Institute of Chemical Technology in Prague

Faculty of Chemical Engineering

Department of Computing and Control Engineering

CONTENT BASED IMAGE SEGMENTATION

MASTERS THESIS

Author: Eva Hošťálková

Supervisor: Prof. Ing. Aleš Procházka, CSc.

Prague May 2006

Acknowledgement

I would like to thank my supervisor Professor Aleš Procházka for his kind help and advice.

I do hereby declare that I am the sole author of this thesis.

Prague, May 2006 Signature .....

Summary

This thesis deals with image segmentation as a widely utilised instrument of image processing. Two segmentation tools are introduced here, i.e. the watershed transform and feature-based image segmentation (FBIS). Chap. 1 supplies a brief introduction to primary problems of content-based image retrieval (CBIR). One of these problems is FBIS, whose applications are listed here. Chap. 2 presents the one-dimensional and two-dimensional Fourier transform (FT) and space domain filters, both employed in signal and image denoising. Fundamentals of wavelet theory are also stated here. Chap. 3 discusses use of the wavelet transform (WT) in signal and image denoising and texture-based image characteristics computation. Chap. 4 describes the watershed segmentation in conjunction with the distance transform, gradient magnitude, and marker-control methods. The effect of the amount of additional noise on segmentation is demonstrated in examples. Chap. 5 is devoted to feature-based image segmentation (FBIS) carried out on simulated and real magnetic resonance (MR) images. Same as in Chap. 4, the influence of image noise on FBIS is discussed.

All programmes are written in Matlab using particularly the commands of the Image Processing Toolbox and Wavelet Toolbox. Segments of the source code are displayed as figures.

Souhrn

Diplomová práce se zabývá metodami segmentace obrazů, která je významnou součástí teorie zpracování signálů. Uvádíme dvě metody segmentace, a to *watershed* transformaci a segmentaci založenou na charakteristikách obrazů (FBIS). Kap. 1 je zaměřena na základní problematiku vyhledávání obrazů v databázích na základě jejich obsahu (CBIR), jejíž součástí je i segmentace obrazů. Kap. 2 uvádí jedno-dimensionální a dvou-dimensionální Fourierovu transformaci (FT) a prostorovou filtraci jako metody odstranění šumu ze signálů nebo obrazů. Jsou zde též uvedeny základy *wavelet* transformace (WT). Kap. 3 ukazuje využití WT pro odstraňování šumu pomocí prahování, a pro výpočet koeficientů charakterizujicích texturu. Kap. 4 popisuje *watershed* segmentaci ve spojení s distanční transformací a dalšími technikami. Názorně je prezentován vliv množství rušivé složky v obrazu na výsledky segmentace. Kap. 5 se zabývá FBIS prováděné na simulovaných i reálných obrazech magnetické resonance (MR). Zde stejně jako v kap. 4, zkoumáme vliv velikosti šumu na celkovou segmentaci.

Všechny programy jsou napsány v prostředí Matlabu využívaje převážně *Image Processing Toolbox* a *Wavelet Toolbox*. Části programů jsou uvedeny v textu jako obrázky.

Contents

List of Figures

List of Tables

.

1

Introduction

1.1 Problems of Content-Based Image Retrieval (CBIR)

Content-based image retrieval (CBIR) [10] is a technique for searching images in large image databases. This method is based on the visual information contained in images so-called *low-level characteristics*.

Image retrieval has been developed since the late 1970s. At that time, *text-based search* [10] was employed as a traditional database technique using standard Boolean queries. This technique requires images to be annotated with text representing semantic interpretation of an image content so-called *high-level characteristics*. Text annotation needs to be done manually, and thus becomes expensive for very large image databases. And furthermore, this semantic description is context-sensitive, incomplete and often subjective. This forms one of the fundamental problems of image retrieval, which is to match high-level semantic characteristics and low-level visual characteristics. It is also difficult for the user to choose right keywords and their structure to define a query.

With the boom of digital images in the early 1990s, a new approach towards image retrieval had to be developed to substitute the text-based retrieval. The CBIR method utilises visual characteristics of image content such as colour, texture, shape and spatial information, which are low-level characteristics.

CBIR systems work in the following manner [10]. As a query definition, the user sketches a rough outline of an image with a graphic editing tool or provides the system with an example image or group of images with similar features.. The system computes feature vectors for the query, and then searches through the database of indexed images and calculates their similarity measure to the user input. According to the similarity measure, the system generates a ranked list of search results. The user marks the relevant and irrelevant search results in order to influence the next stage of retrieval. This action is repeated several times until the desired image is found. *Users' relevance feedback*, as this loop is called, modifies the retrieval process to be more efficient.

Images in the database need to be indexed with their features so as the CBIR system is able to search through the database. Image features are either *global* or *local*, which characterise individually each object in an image. The second option is more natural for users, but on the other hand, requires *image segmentation*.

For large databases, segmentation needs to be done automatically without human supervision during the segmentation process. This represents another fundamental problem of CBIR. There is no universal algorithm for image segmentation, which would suit all kinds of images. We can either employ region segmentation for finding homogenous regions or complete object segmentation for identifying objects, which are semantically meaningful.

After segmentation, we may extract *features* of the resulting segments may be extracted and subsequently classify the segments according to their features by self-organising *neural networks*

(NN). However, feature based classification using NN is beyond the scope of this work.

This thesis introduces two image segmentation tools, i.e. *watershed segmentation* and *feature-based image segmentation* (FBIS) discussed in Chap. 4 and 5, respectively. FBIS [3] is a segmentation technique engaging visual descriptors of images called features. Any rule that captures the desired information from an image can be used as a feature. Features need to be chosen in accordance to their ability to distinguish similarities and dissimilarities between structures composing objects. Chap. 5 focuses on FBIS in more detail.

1.2 Applications of Feature-Based Image Segmentation (FBIS)

Apart from image retrieval, feature-based image segmentation (FBIS) using texture decriptors is utilised in a range of interdisciplinary applications, such as microscopic, satellite, and biomedical imagery. To be more specific, in microscopic imagery, FBIS is employed to examine fabric texture of ropes used in mountaineering, which is highly important to safety. In satellite images, this technique is used to detect cumulus cloud fields [3]. And lastly in magnetic resonance (MR) imagery, this method is implemented to identify various body tissues, which can be subsequently modelled in three dimensions using two sets of MR images in two perpendicular planes. Fig. 1.1 pictures an MR image of the spine and its cut.

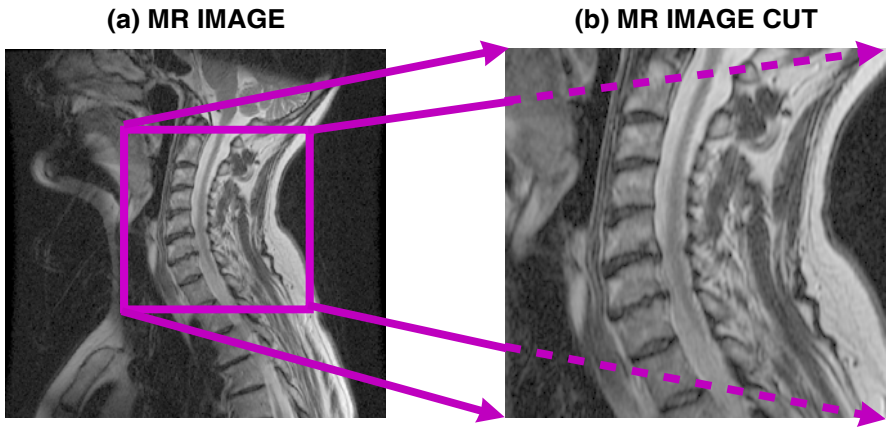


FIGURE 1.1. MR image of spine and its cut.

.

2

Mathematical Methods of Image Processing

In this chapter, we describe signal processing methods in the time and frequency domain, and multiresolutional analysis for one dimensional and two dimensional signals. Signal denoising is the application that we particulary focus on.

2.1 Time Domain Signal Processing

Before approaching the filter theory representing the fundamentals of digital signal processing, let us note that in this work, terms *time* and *space* are used as synonyms. As for image processing, the term *space* is usually preferred.

Filters are used in almost every electronic system. Originally, continuous-time or *analogous* filters were preferred to *digital* ones. However, vast development of computers implied the necessity of digital filters. Analogous filters [16] are specialised electronic circuits composed of resistors, capacitors, operation amplifiers etc., which process current or voltage analogous signals. Whereas digital filters [16] implement filtering of digitised signals as numerical computations using digital processors, such as general purpose PCs or specialised digital signal processor chips. At first, digital filter design was based on deriving from analogous ones. Nowadays, the design is independent on continuous-time systems.

Digital filters are employed in a wide range of applications [14], such as noise reduction, channel equalization, radar, audio and video processing, biomedical signal processing, financial data analysis etc. In these applications, the following basic functions of digital filters [14] are utilised. The first one is constraining a signal into a given frequency band or channel (e.g. anti-aliasing filters), second, decomposing a signal into frequency bands (e.g. filter banks, frequency multiplexers), third, modifying frequency spectrum of a signal (e.g. audio graphic equalizers, channel equalizers), and last, modelling of the input-output relationship of a system (e.g. music synthesizers).

We can classify filters [14], for instance, as *linear* or *non-linear* filters, *adaptive time-varying* or *non-adaptive time-invariant* filters. In this work, we consider only *linear time-invariant* filters, which are described by a linear difference equation with constant coefficients, see Eq. (2.1). We also distinguish finite impulse response (*FIR*) filters and infinite impulse response (*IIR*) filters. These both types of filters are to be described below in Subsec. 2.1.1.

According to the band of frequencies allowed through the filter, filters are classified into the following categories [14]. First, *low-pass* filters let pass through only the frequencies lower than the cut-off frequency w_c. Second, *high-pass* filters allow through only the frequencies higher than w_c. Third, *band-pass* filters and *band-stop* filters preserve or attenuate, respectively, only such frequencies in a signal, which fall into the band between the lower and the higher cut-off frequency. The last category to be mentioned is *filter banks*. Filter banks split a signal into frequency bands, uniformly or non-uniformly spaced in frequency.

2.1.1 Methods of Filters Description

We use several methods to describe filters. One of them is the time domain *difference equation* relating the input and output of a filter. This equation describes the fact, that the current output sample is a weighted combination of the input samples and the previous output samples. For a linear time-invariant filter [14]

$$y(m) = \sum_{k=1}^{N} a_k \, y(m-k) + \sum_{k=0}^{M} b_k \, x(m-k) \tag{2.1}$$

where a_k and b_k are filter coefficients calculated to obtain the desirable frequency response of the filter, N and M are the numbers of these coefficients, respectively, whilst N is called the *filter order*. $y(m)$ and $x(m)$ are the output and input signal, respectively, with the time index m.

Another way of representing filters is the *impulse response*. In this case, the input signal is an impulse that is non-zero just at a given time m. The impulse response is a very useful description method, because any digital signal can be represented by a series of shifted and scaled impulses, and the impulse covers all frequencies with equal energy.

Another means of filters description is the *transfer function* [14], that is the ratio of the z-transferred function of the output and input of a filter

$$H(z) = \frac{Y(z)}{X(z)} \tag{2.2}$$

For a linear time-invariant filter, the transfer function runs as follows

$$H(z) = \frac{\sum_{k=0}^{M} b_k \, z^{-k}}{1 - \sum_{k=1}^{N} a_k \, z^{-k}} \tag{2.3}$$

The transfer function provides us with important filter characteristics. These are so-called *poles* and *zeros*, i.e. roots of the denominator and nominator of the transfer function, respectively. The former introduce resonances and the latter introduce deeps in frequency. Provided the coefficient vectors $a_k, b_k \in \mathbf{R}$, the poles and zeros occur in complex conjugate pairs [14].

The *frequency response* may also be used for filters description. It is a complex variable based on how filters change the magnitude and phase of the input, as reflected by the phase and magnitude response, respectively. The frequency response can be obtained either by calculating the *Fourier transform* (see Sec. 2.2) of the impulse response, or by substituting for $z = e^{jw}$ in the z-transfer function from Eq. (2.2) given as [14]

$$H(e^{jw}) = \frac{Y(e^{jw})}{X(e^{jw})} \tag{2.4}$$

2.1.2 FIR and IIR Filters

Finite-duration impulse response (*FIR*) filters can be described by the following difference equation [14]

$$y(m) = \sum_{k=0}^{M} b_k \, x(m-k) \tag{2.5}$$

This relation distinctly shows that FIR filters have no *feedback*, in other words, the output is independent on the previous outputs. Thus FIR filters are also called *non-recursive* filters.

The FIR filter transfer function given by the next equation contains no poles [14]. Consequently, FIR filters are sometimes called *all-zero* filters.

$$H(z) = \sum_{k=0}^{M} b_k \, z^{-k} \tag{2.6}$$

Infinite-duration impulse response (*IIR*) filters are also called *recursive* filters. The output is a function not only of the input, but also of the previous outputs. This relationship produces the feedback in the difference equation given by [14]

$$y(m) = \sum_{k=1}^{N} a_k \, y(m-k) + \sum_{k=0}^{M} b_k \, x(m-k) \tag{2.7}$$

The transfer function expressed by Eq. (2.8) contains poles, and can also contain a number of zeros [14]. Therefore, IIR filters are sometimes called *zero-pole* filters. The number of finite poles must equal or exceed the number of finite zeros. If having no zeros, IIR filters are called *all-pole* filters.

$$H(z) = \frac{\sum_{k=0}^{M} b_k \, z^{-k}}{1 - \sum_{k=1}^{N} a_k \, z^{-k}} \tag{2.8}$$

In comparison with FIR filters, IIR filters require less coefficients to achieve the same desired frequency response. This fact implies smaller memory and computational requirements. On the other hand, FIR filters are always stable, whereas IIR filters can become unstable, for example, when the poles are outside the unit circle in the pole-zero plane.

FIR and IIR filters may be combined into *direct*, *cascade*, or *parallel* structures [14]. These structures differ in complexity, cost of implementation, computational efficiency and stability. The direct form is the simplest one, however; IIR filters cannot be employed in this structure due to the problems with their stability.

2.1.3 Filter Applications

The first of the filter applications demonstrates removing of the higher-frequency component from the signal composed of two frequencies using the IIR filter of order 6 designed with the Matlab command *butter* [12], and two FIR filters of orders 20 and 36 designed with the command *fir1* [12]. In Fig. 2.1, we can see the comparison between the outputs of the above mentioned filters produced by the command *filter* [12]. For FIR filters, there is a considerably long space delay due to the high filter orders. The IIR filter has order 6, and therefore the output delay is not so obvious.

Fig. 2.2 displays the frequency spectrum of the input signal and the frequency response of the IIR and both FIR filters. To achieve similar frequency response, the orders of the FIR filters have to be several times higher than that of the IIR filter. The Matlab code generating the above mentioned filters and their frequency response is quoted in Fig. 2.3.

A practical example of the *Electroencephalograph* (EEG) signal denoising is displayed in Fig. 2.4 and 2.5. The EEG has the sampling frequency of 200 Hz. We aim to remove the noise component

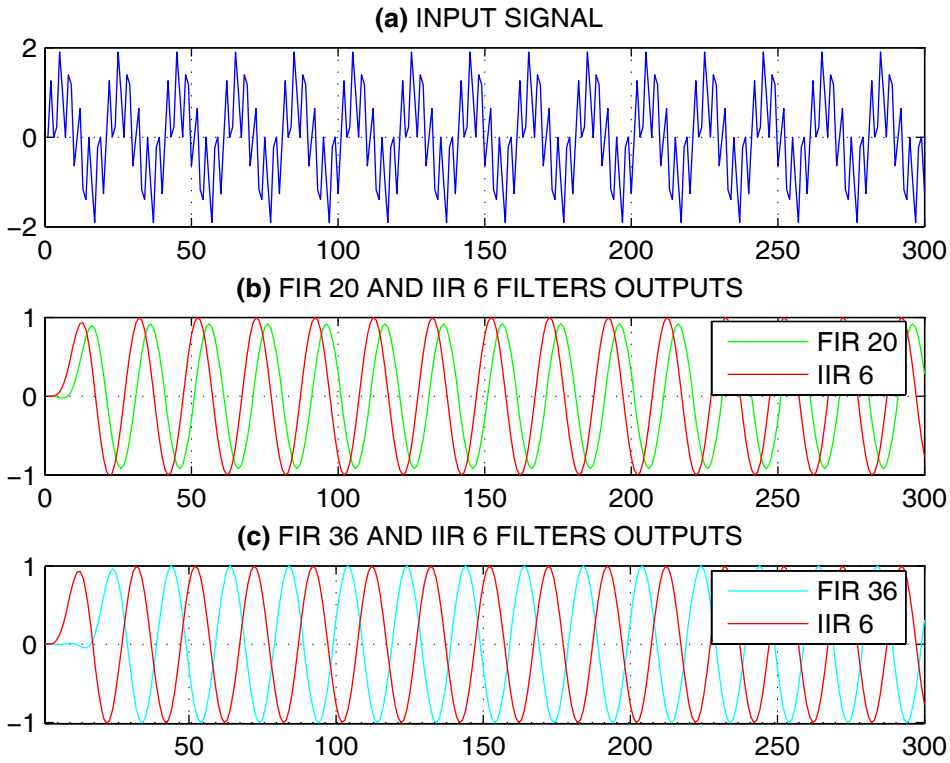


FIGURE 2.1. 1D signal denoising using FIR and IIR filters of different orders.

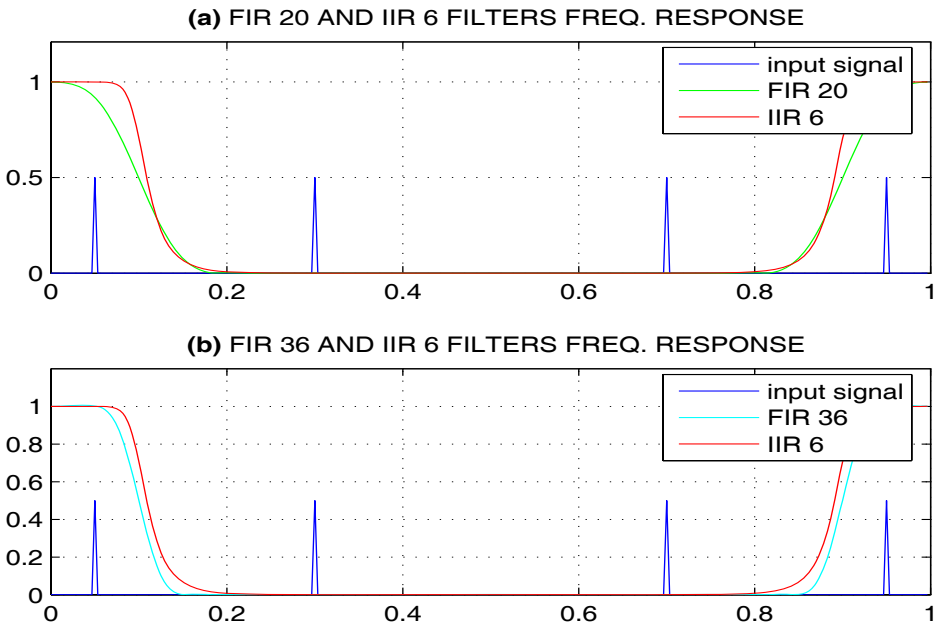


FIGURE 2.2. FT of 1D signal, frequency response of FIR and IIR filters of different orders.

of 50 Hz corrupting the signal, which originates from electrical interference from the main supply [16]. The band-stop FIR filter of order 100 serves obviously well for this purpose.

In two-dimensional space, IIR filters cannot be used because of their instability. Fig. 2.6 shows the 2D signal denoising using the FIR filter. The 2D input signal containing two frequency

```
% DE-NOISING IN 1D SPACE DOMAIN
% SIGNAL GENERATION
  N=300; n=[0:N-1]'; x=sin(2*pi*0.05*n)+sin(2*pi*0.3*n);
% LOW-PASS FILTERS DESIGN AND FILTERING
  wc=0.2   % cut-off freq.
  a=1, b1=fir1(20,wc)   % FIR order 20, hamming window
  y1=filter(b1,a,x);   % FIR 20 filter output
  b2=fir1(36,wc)   % FIR order 36, hamming window
  y2=filter(b2,a,x);   % FIR 36 filter output
  [b3,a3]=butter(6,wc)  % IIR order 6, butterworth window
  y3=filter(b3,a3,x);   % IIR 6 filter output
% FREQUENCY RESPONSE
  [H1 w1]=freqz(b1,1,N,'whole');   % FIR order 20
  [H2 w2]=freqz(b2,1,N,'whole');   % FIR order 36
  [H3,w3]=freqz(b3,a3,N,'whole');   % IIR order 6
```

FIGURE 2.3. Matlab code for 1D signal denoising using FIR and IIR filters of different orders.

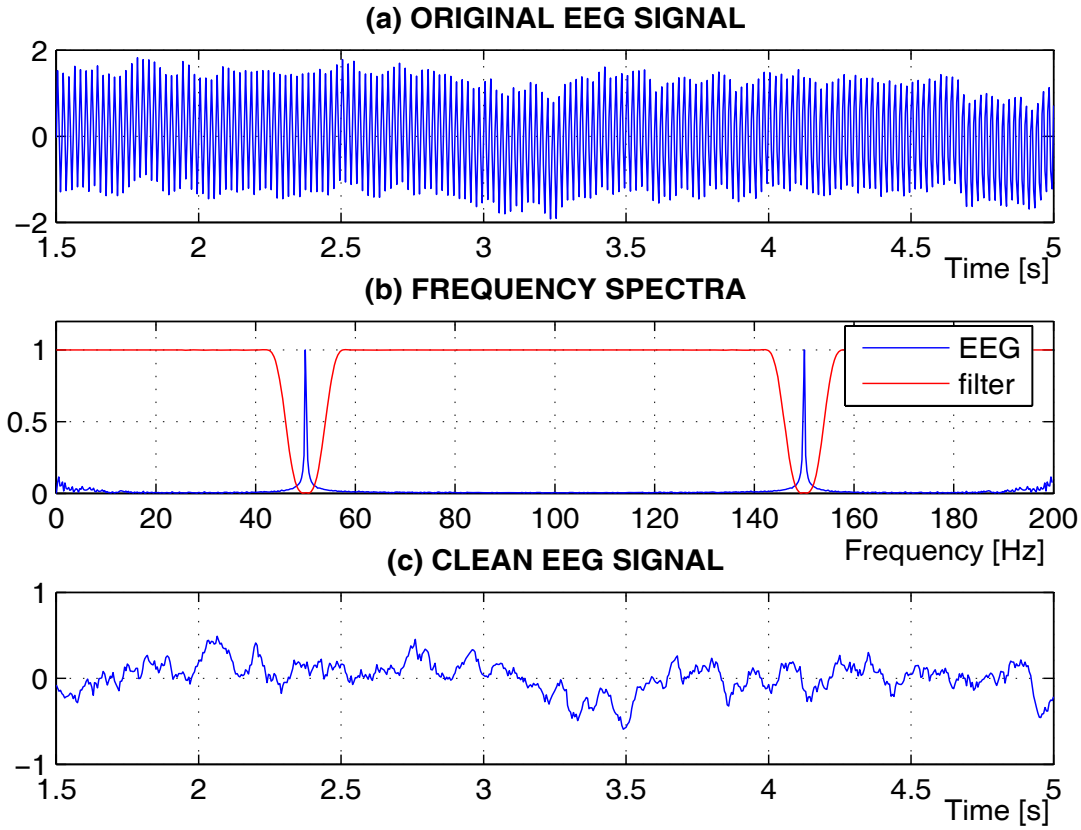


FIGURE 2.4. EEG signal denoising using FIR filter of order 100.

components is shown in Fig. 2.6a, the FT of the input signal in Fig. 2.6b, the FT of the signal and the frequency response of the FIR filter in Fig. 2.6c, the FT of the output signal Fig. 2.6d, and finally, the signal after windowing in Fig. 2.6e.

The 2D FIR filter is designed by the Matlab function *fwind1* [11] based on the desired filter

```
% EEG SIGNAL DENOISING USING FIR FILTER
  load EEGshort  % load EEG signal
  [m,n]=size(EEGshort), Fs=200  % sampling frequency
  x=EEGshort(5,200:800);  % take chanel 5 (out of m=19 channels) samples(200:800)
  x=(x-mean(x))/std(x);  % remove trend
  N=length(x), n=[1:N];
  X=fft(x);  % FT for frequency plot
  X=(X-min(X(:)))/(max(X(:))-min(X(:)));
% BAND-STOP FIR FILTER OF ORDER 100
  Wn=[0.4 0.5];   %cut-off freq.
  a=1, b=fir1(100,[46 54]/(Fs/2),'stop');  % filter coef., hamming window
  y=filter(b,a,x);  % filtering
  [H w]=freqz(b,1,N,'whole');  % filter freq. response for frequency plot
```

FIGURE 2.5. Matlab code for EEG signal denoising using FIR filter of order 100.

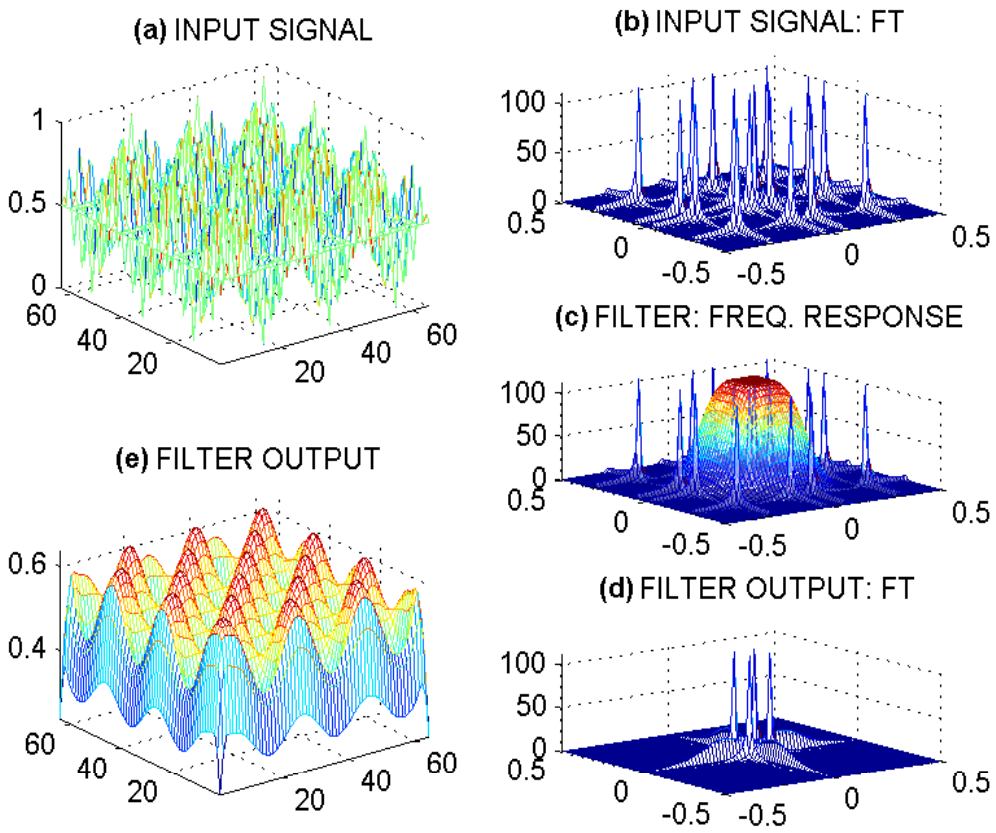


FIGURE 2.6. 2D signal denoising using FIR filter.

frequency response and a chosen window shape. In this case, we use the Hamming window of length 15 as noted in the enclosed Matlab code in Fig. 2.7.

```
% DE-NOISING IN 2D SPACE DOMAIN
% SIGNAL GENERATION
  N=64; n=[0:N-1]'; M=64; m=[0:M-1]';
  x=sin(2*pi*0.05*n)+sin(2*pi*0.3*n); X=x*x';
  X=(X-min(min(X)))/(max(max(X))-min(min(X)));   % normalization
% FIR FILTER DESIGN AND FILTERING
  [f1,f2]=freqspace(N,'meshgrid'); r=sqrt(f1.^2 + f2.^2);   % circle
  Hd=ones(N);  % desired freq. response Hd
  Hd((r<0)|(r>0.4))=0;   % 0.4 corresponds to wc=0.2
  h=fwind1(Hd,hamming(15));   % filter coef. vector h
  Y=filter2(h,X);   % filter output
% DFT: FILTER VERIFICATION
  XX=fft2(X-mean(mean(X))); Xs=fftshift(XX);
  YY=fft2(Y-mean(mean(Y))); Ys=fftshift(YY);
```

FIGURE 2.7. Matlab code for 2D signal denoising using FIR filter.

2.2 Frequency Domain Signal Analysis

The objective of signal analysis in the frequency domain is to determine what frequencies a given signal is composed of, so that we can modify the *frequency spectrum* of the signal for our purposes. A powerful tool for signal frequency analysis is the *Fourier transform*.

2.2.1 Discrete Fourier Transform (DFT)

Continuous Fourier Transform

The *Fourier transform* (FT for short) is a result of Jean Babtiste Joseph Fourier's work dating back to the early 19^{th} century. The FT coefficients are calculated as linear combinations of the *basis functions*, which in the case of the FT are sines and cosines, in other words *complex exponentials*. We can therefore say, that the FT breaks down a signal into constituent sinusoids of different frequencies [5]. The basis functions are *orthogonal* which property is a principal assumption for signal reconstruction from the FT coefficients by the means of the *inverse Fourier transform*. *Orthogonality* is explained in detail in Sec. 2.3.

For the radian frequency $w = 2\pi f$ and time t, the FT is defined as:

$$X(\omega) = \int_{-\infty}^{+\infty} x(t)\, e^{-j\omega t}\, dt \tag{2.9}$$

In Eq. (2.9), the frequency spectrum $X(\omega)$ of the signal $x(t)$ may be interpreted as a linear combination of complex exponentials. $X(\omega)$ is continuous and thus the number of the exponentials is indefinite [4]. The inverse FT is given by:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\omega)\, e^{j\omega t}\, d\omega \tag{2.10}$$

Eq. (2.9) and Eq. (2.10) apply only to signals $x(t)$ of finite energy:

$$Energy = \int_{-\infty}^{+\infty} |x(t)|^2\, dt < \infty \tag{2.11}$$

This implies that $x(t)$ must be *aperiodic*, since periodic signals have infinite energy. To obtain frequency analysis of a *periodic* signal, we need to employ the Fourier series with coefficients a_k and fundamental frequency ω_0 [4]:

$$x(t) = \sum_{k \in \mathbf{Z}} a_k\, e^{jk\omega_0 t} \tag{2.12}$$

If ω_0 increases, the spectral lines become more compact, and in the limit $\omega_0 \to \infty$, the signal becomes aperiodic and its spectrum continuous.

Sinusoids as the FT basis functions are infinitely differentiable. In case that $x(t)$ is not smooth, such as a rectangular pulse, a discontinuity in a signal introduces ripples in the FT in the vicinity of this point. These ripples have a maximum overshoot of 9% of the discontinuity height. This effect is called the *Gibbs phenomenon*. The increase of the number of harmonics N (see Eq. (2.12)) causes compression of the ripples towards the discontinuity, however, it does not decrease the 9%-overshoot [14].

The FT properties, such as linearity, duality etc., are analogous to the *discrete Fourier transform* properties. They are widely described and derived in literature, see [14, 4]. Let us just mention the property of *symmetry*. The FT $X(\omega)$ of a real function $x(t)$ is symmetric:

$$X(-\omega) = X^*(\omega) \tag{2.13}$$

where $X^*(\omega)$ denotes the complex conjugate of $X(\omega)$. As a result of this relation, $\Re\{X(\omega)\}$ is an even function and $\Im\{X(\omega)\}$ is an odd function. Hence $X(\omega)$ for positive ω provides us with full spectral information of real signals [4].

Eq. (2.13) implies that for a *real even* function $x(t) = x(-t)$, $X(\omega)$ is also a *real even* function:

$$X^*(\omega) = X(-\omega) = \int_{-\infty}^{+\infty} x(t)\, e^{j\omega t}\, dt = \int_{-\infty}^{+\infty} x(-t)\, e^{-j\omega t}\, dt = X(\omega) \tag{2.14}$$

Analogously, for a *real odd* function $-x(t) = x(-t)$, $X(\omega)$ is also a *real odd* function. We can break down a function into its even and odd component, the real and the imaginary part, respectively, of its FT will correspond to them. This property applies to the Fourier series as well [4].

Discrete Fourier Transform

Because of a wide use of computers, continuous signals and transforms have to be converted to discrete ones. In practise, the most spectral analysis is based on the *discrete Fourier transform* (DFT).

We can obtain a discrete-time signal by sampling its continuous-time version. According to the sampling theorem, also called the *Nyquist's theorem*, the choice of the sampling frequency f_s is constrained. f_s has to be high enough to cover all frequencies contained in the original signal. This theorem states that a continuous-time signal can be completely recovered from its samples, if and only if f_s is at least twice the Nyquist's rate [14]

$$f_s \geq 2 f_{max} \tag{2.15}$$

where f_{max} is the maximum frequency contained in the original signal or the bandwidth of the signal. In case that f_s does not satisfy the Nyquist's theorem, the phenomenon called *aliasing* takes place. *Aliasing* is a distortion caused by overlapping of two replications of the spectral envelope of a periodic frequency spectra corresponding to the sampled signal [4]. Aliasing is sometimes called *spectral leakage*.

When an aperiodic signal of length N is sampled, the FT is periodic, but continuous. The DFT is derived from sampling the FT of discrete signals. We sample one period 2π of the FT to obtain N uniformly spaced DFT coefficients. Both the FT and the DFT are invertible. Due to the fact that the inverse FT of a discrete function is a periodic function, for the DFT computation, we assume the original non-periodic signal to be periodic with the period N [14]. Eq. (2.16) is the DFT formula, where $x(n)$ is a time-series and $X(k)$ is its DFT, for $k = 0, \ldots, N-1$.

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j\,\frac{2\pi}{N}\,nk} \tag{2.16}$$

The inverse DFT for $n = 0, \ldots, N-1$ is defined as follows:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, e^{j\,\frac{2\pi}{N}\,nk} \tag{2.17}$$

Fast Fourier Transform Algorithm

Originally, the DFT was computed by estimating the correlation function for a given signal and then carrying out the FT of this function to obtain the approximation spectrum of the signal. This method was commonly used until 1960s [7].

That was the time, when the *fast Fourier transform* (FFT) was developed. The FFT is an efficient algorithm for computing the FT spectra of time series [7] and is utilised also in Matlab as the *fft* function [12]. This algorithm employs decimation in both time and frequency. The basic idea is to modify the transform of the length N into two transforms, i.e. the transform of $\frac{N}{2}$ even samples and the transform of $\frac{N}{2}$ odd samples. The FFT reduces the number of computations from $2N^2$ to $2\log_2 N$ [15].

2.2.2 Two-Dimensional Discrete Fourier Transform (2D DFT)

The one-dimensional DFT can be extended to two dimensions preserving most of its properties. The 2D DFT and its inverse version are defined in Eq. (2.18) and Eq. (2.19), respectively. For $k = 0, \ldots, M-1$ and $l = 0, \ldots, N-1$:

$$X(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x(m,n)\, e^{-j2\pi\,(\frac{mk}{M}+\frac{ln}{N})} \tag{2.18}$$

For $m = 0, \ldots, M-1$ and $n = 0, \ldots, N-1$:

$$x(m,n) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} X(k,l)\, e^{j2\pi\,(\frac{mk}{M}+\frac{ln}{N})} \tag{2.19}$$

2.2.3 Windowing

Windowing can be applied to signals either in the time or frequency domain. By windowing a signal in the *time domain*, we constrain the length of otherwise infinite length aperiodic signals, or signals of an unknown period. This length constraint limits signal energy to a finite value and thus enables the FT. By windowing a signal in the *frequency domain*, we can modify its spectrum by removing the undesirable frequency components. This technique is mostly used for signal *denoising* discussed in Subsec. 2.2.5.

Windowing in discrete time n means multiplying a signal $x(n)$ with a window $w(n)$ [14]:

$$x_w(n) = x(n)w(n) \tag{2.20}$$

where $w(n)$ can be for instance the unit-amplitude rectangular window of N-samples duration [14]:

$$w(n) = \begin{cases} 1 & 0 \le n \le N-1 \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

This multiplication corresponds to the convolution in the frequency domain [14]

$$X_w(k) = W(k) * X(k) \tag{2.22}$$

As a consequence of windowing, there are two main phenomena affecting the resulting FT $X_w(k)$. These are firstly the *finite window length effect* and secondly the *end-point effect*. To explain the first mentioned *finite window length effect*, we have to derive the FT of the rectangular

window $w(n)$ using the convergence formula for the partial sum of a geometric series [14]

$$W(k) = \sum_{n=0}^{N-1} w(n)\, e^{-j\,\frac{2\pi}{N}\,nk} = \frac{1 - e^{-j2\pi k}}{1 - e^{-j\,\frac{2\pi}{N}\,k}} = e^{-j\,\frac{N-1}{N}\,\pi k}\,\frac{\sin(\pi k)}{\sin(\frac{\pi k}{N})} \tag{2.23}$$

The shape of the spectrum $W(k)$ derived in Eq. (2.23) resembles the *sinc* function, i.e. *sine cardinal* function, which is the FT of the rectangular window in continuous time. In contrast, the FT of the rectangular window in discrete time is periodic, because of the sinus function in the denominator [4]. The periodicity of $W(k)$ can give way to mutual influence of the neighbouring replications of the spectra.

The shorter the width of $w(n)$ the wider the main lobe of $W(k)$ and therefore the poorer the frequency resolution. Consequently, the convolution of $W(k)$ with $X(k)$ given by Eq. (2.22) causes some spreading of signal energy along the frequency axis in $X_w(k)$. This is called the *finite window length effect*.

To demonstrate the finite window length effect, we have, for instance, a signal composed of two sinusoids of the frequencies f_1,f_2 and a window of the length N. Then the frequency corresponding to the window $\frac{1}{N}$ must be smaller than the difference $|f_1 - f_2|$, otherwise the two spectral components are not distinguishable [4].

The *end-point effect* represents another problem. The DFT algorithm assumes the input signal periodic with the period equal to the length of the window N. For a windowed sinus wave, the DFT depends on the number of sinus periods being windowed. If computing the DFT of an integer multiple of signal periods, we obtain the same spectrum as that of an infinite length sinusoid, i.e. the spectrum with only one frequency of a non-zero value. However, if we input an non-integer number of signal periods, the input signal is not assumed to be a pure sinusoid and the end-point discontinuities occur in the output signal [14].

The impacts of the finite length window and end-point discontinuities are, firstly, signal energy leakage over a wider frequency band, and secondly, the main-lobe of a small magnitude signal component can interfere with the side-lobes of a greater magnitude component. As a result, the smaller magnitude signal can be concealed.

To solve the end-point effect problem, we use windows gently dropping to zero, such as the *Hanning* and *Hamming* windows characterized by Eq. (2.24) and Eq. (2.25), respectively [14]. The price we pay for the smaller end-point effect is approximately twice as large the main lobe width in comparison to the rectangular window [4].

$$w_{Han}(n) = 0.5 - 0.5\,\frac{\cos 2\pi n}{N} \quad \text{for} \quad 0 \le n \le N-1 \tag{2.24}$$

$$w_{Ham}(n) = 0.54 - 0.46\,\frac{\cos 2\pi n}{N} \quad \text{for} \quad 0 \le n \le N-1 \tag{2.25}$$

2.2.4 Short-Time Fourier Transform (STFT)

The prior assumption of the FT is signal stationariness [14], i.e. the signal statistics, such as the power and the power spectrum, do not vary much over time. The FT assumes that all detected frequencies occur at all times in a signal, which is not so for non-stationary signals. In other words, the FT does not provide us with any time information about the occurrence of the frequencies present in a signal. And thus the inverse FT would reconstruct the signal

assuming, that all frequencies are present in the signal at all times. This shortcoming is partially compensated by the *short-time Fourier transform* (STFT) also called the *windowed Fourier transform* [6, 2], introduced by Dennis Gabor in 1946.

The STFT employs the windowing technique. The width of the time window is established short enough so as each chosen segment of a signal constrained by the window can be assumed stationary. The STFT is calculated as the *inner product* (see Sec. 2.3) of a signal and a windowed exponential in the time domain. The STFT $X(u, \xi)$ of a signal $x(t)$ is given by the following equation [6]:

$$X(u, \xi) = \int_{-\infty}^{+\infty} x(t)\, w(t - u) e^{-i\xi t} dt \tag{2.26}$$

where u denotes the position of the centre of the window and ξ is the corresponding radian frequency.

The STFT computation runs as follows. We start with the window's centre situated at the beginning of the signal $x(t = 0)$. Then we multiply the signal by the window function $w(t)$ and compute the FT of the product to obtain the STFT coefficient. After that, we shift the window for some u_1 samples and obtain another STFT coefficient. We repeat this procedure until we reach the end of the signal. The length of the shifting interval u_1 is adjusted in order to make the window positions overlap so as the continuity is preserved.

The narrower the window the better the time resolution, but the poorer frequency resolution. Establishing the appropriate window width is quite difficult. The size of the window is fixed, which produces constant resolution at all times and frequencies. Changing the size would increase the accuracy either in time or in frequency. If we reduce the window width to the Dirac $\delta(t - u)$, we will obtain a perfect resolution in time. On the other hand, we will ruin the frequency resolution, since the FT of the Dirac is the function $e^{-iu\omega}$ which is uniformly spread over all frequencies [13].There is a trade-off between the time and frequency resolution called the *Heisenberg's uncertainty principle* [14] stated by

$$\Delta T\, \Delta f = 1 \tag{2.27}$$

where the frequency resolution is Δf and the time resolution $\Delta T = N\, T_s$. N is the discrete window length, i.e. the number of samples of the window, and T_s is the sampling period, i.e. the continuous time interval between successive samples.

2.2.5 DFT Applications

Prior to windowing, in Matlab, we have to modify the frequency spectrum using the command *fftshift*. This command moves the zero-frequency point of the FT into the centre of the spectrum. Fig. 2.8 displays the DFT of the signal composed of two frequencies without and with *fftshift*, respectively.

In the two-dimensional case, the *fftshift* function moves the zero-frequncy point of the FT into the centre of the spectrum, by swopping the first with the third quadrant, and the second with the forth quadrant, as depicted in Fig. 2.9.

Fig. 2.11 shows removing of a high frequency component from the signal using a rectangular frequency window. In Fig. 2.11b, the blue full line displays the FT of the signal containing two different frequencies, and the red dash line displays the rectangular window. The Matlab code is displayed in Fig. 2.10.

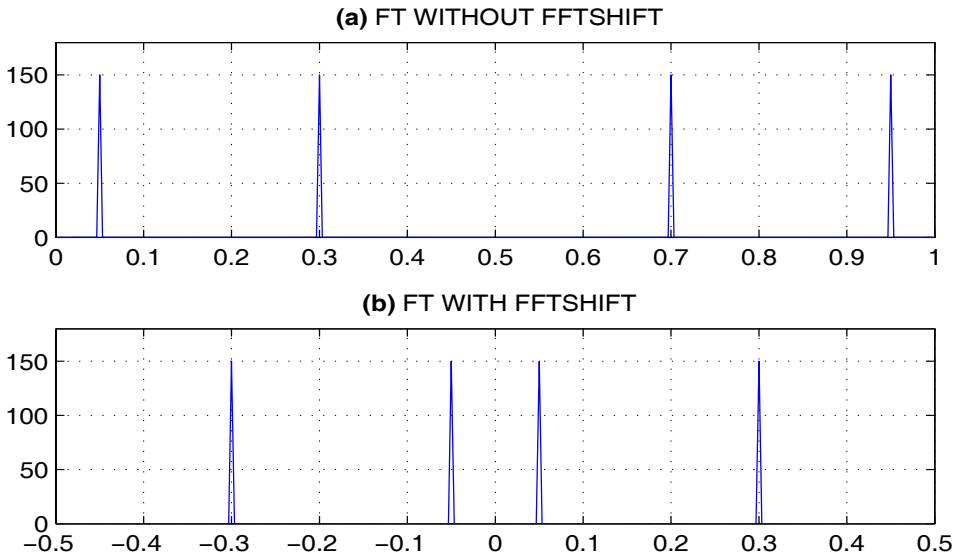


FIGURE 2.8. FT of 1D signal with and without *fftshift* command.

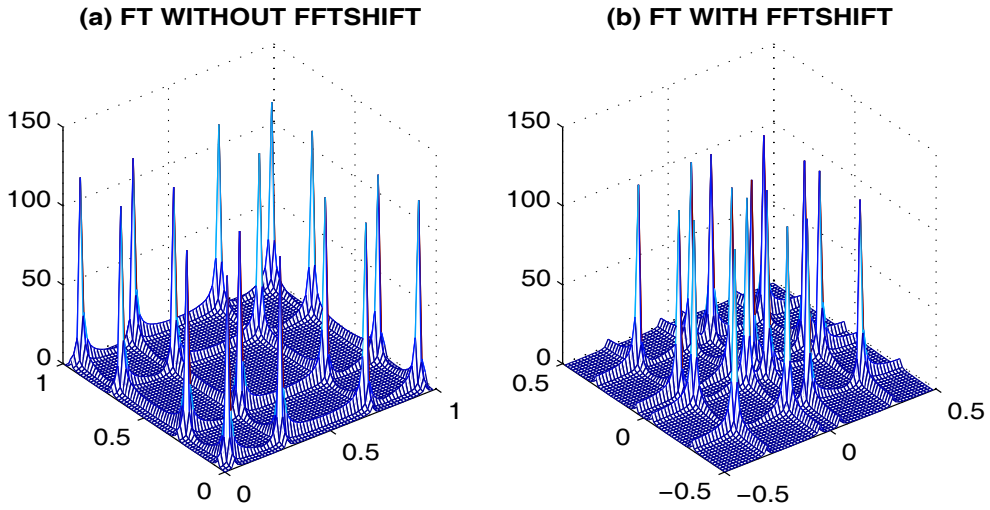


FIGURE 2.9. FT of 2D signal with and without *fftshift* command.

```
% 1D DFT: DENOISING IN FREQUENCY DOMAIN
% SIGNAL GENERATION
  N=300; n=[0:N-1]'; x=sin(2*pi*0.05*n)+sin(2*pi*0.3*n);
% DFT AND FFTSHIFT
  X=fft(x-mean(x)); Xs=fftshift(X);
% WINDOWING
  wc=0.2, NN=wc*N; NN=round(NN)  % wc=corner frequency, NN=window length
  nn=[0:NN-1]'; W=[ones(NN,1);zeros(N-2*NN+1,1);ones(NN-1,1)]; % window design
  Y=X.*W;
% INVERSE DFT
  y=ifft(Y); y=y+mean(y);
```

FIGURE 2.10. Matlab code for 1D signal denoising by frequency windowing.

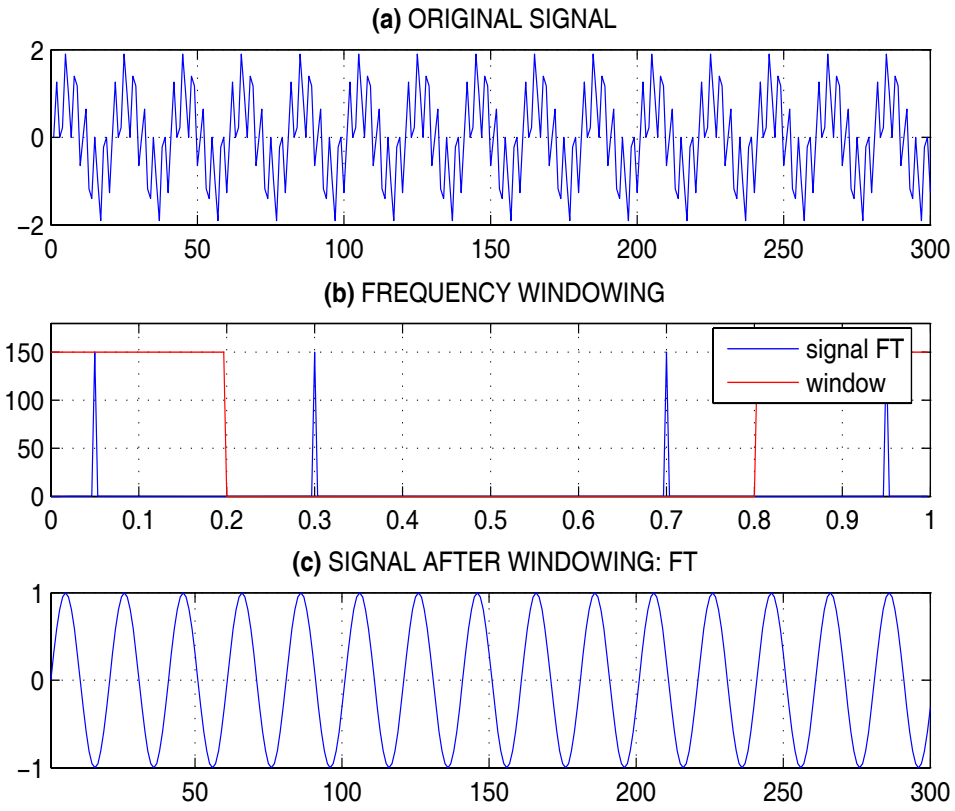


FIGURE 2.11. 1D signal denoising by frequency windowing.

Fig. 2.13 pictures removing of a high frequency component from the 2D signal using a rectangular frequency window extended to two dimensions. The figure shows clockwise the signal containing two frequencies, the FT of the signal, the FT of the signal being windowed, the FT of the signal after windowing, and finally, the signal after windowing. Fig. 2.12 shows how the Matlab code runs.

```
% 2D DFT: DENOISING IN FREQUENCY DOMAIN
% SIGNAL GENERATION
  N=64; n=[0:N-1]'; M=64; m=[0:M-1]';
  x1=sin(2*pi*0.05*n)+sin(2*pi*0.3*n);
  x2=sin(2*pi*0.05*m)'+sin(2*pi*0.3*m)'; X=x1*x2;
  X=(X-min(min(X)))/(max(max(X))-min(min(X)));   % normalization
% DFT AND FFTSHIFT
  XX=fft2(X-mean(mean(X))); Xs=fftshift(XX);
% WINDOWING
  W=ones(N,M); a=18; W([1:a N-a+2:N],:)=0; W(:,[1:a N-a+2:N])=0;  % window design
  Xw=Xs.*W;
% INVERSE DFT
  Xws=ifftshift(Xw); Z=ifft2(Xws+mean(mean(Xws))); Z=real(Z);
  Z=(Z-min(min(Z)))/(max(max(Z))-min(min(Z)));   % normalization
```

FIGURE 2.12. Matlab code for 2D signal denoising by frequency windowing.

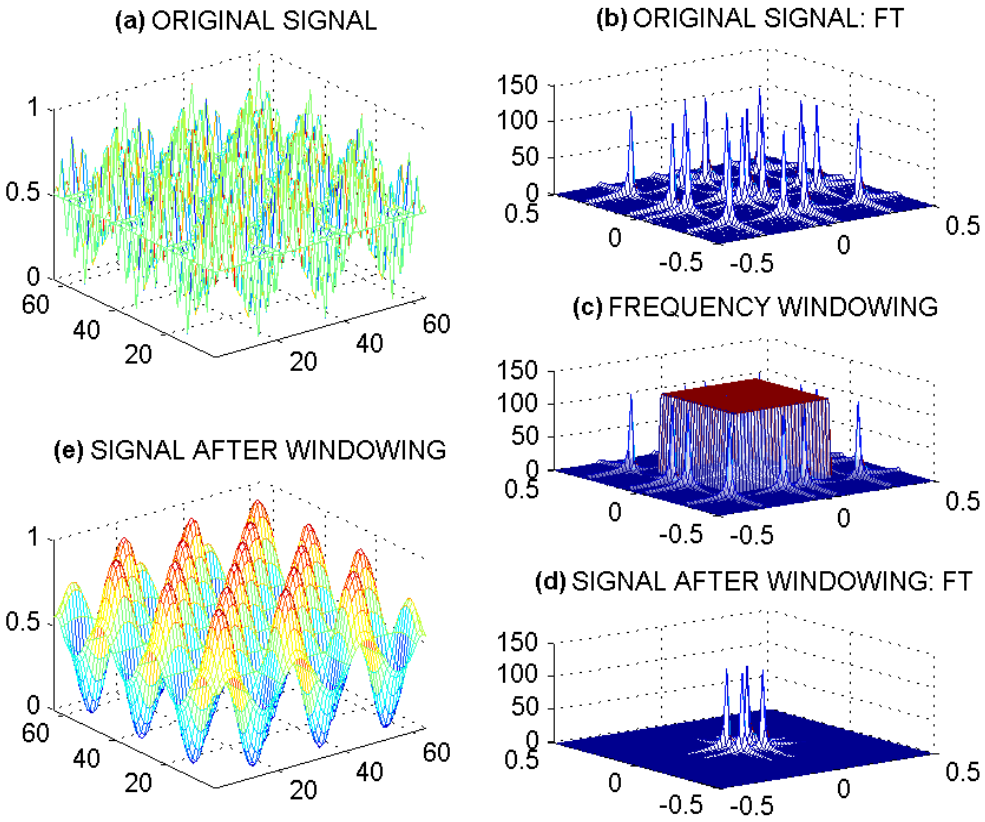


FIGURE 2.13. 2D signal denoising by frequency windowing.

2.3 Time-Scale Signal Analysis

2.3.1 Multiresolutional Signal Analysis

Wavelets and Multiresolutional Analysis (MRA)

In the 19th century, Fourier invented signal frequency analysis. In 1909, Alfred Haar laid the foundations of *wavelet analysis*, although he did not use this term. The word *wavelet* means "a small wave", i.e. an oscillatory function of an effectively limited length. Wavelets have *compact support*, which means that they are zero outside of a bounded domain called a compact set [15, 5].

Wavelets are either real or complex functions, and are usually irregular and asymmetric. Only very few wavelets have an analytical expression. Mostly, they are expressed as piecewise polynomials, such as the Haar wavelet, the Morlet wavelet, the Mexican Hat wavelet or the Daubeschies wavelets [5].

In contrast to the FT, wavelets are utilised for non-stationary signals with transitory phenomena. The frequency response of these signals varies in time, i.e. these signals have quick local variations. By the FT analysis, a signal is decomposed into sine waves of various frequencies. Whereas, in the wavelet analysis, a signal is decomposed into scaled and shifted versions of the *mother wavelet* [5].

The convolution of a signal with a scalable modulated window produces the *Multiresolutional Analysis* (MRA) of the signal. In other words, MRA is a time-scale representation of the analysed signal for every resolution.

A signal is analysed at different frequencies with different resolutions. At high frequencies, we obtain a good time resolution and a low frequency resolution. On the contrary, at low frequencies, we obtain a good frequency resolution, but a poor time resolution. This trade-off is mostly convenient in practise because the majority of signals we work with contain low frequency components of long duration (and so a good time resolution is not necessary), and high frequency components of short duration (a good time resolution is provided). These high frequency components, so-called *details*, represent the main information contained in the signal.

Continuous Wavelet Transform (CWT)

A real *mother wavelet* function $\psi(t)$ is a real square integrable function, i.e. $\psi(t) \in \mathbf{L^2(R)}$ with a zero average and a norm $\|\psi\| = 1$, centred in the neighbourhood of $t = 0$. The term *mother* reflects the fact, that other wavelets belonging to a given wavelet family are derived from ψ by scaling and translating [6] given by

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\,\psi\left(\frac{t-u}{s}\right) \tag{2.28}$$

In Eq. (2.28), the *translation* $u \in \mathbf{R}$ represents the location of $\psi(t)$ of a certain *scale* $s \in \mathbf{R}^+ - \{\mathbf{0}\}$, which is shifted along the signal in time (or space). For each shift, we calculate one wavelet coefficient as a measure of similarity between the wavelet and the corresponding segments of the signal. If the signal energy and the wavelet energy are normalised to 1, the wavelet coefficients become the *correlation coefficients*. When we reach the end of the signal, we change s, return to the beginning of the signal and repeat the computation procedure all over again [5].

The continuous wavelet transform (CWT) of a signal $x \in \mathbf{L^2(R)}$ for a translation u and scale

s is given by [6]

$$W_x^{\psi}(s,u) = \langle x, \psi_{u,s} \rangle = \int_{-\infty}^{+\infty} x(t) \frac{1}{\sqrt{s}} \psi^* \left(\frac{t-u}{s} \right) dt \tag{2.29}$$

where the angle brackets denote the *inner product*. By definition, the L^2*-inner product* of two complex functions f and g on a measure space X with respect to the measure μ is given by [15]

$$\langle f, g \rangle_{L^2} = \int_X f \, g^* \, d\mu \tag{2.30}$$

where the star symbol denotes the complex conjugate. Eq. (2.29) can be also interpreted as a convolution of a signal and a dilated band-pass filter, since the wavelet function ψ has a band-pass like spectrum $\hat{\psi}(\omega)$ as described below when discussing wavelet properties.

The *scale* s is inversely proportional to radian frequency ω. As a result, low frequencies correspond to high scales. When using high scales, ψ is *dilated* or stretched out. By wavelet analysis at high scales, we extract *global information* from a signal. On the contrary, high frequencies correspond to low scales. ψ is contracted or compressed. And by wavelet analysis at low scales, we extract *detail information* from a signal. Fig. 2.14 displays scaling of the wavelet having an analytical expression designed by Prof Newland. Dilation in time compresses the frequency spectrum $\psi\left(\frac{t}{s}\right) \leftrightarrow |s| \, \hat{\psi}(s\omega)$ [6].

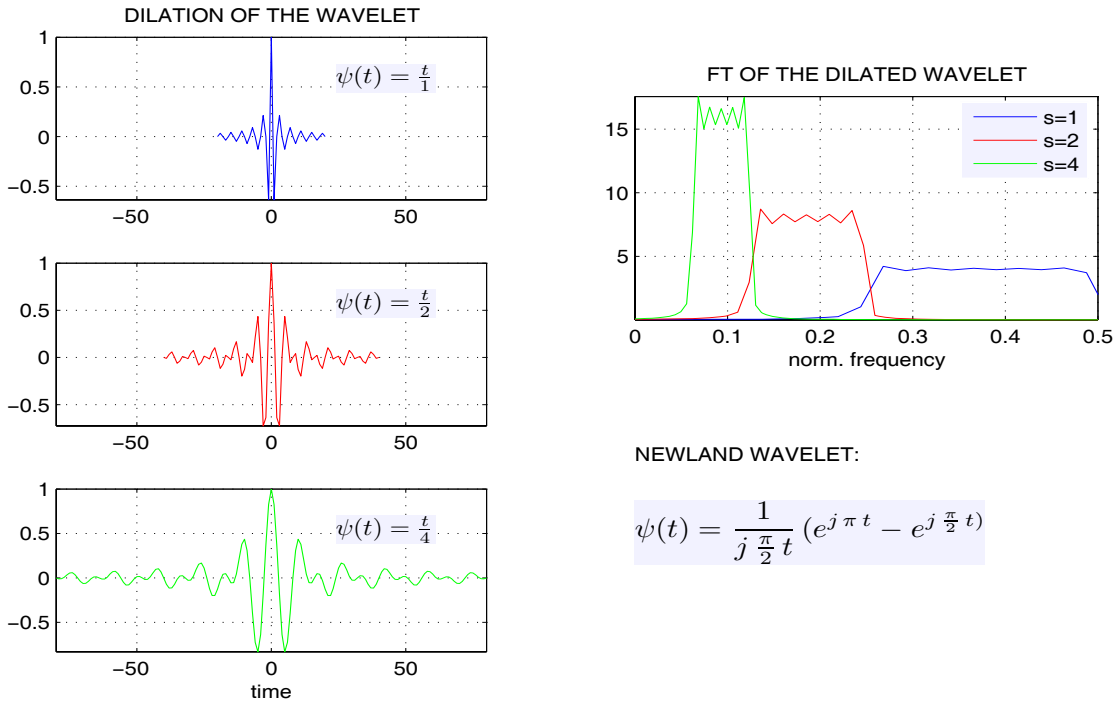


FIGURE 2.14. Time dilation of Newland wavelet and compression of its frequency spectrum.

Signals are usually band-limited, which is equivalent to having finite energy, and therefore we need to use just a constrained interval of scales. We start the CWT computation with $s = 1$, i.e. with the most dilated ψ, and continue by increasing the scale, i.e. compressing ψ. This way, we start the analysis at high frequencies and proceed towards lower frequencies.

The CWT coefficients are a measure of similarity (correlation) in the frequency content between

a wavelet and a signal. The coefficients have large values at a certain location, if a signal contains a current scale at this location.

The CWT is reversible if the admissibility condition is fulfilled. The basis functions do not even have to be orthonormal. For a signal x of finite energy, the *inverse CWT* [2] runs

$$x(t) = \frac{1}{C_\psi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} W_x^\psi(s,u) \frac{1}{\sqrt{s}}\, \psi\left(\frac{t-u}{s}\right) \frac{ds\, du}{s^2} \tag{2.31}$$

Where C_ψ is a constant depending on the kind of the wavelet function ψ [2]

$$C_\psi = 2\pi \int_{-\infty}^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|}\, d\omega \tag{2.32}$$

By saying that x has finite energy, we mean that its L^2-form is finite, see Eq. (2.33). Natural signals usually satisfy this condition [13].

$$\int_0^\infty |\, x(t)|^2\, dt < \infty \tag{2.33}$$

In the CWT, u and s are continuous. In fact, because of a computers use, u and s are sampled with a small enough step size, that they can still be considered continuous.

Wavelet Properties

The first property we shall discuss here is the *compact support* of the wavelet function. We say that a wavelet has compact support, if the wavelet if zero outside a closed interval. This characteristic corresponds to the impulse response of the *FIR filter*. However, wavelets need not have compact support. They can also correspond to the *IIR filters* [2].

Other desired properties are *orthogonality* and *orthonormality* of wavelets. Two functions $\psi_k(t)$ and $\psi_l(t)$ are *orthogonal* to each other over the interval [a,b] if their *inner product* is zero.

$$\langle \psi_k(t), \psi_l(t) \rangle = \int_a^b \psi_k(t)\, \psi_l^*(t)\, dt = 0 \tag{2.34}$$

A set of functions $\{\psi_k\}$, $k = 1, 2, 3, ...$ is *orthonormal* over the interval [a,b], if they are normalised and orthogonal to each other, as expressed by the following expression.

$$\langle \psi_k(t), \psi_l(t) \rangle = \int_a^b \psi_k(t)\, \psi_l^*(t)\, dt = \delta_{kl} \tag{2.35}$$

where δ_{kl} denotes Kronecker delta function defined as

$$\delta_{kl} = \begin{cases} 1 & \text{for } k = l \\ 0 & \text{for } k \neq l \end{cases} \tag{2.36}$$

For orthonormal basis, the *wavelet transform* (WT) coefficients are given by a linear combination of the basis functions. In some applications, orthonormal bases are not available, and therefore we employ *biorthogonal* bases, which are two bases orthogonal to each other, but each do not form an orthogonal set. Sometimes, even biorthogonal bases are not available, so we use *frames* [8].

The *admissibility condition* stated in Eq. (2.37) is one of the basic assumptions of no information loss during signal analysis and subsequent synthesis [13].

$$\int \frac{|\,\hat{\psi}(\omega)|^2}{|\,\omega|}\,d\omega < +\infty \tag{2.37}$$

where $\hat{\psi}(\omega)$ is the FT of $\psi(t)$. As an implication, $\hat{\psi}(\omega)$ vanishes at $\omega = 0$, which shapes the wavelet spectrum to a band-pass [13]:

$$|\,\hat{\psi}(\omega)|^2|_{\,\omega=0} = 0 \tag{2.38}$$

As a result of Eq. (2.38) in the time domain, the average value of $\psi(t)$ equals zero, which means that the wavelet is oscillatory [13]:

$$\int \psi(t)\,dt = 0 \tag{2.39}$$

The *regularity condition* is linked to the *smoothness* of the wavelet and the *decay* of $|\hat{\psi}(\omega)|$ for large ω, i.e. fine scales, which is desired to be fast [6, 13]. We shall describe regularity using the concept of *vanishing moments*. A function ψ has p vanishing moments if [6]

$$\int_{-\infty}^{\infty} t^k\,\psi(t)\,dt = 0 \quad \text{for } 0 \le k < p \tag{2.40}$$

The k-th vanishing moment M_k is given by [6]

$$M_k = \int_{-\infty}^{\infty} t^k\,\psi(t)\,dt \quad \text{for } k \in \mathbf{Z}^* \tag{2.41}$$

where $\mathbf{Z}^*$ stands for the nonnegative integers. We can express the wavelet transform of a signal x using the Taylor series until order n at a time $t = 0$ and for $u = 0$ [13]

$$W_x^{\psi}(0,s) = \frac{1}{\sqrt{s}}\left[\sum_{k=0}^{n} x^{(k)}(0)\int \frac{t^k}{k!}\,\psi\left(\frac{t}{s}\right)dt + O(n+1)\right] \tag{2.42}$$

where $x^{(k)}$ stands for the k-th derivative of x and $O(n+1)$ denotes the remainder of the expansion. We can rewrite this equation using the moments M_k [13]

$$W_x^{\psi}(0,s) = \frac{1}{\sqrt{s}}\left[\sum_{k=0}^{n} x^{(k)}(0)\frac{M_k}{k!}\,s^{k+1} + O(s^{n+2})\right] \tag{2.43}$$

As implied by the admissibility condition, $M_0 = 0$. If all the other moments vanish to zero then the wavelet coefficients W_x^{ψ} decay as quick as s^{n+2} for x n-times differentiable at $t = 0$ [13]. To sum this concept up, if the signal x is regular and the wavelet ψ has enough vanishing moments then the wavelet coefficients are small at fine scales [6].

The number of vanishing moments p and the support size of orthogonal wavelets are actually independent except the constriction saying that the support is at least equal $2p-1$. The minimal support size $2p-1$ applies to Daubeschies wavelets [6].

Wavelets themselves are often used to study regularity of signals [5].

Scaling Function

If we have wavelet coefficients only for scales $s < s_0$, for signal reconstruction, we need a complement to these coefficients for $s > s_0$, which is provided by the *scaling function* ϕ. The function ϕ was introduced by Stephane Mallat in 1989. The modulus of its Fourier transform $\hat{\phi}(\omega)$ is given by [6]

$$|\,\hat{\phi}(\omega)|^2 = \int_1^{+\infty} |\,\hat{\psi}(s\omega)|^2 \,\frac{ds}{s} = \int_\omega^{+\infty} \frac{|\,\hat{\psi}(\xi)|^2}{\xi}\, d\xi \tag{2.44}$$

where $\xi = s\omega$. Similarly to the wavelet function ψ, we can state the admissibility condition for the scaling function ϕ [6]

$$\int \phi(t) = 1 \tag{2.45}$$

As a consequence, the 0^{th} moment of ϕ cannot obviously vanish to zero. The scaling function represents a *low-pass filter* or an averaging filter and all the dilated wavelets form *band-pass filters*, hence all these functions compose a *filter bank*.

The scaling function is associated with the wavelet function, but not for every wavelet [5].

Discrete Wavelet Transform (DWT)

The CWT provides us with lots of redundant information, which is costly in terms of computation time, but on the other hand, all information is very clearly visible. The *discrete wavelet transform* (DWT) requires less space utilising the *space-saving coding* based on the fact that wavelet families are orthogonal or biorthogonal bases, and thus do not produce redundant analysis. In addition, to save some more memory space, we can neglect the coefficients of a very low value without a significant information loss.

The DWT introduced by Eq. (2.46) corresponds to the CWT sampled usually on a *dyadic* grid, which means powers of two, so as $s = 2^j$ and $u = k\,2^j$, where t is continuous time and $j, k \in \mathbf{Z}$ [13].

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}}\,\psi\left(\frac{t - k\,2^j}{2^j}\right) \tag{2.46}$$

In contrast to WT, the FT basis functions $e^{i\omega t}$ are perfectly localised in frequency but they extend over all time. Wavelets are not at a single frequency, or even a finite range, but they are limited to finite time. As we rescale, the frequency goes up by 2^j and the time interval goes down by 2^j. This suggests that the product of the frequency and time interval is a stable quantity [2], see the Heisenberg's uncertainty principle in Subsec. 2.2.4.

In practise, the DWT is obtained by passing a signal successively through low-pass and high-pass filters. This computation algorithm is called the *subband coding*. Passing a signal x through a filter corresponds to the convolution of x with the impulse response of the filter l_d [8]

$$x[n] * l_d[n] = \sum_{k=-\infty}^{\infty} l_d[k]\, x[n-k] \tag{2.47}$$

where n denotes discrete time sampled from t. By the filtering process, the resolution is altered. The scale is changed by subsampling, i.e. either *upsampling* or *downsampling*. Upsampling by 2 means adding a new sample (a zero or an interpolated value) between every 2 samples. Downsampling by 2 is implemented by excluding every other sample.

When a signal sampled at the Nyquist's rate, see Subsec. 2.2.1, it is passed through a half-band low-pass filter and the highest frequency component f_{max} is halved $f_{max} = \pi \rightarrow \frac{\pi}{2}$. Conse-

quently, in keeping with the Nyquist's rule, the minimal sampling frequency f_s of the signal can decrease by half $f_s = 2\pi \to \pi$. We achieve to halve f_s and simultaneously to double the scale by downsampling by 2. Through the filtering, we loose half of the samples and also the higher half of the frequency spectrum of the signal. As a result, the time resolution is halved and the frequency resolution is doubled. This procedure can be described by the following equation, where A_1 is the low-pass filter output [8] equivalently the approximation coefficients of the decomposition level 1.

$$A_1[n] = \sum_{k=-\infty}^{\infty} l_d[k]\, x[2n-k] \tag{2.48}$$

The *subband coding* algorithm, designed by S. Mallat in 1988, is shown in Fig. 2.15. The outputs of the high-pass and the complementary low-pass filter are called *approximations* and *details*, respectively. In order to work efficiently, the length of the input x should be a power of 2 (i.e. 2^j) or at least a multiple of a power of 2 (i.e. $k\,2^j$). The length of x equals the number of the DWT coefficients. However, the filters are not ideally narrow-band, and thus due to the convolution, we obtain some more coefficients before the beginning and beyond the end of the signal. These coefficients need to be neglected.

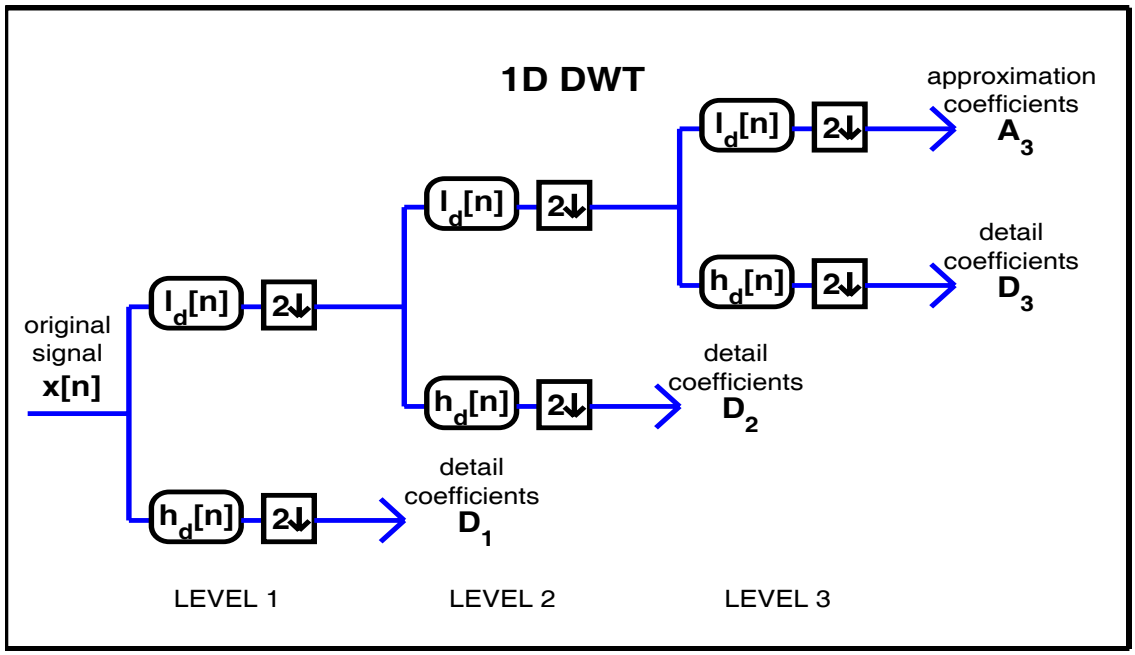


FIGURE 2.15. 1D DWT 3-level decomposition scheme.

Fig. 2.15 displays the *wavelet decomposition tree*. The decomposition is indefinite in theory, however, for real computations, it ends when only one detail coefficient is left. The number of decomposition levels depends on the kind of the signal or on a chosen criterion, such as *entropy* [5].

The impulse response of the low-pass filter l_d forms the scaling function [6] and produces the approximations A. The high-pass filter h_d represents the wavelet function and generates the details D. The impulse responses of both filters are dependent on each other as stated in the following equation [8].

$$h_d[L-1-k] = (-1)^k\, l_d[k] \tag{2.49}$$

where L is the filter length. The two following equations describe the DWT analysis, which means filtering the signal and downsampling the result by 2 [8].

$$D_1[n] = \sum_{k=-\infty}^{\infty} h_d[k]\, x[2n-k] \tag{2.50}$$

$$A_1[n] = \sum_{k=-\infty}^{\infty} l_d[k]\, x[2n-k] \tag{2.51}$$

Half-band filters form orthonormal bases, and therefore make the reconstruction easy. The synthesis consists of upsampling by 2 and filtering [8]:

$$x[n] = \sum_{k=-\infty}^{\infty} \left(D_1[k]\, h_r[2k-n] + A_1[k]\, l_r[2k-n]\right) \tag{2.52}$$

The *reconstruction* filters l_r and h_r and identical with the decomposition filters l_d and h_d, respectively, except the reverse time course. These two pairs of filters are called the *Quadrature Mirror Filters* (QMF). The decomposition QMF determine the wavelet's shape by upsampling the high-pass filter coefficients vector and convolving the result with the original filter. This is repeated in several iterations. The scaling function is derived analogously from the low-pass filter [5].

The filter choice is of a great importance. Filters are never ideally half-band, which fact makes synthesis difficult. We need use non-ideal filters forming the QMF to cancel out the aliasing, which originates from the downsampling during the decomposition. These filters attain to produce perfect signal reconstruction from the DWT coefficients provided that the signal is of finite energy, and that the wavelet satisfies the admissibility condition as usual wavelets do [5].

Wavelet Packets

The *wavelet packets* analysis are similar to the wavelet analysis, except that details coefficients vectors are also decomposed as shown in Fig. 2.16 [5]. The number of decomposition levels may be determined for example by an entropy-based criterion.

Wavelet Families

The two main wavelet functions, we use in this thesis are firstly the *Haar* wavelet and secondly the *Daubeschies* wavelet of order 2 abbreviated as *db*2. The *Haar* wavelet is the oldest and the simplest wavelet function. It is compactly supported, orthogonal and discontinuous, which makes an exception from other wavelet functions. The Haar wavelet is sometimes denoted as *db1* wavelet. In this work, it is used for edges detection in images. The Haar wavelet ψ and its scaling function ϕ shown in Fig.2.17 have an explicit expression given as [5]

$$\psi(t) = \begin{cases} 1 & \text{for } t \in \langle 0, 0.5) \\ -1 & \text{for } t \in \langle 0.5, 1\rangle \\ 0 & \text{otherwise} \end{cases} \tag{2.53}$$

$$\phi(t) = \begin{cases} 1 & \text{for } t \in \langle 0, 1) \\ 0 & \text{otherwise} \end{cases} \tag{2.54}$$

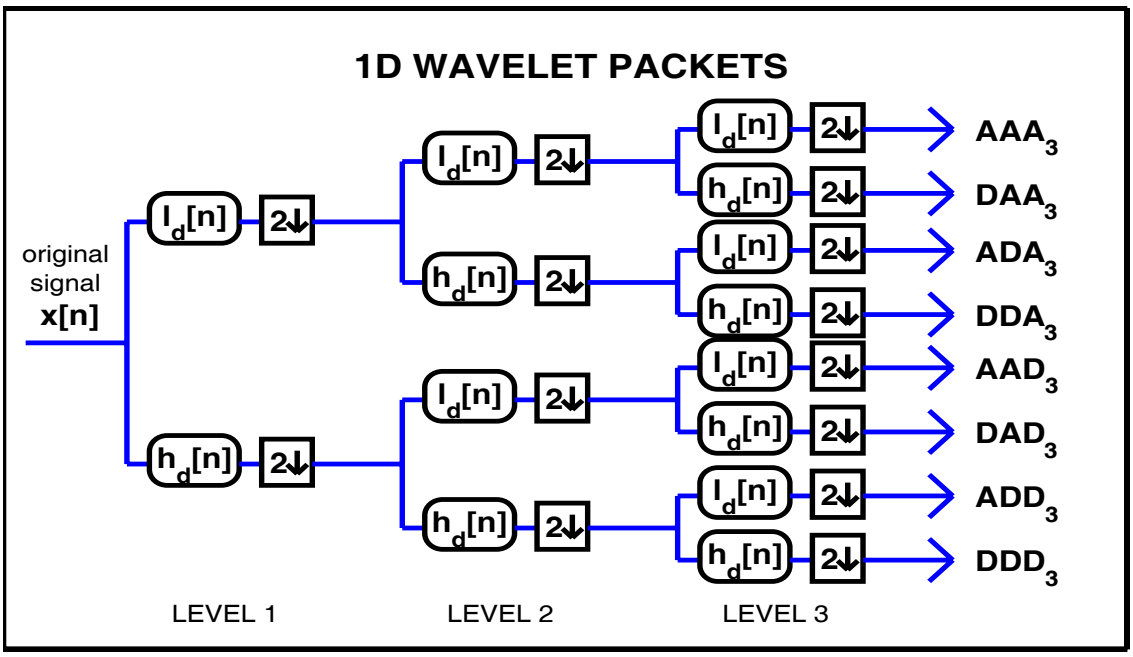


FIGURE 2.16. 1D wavelet packets 3-level decomposition scheme.

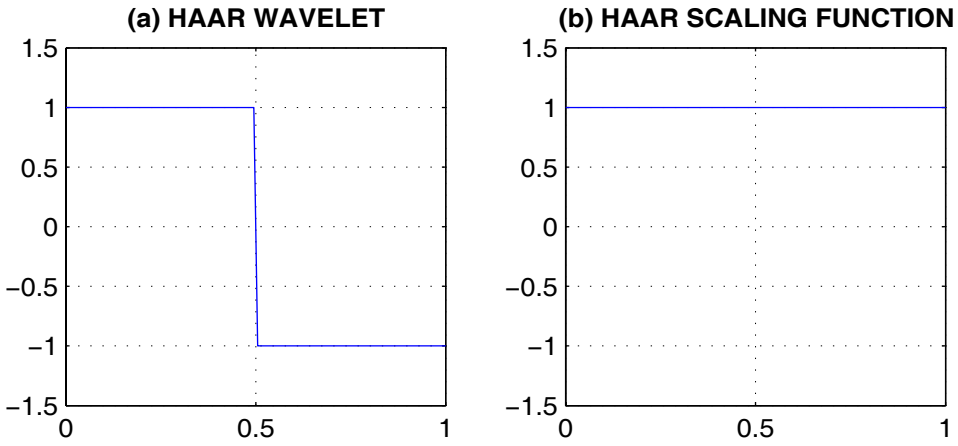


FIGURE 2.17. Haar wavelet and its scaling function.

Daubeschies wavelets (DbN) are compactly supported orthogonal wavelets, where N denotes the number of vanishing moments. *db1*, i.e. the Haar wavelet, is discontinuous and with increasing N, the regularity increases. Some of dbN are less some are very far from symmetry. Except the Haar wavelet, dbN have no explicit expression. They are given by $2N$ non-zero coefficients. In this work, *db2* wavelet is used for image denoising. Fig. 2.18 showing *db*2, its scaling function and the corresponding frequency spectra is created according to Matlab help for a function *orthfilt* [5], which computes the decomposition and reconstruction filters associated with the scaling filter corresponding to a wavelet.

Wavelets Applications

In signal processing, wavelets are used for many purposes [5]. Such as denoising, detecting trends, breakdown points, discontinuities in higher derivatives and self-similarity in signals.

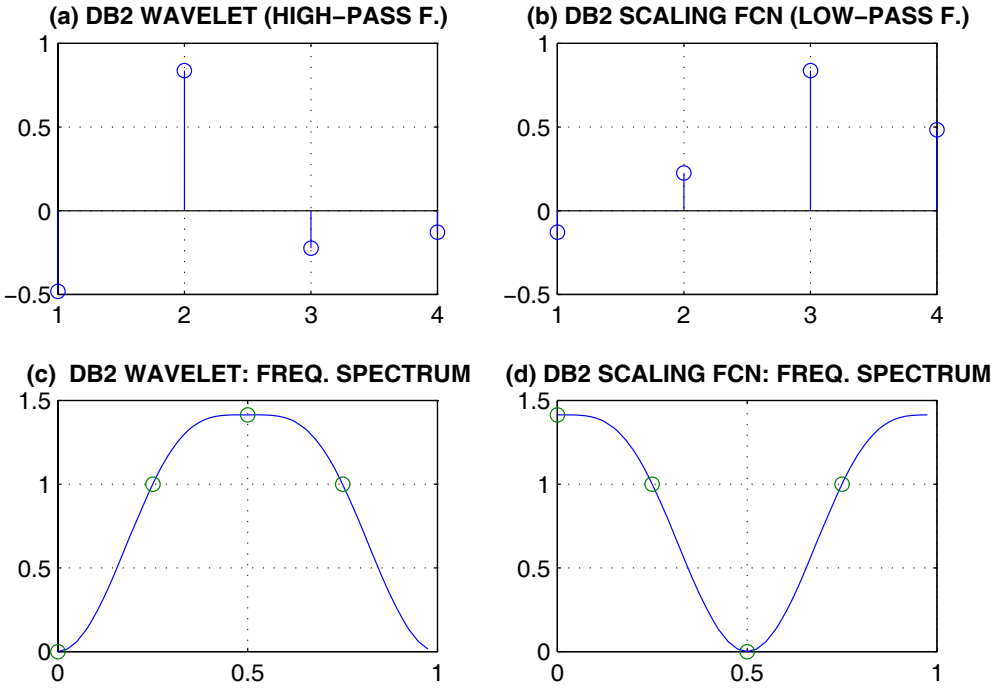


FIGURE 2.18. Daubeschies wavelet of order 2, its scaling function and frequency spectra.

Fig. 2.19 and the enclosed Matlab code in Fig.2.20 demonstrate wavelet use on impulse detection, i.e. detection of a discontinuity in frequency.

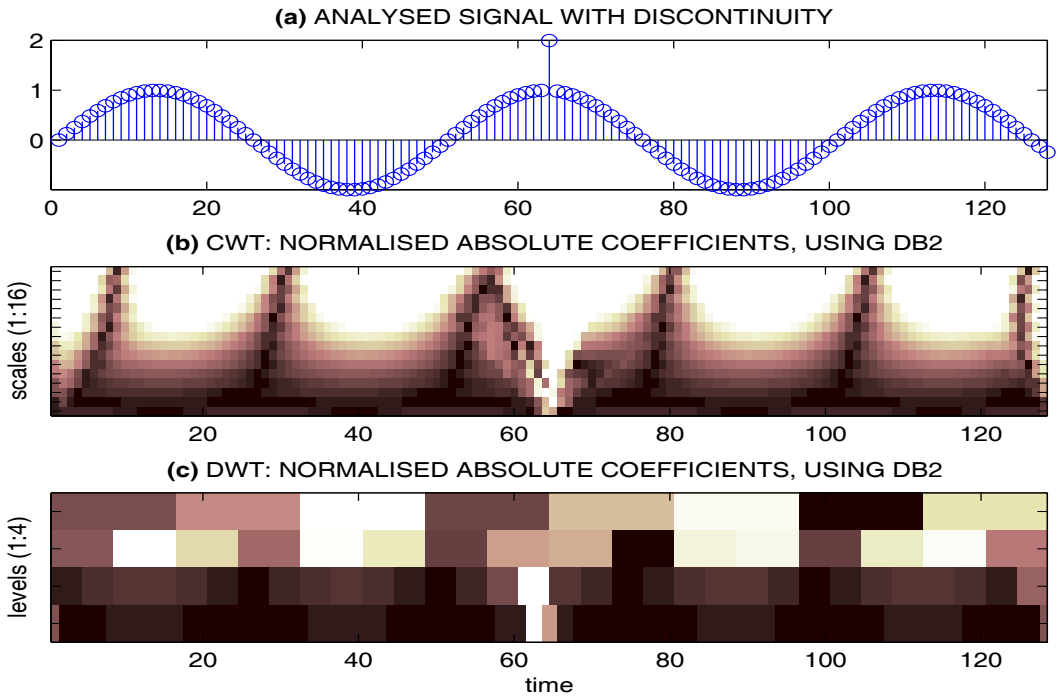


FIGURE 2.19. Discontinuity detection using CWT and DWT.

In image processing, wavelets are used for instance for edges detection, watermarking, texture

```
% CWT & DWT: SIGNAL ANALYSIS - DISCONTINUITY DETECTION
% ANALYSED SIGNAL WITH A DISCONTINUITY (IMPULSE)
  N=128; n=[0:N-1]; x=sin(2*pi*0.02*n); x(64)=x(64)+1;
  figure(1),subplot(311), stem(x)
% CWT COMPUTATION AND COEFFICIENTS PLOT
  subplot(312), c=cwt(x,1:16,'db2','plot');
  set(gca,'YTickLabel',[])  % set y-axis tick label, gca=get current axis
% DWT COMPUTATION
  [c,l]=wavedec(x,4,'db2');  % Levels 1 to 4 correspond to scales 2, 4, 8, 16.
% DWT COEFFICIENTS EXPANSION FOR PLOT
  cfd=zeros(4,N);
  for k=1:4
     d=detcoef(c,l,k);  % extract the detail coefficients at level k from
            % the wavelet decomposition structure [c,l], d is a vector
     d=d(ones(1,2^k),:);  % d -> matrix (the same values in each column)
     cfd(k,:)=wkeep(d(:)',N);  % extract the vector cfd (of length N) from
            % the central part of the vector d as its ; cfd is a row vector
  end
% DWT COEFFICIENTS PLOT
  subplot(313), colormap(pink(64))
  img=image(flipud(wcodemat(cfd,64,'row')));  % flipud = Flip matrices up-down
            % wcodemat = extended pseudocolor matrix scaling (row-wise)
  set(get(img,'parent'),'YtickLabel',[]); % specify y-axis tick labels (empty)
```

FIGURE 2.20. Matlab code for discontinuity detection using CWT and DWT.

detection (see Chap. 5), compression, denoising, and coding of interesting features for subsequent classification [5]. Image and signal *denoising* by *thresholding* DWT coefficients is discussed in the following chapter.

2.3.2 Multiresolutional Image Analysis

To compute the two-dimensional DWT of an image, we decompose the approximations at level j to obtain four matrixes of coefficients at level $j+1$. These four matrixes for single level decomposition using $db2$ are displayed in Fig. 2.21 denoted as the *approximations* A_1 and the *horizontal* H_1, *vertical* V_1 and *diagonal details* D_1 of level 1.

As shown in the scheme in Fig. 2.22, first, we convolve the rows of the image, or generally the matrix of the approximations at level j, with a low-pass and a high-pass decomposition filter $l_d[n]$ and $h_d[n]$, respectively. Then we downsample both resulting matrixes by 2 keeping every even column. Second, we filter each of the matrixes by their columns using the previously mentioned filters. Then we downsample all four resulting matrixes by 2 keeping every even row to obtain four matrixes of one-level decomposition coefficients, or generally four matrixes of $(j+1)$-level coefficients [5]. We can also reconstruct the image by using these coefficients matrixes, upsampling by 2 and the reconstruction filters $l_r[n]$ and $h_r[n]$. .

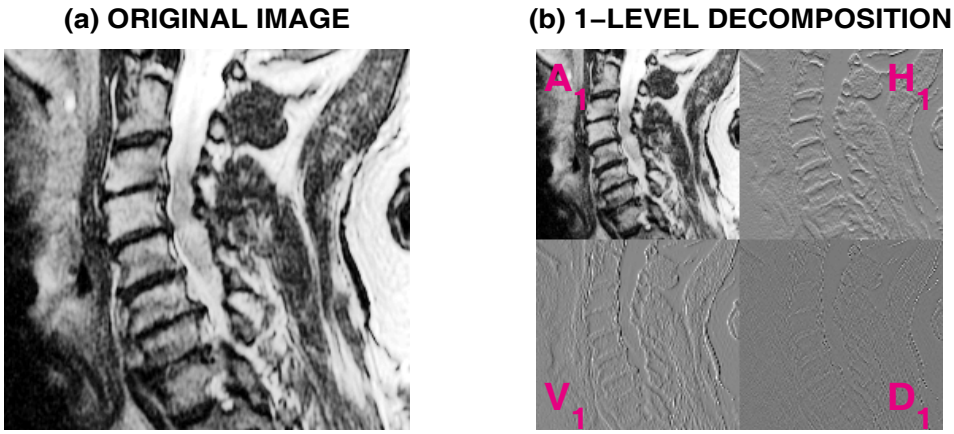


FIGURE 2.21. 2D DWT 1-level decomposition.

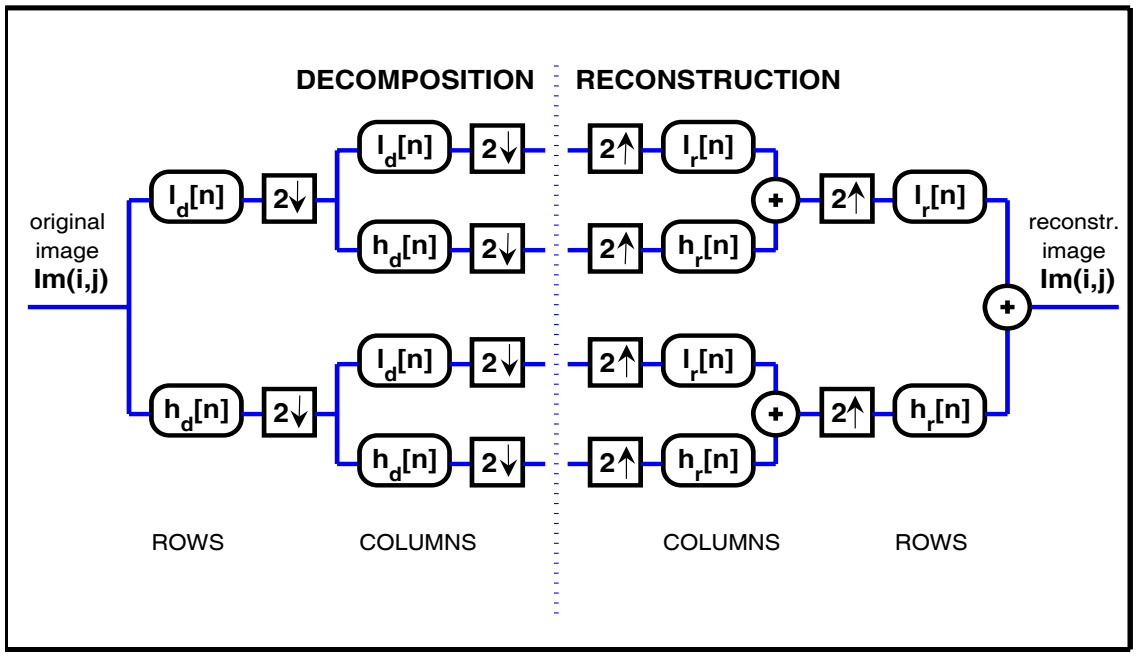


FIGURE 2.22. 2D DWT 1-level decomposition and reconstruction scheme.

3

Wavelet Transform in Signal and Image Processing

3.1 Signal Decomposition, Thresholding, and Reconstruction

Signal *denoising* using the DWT consists of the three successive procedures named in the title of this section, i.e. signal *decomposition*, DWT coefficients *thresholding*, and signal *reconstruction*. Firstly, we carry out the wavelet analysis of a noisy signal up to a chosen level N. Secondly, we perform thresholding of the detail coefficients at levels from 1 to N. Lastly, we synthesize the signal using the altered detail coefficients of levels from 1 to N and approximation coefficients of level N [5].

First of all, we introduce the model of a noisy 1D signal $x_n(n)$ and an image $I_n(i,j)$ [5]

$$x_n(n) = x(n) + s\,e(n) \tag{3.1}$$

$$I_n(i,j) = I(i,j) + s\,e(i,j) \tag{3.2}$$

where n, i and j are equally spaced time or space coefficients, x and I are clean signals, which are to be recovered from x_n and I_n, respectively, by removing the noise e. However, it is generally impossible to remove all the noise without corrupting the clean signal. In the simplest case, e is supposed to be a *Gaussian white noise* of amplitudes from 0 to 1, and s is a noise level assumed to equal 1.

As for *thresholding*, we can settle either a *global* threshold of a constant value for all levels or a *level-dependent* threshold vector of length N. According to *D. Donoho's method*, the threshold estimate δ for denoising with an orthonormal basis is given by [2]

$$\delta = \sigma\,\sqrt{2\log L} \tag{3.3}$$

where the noise is Gaussian with standard deviation σ of the DWT coefficients and L is the number of samples or pixels of the processed signal or image. This estimation concept is used by Matlab.

From another point of view, thresholds can be either *soft* or *hard* as shown in Fig. 3.1 and Fiq. 3.2 and given by the following expressions [2]

$$y_{hard}(n) = \begin{cases} x_n(n) & \text{for } |\,x_n(n)| > \delta \\ 0 & \text{for } |\,x_n(n)| \le \delta \end{cases} \tag{3.4}$$

$$y_{soft}(n) = \begin{cases} sign(x_n(n))\,(|\,x_n(n)| - \delta) & \text{for } |x_n(n)| > \delta \\ 0 & \text{for } |\,x_n(n)| \le \delta \end{cases} \tag{3.5}$$

Hard thresholding zeroes out all the signal values smaller than δ. Soft thresholding does the same thing, and apart form that, subtracts δ from the values larger than δ. In contrast to hard thresholding, soft thresholding causes no discontinuities in the resulting signal. In Matlab, by default, soft thresholding is used for denoising and hard thresholding for compression [5].

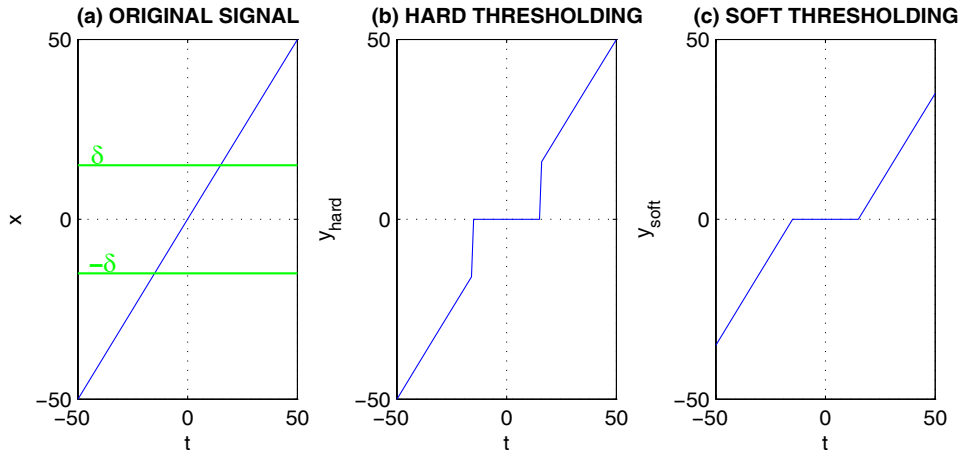


FIGURE 3.1. Soft and hard thresholding for linear signal.

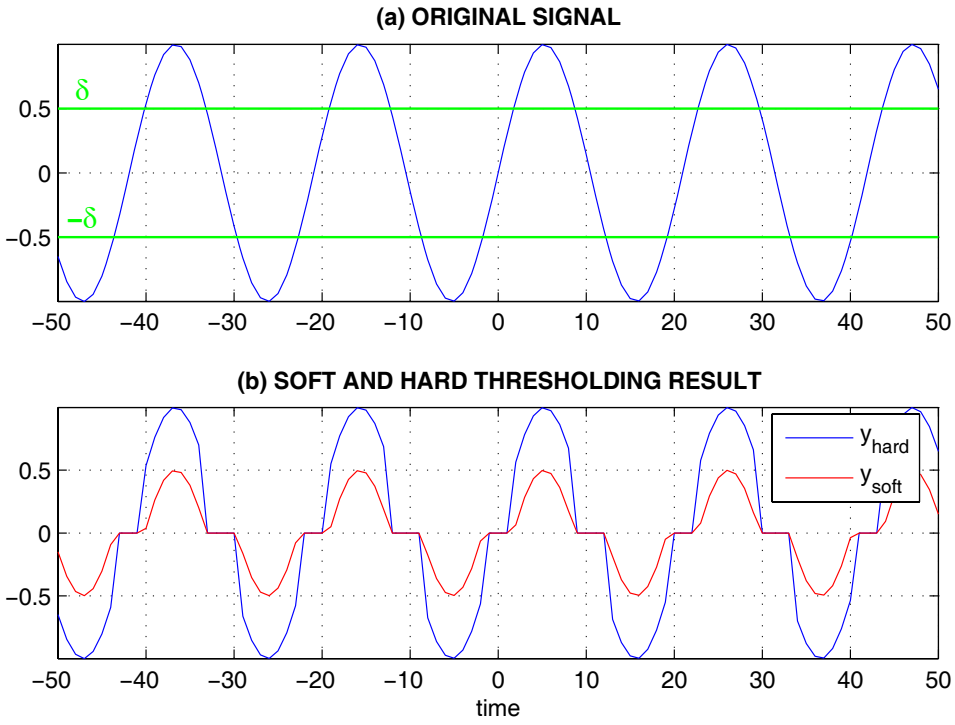


FIGURE 3.2. Soft and hard thresholding for sinusoidal signal.

An example of *Electrocardiogram* (ECG) signal denoising is displayed in Fig. 3.3. Removing of artificially added white noise is carried out by soft global thresholding of the DWT coefficients either up to level 2 or 4. In the former case, the main peaks' magnitudes of the denoised signal nicely correspond to those of the original one. However, the clean signal still contains a big amount of noise. In the latter case, the main peaks' magnitudes of the denoised signal are depressed. The clean signal contains less noise, but on the other hand, it lacks some of the detail information. The Matlab code is enclosed, see Fig. 3.4. Fig. 3.5 shows a plot of the DWT coefficients of levels 1 to 4 and the global threshold δ. The magenta vertical lines divide five sets of the DWT coefficients, i.e. from left to right, approximations at level 4 A_4, details at level 4 D_4, details at level 3 D_3 etc.

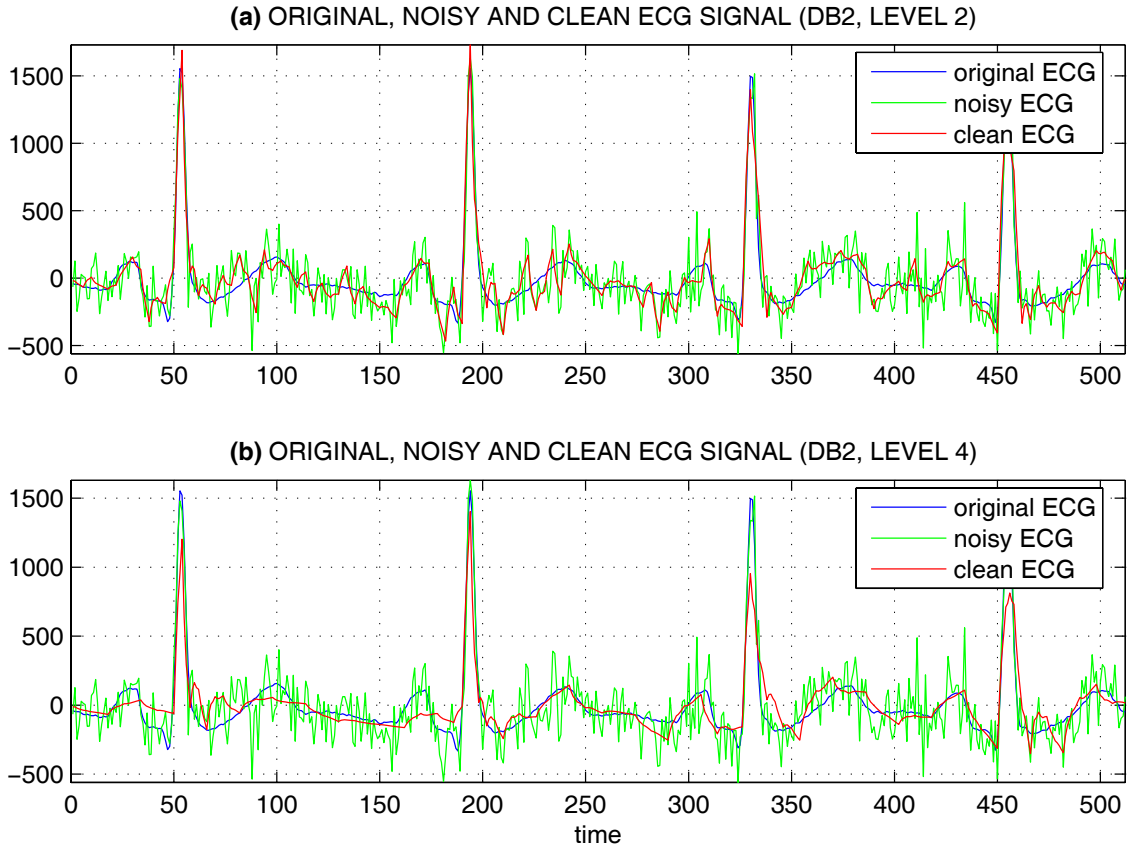


FIGURE 3.3. ECG signal denoising by thresholding DWT coefficients up to levels 2 and 4.

```
% ECG SIGNAL DENOISING BY THRESHOLDING DWT COEFFICIENTS
  ecg=load('ECG01.TXT'); ecg=ecg(100:611);  % load ecg signal, take 512 samples
  L=length(ecg); ecg=detrend(ecg);  % remove a linear trend
  ecgN=ecg; ecgN=ecg+150*randn(L,1);  % add random noise
% DWT USING 'DB2' UP TO LEVELS 2 AND 4 (GLOBAL THRESHOLDING)
  [THR,SORH,KEEPAPP]=ddencmp('den','wv',ecgN)  % find default values
                                    % for denoising -> soft threshold
  [ecgC1,CecgC,LecgC,PERF0,PERFL2]=wdencmp('gbl',ecgN,'db2',2,THR,SORH,KEEPAPP);
  [ecgC2,CecgC,LecgC,PERF0,PERFL2]=wdencmp('gbl',ecgN,'db2',4,THR,SORH,KEEPAPP);
```

FIGURE 3.4. Matlab code for ECG signal denoising by thresholding its DWT coefficients up to levels 2 and 4.

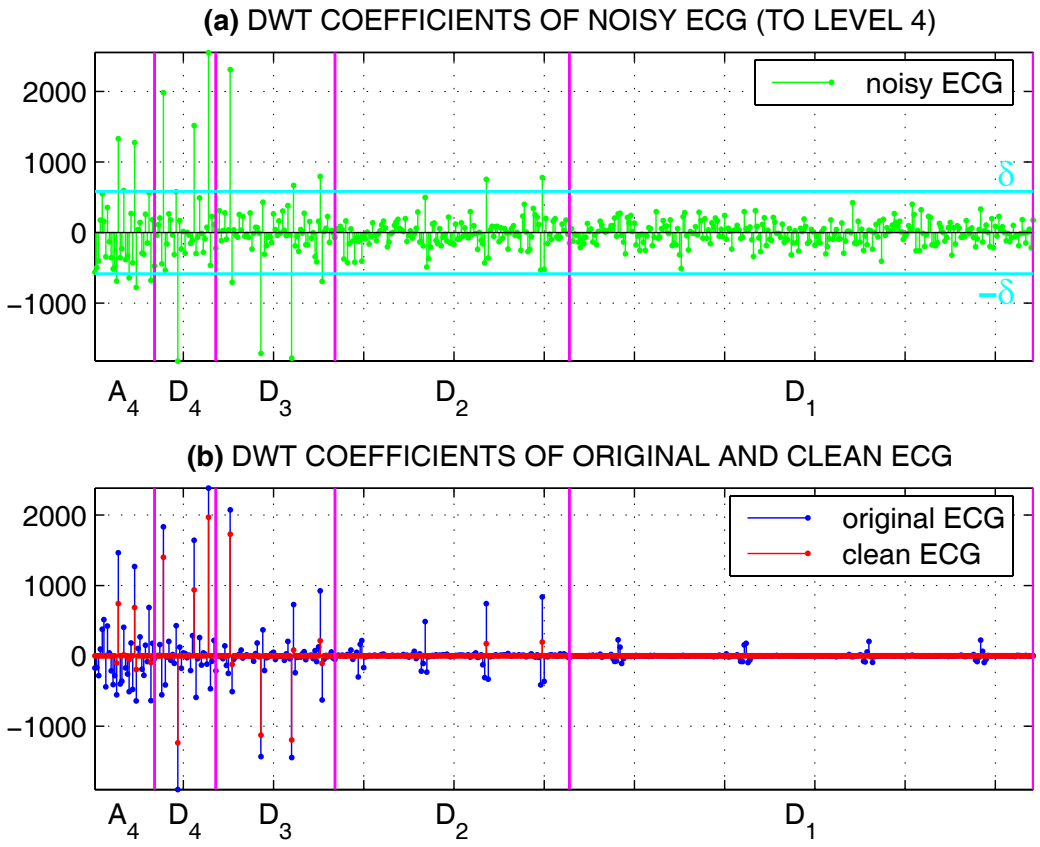


FIGURE 3.5. ECG signal DWT coefficients up to level 4 and their thresholding.

3.2 Image Denoising

Denoising by thresholding DWT coefficients in two-dimensional space is analogical to that in one dimensional space discussed in the previous section. A practical example demonstrates denoising of the MR image of the spine. In this case, wavelet decomposition runs to level 2 using *db*2 wavelet. We employ soft global thresholding of detail coefficients (see Fig. 3.8) and Donoho's threshold level estimate given by Eq. (3.3). Decomposition of the given MR image and the thresholding result are displayed in Fig. 3.6. The main part of the Matlab m-file without potting commands is shown in Fig. 3.7.

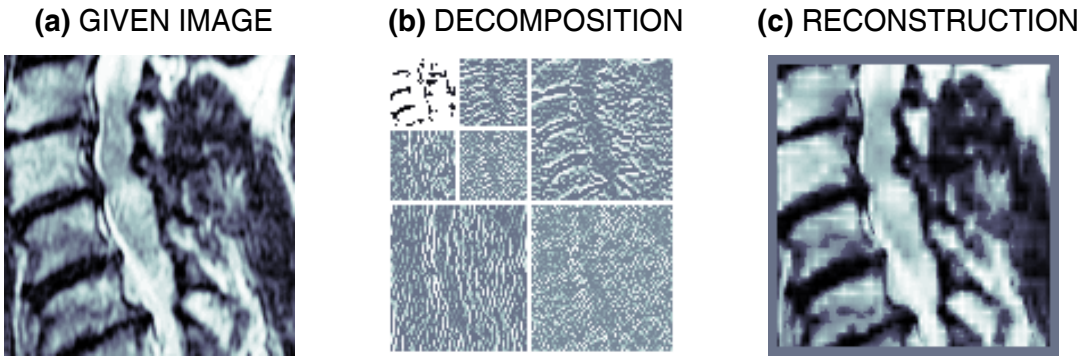


FIGURE 3.6. MR image denoising by thresholding DWT coefficients up to level 2.

```
% MR IMAGE DENOISING BY THRESHOLDING DWT COEFFICIENTS
  load('MRpater004.mat'), A=im2double(A);  % given image definition
  A=A(64:191,64:191);  % image cut
% DWT DECOMPOSITION TO LEVEL 2
  level=2, wavelet='db2'  % decomposition parameters
  [c,s]=wavedec2(A,2,'db2');
  s2=s(2:level+1,1)'; s2=[s2; s2; s2]; s2=[s(1);s2(:)]; ss=s2.^2;
% THRESHOLDING DETAIL COEFFICIENTS
  THR=ddencmp('den','wv',A); % global estimate of thresholds
  k=find(abs(c(s(1,1):length(c)))<=THR); cd(k)=0; % threshold only details
  k=find(abs(c(s(1,1):length(c)))>THR);
  cd(k)=sign(c(k)).*(abs(c(k))-THR); % soft thresholding
% IMAGE RECONSTRUCTION
  Z=waverec2(cd,s,wavelet); [m,n]=size(Z); mZ=mean2(Z);
  Z([1:4,m-3:m],:)=mZ; Z(:,[1:4,n-3:n])=mZ;  % image frame
```

FIGURE 3.7. Matlab code for MR image denoising by thresholding DWT coefficients up to level 2.

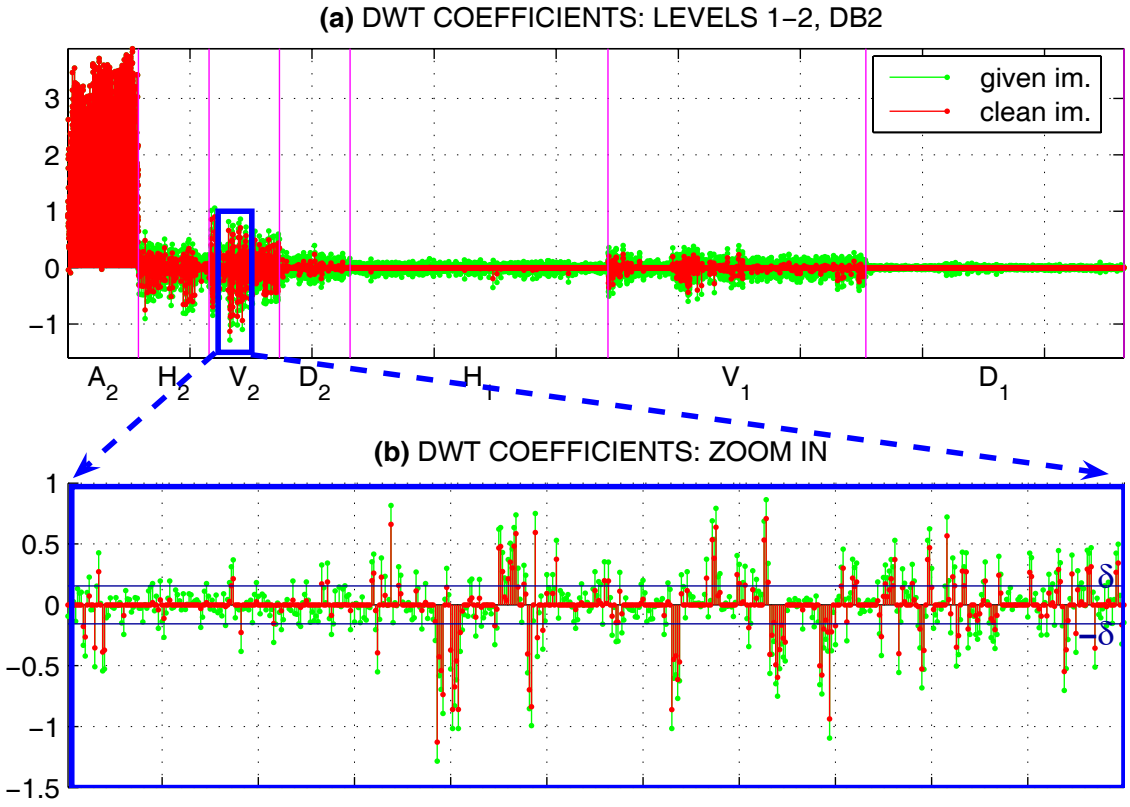


FIGURE 3.8. MR image denoising DWT coefficients up to level 2 and their thresholding..

3.3 Wavelet Transform (WT) in Signal and Image Feature Extraction

Wavelet decomposition coefficients to a selected level can be used for signal or image specification, which is based on assigning certain values called *features* to a signal or image. This is highly utilised in *feature based image segmentation* (FBIS) introduced in Chap. 5.

Due to the wavelet transform (WT), we obtain multiscale representations characterising the image content over a number of resolutions. For finer scales, the WT coefficients represent details in an image. For coarser scales, the coefficients represent the dominant visual information of an image and demand less computation time. This enables rough estimates of the contents by computing larger scale coefficients, and thus increases computational efficiency [10].

Another advantage of multiscale representation is the possibility of *texture* modelling. Whereas colour is point-specific, texture occurs in a range of scales. The output of multiscale representation, i.e. texture features vectors, can be modelled by Markov-based frameworks. For instance, the *Hidden Markov tree* (HMT) models comprehend the WT coefficients as realisations of a set of statistical distributions. These models are commonly used for grey-scale image segmentation based on wavelet decomposition. They capture substantial statistical relations of the DWT coefficients. Apart from this application, the HMT is widely used in signal processing for signal estimation, denoising, classification and audio signals and image segmentation.

The WT is advantageous also for its good performance in compression [10] and the possibility of choosing the wavelet function most suitable for a given application. However, the choice

of the wavelet is not of too great influence on texture analysis [1].

The discrete wavelet transform (DWT) as a texture feature extraction method [10] has two particular disadvantages. First, it is the *lack of shift invariance*, i.e. a small shift in the input entails large changes in the output. The undecimated form of DWT may be the solution, however, it causes big redundancy, and thus increases computation cost. Second, it is the *poor directional selectivity*. The DWT features are unable to distinguish diagonal edges orientation ($\pm 45°$).

To overcome these both drawbacks, *pyramid-structured WT* (PWT) and *tree-structured WT* (TWT) [1] have been developed. The PWT recursively decomposes the approximation coefficients, however most of the textures have the important information in medium frequency bands. The TWT decomposes also details (horizontal, vertical, and diagonal) when necessary. For instance the *dual-tree complex wavelet transform* (DT-CWT) represents a very efficient feature extraction tool utilised in image retrieval [10]. However, we apply only the DWT in this thesis.

From the wavelet coefficients, the feature vectors can be obtained through various procedures. We can for instance compute the *mean* and *standard deviation* of the energy distribution of each subband at each decomposition level. It is also possible to use summed squared diagonal detail coefficients (see Fig. 2.21) of the highest decomposition level to characterise fast frequency components in images. This method of feature computation proves to be efficient as also presented in Chap. 5.

.

4

Image Segmentation Using the Watershed Transform

Image segmentation is one of the basic initial steps in image processing. This chapter deals with the *watershed transform*, which is a widely used tool for image segmentation, along with some other associated techniques. Sec. 4.5 studies the negative impact of noise on watershed segmentation and the improvement after image denoising in frequency and time domain.

To clarify the purpose of watershed segmentation, it may be used as outlined in Chap. 1 prior to feature based image *classification* in the following manner. We extract features of the originated segments using the discrete Fourier transform (DFT) or alternatively the discrete wavelet transform (DWT), both described in Chap. 2. As a result, each segment is characterized with 2 or 3 features invariant with respect to translation and rotation. Subsequently, the segments are classified in accordance with their features by self-organising *neural networks* (NN). However, feature based classification using NN is beyond the scope of this work, which only presents watershed segmentation as an alternative and complement of *feature based image segmentation* (FBIS) discussed in Chap. 5.

4.1 Watershed Transform

The watershed transform [9] interprets grey-scale images as three dimensional surfaces. Same as in geography, *watershed ridge lines* represent boundaries dividing drainage areas of different rivers called the *catchment basins*. Rain falling within these boundaries is collected in the actual basin. When falling on a ridge line, there is the same likelihood for the rain to be drained into either of the basins sharing the boundary. The catchment basins are the regions in the image we aim to identify by segmentation.

For watershed segmentation using Matlab, the function *watershed* [11] may be employed. This function finds the watershed ridges in images, assigns zeros to them, and labels the pixels of each region with the same integer value. Pixels belonging to the first segment are labeled with ones, in the second with twos and so on.

Prior to the watershed transform, images need to be preprocessed by one or more chosen methods described below. In this work, we use particularly the *distance transform*.

4.2 Distance Transform

The distance transform (DT) precedes the watershed transform to preprocess images [9]. As an input this transform requires binary (black-and-white) images. If necessary, images are converted to the binary form using a threshold of a certain level.

In Matlab, the DT is computed by the command *bwdist* [11]. This command calculates the Euclidean distance from every pixel to the nearest non-zero-valued pixel. One-valued pixels (white

colour) are assigned with zeros (black colour). Fig. 4.1 pictures the distance transform of the matrix F of size 16×16. Before computing the DT, a grey-scale matrix is firstly converted into black-and-white. To reveal the DT computation, we show the 5×5 cut of the matrix F, i.e. the black-and-white cut F_{bw} and its distance transform F_{dt}. Fig. 4.2 displayes the Matlab code computing the DT of F and defining the position of the cut.

$$F_{bw} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad F_{dt} = \begin{bmatrix} 0.2828 & 0.1414 & 0 & 0 & 0.1414 \\ 0.2828 & 0.1414 & 0 & 0 & 0.1414 \\ 0.1414 & 0.1414 & 0 & 0.1414 & 0.1414 \\ 0 & 0 & 0 & 0 & 0 \\ 0.1414 & 0.1414 & 0.1414 & 0 & 0.1414 \end{bmatrix}$$

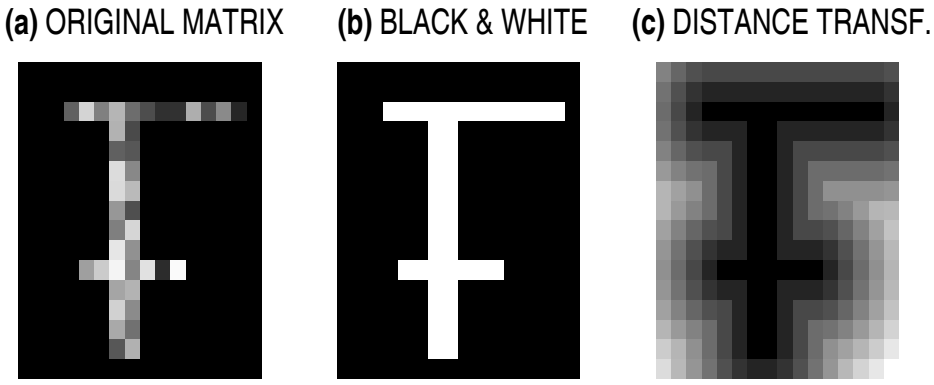


FIGURE 4.1. Distance transform.

```
% DISTANCE TRANSFORM: SIMPLE EXAMPLE
% MATRIX F (16x16)
  F=zeros(16,16); F(3,4:15)=rand(1,12)';
  F(4:15,7:8)=rand(12,2); F(11,5:11)=rand(1,7);
% CONVERSION TO B&W
  Fbw=F>=0.1   % threshold 0.1
% DISTANCE TRANSFORM
  Fdt=bwdist(Fbw)
% NORMALISATION
  Fdt=(Fdt-min(Fdt(:)))/(max(Fdt(:))-min(Fdt(:)));
% CUT OF MATRIX F
  Fcut=F(8:12,5:9), Fbwcut=Fbw(8:12,5:9), Fdtcut=Fdt(8:12,5:9)
```

FIGURE 4.2. Matlab code for distance transform.

When the DT finished, we can approach the watershed transform. In case that the objects to be identified are white, we have to obtain the *complement image*. Consequently, white pixels become black, in geographical interpretation, mountains in the original binary image become catchment basins. The computation process composes of the following steps. We compute firstly the DT, secondly the negative of the DT, and finally the watershed transform to produce

the label matrix. In this matrix, positive integers represent catchment basins and zeros symbolise the watershed ridges. When superimposed on the original image, we can display the segmented image. This technique is displayed in the top raw of Fig. 4.8.

Sometimes, objects can be split incorrectly, which is called *oversegmentation*. This phenomenon takes place in Fig. 4.8e. A couple of methods of overcoming oversegmentation are discussed below.

4.3 Gradients

Computation of the *gradient magnitude* [9] is one of the techniques for grey-scale images pre-processing prior to watershed segmentation. This technique aims to reduce oversegmentation.

In gradient magnitude images, pixels along the object edges are of high values, and other pixels are of low values. To achieve a good segmentation performance, the gradient image needs to be smoothed by filling gaps between objects and smoothing their outer edges.

One of the ways to compute gradient magnitude is linear filtering. We can employ, for instance, the *Sobel edge-emphasising filter*, which is either horizontal h or vertical h'. This filter uses a smoothing effect by approximating either the vertical or the horizontal gradient in an image, respectively.

$$h = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad h' = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

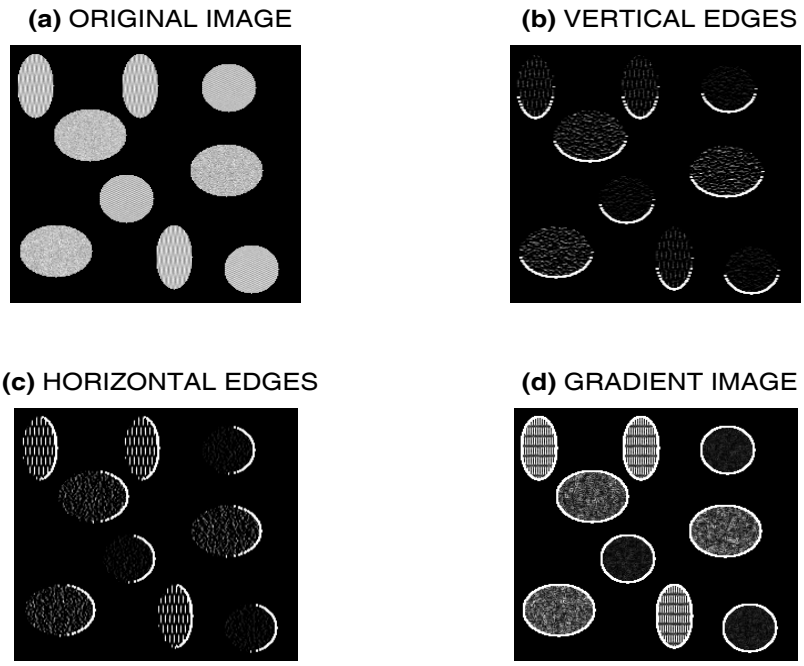


FIGURE 4.3. Gradient magnitude computed by convolution with Sobel filters.

Fig. 4.3b,c displays the results I_h and I_v of the convolution of the original image and the edge

```
% GRADIENT MAGNITUDE IMAGES
% HORIZONTAL AND VERTICAL SOBEL FILTERS
  hh=fspecial('sobel') hv=fspecial('sobel')'
% LOAD IMAGE
  A=load('SimImage.mat'); C=struct2cell(A); I=C{1};
% GRADIENTS
  Ih=imfilter(I,hh,'replicate');  % emphasise vertical edges
  Iv=imfilter(I,hv,'replicate');  % emphasise horizontal edges
  Ig=sqrt(Ih.^2+Iv.^2);    % combined gradient image
```

FIGURE 4.4. Matlab code for gradient magnitude computation by convolution with Sobel filters.

emphasising filters h and h', respectively. The resulting gradient image I_g is the combination of the two images with emphasised edges given by $Ig = \sqrt{I_h^2 + I_v^2}$. The programme code is enclosed in Fig. 4.4.

Problems that might occur are oversegmentation (even after smoothing) and association of the catchment basins with the objects. One of the methods of solving these problems is described in the following section.

4.4 Marker-Controlled Watershed Segmentation

The watershed transform applied directly to gradient images can result in a severe oversegmentation. To eliminate oversegmentation, we need some additional knowledge to determine the number of allowable regions. One of the methods using additional knowledge is *marker-controlled segmentation* [9].

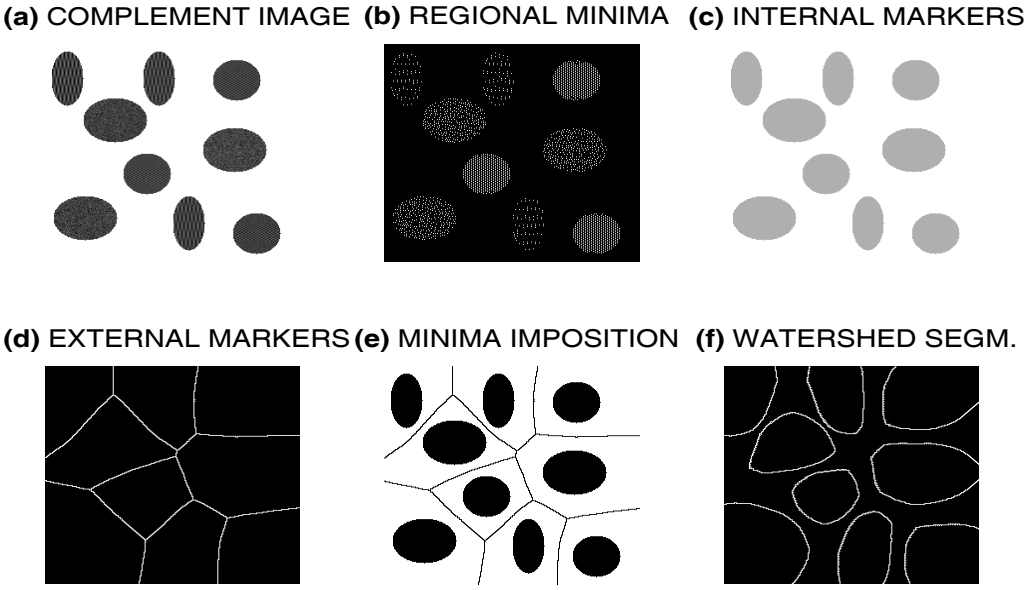


FIGURE 4.5. Marker-controlled watershed segmentation.

Markers are used for modifying gradient images. There are two types of markers, i.e. *internal* and *external* markers. Internal markers are imposed inside the objects to be identified, external markers are imposed outside the objects, i.e. in the background of an image. Markers can be computed by various methods such as linear filtering, nonlinear filtering, or morphological processing. The method choice is determined by the nature of the processed image. There is a range of methods of much higher complexity that the one chosen in this work. These complex techniques render more additional knowledge of the processed image.

Fig. 4.6 contains the programme code for marker-controlled watershed segmentation. The remainder of this section is dedicated to its explanation.

```
% MARKER CONTROLLED SEGMENTATION
% COMPLEMENT IMAGE
  A=load('SimImage.mat'); % load original image
  C=struct2cell(A); I=C{1}; N=size(I)
  Ic=ones(N,N)-I;  % complemental image (objects need to be the minima)
% INTERNAL MARKERS
  rm=imregionalmin(Ic);  % rm=binary image; regional minima location (ones)
  im=imextendedmin(Ic,0.5);  % extended minima transform, threshold=0.5
  Iim=Ic;   % internal markers (inside objects)
  Iim(im)=175/256;  % extended minima locations shown as grey regions
% EXTERNAL MARKERS
  Lim=watershed(bwdist(im));  % watershed transform of the DT of the int. markers
  em=Lim==0; % external markers: in the background- midway between the int. markers
% MINIMA IMPOSITION TECHNIQUE
  Ic2=imimposemin(Ic,im|em); % -> reg. minima only in marked locations, other
        % pixels pushed up; im|em is a mask marking the desired minima locations
% WATERSHED SEGMENTATION
  L=watershed(Ic2); I2=L==0;  % ridge lines
```

FIGURE 4.6. Matlab code for marker-controlled watershed segmentation.

As noted above, we endeavour to place *internal markers* inside each object of interest, that means into *regional minima* of a gradient image. Regional minima are defined as connected components of pixels with the same intensity value, whose pixels of external boundaries all have a higher intensity value. To obtain regional minima, we can use the Matlab function *imregionalmin* [11].

Images can be highly oversegmented, as a consequence of too many regional minima contained in a given gradient image. The large number of the minima might be caused by noise or other distortions, and can be destructive to segmentation. One of the methods of computing internal markers, which solves this problem, is the *extended minima transform* implemented by *imextendedmin* function [11]. This transform extends the minima locations. The pixels falling to these locations are of lower intensity than their immediate surrounding by an adjusted threshold. Consequently, a large number of redundant minima is suppressed. As noted in the programme in Fig. 4.6, we assign the same grey-color intensity value to all pixels in the extended minima locations, and thus exhibit the internal markers.

To calculate *external markers*, we need to identify the background in the gradient image, i.e.

to find pixels midway between the internal markers. Hence we compute the watershed transform of the distance transform of the internal markers image. The resulting ridge lines are the external markers.

As the next step, the gradient image is modified with the *minima imposition* procedure using both internal and external markers, realised by the command *imimposemin* [11]. In the gradient image, minima exist only in marked locations, other pixels' intensity has been increased to suppress other regional minima. Finally, we implement the watershed transform to the modified gradient image.

Fig. 4.5 displays the marker-controlled watershed segmentation applying the above described techniques. The objects to be labeled by segmentation need to be black, i.e. the minima of the image. Therefore, we work with the complement to the original grey-scale image. The result of segmentation is in the midway between internal and external markers. This image is not of the kind, that requires marker-controlled segmentation. That is why the contour of the objets is very rough. However, this is only a demonstration of the marker-controlled technique.

4.5 Impact of Noise on Watershed Segmentation

Noise can have a severe impact on image segmentation resulting in oversegmentation. Fig. 4.7 demonstrates addition of high-frequency noise. In this section, the additional noise is realised by two-dimensional sinusoids running vertically and horizontally. This kind of noise enables us to perform successful denoising using FIR filters and frequency windows. For noise of not such a very constrained frequency spectrum, e.g. white noise, we need to employ other denoising techniques, such as wavelet transform described in Chap. 3.

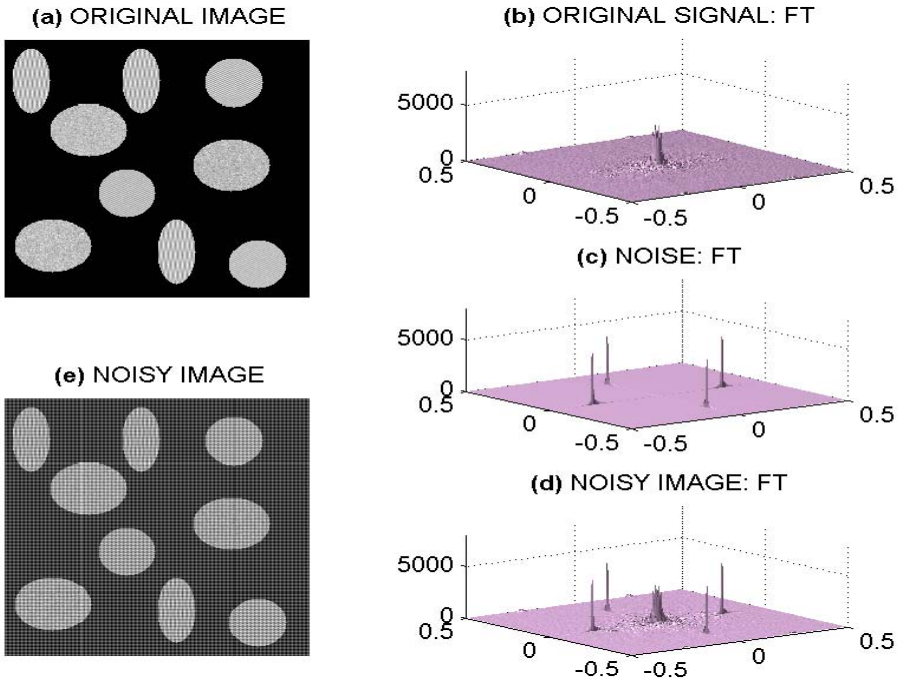


FIGURE 4.7. Adding noise to image.

Fig. 4.8 points out the corruptive effect of noise on image segmentation. Fig. 4.9 shows the corresponding Matlab code for noise addition and the distance transform followed by the watershed transform of the original and noisy image. In order to prevent oversegmentation, it is necessary to reduce the amount of noise in an image. The following subsections demonstrate denoising in the frequency and space domain, and the consequent improvement in segmentation results.

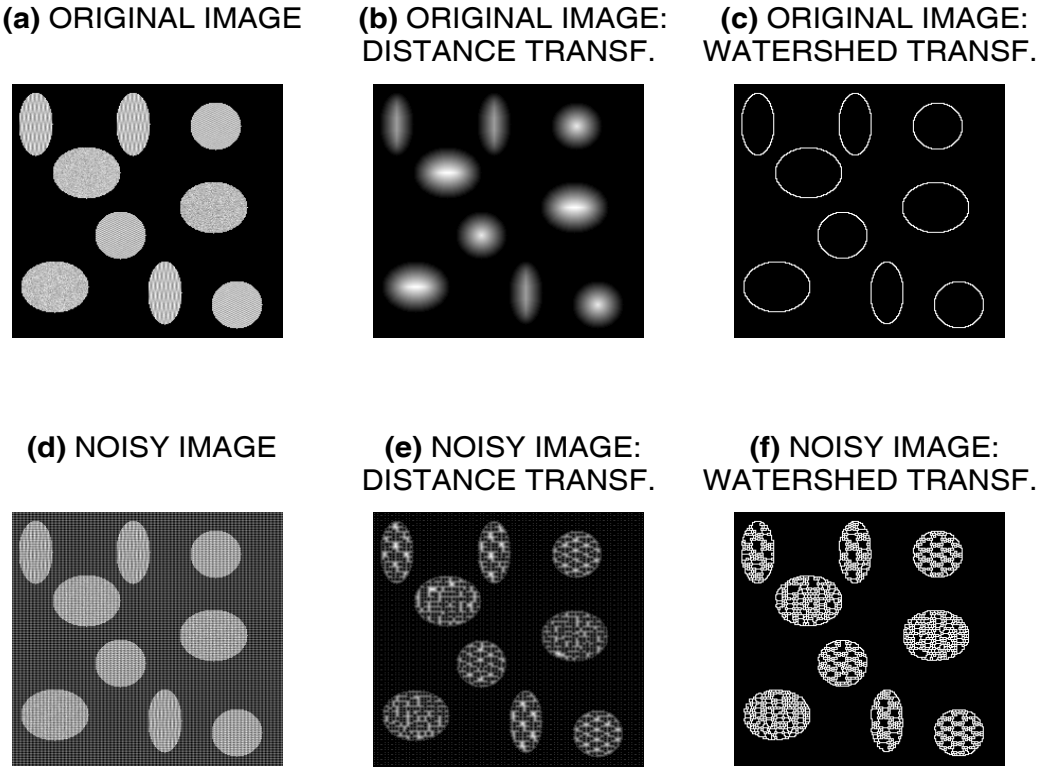


FIGURE 4.8. Destructive impact of noise on watershed segmentation causing oversegmentation.

```
% IMPACT OF NOISE ON WATERSHED SEGMENTATION
  A=load('SimImage.mat'); C=struct2cell(A); IM=C{1}; % load original image
  bw=im2bw(IM,graythresh(IM));  % image conversion to binary
  bwc=~bw;   % complement to binary
% DISTANCE TRANSFORM
  D=bwdist(bwc); D1=(D-min(D(:)))/(max(D(:))-min(D(:))); % normalisation
% WATERSHED TRANSFORM
  D=-D;   % D is negative distance transform (complement to D)
  D(bwc)=-Inf;  % each zero pixel in matrix D = -Inf
  LL=watershed(D);  % LL is a label matrix
  w=LL==0; % boundary display: assigning ones to zeros in LL, other pixels=0
  [L,num(1)]=bwlabel(bw);  % label image, num = no of connected objects
  [N,N]=size(IM);
  for i=1:N   % noise
     n=[0:N-1]'; a(i,:)=[sin(2*pi*0.3*n)]'; b(:,i)=[sin(2*pi*0.4*n)];
  end
  x=a+b; x=(x-min(m x)))/(max(max(x))-min(min(x))); % normalisation
  IMnoisy=IM+x;  % noise addition
  IMnoisy=(IMnoisy-min(min(IMnoisy)))/(max(max(IMnoisy))-min(min(IMnoisy)));
  bw2=im2bw(IMnoisy,graythresh(IMnoisy));  % conversion to binary
  bwc=~bw2; % complement to binary
% DISTANCE TRANSFORM
  D=bwdist(bwc); D2=(D-min(D(:)))/(max(D(:))-min(D(:)));
% WATERSHED TRANSFORM
  D=-D; D(bwc)=-Inf; LL2=watershed(D); w2=LL2==0;
  [L2,num(2)]=bwlabel(bw2); % label image
```

FIGURE 4.9. Matlab code for marker-controlled watershed segmentation.

4.5.1 Denoising in Frequency Domain

To prevent oversegmentation, additional noise has to be removed before segmentation takes place. Frequency windowing described in detail in Subsec. 2.2.5 is shown in Fig. 4.10.

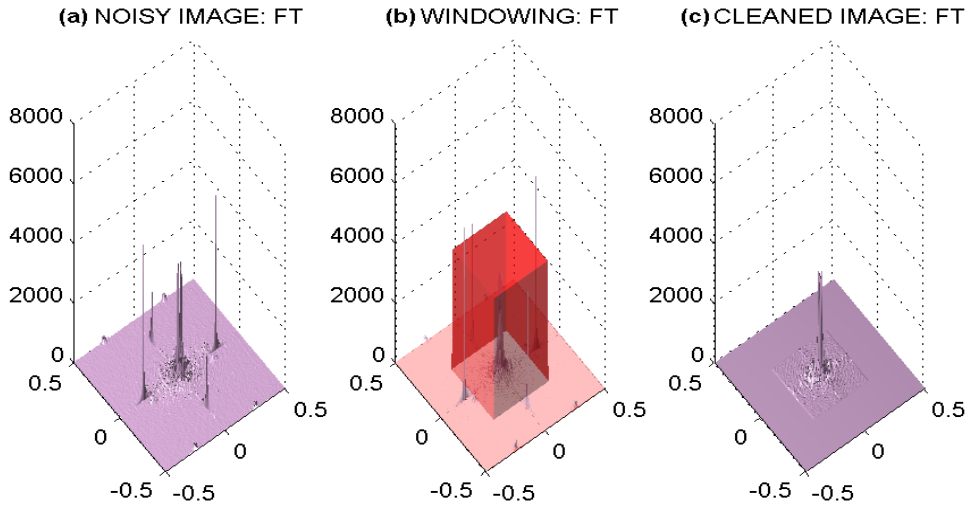


FIGURE 4.10. Image denoising using frequency windowing.

Thanks to denoising, we achieve a very good segmentation result as apparent from Fig. 4.11 comparing the noisy and clean image segmentation.

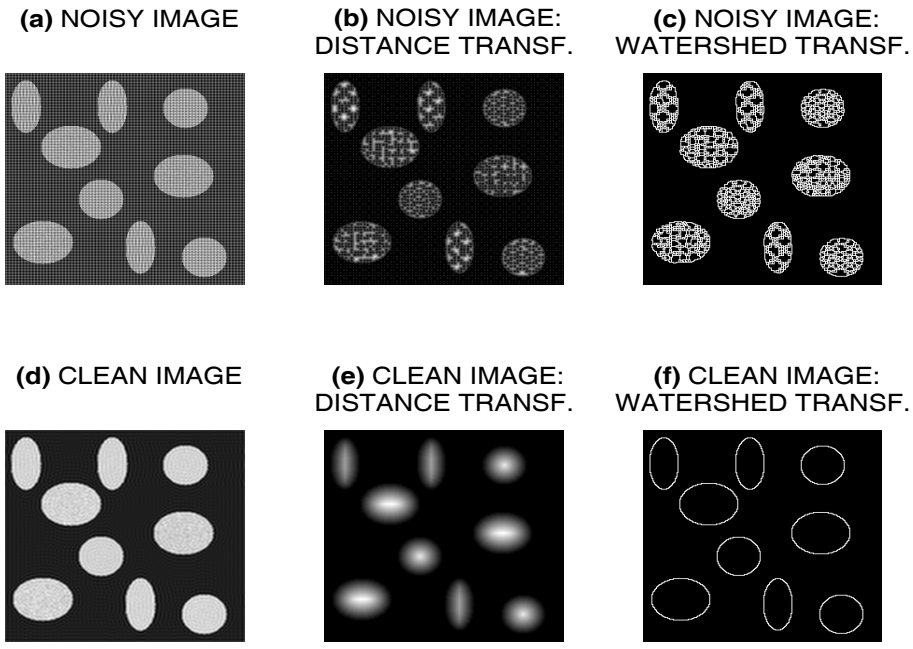


FIGURE 4.11. Effect of denoising using frequency window on watershed segmentation.

4.5.2 Denoising in Space Domain

Additional noise can also to be removed from images by filtering in the space domain. Fig. 4.12 pictures frequency characteristics of the two-dimensional FIR filters of order 5 and 15 designed on the same basis as in Subsec. 2.1.3.

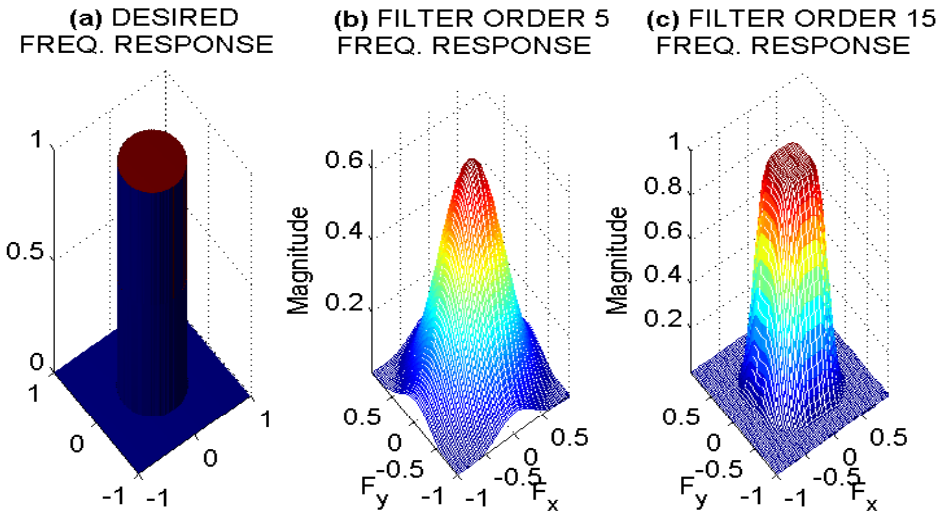


FIGURE 4.12. 2D FIR filters design.

Fig. 4.13 displays the improvement in segmentation after denoising using these two FIR filters. Same as above, watershed segmentation is preceded by the distance transform. In case of denoising with the filter of order 5, some oversegmentation is still apparent.

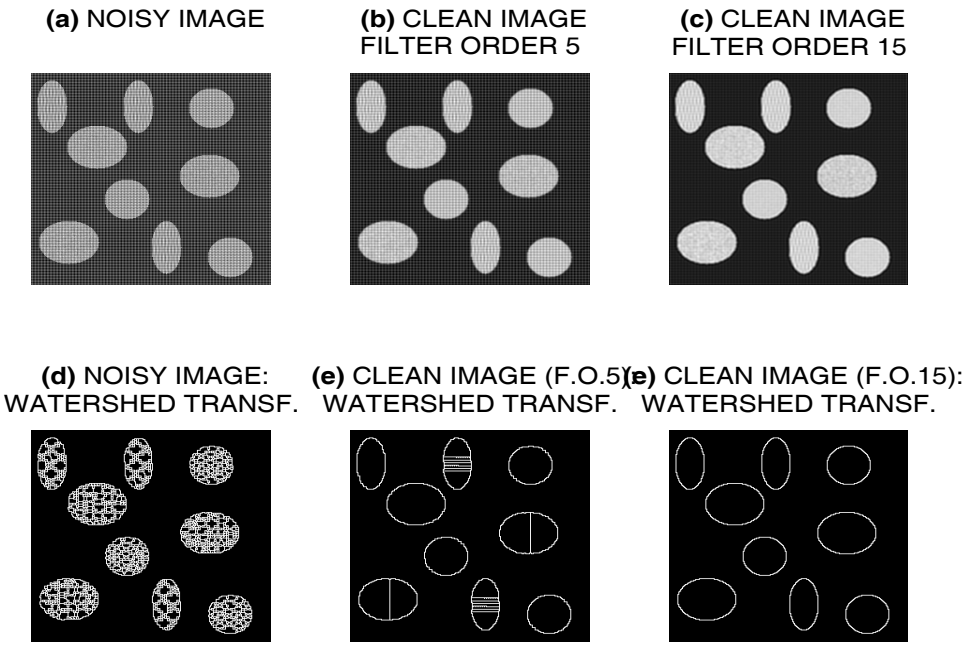


FIGURE 4.13. Effect of denoising using two FIR filters of different orders on watershed segmentation.

4.5.3 Comparison of Denoising Methods

This subsection brings the comparison of the denoising approaches dealt with in the two previous subsections, i.e. frequency windowing and space-domain filtering using the FIR filter of order 5 and 15. To evaluate the amount of noise in the image, we calculate the *signal to noise ratio SNR* in Decibels given as

$$SNR = 10 \log_{10} \left(\frac{\sum_{i=1}^{M} \sum_{j=1}^{N} I(i,j)^2}{\sum_{i=1}^{M} \sum_{j=1}^{N} (I(i,j) - I_n(i,j))^2} \right) \tag{4.1}$$

where I is the reference original image of size $M \times N$, I_n is the noisy image, and i, j are the raw and column indexes, respectively. Tab. 4.1 lists the denoising methods according to the increasing SNR values. Two of these methods have achieved the same number of segments as that for the original image.

TABLE 4.1. Comparison of denoising methods according to segmentation results.

Denoising Method	SNR [dB]	No of Segments
None	28.48	1378
FIR 5	32.40	23
Freq. Window	33.78	9
FIR 15	34.00	9
Original Image	196.59	9

.

5

Feature Based Image Segmentation

5.1 Principles of Feature Based Image Segmentation (FBIS)

As already mentioned, image segmentation is widely employed in image processing specifying boundaries between regions or objects of interest. *Feature-based image segmentation* (FBIS) is a segmentation technique engaging visual content descriptors of images called *visual features*. Visual features include *colour*, *texture*, *shape*, and *spatial* information. Features should be ideally invariant to the conditions of the imaging process, e.g. illumination of the scene, viewing angle etc. However, the greater is the invariance quality, the poorer is the discrimination ability.

The descriptors are either *global* (for whole images) or *local* (for objects or regions in images). To obtain a finer resolution for global features, we can apply the partition method, which cuts an image into parts of the same size and shape. Each of the parts is represented by global features. Local features are computed for objects or regions resulting from image *segmentation*.

Colour Features

An advantage of colour features is their point-like nature, which makes them independent on image size and rotation. Before approaching colour features, we need choose *colour space*. Colour space should be perceptually uniform, which means that the Euclidian distance between two colour triples equals the difference between the two colours perceived by the human eye [10].

The *RGB* space developed upon the colour receptors in the eye [10] is probably the most popular space for image display. This space comprises of the *red*, *green*, and *blue* component, whose values are added together to form each pixel value. We therefore call this space *additive*. The *CMY* space [1] used for printing comprises of *cyan*, *magenta* and *yellow* component. This space is *subtractive*. Both the RGB and CMY space are *device-dependent* and perceptually *non-uniform*. However, they are convertible into the following colour spaces.

The *L*a*b** and *L*u*v** [10] space are *device-independent* and *uniform*. L is a *luminance* (or lightness) component and a, b or u, v are *chromatic* components. L*a*b* is colour subtractive, whereas L*u*v* is colour additive. Both can be obtained from a *non-linear* transform of the RGB space. They are derived from human perception models, and hence most appropriate for image segmentation and retrieval.

Another *non-linear* colour space is *HSV* [1] used in computer graphics. HSV stands for *hue*, *saturation* and *value* (brightness). The hue is invariant to illumination and viewing angle, and therefore is also convenient for image segmentation and retrieval.

The *opponent* colour space [1] has the colour axes given as (R-G, 2B-R-G, R+G+B). The first two axes represent *chromacity* (colour) and do not vary with the changes of illumination. The third axis isolates the *brightness* component. Human perception is more sensitive to brightness than to colour, and hence the chromacity data can be down-sampled.

The first type of colour features dealt with hereby are *colour moments* reflecting colour distri-

butions in images. They are used especially for images containing solely the object. The first order moment is called *mean*, the second is *variance*, and the third is *skewness* [1]

$$\mu_i = \frac{1}{N}\sum_{j=1}^{N} f_{ij} \tag{5.1}$$

$$\sigma_i = (\frac{1}{N}\sum_{j=1}^{N}(f_{ij}-\mu_i)^2)^{\frac{1}{2}} \tag{5.2}$$

$$s_i = (\frac{1}{N}\sum_{j=1}^{N}(f_{ij}-\mu_i)^3)^{\frac{1}{3}} \tag{5.3}$$

where f_{ij} is the value of the i-th colour component of the pixel j and N is the total number of pixel in an image. The skewness is additional to the first two moments, since it can sometimes be sensitive to scene conditions. We have got three moments for three colour components, what gives nine features characterising the colour content of an image. These features are easily computable, and thus may be evaluated prior to more complex colour descriptors.

Another easily computable colour representation is the *colour histogram* [1] used either as a global or local descriptor invariant to translation, rotation about the view axis, and just slightly dependent on the scale, occlusion and viewing angle. For each colour space component, histogram shows the distribution of pixels among quantised bins. With the increasing number of bins, the discrimination ability grows, but on the other hand, the computation cost increases. Furthermore the large number of bins makes histogram unsuitable for creating effective database indexes. This problem can be solved by taking into account only a certain number of the largest bins. The small ones are highly impacted by noise anyway.

Matlab provides us with the command *hist* [11] to plot the histogram of a given image as shown in Fig. 5.1 for the grey-scale MR image of the spine.

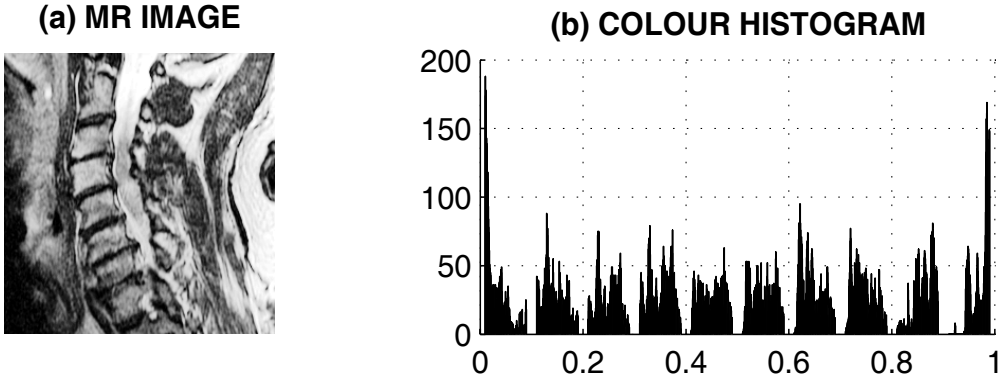


FIGURE 5.1. Colour histogram of MR image.

For very large databases, histogram-based description can result in too many matches. This high number of matches can be reduced by some additional information. Histogram itself provides no spatial information of pixels. We can introduce this information into this representation by simple image partition or image segmentation, but this requires more memory and computation

time. One method of introducing spacial information into the histogram is the *colour coherence vector* (CCV) [1]. Each histogram bin is divided into two parts, i.e. *coherent* and *incoherent*. The former represents large uniformly coloured regions. For the both descriptors, i.e. the CCV and histogram, HSV colour space is more suitable than L*a*b* or L*u*v*.

Colour Correlogram or autocorrelogram [1] has better results than the previous two. It combines colour distribution of pixels with spatial correlation of pairs of colours or identical colour in case of autocorrelogram.

Texture Features

Texture is a local pattern of intensity variation in an image produced by various nature of imaged objects's surface [3]. Texture cannot be described by means of colour or mean intensity. In some cases, it is solely texture features, which can distinguish between neighbouring objects displayed in an image. A good texture feature should be translation invariant as much as possible, since texture is unaffected by shifting the identified object [10].

Texture features are utilised in such disciplines as pattern recognition or computer vision. Methods of patern description are either *structural* or *statistical* [1]. The former, such as morphological operator and adjacency graph, work with structural primitives and their placement rules, and thus perform well for very regular textures. The latter, such as Fourier power spectra, and multiresolution filtering using the wavelet transform (WT), describe statistical distribution of the image intensity. The WT as an instrument of pattern characteristics is probably the most commonly used technique in CBIR [10]. Dependencies of the WT coefficients are usually statistically modelled employing multiresolution frameworks such as *Markovian statistical models* [10]. See Sec. 3.3 for more information.

Spatial Features

Spatial features characterise the layout of the objects or regions in an image, and thus prior segmentation is required. These features may be simple measures such as object size and its centre of gravity. Spatial characteristics should be invariant to translation, rotation and scaling.

They are either *region-based*, such as statistical moments, or *boundary-based*, such as rectilinear shapes, polygonal approximation, and Fourier-based shape descriptors [1]. Fourier descriptors are computed from the pixels of the regions' or objects's boundaries.

5.2 Results for Simulated Images

Throughout this section, we process the simulated image composed of four regions of different texture shown in Fig. 5.2 and produced by the source programme code displayed in Fig. 5.3. Using FBIS, we endeavour to classify pixels of the image into the four corresponding categories.

For segmentation, three feature extraction techniques are employed. The first of them is the *summed squared db2 diagonal detail coefficients* (see Fig. 2.21) of the highest DWT decomposition level. The second one is *intensity histogram* and the last one is the *standard deviation*.

For each pixel, we calculate one feature from a selected square neighbourhood of the pixel. The current pixel is called the *root pixel*, since it lies in the middle of the square neighbourhood. Computation is carried out by the Matlab function *nlfilter* [11]. This function applies

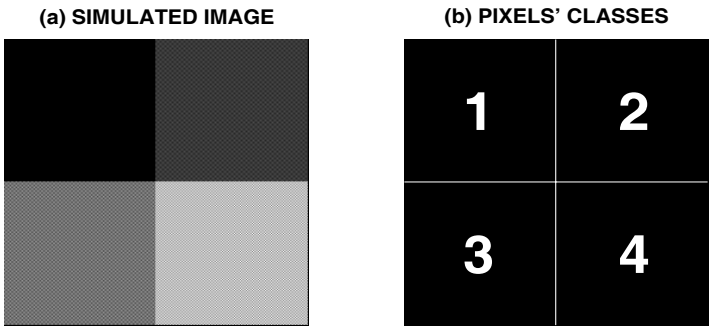


FIGURE 5.2. Simulated image and its pixels's classes visualisation.

```
% FUNCTION 'SIMULATE IMAGE'
  function A=SimImage()
  A=zeros(256);R=11;
% CLASS 2
  f1=0.25; f2=0.25; N=128; M=128; r1=1; c1=128;
  B=class(N,M,f1,f2)*0.3; A(r1:r1+N-1,c1:c1+M-1)=B;
% CLASS 3
  f1=0.3; f2=0.3; r1=128; c1=1;
  B=class(N,M,f1,f2)*0.6; A(r1:r1+N-1,c1:c1+M-1)=B;
% CLASS 4
  f1=0.4; f2=0.4; r1=128; c1=128;
  B=class(N,M,f1,f2); A(r1:r1+N-1,c1:c1+M-1)=B;
% FUNCTION 'GENERATE TEXTURE'
  function B=class(N,M,f1,f2)
    B=sin(2*pi*f1*[1:N]')*sin(2*pi*f2*[1:M]);
    % B=B+0.5*randn(N,M); % noise addition
    B=(B-min(B(:)))/(max(B(:))-min(B(:)))/2+0.5;
```

FIGURE 5.3. Matlab code for simulated image generation.

one of the above techniques to each square sliding block. It zero-pads the block at the edges, if necessary. The source code for FBIS into our levels is enclosed in Fig. 5.5.

Fig. 5.4 pictures the segmentation results. Obviously, the DWT method of features computation performs best. To evaluate the success of each method, we compute the percentage p of correctly classified pixels given by

$$p = \frac{correct}{total} 100\% \tag{5.4}$$

where *correct* is the number of correctly classified pixels in the area and *total* is the total number of pixels in the area. The algorithm for p evaluation is also contained in the code in Fig. 5.5. Tab. 5.1 displays results for each feature extraction technique for a chosen area (1, 2, 3 or 4) and for the whole image.

FIGURE 5.4. FBIS of simulated image using DWT features, intensity features, and standard deviation features.

TABLE 5.1. Comparison of FBIS methods according to percentage of correctly classified pixels in original simulated image.

FBIS Methods →	DWT Coefficients	Intensity	Standard Deviation
class 1	93.85 %	100.00 %	93.85 %
class 2	97.49 %	93.11 %	90.10 %
class 3	97.30 %	84.20 %	90.10 %
class 4	99.06 %	89.89 %	100.00 %
total	96.92 %	91.80 %	93.51 %

```
% FEATURE BASED IMAGE SEGMENTATION
  R=SimImageF; Region=[7 7];  % load image; 7x7 region for feature extraction
% FEATURES EXTRACTION
% nlfilter: perform general sliding-neighborhood operations
  F1=nlfilter(R,Region,'feature1e'); F1=classify(F1); % DWT coefficients features
  F3=nlfilter(R,Region,'feature3');  % intensity features
  F4=nlfilter(R,Region,'feature4');  % standard deviation features
% PERCENTAGE OF CORRECT PIXELS CLASSIFICATION
  [M,N]=size(R); val=[0.25 0.5 0.75 1];  % pixels' values in classified images
  image_total=M*N;  % total no of pixels in the image
  class_total=M/2*N/2; % total no of pixels belonging to aech single class
  r=[0 0 M/2 M/2]; c=[0 N/2 0 N/2];  % raw and column index
  FF={F1 F3 F4};
  for j=1:3, for i=1:4   % classes individually
     FF1=FF{j}(1+r(i):M/2+r(i),1+c(i):N/2+c(i)); FF1=FF1==val(i);
     k(j,i)=sum(FF1(:)); % no of correctly classified pixels in each class
     Fclass(j,i)=k(j,i)/class_total*100; end  % [%]
  end
  Ftotal=sum(k')/image_total*100  % [%], whole image
% FUNCTION 'COMPUTE DWT FEATURES'
  function s=feature1e(B)
    [cA,cH,cV,cD] = dwt2(B,'db2'); % DWT: 'db2', single level decomp.
    s=sum(sum(cD.^2));
% FUNCTION 'CLASSIFY DWT FEATURES'
  function C=classify(B)
    v=[0 0.00001 0.05 0.45]  % for SimImage (Region=[7 7]), 'db2'
    %v=[0 0.00005 0.013 0.07]  % for noisy SimImage (Region=[7 7]), 'db2'
    %v=[0 0.000000023 0.00016 0.015]  % or clean SimImage, THR/2
    %v=[0 0.001 0.0025 0.0035]  % for MRpater004 (Region=[5 5]); 'db2'
    N=length(v); C=zeros(size(B));
    for i=1:N-1, A=B>=v(i) & B<v(i+1); A=A*i/N; C=C+A; end
    A=B>=v(N); C=C+A;
% FUNCTION 'COMPUTE INTENSITY FEATURES'
  function s=feature3(B)
    N=4; s=0; v=[0:1/N:1]*0.8;  % intensity boundaries
    [MH,x1]=hist(B(:),N); rc=mean(x1(MH==max(MH)));
    for i=1:N-1, if rc>=v(i) & rc<v(i+1), s=i/N; end, end
    if rc>=v(N), s=1; end
% FUNCTION 'COMPUTE STANDARD DEVIATION FEATURES'
  function s=feature4(B)
    N=4; F1=std(B(:));  % standard deviation
    v=[0 0.01 0.05 0.098];  % std boundaries for SimImage (Region=[7 7])
    %v=[0 0.01 0.03 0.05];  % std boundaries for noisy image SimImage
    %v=[0 0.0008 0.007 0.025];  % std boundaries for denoised SimImage, THR/2
    %v=[0 0.05 0.1 0.15];  % std boundaries, for MR image (Region=[5 5])
    if F1>=v(N), s=1; end, for i=1:N-1, if F1>=v(i) & F1<v(i+1), s=i/N; end, end
```

FIGURE 5.5. Matlab code for FBIS of simulated image using DWT features, intensity features, and standard deviation features.

5.2.1 Impact of Noise on FBIS

This section focuses on the impact of white noise added to the simulated image. Noise addition is noted as a commentary in the Matlab code displayed in Fig. 5.3. The segmentation procedure is analogical to that described in the previous section. The only difference are the newly adjusted values of the boundaries between the classification bins for the DWT and standard deviation features. These new values are displayed as a commentary in Fig. 5.5.

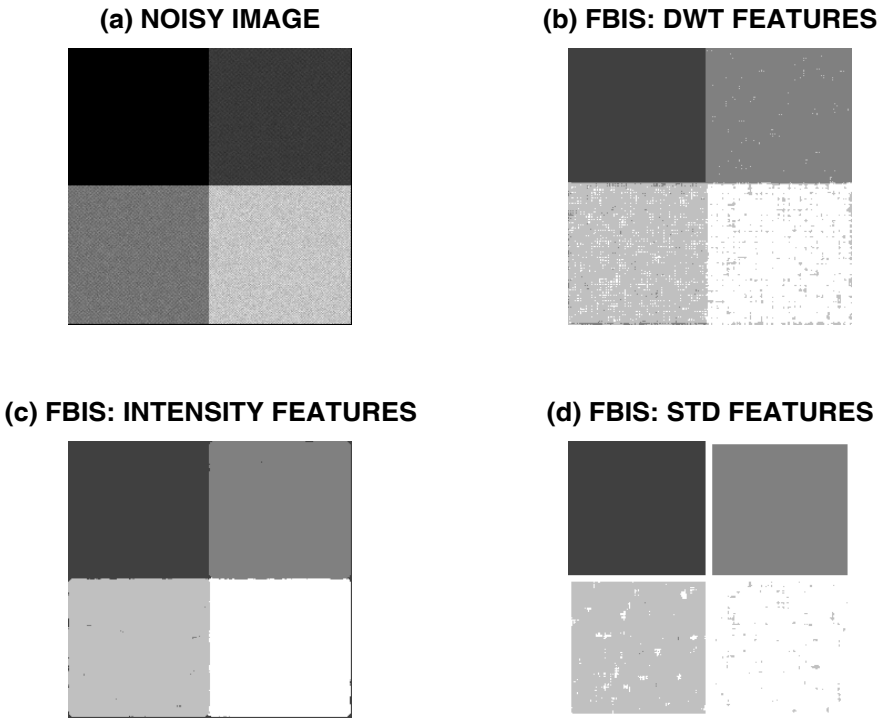


FIGURE 5.6. FBIS of noisy simulated image using DWT features, intensity features, and standard deviation features.

Fig. 5.6 and Tab. 5.2 demonstrate the segmentation outcome for the noisy image. For the DWT and standard deviation methods, the percentage of successful classification decreases as expected. However, it surprisingly increases for intensity histogram.

TABLE 5.2. Comparison of FBIS methods according to percentage of correctly classified pixels in noisy simulated image.

FBIS Methods →	DWT Coefficients	Intensity	Standard Deviation
class 1	93.85 %	98.63 %	93.85 %
class 2	96.84 %	98.59 %	90.10 %
class 3	90.04 %	98.65 %	87.63 %
class 4	93.28 %	98.27 %	98.20 %
total	93.50 %	98.54 %	92.44 %

5.2.2 Effect of Denoising on FBIS

For denoising of the simulated image, we use the programme displayed in Fig. 3.7. As shown in Fig.5.7, the *db*2 detail coefficients up to the second level are thresholded with the soft global threshold of half the value of the estimate given by Eq. (3.3).

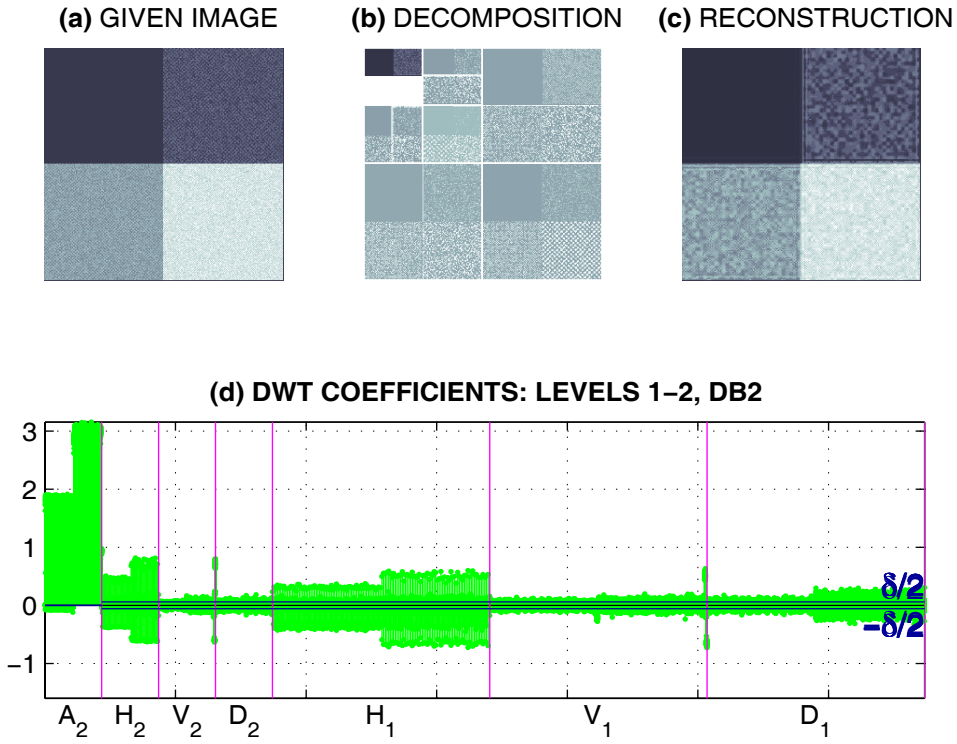


FIGURE 5.7. Denoising of noisy simulated image using *db*2 wavelet decomposition to level 2 and thresholding.

Fig. 5.8 and Tab. 5.3 display the effect of denoising on image segmentation. The percentage of correctly classified pixels for the DWT method increases in keeping with our expectations. For the standard deviation, the percentage decreases since the boundaries dividing the different pattern areas are now wider due to the denoising process. Intensity features behave slightly oddly, same as in the previous section.

TABLE 5.3. Comparison of FBIS methods according to percentage of correctly classified pixels in denoised simulated image.

FBIS Methods →	DWT Coefficients	Intensity	Standard Deviation
class 1	88.12 %	97.86 %	83.48 %
class 2	94.88 %	98.14 %	84.23 %
class 3	95.75 %	98.91 %	87.56 %
class 4	97.83 %	98.39 %	99.04 %
total	94.14 %	98.32 %	88.58 %

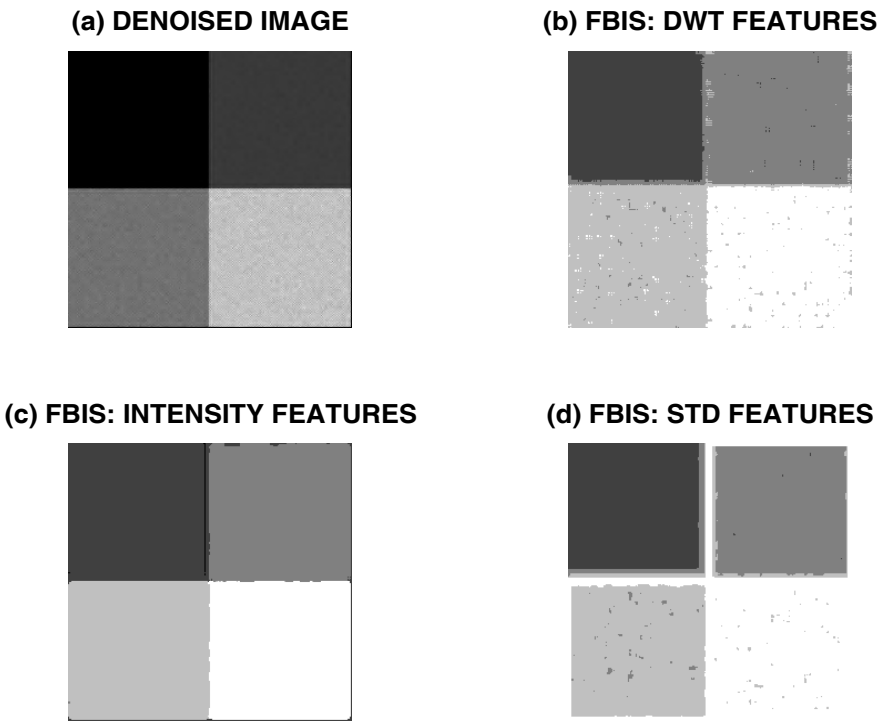
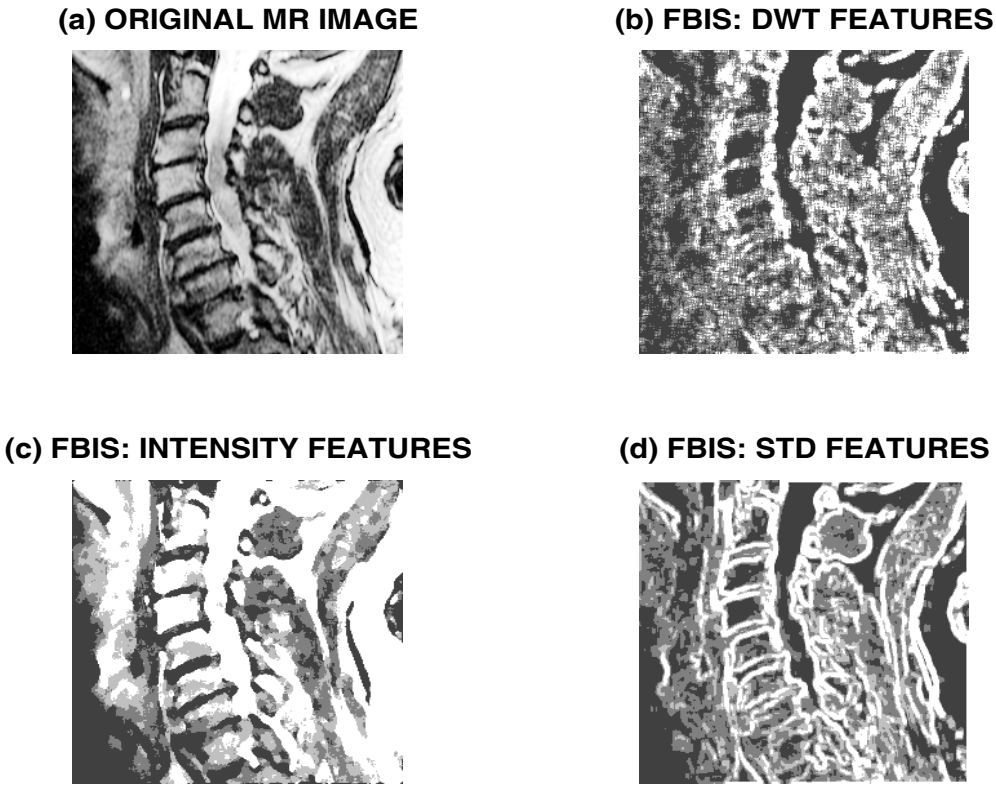


FIGURE 5.8. FBIS of denoised simulated image using DWT features, intensity features, and standard deviation features.

5.3 Results for Magnetic Resonance (MR) Images

Fig. 5.9 pictures a real MR image segmented using the three feature extraction methods discussed above. Different from the simulated image, the neighbourhood region is reduced to 5×5 pixels.



FIGUR E 5.9. FBIS of MR image using DWT features, intensity features, and standard deviation features.

To evaluate the percentage of successfully classified pixels, we need consult a specialist. Therefore, we leave Fig. 5.9 only to visual examination. Apparently, the DWT method does not perform very well. For further research, a more appropriate wavelet may be by sought. .

6

Conclusions

This thesis is dedicated to problems of image segmentation as an important preliminary procedure of image processing. Two segmentation tools, i.e. the watershed transform and feature-based image segmentation (FBIS), are introduced and applied on both simulated and real biomedical images.

The former requires image preprocessing in order to avoid oversegmentation. Therefore, the watershed segmentation is used in conjunction with the distance transform, gradient magnitude, and marker-control methods also described and demonstrated hereby.

The latter technique exploits visual features capturing the desired information in images to distinguish the objects or regions of interest. We focus mainly on texture features obtained from the Daubeschies of order 2 wavelet decomposition coefficients. These descriptors perform very well for simulated images. However, their application to magnetic resonance (MR) images is not very successful. On the other hand, these results might be improved with the use of another wavelet function.

FBIS is the most frequently used segmentation tool in content-based image retrieval (CBIR). A brief outline of how the CBIR systems work is provided.

A considerable part of this work deals with the negative impact of noise on image segmentation and the improving effect of image denoising. For noise removal, we use, firstly, frequency windowing employing the Fourier spectra, secondly, space filtering utilising finite impulse response (FIR) filters, and lastly, thresholding of the wavelet decomposition coefficients. The choise of the denoising method depends on the nature of the noise and signal.

In future, I will study various wavelet decomposition methods applied both to texture content extraction and image denoising. Denoising employs chosen techniques of thresholding of the wavelet coefficients. Another part of my future work will be focused on the possitive effect of denoising on FBIS utilising particularly texture features.

Texture-based segmentation of MR images enables us to create three-dimensional models of classified body tissues. The segmentation results will be discussed with a specialist to find out whether they are semantically meaningful and facilitate medical diagnosing. For this practical application, a well-arranged graphical user interface (GUI) will be designed.

The results of my work will be placed on Matlab Web Server for remote data processing administrated by the Department of Computing and Control Engineering at the Institute of Chemical Technology (ICT) in Prague. This server provides free download of executable files created in Matlab and running independently on Matlab platform.

7

References

[1] D. D. Feng F. Long, H. Zhang. Fundamentals of content-based image retrieval.

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

[3] T. H. Hinke S. J. Graves J. A. Rushing, H. Ranganath. Image Segmentation Using Association Rule Features. *IEEE Trans. on Image Processing*, 11(5), 2002.

[4] P. Sovka J. Uhlíř. *Číslicové zpracování signálů*. Vydavatelství ČVUT, 2. vydání, Praha, 2002.

[5] G. Oppenheim J. M. Poggi M. Misiti, Y. Misiti. *Wavelet Toolbox*. The MathWorks, Inc., Natick, Massachusetts 01760, April 2001.

[6] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, San Diego, USA, 1998.

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

[8] R. Polikar. Wavelet tutorial. eBook, March 1999. http://users.rowan.edu.

[9] S. L. Eddens R. C. Gonzales, R. E. Woods. *Digital Image Processing Using Matlab*. Pearson Prentice Hall, 2004.

[10] C. W. Shaffrey. *Multiscale Techniques for Image Segmentation, Classification and Retrieval*. PhD thesis, University of Cambridge, August.

[11] Inc. The MathWorks. *Image Processing Toolbox for Use with Matlab*. The MathWorks, Inc., 3 Apple Hill Drive, Natick, Massachusetts 01760-2098, April 2001.

[12] Inc. The MathWorks. *Signal Processing Toolbox for Use with Matlab*. The MathWorks, Inc., 3 Apple Hill Drive, Natick, Massachusetts 01760-2098, April 2001.

[13] C. Valens. A really friendly guide to wavelets. eBook, 2004. http://perso.wanadoo.fr.

[14] S. V. Vaseghi. *Advanced Signal Processing and Digital Noise Reduction*. Wiley & Teubner, West Sussex, U.K., first edition, 1996.

[15] E. W. Weisstein. Mathworld. eBook, 2003. http://mathworld.wolfram.com.

[16] www.dsptutor.freeuk.com. Digital signal processing. eBook, 2006.