(boundary{i});
    f= max(boundary{i});
    x1= floor((e(:,1)+f(:,1))/2);
    y1= floor((e(:,2)+f(:,2))/2);

    R1{i}= R1{i}(e(:,1):f(:,1), e(:,2):f(:,2));
    h1=text(y1,x1,num2str(i-1));%labelling the image objects.
    set(h1,'FontSize',14,'color',pal(Ac(i-1)));
end


% Typical images
CTyp=ClassAnal(4,1:S);
len=1/S;
figure(4)
clf
axes('Position',[(1-1)*len-0.1*len 0.5 1.2*len 1.2*len]); hold on
imshow(R1{CTyp(1)}); axis image; hold off
title('{\bf (a)}')

axes('Position',[(2-1)*len-0.1*len 0.5 1.2*len 1.2*len]); hold on
imshow(R1{CTyp(2)}); axis image; hold off
title('{\bf (b)}')

axes('Position',[(3-1)*len-0.1*len 0.5 1.2*len 1.2*len]); hold on
imshow(R1{CTyp(3)}); axis image; hold off
title('{\bf (c)}')
```

```
function [W1,b1,ClassPattern,ClassAnal,net,Ac,f3]...
         = IM04_CLASSIFICATION(PAT,S,color,f3)
% Neural network classification of pattern matrix P into S classes
%   PAT - matrix (2,N) of image features
%   S - number of classes
%   color - color output ('yes' | 'no')
%   f3 - figure handle
%   W1,b1 - final neural network coefficients
%   ClassPattern - classes of image patterns [index; Class; Pattern]
%   ClassAnal - analysis of classes [ClassIndex; ClassLength; ClassSTD;
%   ClassTypical]
%   net - neural network definition

if color(1)=='y'
    pal=['b' 'r' 'm' 'k' 'y' 'g' 'c' ];
else
    pal=['k' 'k' 'k' 'k' 'k' 'k' 'k' ];
end

classname=['A' 'B' 'C' 'D' 'E' 'F' 'G'];

klr=input('Kohonen training rate (=0.01): ');
% net = newc(minmax(PAT),S,klr,0.001);
net = newc(minmax(PAT),S,klr,0);
net.trainparam.epochs=input('The number of training cycles (=200): ');
net.b{1,1}=zeros(S,1)+eps; % Biases set to zero
% net.biases{1}.learnParam.lr = 0;
net = train(net,PAT);
W1=net.IW{1,1}; b1=net.b{1,1};

A = sim(net,PAT);
Ac = vec2ind(A);
ClassPattern=[1:length(PAT(1,:)); Ac; PAT];

% Analysis of image classification
for i=1:S
    CC=ClassPattern(2,:)==i;
    if sum(CC)~=0
        ClassI = ClassPattern(:,CC);
        D=dist(W1(i,:),ClassI([3 4],:));
        ClassLength(i)=length(D); ClassSTD(i)=std(D);
        [CT,j]=min(D); ClassTypical(i)=ClassI(1,j);
    else
        ClassLength(i)=0; ClassSTD(i)=0; ClassTypical(i)=0;
    end
end

disp('[ClassIndex; ClassLength; ClassSTD; ClassTypical]')
ClassAnal=[[1:S]; ClassLength; ClassSTD; ClassTypical]
```

```
Y=flipud(ClassAnal); YTemp=sortrows(Y'); ClassAnal=flipud(YTemp'); %%%

% Visualization of image features and network coefficients
f3=figure(3); set(f3,'Units','Normalized','Position',[0.395 0.42 0.39 0.39]);
v=minmax(PAT); d=diff(v')*0.05;
% axis([v(2,1)-d(2) v(2,2)+d(2) v(1,1)-d(1) v(1,2)+d(1)]);
axis([min(PAT(2,:)) max(PAT(2,:)) min(PAT(1,:)) max(PAT(1,:))]);

for i=1:length(PAT(1,:))
    t1=text(PAT(2,i),PAT(1,i),num2str(i),'Color',pal(Ac(i)));
end
hold on
for i=1:S
    ii=ClassAnal(1,i);
    t2=text(W1(ii,2),W1(ii,1),...
        classname(i),'Color',pal(ii),...
        'FontSize',16,'FontWeight','bold');
end
hold off
grid on; set(gca,'Box','on'); title('{\bf (b)} IMAGE TEXTURE CLASSIFICATION')
xlabel('W(:,2), P(2,:)'); ylabel('W(:,1), P(1,:)');


function [W,f5]=IM04_DECOMPOSITION(wavelet,level,GN)

% Visualization of selected image and its wavelet decomposition
% coefficients
% wavelet - wavelet decomposition function
% level - wavelet decomposition level
% GN - image matrix
% W(level).matrix - structured array of decomposition images at a given
% level
% matrix: A-approximation, H-horizontal, V-vertical, D-diagonal detail
% coefficients
% Uses IMDEC.M and IMDECSHOW.M


f5=figure(5);

h=axes('Position',[0.05 0.45 0.4 0.4]);
imshow(GN);
title('{\bf (a)} GIVEN IMAGE')
axis tight; v=axis;

h3=axes('Position',[0.1 0.06 0.77 0.3]);
[W,c,ind]=IM_dec(GN,level,wavelet);
plot(c,'g');  grid on;
set(gca,'XtickLabel',[]);
axis tight; v=axis;
```

```
dv=v(3)-0.08*(v(4)-v(3));
title(['{\bf (c)} IMAGE SCALING AND WAVELET COEFFICIENTS: LEVELS 1-',...
num2str(level)])
ylabel(['WAVELET: ',wavelet])

for i=1:3*level+1
    line([ind(i+1); ind(i+1)],[v(3);v(4)])
end

text(v(1), dv, [num2str(level),',','0']);
for i=1:level
    for j=1:3
        text(ind(3*(i-1)+j+1), dv, [num2str(level-i+1),',',num2str(j)]);
    end
end

h=axes('Position',[0.52 0.45 0.4 0.4]);
A1=IM_decshow(W);
title('{\bf (b)} IMAGE DECOMPOSITION')
axis tight; v=axis;


function [W,PAT,f2]=IM04_FEATURE_SSQ(wavelet,level,GN)
% Image Feature Extraction using wavelet decomposition
%   wavelet - wavelet decomposition function
%   level - wavelet decomposition level
%   GN - multidimensional array of images
%   W(level).matrix - structured array of decomposition images at a given
%   level
%   matrix: A-approximation, H-horizontal, V-vertical, D-diagonal
%   detail coefficients
%   PAT - pattern matrix for image classification
%   f2 - image handle
% Uses IMDEC.M and IMDECSHOW.M

N=length(GN);
disp(['Number of images: ', num2str(N)]);
fprintf('\n')

F1(1)=input('Level of the first feature (=1,2,..): ');
F1(2)=input('Property of the first feature (=''H|V|D'')): ');
F2(1)=input('Level of the second feature (=1,2,..): ');
F2(2)=input('Property of the second feature (=''H|V|D'')): ');
fprintf('\n')
fprintf('\n')

for ii=1:N
    % Definition of input values
    %GN{ii}=nnzmatrix(GN{ii});
```

```
    [XL,YL]=size(GN{ii});
    disp(['Image size: ', num2str(XL),'x',num2str(YL)]);

    % Evaluation of image features
    for v=1:level
        G1=GN{ii};
        S=G1-mean2(G1); S=(S-min(S(:)))/(max(S(:))-min(S(:)));
        [W,c,ind]=IM_dec(S(:,:),v,wavelet);
        FH=sumsqr(W(v).H(:));
        FV=sumsqr(W(v).V(:));
        FD=sumsqr(W(v).D(:));

        PH(v)=(FH);
        PV(v)=(FV);
        PD(v)=(FD);
    end
    eval(['P1=P',F1(2),'(',num2str(F1(1)),');']);
    eval(['P2=P',F2(2),'(',num2str(F2(1)),');']);
    PAT([1 2],ii)=[P1; P2];

end

% Visualization of image features
f2=figure(2); axis([min(PAT(2,:)) max(PAT(2,:)) min(PAT(1,:)) max(PAT(1,:))]);
for i=1:N
    text(PAT(2,i),PAT(1,i),num2str(i));
end
grid on; title('CLUSTERS OF IMAGE FEATURES')
xlabel('Feature(2,:)'); ylabel('Feature(1,:)');


function [W,PAT,f2]=IM04_FEATURE_MSTD(wavelet,level,GN)
% Image Feature Extraction using wavelet decomposition
%   wavelet - wavelet decomposition function
%   level - wavelet decomposition level
%   GN - multidimensional array of images
%   W(level).matrix - structured array of decomposition images at a given
%   level
%   matrix: A-approximation, H-horizontal, V-vertical, D-diagonal
%   detail coefficients
%   PAT - pattern matrix for image classification
%   f2 - image handle
% Uses IMDEC.M

N=length(GN);
disp(['Number of images: ', num2str(N)]);
fprintf('\n')

F1(1)=input('Level of the first feature (=1,2,..): ');
```

```
F1(2)=input('Property of the first feature Mean (=''M'')): ');
F2(1)=input('Level of the second feature (=1,2,..): ');
F2(2)=input('Property of the second feature STD (=''S'')): ');
fprintf('\n')
fprintf('\n')

for ii=1:N
    % Definition of input values
    % GN{ii}=nnzmatrix(GN{ii});
    [XL,YL]=size(GN{ii});
    disp(['Image size: ', num2str(XL),'x',num2str(YL)]);
    % Evaluation of image features
    for v=1:level
        G1=GN{ii};
        S=G1-mean2(G1);
        S=(S-min(S(:)))/(max(S(:))-min(S(:)));
        % DWT decomposition
        [W,c,ind]=IM_dec(S(:,:),v,wavelet);
        % detail coefficients at level v
        GG=[W(v).H(:); W(v).V(:); W(v).D(:)];
        % Mean
        PM(v)=mean(GG);
        % Standard deviation
        PS(v)=std(GG);
    end
    eval(['P1=P',F1(2),'(',num2str(F1(1)),');']);
    eval(['ST1=P',F2(2),'(',num2str(F2(1)),');']);
    % pattern matrix
    PAT([1 2],ii)=[P1; ST1];
end

% Visualization of image features
f2=figure(2); axis([min(PAT(2,:)) max(PAT(2,:)) min(PAT(1,:)) max(PAT(1,:))]);
for i=1:N
    text(PAT(2,i),PAT(1,i),num2str(i));
end
grid on; title('CLUSTERS OF IMAGE FEATURES')
xlabel('Feature(2,:)'); ylabel('Feature(1,:)');



function IM04_CLASS(W1,b1,f3)
% function IM04_CLASS(W1,f3)
% Neural network boundaries
%   f3 - figure handle
%   W1,b1 - final neural network coefficients

figure(f3);
S=length(W1(:,1));
```

```
v=axis; PVX=[v(1),v(2)];
for q=1:S-1             % Line q definition
    for n=q+1:S
        PXO=[];
        PY=(sum(W1(q,:).^2)-sum(W1(n,:).^2)-2*PVX*diff(W1([n q],2)))/...
            (2*diff(W1([n q],1)));
        hty=line(PVX,PY,'color','b','LineStyle',':','LineWidth',0.2);
        % Evaluation of intersection of boundaries
        for m=1:S
            if m~=n & m~=q
            PXO=[PXO (diff(W1([m,q],1))*(sum(W1(q,:).^2)-sum(W1(n,:).^2))...
                  -diff(W1([n,q],1))*(sum(W1(q,:).^2)-sum(W1(m,:).^2)))/...
                  (2*(diff(W1([n,q],2))*diff(W1([m,q],1))...
                      -diff(W1([m,q],2))*diff(W1([n,q],1))))];
            end
        end
        PXO=PXO( PXO > v(1) & PXO < v(2) );
        PXO=[v(1) PXO v(2)]; PXO=sort(PXO);
        PYO=(sum(W1(q,:).^2)-sum(W1(n,:).^2)-2*PXO*diff(W1([n q],2)))/...
            (2*diff(W1([n q],1)));
        if S>2
            for i=2:length(PXO);        % Boundary lines segmentation
                L=[];
                for m=1:S               % Intersection classification
                  if m~=n & m~=q
                   PX=(PXO(i-1)+PXO(i))/2;
                   PY=(sum(W1(q,:).^2)-sum(W1(n,:).^2)-...
                      2*PX*diff(W1([n q],2)))/(2*diff(W1([n q],1)));
                      L=[L dist(W1(q,:),[PY;PX]) < dist(W1(m,:),[PY;PX]) ];
                    end
                end
                if ~isempty(L) & prod(L)>0
                    line(PXO(i-1:i),PYO(i-1:i),'color','k',...
                        'LineStyle','-','LineWidth',2);
                end
            end
        end
    end
end
```

Appendix E
Index

Index

Appendix F
Selected Papers

[1] A. Procházka, M. Pánek, V. Musoko, M. Mudrová and J. Kukal. **Decomposition and Reconstruction Methods in Biomedical Image De-noising.** *International EURASIP Conference*, BIOSIGNAL 2002.

[2] V. Musoko, M. Kolínová and A. Procházka. **Image Classification Using Competitive Neural Networks.** *12th Scientific Conference MATLAB2004*, Humusoft, 2004.