A study of wavelets combined, if desired, with other methods for use in edge sharpening of ultrasound images.

by

Amund Solbakken
Tor Erik Sommerland

Masters Thesis in
Information and Communication Technology

Agder University College

Grimstad, May 2003

Abstract

The increased use of laparoscopic surgery has resulted in a growing interest in use of ultrasounds. Ultrasounds do not have any known side effects, and it provides the surgeon with real-time imaging. In spite of its advantages, ultrasounds also have some disadvantages. The most obvious is that ultrasound images contain much noise. Another problem is that edges of vital organs may appear diffuse. In this thesis we will look at ways of sharpening edges by using a relatively new mathematical method called wavelet.

With wavelets we analyse signals according to scale. This idea started with Fourier in the 1800's, when he used sines and cosines to represent functions. After Fourier, mathematicians where led from frequency to scale analysis of functions. But it was not until the 1980's that wavelets really came in to play. The work done by Ingrid Daubechies in 1990 was a milestone in the world of wavelets; she created a set of orthonormal basis functions.

In this thesis our motive was to sharpen edges of blood vessels in ultrasound images. These edges may be both dark and diffuse, and our goal was to make them narrower and brighter. To find out if wavelets could be used for this purpose we developed a prototype application that implements the wavelet-based methods. With this application we performed some tests on very simple synthetic images containing edges. Our results imply that the Mexican hat wavelet is exceptionally well suited for finding and sharpening edges. We were able to "attune" the Mexican hat to the edge so that the centre of the edge became many times brighter, at the same time as the noise in the image was reduced. We were able to both located the two sides of the edge so that it could be made narrower manually, in addition to make the edge slimmer by means of wavelet transformation.

Preface

This thesis is written for SINTEF Unimed in Trondheim, Norway, and is a master thesis in information and communication technology (ICT).

In December 2002 we decide that we would go for this thesis presented to us by Per Henrik Hogstad (HiA), on behalf of SINTEF Unimed in Trondheim. They had several theses were wavelets was the main theme. Our first whish was to study wavelets for use in reduction of noise in ultrasound images. However, we were later requested to change our thesis definition in direction of edge sharpening. We instantly agreed to this, and this became our task.

During January and February 2003, Per Henrik Hogstad gave us, along with some other groups which also worked with wavelets, several lectures in Linear algebra, Fourier and Wavelet theory. These lectures have been of great help while working on this project. Hogstad has also been our supervisor during this period, and we have received guidance from him on many occasions. In addition we have had Thomas Langø as a contact at the SINTEF Unimed in Trondheim.

As mentioned we started with some preliminary studies of theory in January 2003. Linear algebra and Fourier must be said to be a repetition of mathematics we already knew, but it did good to refresh our knowledge of these subjects. The wavelet theory was completely new to us, and has definitely ignited a spark of interest and fascination in us.

Documentation of this thesis has been parallel to the development of a prototype application through the entire project. This application was intended for use in later projects as well as this one. We have also had several meetings with our supervisor to discuss problems we have encountered during the process.

Table of contents

1 Introduction

1.1 Modern medicine

Today many operations are performed with the so called laparoscopic surgery. This means that the surgeon inserts a number of pipes through the skin and further through the abdomen in the front or in the side of the patients body. Through these pipes the surgeon can thread all the instruments needed for the surgery, including a small camera with light. For several years this kind of operation has been used for surgeries in the abdominal region of the body. The increased use of laparoscopic surgery will benefit both patient and society. Compared to traditional open surgery, laparoscopic surgery is less straining for the patient and therefore shortens the convalescence time. Considering this, it is desirable to develop the technology of laparoscopic surgery even further.

Scientists and surgeons in Trondheim have together developed a new IT based "window" to the interior of the human body. This system transforms advanced X-ray and MR pictures to a three dimensional map that the surgeons can use to navigate while he operates. This 3D map is shown on a screen in the operating room, along with the video provided by the small camera inside the body. In addition three smaller pictures are shown representing the three planes in 3D space. On each of these planes the tip of a special developed pointing device is indicated by a cross. On this map and in the three pictures all vital organs and the operating instruments are shown, making it much easier for the surgeon to navigate the instruments without injuring any organs.

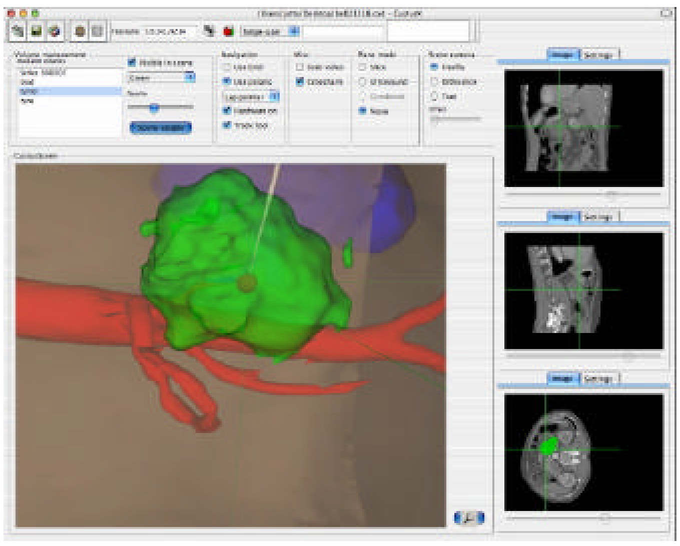


Fig 1.1 (Source [8])

This system is a valuable extra to the surgeon because it gives him the opportunity to look behind, or even through, different organs that may be blocking for the video camera in side the body. It is therefore also possible for the surgeon to take advantage of this system in order to plan the insertion of the pipes in the patient's body before the operation takes place.

This navigation system is already in use in the operating room at the University Hospital in Trondheim, and makes it possible for surgeons to remove tumours with laparoscopic surgery instead of open surgery. It also makes surgeons capable of using laparoscopic surgery in more cases then before. [16]

Next step in the development of this system is adapting it to use of ultrasound instead of X-ray and MR pictures. Today the X-ray and MR based maps show the body before the surgery. With the use of ultrasound, these maps could be updated while the surgery is in progress. This would help the surgeon to better determine if he has completely removed e.g. a tumour.

1.2 Ultrasound

Use of ultrasound instead of X-rays will also be desirable in many other situations. One example is the case of abdominal aorta aneurysme, which is a diverticulum (swelling) on the main aorta inside the abdominal cave. If such a diverticulum should burst, the patient would die shortly after. It is therefore necessary to operate on the patient before this happens. Today this surgery is performed at the University Hospital (St. Olavs Hospital) in Trondheim by inserting a catheter inside the aorta from the groin up to the diverticulum. The surgeon is then able to place prosthesis to support the aorta in the region of the diverticulum. This surgical operation takes about 5 to 6 hours. Today X-rays are used to navigate the catheter from the groin to the swelling. During this operation X-rays are used for about an hour. [2] Unfortunately the use of X-rays can have very harmful consequences for both the patient and, not at least, the surgeon. It would therefore be a great benefit to replace to use of X-rays with ultrasound. Ultrasound is, unlike X-rays, not ionising, and therefore not hazardous to those exposed to it. In addition, ultrasound imaging is real time, so the images can be seen on the spot.

Despite all these advantages there are also some problems attached to the use of ultrasound. One of the main problems is that ultrasound images contain white speckles of noise. It is of great interest to be able to remove most of this noise and sharpen edges so that the more important structures are clearer. This will be necessary so that the surgeon relatively easy can navigate the catheter inside the aorta.

Edge sharpening is also one motive behind this thesis. Since the human vision is very sensitive to sharp edges and other details in images, there will be of interest to make edges of vital organs more clear. When it comes to edge sharpening there are several methods available. One simple, but effective tool is use of gradient operators. This is a well proven method for this job. However, gradient operators [6] will be of little use in this thesis due to its high noise sensitivity. Gradients may prove to be more prominent in the future if techniques to filter out noise are more developed. Other possible methods could be e.g. splines [11].

In this thesis we will concentrate on wavelets as a tool for edge sharpening. With wavelets [9], as a relatively new mathematic method, we hope to be able to find and sharpen edges. The basic procedures attempted in this thesis works by treating images as signals where different parts (noise, edges, etc.) results in different frequencies. Wavelets can be brought into harmony with certain frequencies and to more or less annihilate other frequencies. We thereby hope to overcome the problem with noise to a certain degree so that we are able to sharpen the edges that we shish to make clearer.

2 Digital image processing

2.1. Overview

In this chapter we will give an introduction to digital image processing and computer images as a whole. We will be working with computer images in this master thesis and this chapter will give the necessary background for understanding the concepts behind the techniques implemented in this project. This chapter will explain how images are stored and treated in the world of computers, in addition to giving a brief overview of some colour model concepts. The foundation of the methods that are going to be tested in this project is to treat digital images as signals. This chapter is followed by a chapter containing some theory on signal analysis. The motive behind this chapter and the next is to give an understanding of the main principle behind these methods.

2.2. Computer images

Vision allows humans to perceive and understand the world surrounding us. And computer vision aims to duplicate the effect of human vision by electronically perceiving and understanding images. But the images that are stored and displayed on a computer are significantly different from those seen by the human eye. We live in a three dimensional world. When computers try to analyse objects in 3D space, visual sensors such as TV cameras usually give two dimensional images. This projection into the 2D plane results in a tremendous loss of information. [6]

While natural images are continuous, computer images are discreet digital versions of reality. Images on a computer are made up of small dots called pixels. The resolution of an image, i.e. the ability to distinguish fine spatial details, can be expressed as the number of pixels per inch (ppi). Each pixel has a specific colour of illuminating light. There are three kinds of digital images: Bi-tonal images (images with only two colours, typically black and white), greyscale images and colour images. The bit depth of an image is determined by the number of bits used to define the colour or shade of a pixel. Bi-tonal images only require one bit per pixel to determine which of two possible colours it has. Greyscale images typically have a bit depth of 2 to 8. A colour image is typically represented with a bit depth of 8 to 24. With a bit depth of 8 it is possible to represent 256 different colours or shades, while a bit depth of 24 offers 16.7 million tones.

The numerical value of a pixel in a colour image is associated with a colour space. In 1931, the Commission Internationale de l'Eclairage (CIE) developed a device independent colour model called CIE XYZ. The CIE XYZ was based on the tristimulus theory of colour perception, which states that the human retina has three kinds of colour receptors with peak sensitivities to 580 nm, 545 nm and 440 nm.

These primary colours can be combined in an additive manner to match all colours in the visible spectrum. Other colour models have been derived from CIE XYZ. The most widely used with computer graphics is the RGB colour model, which uses red, green and blue as its primary colours. The RGB colour model allows almost all colours visible by humans to be produced.

2.3 Data representation

A digital image can be thought of as a two dimensional matrix where its elements are called pixels (abbreviation of picture elements). The process of displaying an image involves creating a graphical representation of this matrix where the pixel values are assigned a particular greyscale (monochromatic image) or a particular color.

The pixels of an image can be of several data types. A binary image has just two possible values, often assigned to black and white. A greyscale image has often positive integer values from 0 to a maximum. It is possible to have pixels with negative values, pixels with real numbers, and even pixels with complex numbers. An example of an image with negative pixel values are thermal images with negative temperatures. Images with pixels that are real numbers can be found in images that represent a sinusoid wave with values varying from -1 to +1. Images with complex pixel values can be found in some image transforms like the Discrete Fourier Transform.

When digital images are stored as files in the computer, the files contain not only the pixel values associated with each coordinate, but also an image header which provides additional information such as the widht and height of the image, the data type of the pixel elements, the color model, etc. [6]

3 Signal analyses

3.1 Overview

The fundamental principle behind the methods investigated in this master thesis is the treatment of digital images as signals. In most practical situations signals are functions with time on the horizontal axis and some other measure on the vertical axis. One way to analyse these signals is to dissect them into the frequency components that occur in the signals. The most widely used method for doing this is the Fourier transform, named after the 19th century mathematician.

The Fourier transform allows one to look at a signal from two different perspectives, either as a function of time or as a function of frequency. This is very useful for static signals or in situations where we are only interested in the frequencies. However, if we are going to analyse images, we will be more interested in knowing where a frequency occurs than what the frequency is. For example, an edge in an image will result in a specific frequency component in the Fourier transform of the image, but this just tells us about the existence of this edge, not where it occurs in the image.

There are several ways to rectify this limitation of the Fourier transform. One way is to use the Fourier transformation together with a window function. This window allows one to transform parts of the signal in order to find the frequencies in specific intervals of the signal. The problem with this method, however, is that the narrower this window is (and hence, the better the time resolution), the worse will the frequency resolution be.

The methods used in this project are based on wavelet transforms. This is an improved version of the Fourier transform with window, and it gives high frequency resolution for low frequencies and high location determination for high frequencies. Although the mathematician Haar worked with wavelets already in the 1930s, it was first in the 1989-90s that this method really came in to action. Since then this method has gained access to many fields, such as seismic geology, image processing, communication, quantum physics, music and others.

This chapter will give an introduction to signal analysis and wavelet transforms. Section 3.2 will introduce the Fourier transform, which is the basic mechanism of the wavelet transform. Section 3.3 will move on to Fourier with window, which can be considered to be the forerunner of the wavelet transform. Finally, section 3.4 will focus on some theory that is highly relevant in this project, namely wavelet theory.

3.2 Fourier Transformation

The Fourier transformation transforms a function from the time domain to the frequency domain. The Fourier transformation exploits the fact that all functions can be written as an infinite sum of cosine and sine waves with different frequencies. This remarkable property was shown by the French mathematician J. Fourier in 1822 [1]. If x(t) is a function of time t, then the Fourier transformation of that function results in the function X(f) of frequency f. The Fourier transformation can be expressed as

$$X(f) = \int_{-\infty}^{\infty} x(t)\cdot\psi^{*}(t,f)\,dt$$

The function ? (t,f) is called the base function of the Fourier transformation. This is the complex exponential function

$$e^{(i2\pi ft)} = \cos(2\pi ft) + i\cdot\sin(2\pi ft)$$

This means that the Fourier transformation of x(t) can be written as a sum of two integrals[1]:

$$X(f) = \int_{-\infty}^{\infty} x(t)\cdot\psi^{*}(t,f)\,dt = \int_{-\infty}^{\infty} x(t)\cdot e^{(-i2\pi ft)}dt = \int_{-\infty}^{\infty} x(t)\cos(2\pi ft)\,dt - i\cdot\int_{-\infty}^{\infty} x(t)\sin(2\pi ft)\,dt$$

The inverse of the Fourier transformation is

$$x(t) = \int_{-\infty}^{\infty} X(f)\cdot e^{(i2\pi ft)}dt$$

When we transform the function x(t) we find how much correlation there is between x(t) and the base function ? (t,f). The result of the Fourier transformation is a complex function where the real part says how much cosine with frequency f there is in the function x(t), and the imaginary part says how much sine there is with frequency f. We will look at what this means by focusing on the real part.

The function cos(t) is a periodic function with frequency 1/(2p). If we multiply the argument by 2p we get a function with frequency 1. The frequency of the function cos(2pft) can then be controlled with the parameter f; if f = 1 the frequency of cos(2pft) will be 1, if f = 2 the frequency will be two, etc. There is one thing to note about the integral of a cosine or sine function from -8 to 8 . The integral is zero because the function is symmetric about the horizontal axis (t axis). We will now look at the integral ∫x(t)·cos(2pft)dt and find out what it means.

[1] The reason for the subtraction of the last term, instead of addition, is that base function is actually complex conjugated in the integrand.

Suppose we have a signal that we wish to analyse by finding the frequency components of the signal. Let us now say that the signal can be expressed as the function

$$x(t) = \cos(2\pi t) + \cos(2\pi 3t)$$

This signal contains two frequency components, one cosine signal with frequency 1 and one with frequency 3 (these frequency components are the ones we wish to find, so pretend for a moment that you do not know what they are). This is what the signal looks like when its graph is plotted:

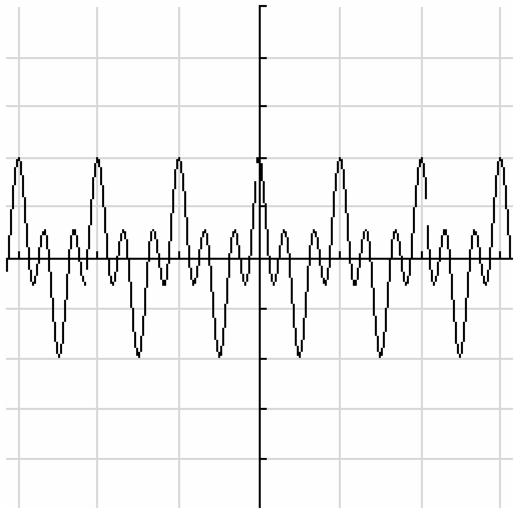


Fig. 3.1: cos(2pt) + cos(2p3t)

Fig. 3.2 below shows what the product x(t)·cos(2pft) looks like for various values of f (the frequency). The graphs show the integrand in the Fourier transformation.

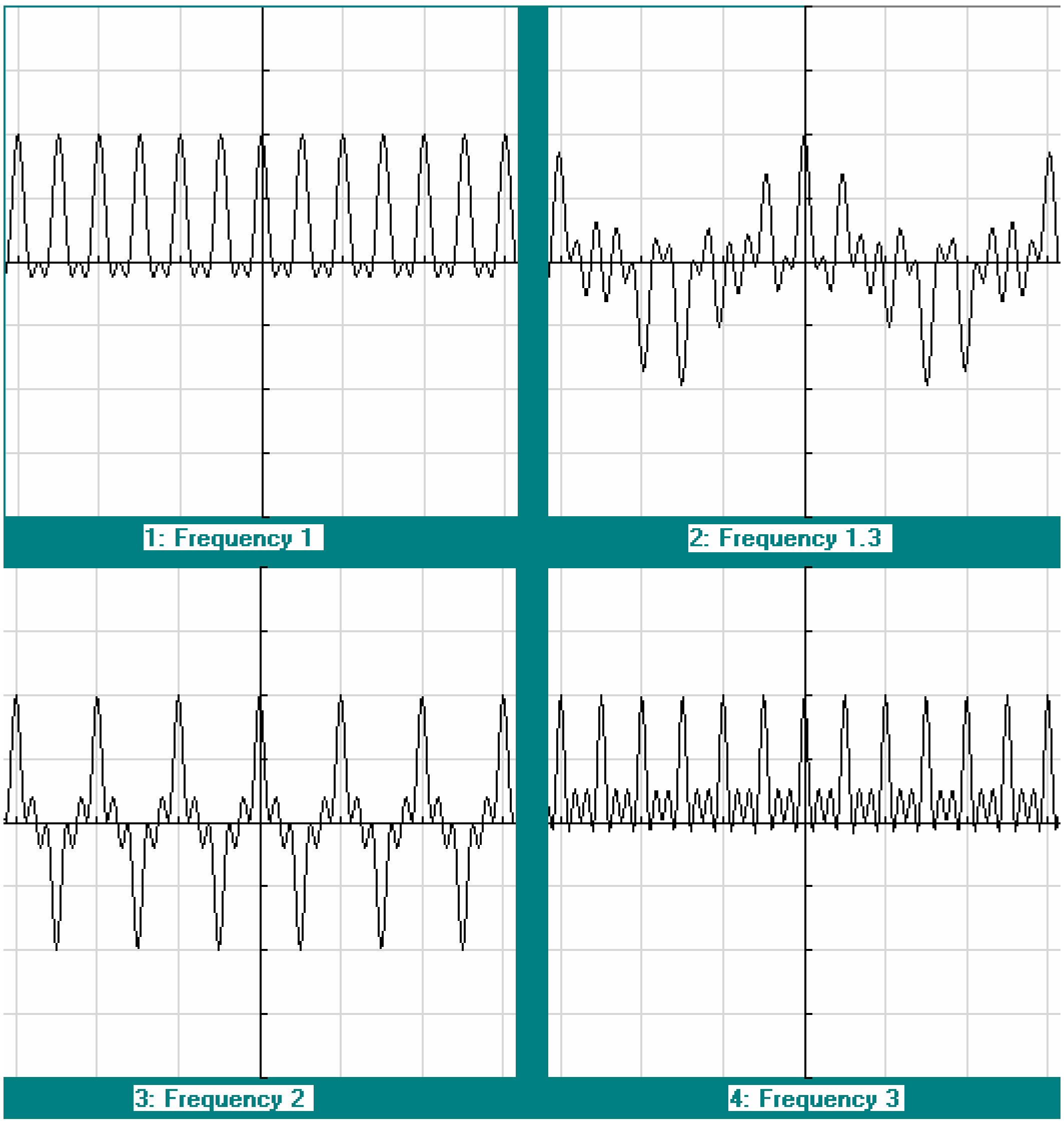


Fig. 3.2: The product (cos(2pt) + cos(2p3t))·cos(2pft) for different values of f

In the first and fourth graph in fig. 3.2 there is strong correlation between the function x(t) and cos(2pft), while in the second and third graph there is a weak correlation. Graph 2 and 3 appear to be equally distributed on both sides of the t-axis. It is obvious that the integral from -8 to 8 of graph 2 and 3 would be very close to zero, while the integrals of the other two graphs would have a high positive value. So the graph of the function X(f) = ?(cos(2pt)+cos(2p3t))·cos(2pft)dt would look like the curve in fig. 3.3 below.

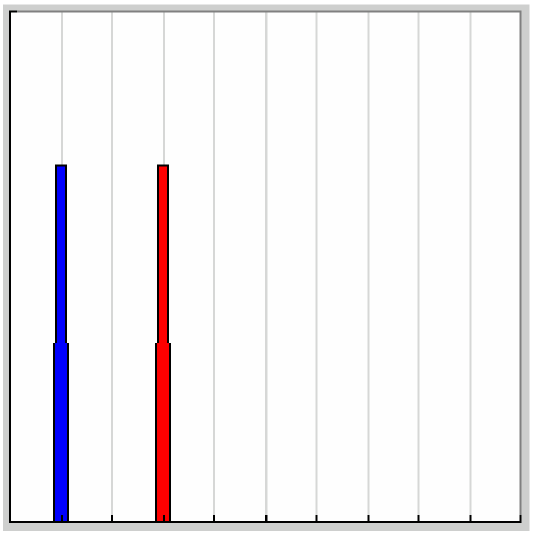


Fig. 3.3: The Fourier transformation of cos(2pt) + cos(2p3t)

The horizontal axis in fig. 3.3 is now f (frequency), because the Fourier transformation is a definite integral from -8 to 8 with respect to the variable t (time). Because the frequency of the base function, cos(2pft), was represented as a variable, f, rather then a fixed value, the resulting integral will be a new function, X(f), where this variable is free. As one can see from the graph of X(f), there are two values of f that (literary) stick out. The blue curve shows the value of X(f) at f = 1, and the red shows X(f) at f = 3. The function that were transformed, x(t) = cos(2pt) + cos(2p3t), has two cosine components, namely one with frequency 1 and one with frequency 3. The amplitude of the blue and red curves tell us how high the amplitudes of the two frequencies are in the original signal, x(t).

The Fourier transformation, although probably the most commonly used, has severe limitations which make inadequate for some purposes. When a function is transformed from the time domain to the frequency domain with the Fourier transformation, the time information will be unavailable. Because the Fourier transformation is defined as an integral from -8 to 8 , a frequency component will affect the transformation equally regardless of where it occurs in the function being transformed. This can best be illustrated with an example.

The two graphs in fig. 3.4 both contain the same frequency components (1 and 3), but they occur at different times. The signal to the left is a stationary signal; its frequency components are the same over the entire time scale. The graph on the right is a non-stationary signal; its frequency components vary over time.

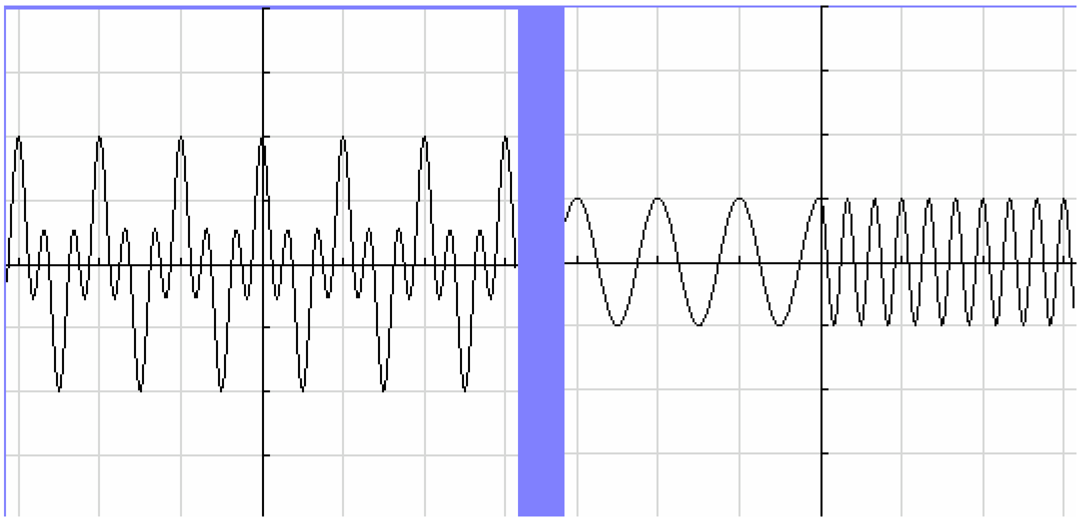


Fig. 3.4: Two functions with the same frequency components occurring at different times

Most signals in practice are non-stationary, and very often we are more interested in finding where a frequency occurs than finding what the frequency is. In order to do so scientist have been using a modified version of the Fourier transformation, called Short Term Fourier transformation (STFT).

3.3 Short Term Fourier Transformation

The problem that STFT solves is that it not only finds what frequency components a signal consists of, but gives some clue to where these components exists in the signal. The basic idea of the STFT is that it assumes that for some interval, hopefully not to small, the frequency of the signal does not change. It then applies the ordinary Fourier transformation to that interval. This interval is then shifted to the right and the same procedure is repeated. This makes the STFT a function of both frequency and time.

The following is the definition of the STFT:

$$STFT?f,t'??\ ?\ ?f,t'??\ \overset{?}{\underset{??}{?}}x?t??w?t\ ?\ t'??e^{??i2?ft?}dt$$

The above equation shows that the STFT is a function of two variables, frequency f and time t'. Please note that there are two time related variables in the definition above, t and t'. The difference between them is that we use t' to move the window function and we integrate over t. The window function is the factor w(t) in the integrand. By subtracting the variable t' from t in w(t)'s argument, we move w(t) t' units to the right.

We will now illustrate the STFT by using the signal in the right plot of fig. 3.4, the one with a frequency of 1 before 0 and a frequency of 3 after (and including) zero.

As a window function we will use w(t) = 1 when -1 <= t <= 1 (and zero elsewhere). We will look at the integrand of the STFT at different times. We will fix the value of f to 1 because we know that the first part of the signal, x(t), has that frequency. fig. 3.5 below summarizes the "experiment".

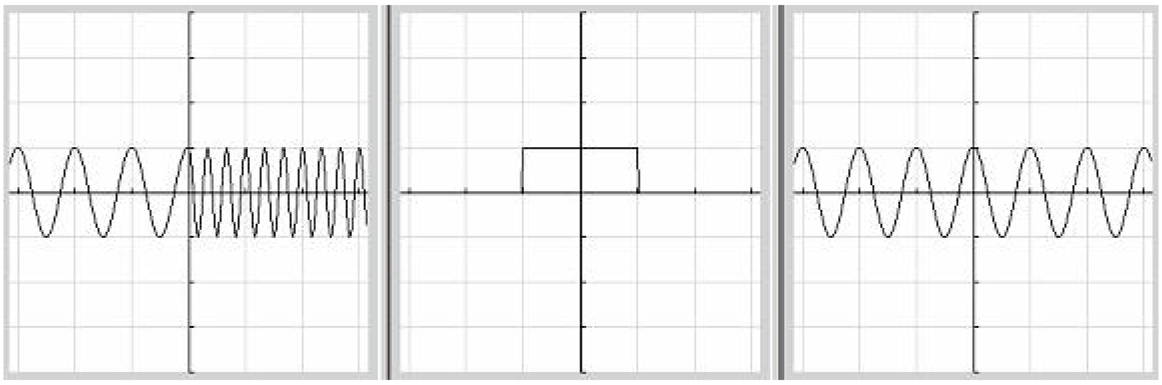


Fig. 3.5: From left to right: x(t), w(t) and the real part of e^(i2pft) at f = 1

The "cartoon" in fig. 3.6 shows how the window function w(t) moves as the value of t' increases. The left graph shows w(t-t') for t'= -1, the one in the middle shows w(t-t') for t'=0 and the right one shows w(t-t') for t'=1. When the window has crossed over to the region containing the frequency 3, the integral of the product should be (close to) zero.

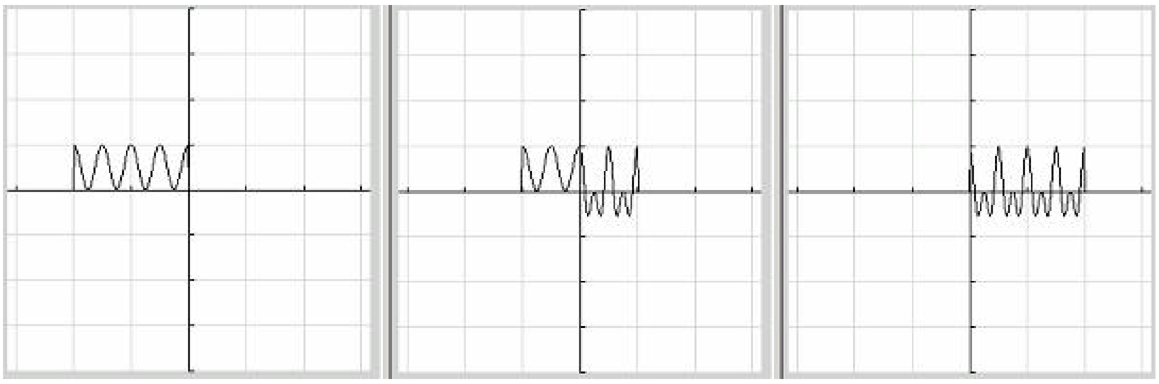


Fig. 3.6: The product of the functions in fig. 3.5 for t' = -1, 0 and 1.

We can see that in order to find the location of a frequency more accurately, we will have to reduce the width of the window. The problem with this is that the more we reduce this window, the less accurately are we able to distinguish the different frequencies. There is no remedy for this; it is a result of Heisenberg's uncertainty principle. But we can make some improvements to this technique by varying the size of the window. This is basically how the wavelet transformation works, and this is the topic of the next section.

3.4 Wavelets

3.4.1 Introduction

Wavelets are mathematical functions that cut data into different frequency components, and then study each component whit resolution matched to its scale. They have advantages over traditional Fourier methods in analysing physical situations where the signal contains discontinuities and sharp edges or spikes. Wavelets were developed independently in the fields of mathematics, quantum physics, electric engineering and seismic geology. [7]

During the last ten years interchanges between the above mentioned scientific fields has led to many new wavelet applications. As examples we can mention image compression, radar applications and earthquake prediction. Other fields that are taking advantages of wavelets are astronomy, acoustics, nuclear engineering, signal and image processing, neurophysiology, music, magnetic resonance imaging, optics, and pure mathematical applications.

The idea behind wavelets is to analyse according to scale. This idea however, is not new. Functions that satisfy certain requirements are called wavelets and can be used to represent other functions. Already in the early 1800's Joseph Fourier discovered that he could use sines and cosines to represent other functions. The problem with Fourier is that both sines and cosines stretch out from minus infinity to infinity (the functions are non local). They will therefore be of little use in approximating sharp edges. When using wavelets we avoid this problem because the mother wavelet used is contained in finite domains, and can be both scaled and translated. This is why wavelets is well suited for approximating edges or discontinuities.

The procedure is to adapt the mother wavelet, a prototype function. Then temporal analysis can be performed with a high frequency version, a version with a small scale, of the mother wavelet. A dilated (stretched) version is more appropriate for frequency analysis. If we further knew what wavelet was best adapted to our data, we could have our data sparsely represented. This is the reason why wavelets are a very good tool for data compression. [4]

3.4.2 Short history on wavelets

The work of what is today known as wavelets starts in the 1930's with efforts in different scientific fields. It did not at that time appear to be coherent theory. Scientists then had their ideas from the well known Fourier, and Fourier is said to have opened a door to a completely mew mathematical universe.

The first ever mention of wavelets occurred in an appendix to the thesis of A. Haar in 1909. At that time he, and other mathematicians, where led from the notion of frequency analysis to scale analysis by exploring the meaning of functions. As with other wavelets, the Haar wavelet has compact support, meaning that it vanishes outside a finite interval. The problem with the Haar wavelet however, is that it is not continuously differentiable.

Several groups worked independently in the 1930's on representations of functions using scale-varying functions. Some of the results suggested that energy might not be conserved, which scared scientists. However they did find a function that can vary in scale and conserve energy. And the effort of these groups provided David Marr, with an effective algorithm for numerical image processing using wavelet in the 1980's.

In 1980 wavelets was broadly defined in the context of quantum physics by Grossman and Morlet, and provided a way of thinking for wavelets based on physical intuition. Stephan Mallat later gave wavelets an additional jumpstart Trough his work with digital signal processing. As Y. Meyer was inspired by the results of Mallat, he constructed the first non-trivial wavelet which unlike the Haar wavelet is continuously differential, but does not have compact support. And then Ingrid Daubechies a few years later, based on Mallat's work, constructed a set of orthonormal basis functions, which have become the corner stone of wavelet applications today. [4]

3.4.3 Basis functions

To explain what a basis function really is we are going to use vectors. Since every two dimensional vector can be described by the two vectors (1, 0) and (0, 1) we call these the basis vectors for (x, y). This is because the sum of x multiplied by (1, 0), and y multiplied by (0, 1), is (x, y). In addition these basis vectors are orthogonal to each other.

Having explained the basis vectors lets imagine that instead of (x, y) we have a function f(x), e.g. a musical tone. By adding sines and cosines using amplitudes and frequencies we can construct this tone. If we in addition set the requirement that the sines and cosines must be orthogonal, meaning that the inner product adds up to zero, we have got our orthogonal basis function for this problem.

If we now have a signal over the domain from 0 to 1, we could divide it into two functions that range from 0 to ½ and from ½ to 1. And so on we can divide it into four step functions from 0 to ¼, ¼ to ½, ½ to ¾, and ¾ to 1. Like this we could go on forever. But the point is that each set of representation code the signal with a particular resolution or scale, and we then have our scale varying basis function. [4]

3.4.4 Continuous Wavelet Transform

To overcome the problem with resolution in short term Fourier transform (STFT) the continuous wavelet transform (CWT) were developed. Like in STFT the signal is multiplied with a function and the transform is computed separately for different segments of the time-domain signal. [1]

We think of a function f(x) in L^2(**R**) consisting of all square integrable functions with a real argument. This need of a square integrable function means that the function has finite energy. This includes the most functions of interest in practical use.

Such a function is for us desirable to split up so that we can find the single components that this function consists of. We wishes to see the function as a sum, or as an integral, of weighted single components. That meaning every component multiplied with respective factor or coefficient which tell how much of each component there is in the function. In the beginning we hope to do a transformation of the function to see if the transformed space can help us to analyse the function. Next we hope to be able to do the inverse transform to reconstruct the function, as a weighted sum of elements from the transformed space.

The Fourier transform is made by multiplying the function f(x) with $e^{-j*2*pi*u*x}$. It can then be shown that the function can be reconstructed by writing f as a sum where the components is weighted $e^{-j*2*pi*u*x}$ with the Fourier transformed values.

These ideas are used in wavelet transformation, but this time we exchange $e^{-j*2*pi*u*x}$ with a more general function **?** $_{a,b}$(x) where we use two indexes **a** (the dilation-index) and **b** (the translation-index). Index **a** is related to the frequency information, analogous to the Fourier transform, while **b** is the translation. Unlike sine and cosine, the basis functions for the Fourier transform, the basis function **?** $_{a,b}$(x) is not necessarily infinite. A function with compact support is needed to go one step further from Fourier, giving the possibility of time- and frequency localisation of detailed information in a signal. [3]

We can generate a doubly indexed family of wavelets [9] from ? by dilating and translating:

$$? \;_{a,b}(x) \; ? \; |a|^{?1/2} \; ? \; ?\frac{x \; ? \; b}{a}?$$

Where **a**, **b** ? *R* and **a** ? 0.

It is then clear that **a** will have a frequency like effect, and that **b** will have a translation effect. A low value of **a** will shrink the function in direction of x, while a higher value will stretch it out. We can then say that **?** $_{a,b}$(x) measures the frequency component 1/**a** of the function f(x) in the position x=**b**.

The function **?** is called the Mother Wavelet, because it allows us to generate all the functions **?** $_{a.b}$ by changing the parameters **a** and **b**. The coefficient $|a|^{-1/2}$ is used for normalisation. This will ensure that the energy is the same for all functions **?** a,b, meaning that the integral $|\mathbf{?}_{a,b}|^2$ is independent of **a** and **b**. It is also worth noticing that **?** it self is a special case of $\mathbf{?}_{a,b}$ namely where **a**=1 and **b**=0, ($\mathbf{?}_{1,0}$).

Until now we have not cared about special behaviours or limitations of these wavelets other than that they will help us rip apart the function f(x). We will see that if there should be an inverse wavelet transform it is a condition that C exists. C in turn is defined as square integral of the Fourier transformed of the wavelet-function divided by the absolute value of the frequency omega.

$$C_? \; ? \; ?_{??}^{?} \frac{|? \,(? \,)|^2}{|? \,|} d? \qquad 0 ? C_? \; ? \; ?$$

This is known as the “Admissibility condition”. Wavelets fulfilling this condition are called Basic Wavelets.

$$?_{??}^{?} \; ? \,(x)dx \; ? \; 0 \qquad x? \,(x),\; x\hat{?} \,(x) \; ? \; L^2(R)$$

Oscillation (Wave)

$$?_{??}^{?} |? \,(x)|^2 dx \; ? \; ? \qquad L^2(R) ? \{f : R ? \; C \mid ?_{??}^{?} |f(x)|^2 dx ? ?\}$$

Finite energy (Let)

Because of the admissibility condition our function will oscillate around the x-axis and the region above and under will be equal. Not surprisingly must our wavelet functions have the same finite energy as the function f(x), meaning that they have to be square integrable. [3]

3.4.5 Haar wavelet

As an example wavelet we will use the Haar wavelet, which must be said to be the simplest wavelet. Despite being simple it is still used. This function has the value 1 in the interval <0, ½] and the value -1 in the interval <½, 1], and 0 everywhere else.

$$? \,(x) \; ? \; ? _{1,0}(x) \; ? \; \begin{cases} 1 & 0 ? x ? \frac{1}{2} \\ ?1 & \frac{1}{2} ? x ? 1 \\ 0 & \text{otherwise} \end{cases}$$

We can easily see that this wavelet is in accordance with the abovementioned conditions.

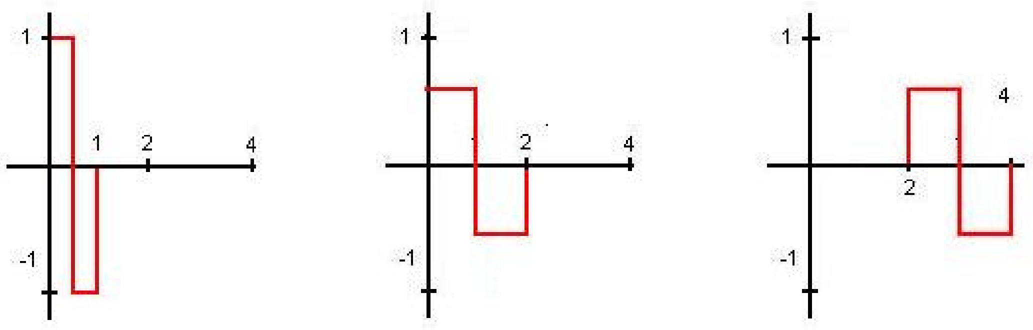


Fig 3.7a) **?** $_{1,0}$ *b)* **?** $_{2,0}$ *c)* **?** $_{2,1}$

If we compare the functions **?** $_{1,0}$ and **?** $_{2,0}$ we see that the latter is stretched compared to the first one. And at the same time the amplitude is less then in **?** $_{1,0}$. This is because of the factor $|\mathbf{a}|^{-1/2}$ in the definition of **?** $_{a,b}$. And if we look at the function **?** $_{2,1}$, we now see that it is translated on the x-axis.

We can now clearly see that this function has compact support and can be both scaled and translated. Because of the compact support of these kinds of functions they are well suited for determining where on the x-axis we can find interesting details on the function we are analysing. At the same time the ability to scale the function gives us the possibility to look for both small and large details as we whish. It can be compared to a binocular which you can move along, or closer or further away from, the object you are analysing. But we do also have an infinite range of wavelet functions giving us the opportunity to select the wavelet best suited for what kind of details we look for in the function to be analysed. [3]

3.4.6 Discrete Wavelet Transform

When using discrete wavelet transform we first set the indexes **a** and **b**, and keep the x as a continuous variable. Usually **a** and **b** is set to, **a** = a_0^{-m} and **b** = $n*b_0a_0^{-m}$, where m is an integer. Here we have several options but the most used is $a_0 = 2$ and $b_0 = 1$. That meaning $a_0 = 2^{-m}$ and $b_0 = n2^{-m}$, m still being an integer.

Such a way of fixing of **a** and **b** is called binary dilation ($a_0=2^{-m}$) and dyadic translation ($b_0=n2^{-m}$). By using the factor 2 in this case is very useful and effective in terms of algorithms and computers. With **a** and **b** set like this we get the following expression for our wavelets: [3]

$$? _{m,n}(x) ? 2^{m/2}? \, 2^m x ? n ?$$

We will not give a more thoroughly discussion on discrete wavelet transformation since we will not use it in our project.

4 Wavelets and edge sharpening

4.1 Overview

The main motive behind this master thesis is the need to improve processing of ultrasound images. A major advantage of ultrasound images over e.g. magnetic resonance imagery (MRI) is that ultrasound images can be viewed in real time. Unlike X-rays, long exposure to ultrasounds is not hazardous, and it would be a great benefit if the use of ultrasound could replace some of the X-ray usage that is necessary today.

There are problems with ultrasound images that make it inadequate for many purposes. Ultrasound images are filled with white speckles of noise that often make it hard to visually interpret the image, and especially hard to digitally process the image. The main focus of this thesis is the on ultrasound images of blood vessels for use in surgery. The primary mission is to locate the edge of the vessel so that it can be sharpened for better visual clarity. Because noise in the image might pose a great challenge to the techniques investigated in this project, a secondary mission will be find ways of overcoming this problem. However, the filtering of noise is not the primary goal of this thesis.

4.2 A basis for evaluating edge sharpening methods

In order to evaluate edge sharpening methods we will have to define what we mean by “sharpening” edges. There are two possible problems with edges of blood vessels in ultrasound images that we hope the methods implemented in this project one day will solve. One problem is that the edge of a blood vessel may look wider on an ultrasound image than what it really is. This effect is caused by dross on the wall of the vessel. The other problem is that the edge may be hard to see because it is dark. These problems lead to two interpretations of “sharpening” that we will use in this project; one is to make the edge narrower and the other is to increase the contrast between the edge and the background. By narrowing the edge we will hope to make the location of the edge in the image more accurate with respect to the real location of the edge. Of course, finding the exact location of the real edge would be preferable, but we are not that optimistic in this project. Our goal is limited to finding the edge in the image, and then to make it narrower and brighter. It is needless to say that if we are able to find the two sides of the edge in the image, then we have come a long way in achieving our goal.

Just so there is no doubt about what our plan is, fig. 4.1 below tries to illustrate the nature of our mission.

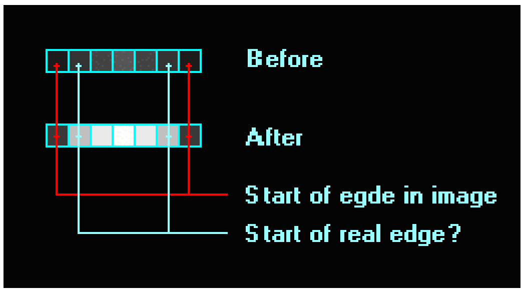


Fig. 4.1: Our goal

We will try to find the start of the edge in the *image*. Finding the *real* edge is considered to be too ambitious in this project. We can then narrow the edge as we wish. In addition we will try to make the edge look brighter, like in the pixel row labelled "After" in fig. 4.1. That is, "after" it has been transformed. We should mention a second possible way of making the edge look sharper, and that is if we somehow transform the edge to be slimmer instead of explicitly guessing the new width based on retrieved information. This possibility does not necessary imply that the edge after transformation is equal to the real edge of the blood vessel. It may be just another way to make it slimmer.

4.3 Edge detection with gradient operator

Human vision is very sensitive when it comes to sharp edges or other details in an image. These details in images are located in the high frequencies. To detect these edges it is possible to use the gradient operator. The gradient G is a vector of two elements, namely G_x in the width direction and G_y in the height direction. From vector theory we then have that the magnitude (G_m) and direction (?) is:

$$G = [G_x G_y] \qquad G_m = \sqrt{G_x^2 + G_y^2} \qquad \theta = tan^{-1}(\frac{G_y}{G_x})$$

The gradient is then calculated in the x- and y direction before the results are combined to give the final result. By using Euclidean distances, or absolute differencec, the partial results each give an approximation to the real gradient. [6]

Al though the gradient operator is a simple and effective tool in the field of edge detection and sharpening it becomes completely useless when dealing with ultrasound images. This has been tested by scientists at SINTEF Unimed in Trondheim, and the results were definitely not good. The use of gradient operators stood no change against the problem with the noise in ultrasound imiges. This method for edge sharpening was therefore considered inappropriate for use with ultrasound images, and therefore abandoned. But because of its simplicity it will be very well suited for less critical applications.

Fig. 4.2 below shows an example of edge detection using the Sobel gradient operator.

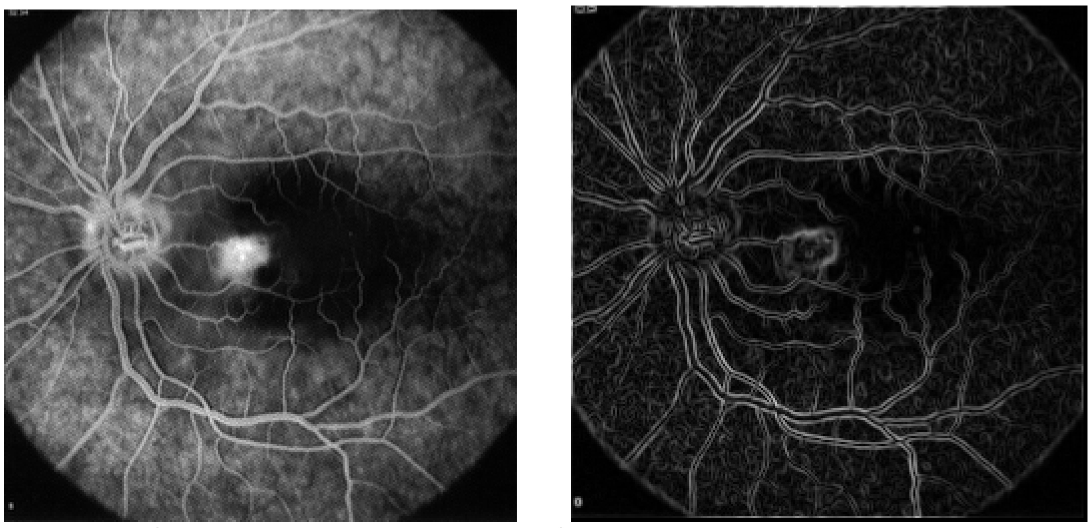


Fig 4.2: Left: Original image Right: Transformed image (Source[6])

Instead of gradient operators we will use the relatively new mathematical method wavelet transform. As mentioned in chapter 3 we have both continuous and discrete wavelet transformation, however in this thesis we will concentrate on using the continuous wavelet transform (CWT).

5 The EdgeX application

5.1 Overview

To conduct the experiments needed in this project a testing environment was needed. When deciding how to create this environment two options were considered, one was writing a testing tool in C++ and the other was writing the testing tool in Java. If the speed of the algorithms had been of high priority, then C++ would have been the best choice. Programs written in Java are a bit slower that programs written in traditional languages like C and C++. The reason for this is that Java programs execute on a layer above the operating system called the Java Virtual Machine (JVM). This has the advantage that a Java program can run on any OS without being recompiled provided there is a Java Runtime Environment (JRE) available for that OS.

The choice fell on Java because we were most experienced with this language. The application that was made was called EdgeX, in lack of a better name, and this chapter is dedicated to describing this application. Section 5.2 will give a brief introduction to EdgeX and give simple illustrations of its usage. Section 5.3 will be more concerned with the design, or architecture, of the application. Finally section 5.4 will give more detailed information about how the image processing techniques of EdgeX are implemented.

5.2 Introduction to the EdgeX application

EdgeX is the prototype that was made to realize the image processing techniques that were tested in this project. This prototype is written entirely in the Java programming language. The classes that make up the EdgeX program can be divided into two main categories, those that are part of the main application and those that implement the image processing techniques. The image processing techniques are implemented by separate modules that are loaded and started at runtime. The main class of the application is a class called EdgeX that acts as a framework for implementing and running these modules. The EdgeX class and a few other classes control the main user interface of the application.

EdgeX is a program with a multiple document interface (MDI), also known as a desktop. This means that the user interface of EdgeX consists of a main window that contains other windows inside it. Initially, when EdgeX is started, there is only one window inside the main window. This window is used to open and display images from files. This window will be present during the entire execution of the EdgeX program, and the user cannot close it. At the time of writing, only two image formats are supported by EdgeX, GIF and JPEG. However, support for other image formats can easily be added if necessary.

Fig. 5.1 below shows what the user interface of the EdgeX application looks like.

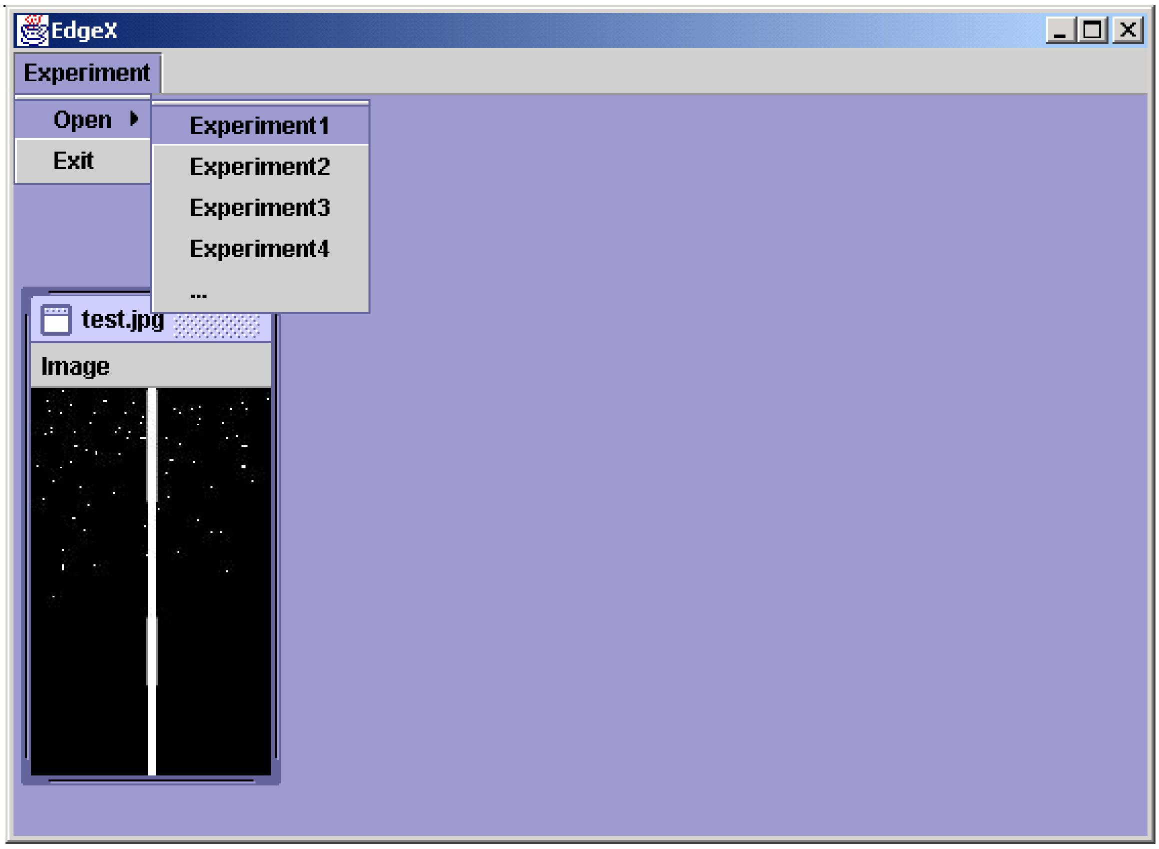


Fig. 5.1: The EdgeX application

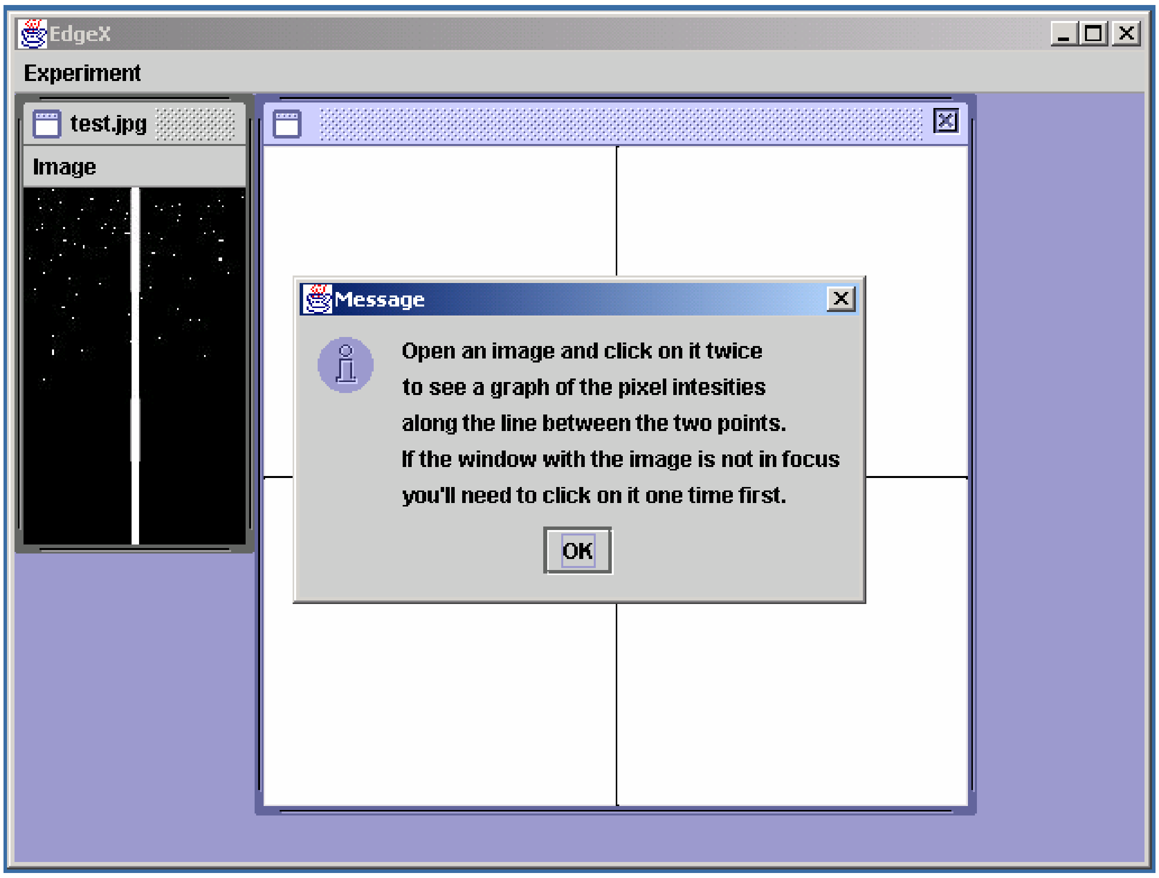
Fig. 5.1 shows the user interface of EdgeX after an image has been opened. The image contains lots of white dots and a vertical white line. This is one of the images that was used to test the EdgeX application. The name of the image file, test.jpg, is displayed at the top of the window containing the image.

In order to process the image that has been opened, an experiment must also be loaded. As shown in fig. 5.1 there is a menu labelled “Experiment” that contains a list of names of experiments that can be loaded. In fig. 5.1 this list contains the options “Experiment1”, “Experiment2”, etc. These particular names are just examples; the real names of the experiments must have a very specific format in order for EdgeX to find the corresponding class files. The name of an experiment is the fully qualified class name of a class implementing a Java interface called “Experiment”. When EdgeX starts, it reads all the names from a simple text file, edgex.ini, and adds the names to the menu. The names must be manually added to the edgex.ini file. If any of the names are invalid, clicking on the name will cause an error message to pop up on the screen.

At the time of writing the edgex.ini file contains the following lines:

```
no.hia.edgex.exps.pxlgraph.PixelGraph
no.hia.edgex.exps.cwt.CWTExperiment
no.hia.edgex.exps.mexhat3d.MH3DExperiment
```

These lines will therefore be options in the experiment menu. The experiment at the top of the list, PixelGraph, is the first “experiment” that was made. The reason for the quotes is that PixelGraph is not a real experiment; it is a demonstration of how experiments can be implemented. Fig. 5.2 below shows what happens if the no.hia.edgex.exps.pxlgraph.PixelGraph name is selected from the experiment menu.



Open an image and click on it twice
to see a graph of the pixel intesities
along the line between the two points.
If the window with the image is not in focus
you'll need to click on it one time first.

Fig. 5.2: The PixelGraph experiment

Now there are two extra windows in EdgeX's main window. The one with a white background displays a coordinate system that is used to plot graphs of mathematical functions. The window with the title "Message" is a modal dialog showing simple instructions on how to use the PixelGraph experiment. When a window is modal it means that it has to be closed before other windows can gain focus and get input (i.e. click Ok to proceed).

The dialog in fig. 5.2 instructs the user to click on the image twice, and fig. 5.3 below shows what happens when this is done.

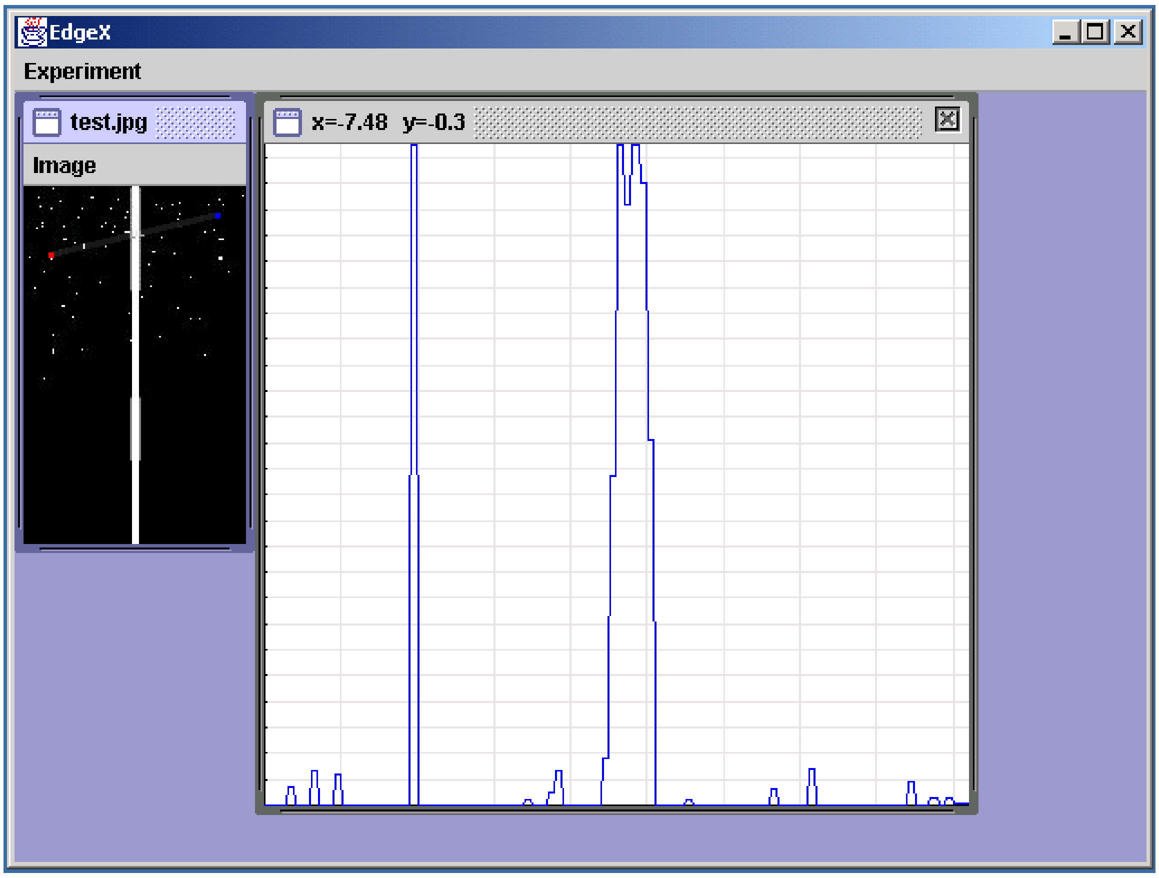


Fig. 5.3: The graph of pixel intensity values along a line

In fig. 5.3 the image test.jpg has two dots, one red and one blue, showing were the mouse clicks occurred. Between the dots is a barely visible transparent line. This line shows approximately were the pixel values of the image are sampled. The coordinate system to the right of the image has a graph of the pixel values as a function of the position on the line. The wide column in the middle of the graph corresponds to the vertical white line in the middle of the image. The x and y values at the top of the graph window show the logical position of the mouse cursor as it hovers over the white area.

The PixelGraph experiment does not implement any image processing techniques, but it illustrates how a typical experiment works. In fact it was made as simple as possible so that other people can use PixelGraph as a reference on how to make their own experiments to be used with EdgeX.

The use case diagram in fig. 5.4 summarises the basic procedures for operating the EdgeX program.

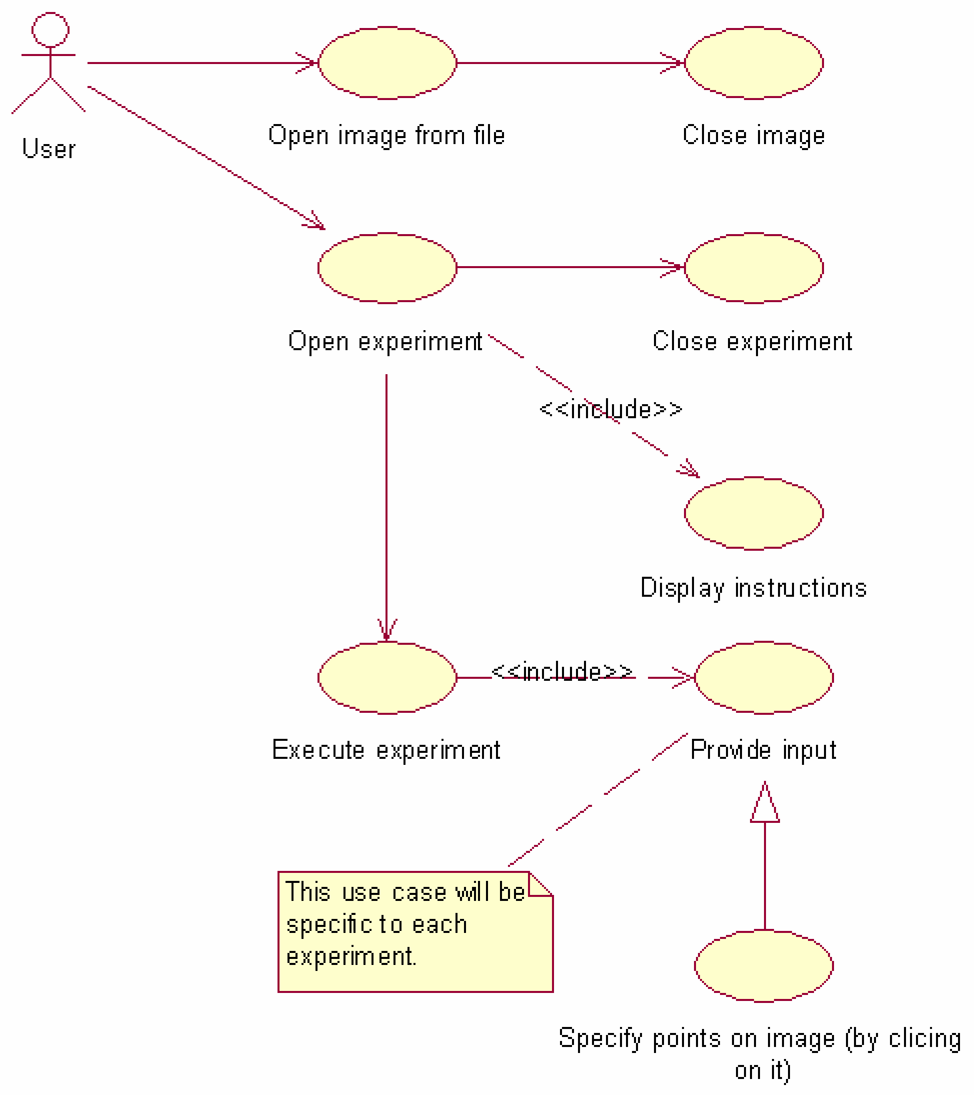


Fig. 5.4: Use case diagram illustrating how the user operates EdgeX

5.3 Application design

The EdgeX prototype is made up of three main Java packages that represent logical groupings of the classes of EdgeX. These packages are:

? *no.hia.edgex:* This is the main package containing the core classes. These classes are considered to be the framework on which the experiments are implemented and executed.

? *no.hia.edgex.util:* This is a sub-package of no.hia.edgex. This package contains various utility classes.

? *no.hia.edgex.exps:* This is a sub-package of no.hia.edgex. The packages containing the experiment implementations were put here for convenience as they were implemented.

The class that is used to start the EdgeX application is a class called (naturally) “EdgeX”. This is located in the no.hia.edgex package along with a few other classes and interfaces. These classes make up what is considered to be the core framework of the application. This framework acts as a layer between the user and the experiments, and all input and output go via the core classes. When the user decides to open an experiment, the experiment is given access to the API of the framework. The purpose of the framework is only to systematize the task of implementing and adding experiments to the project, so the API is limited to the minimum requirements of the experiments.

The use case diagram in Fig. 5.5 illustrates the EdgeX framework’s functionality as seen from the experiment’s point of view.

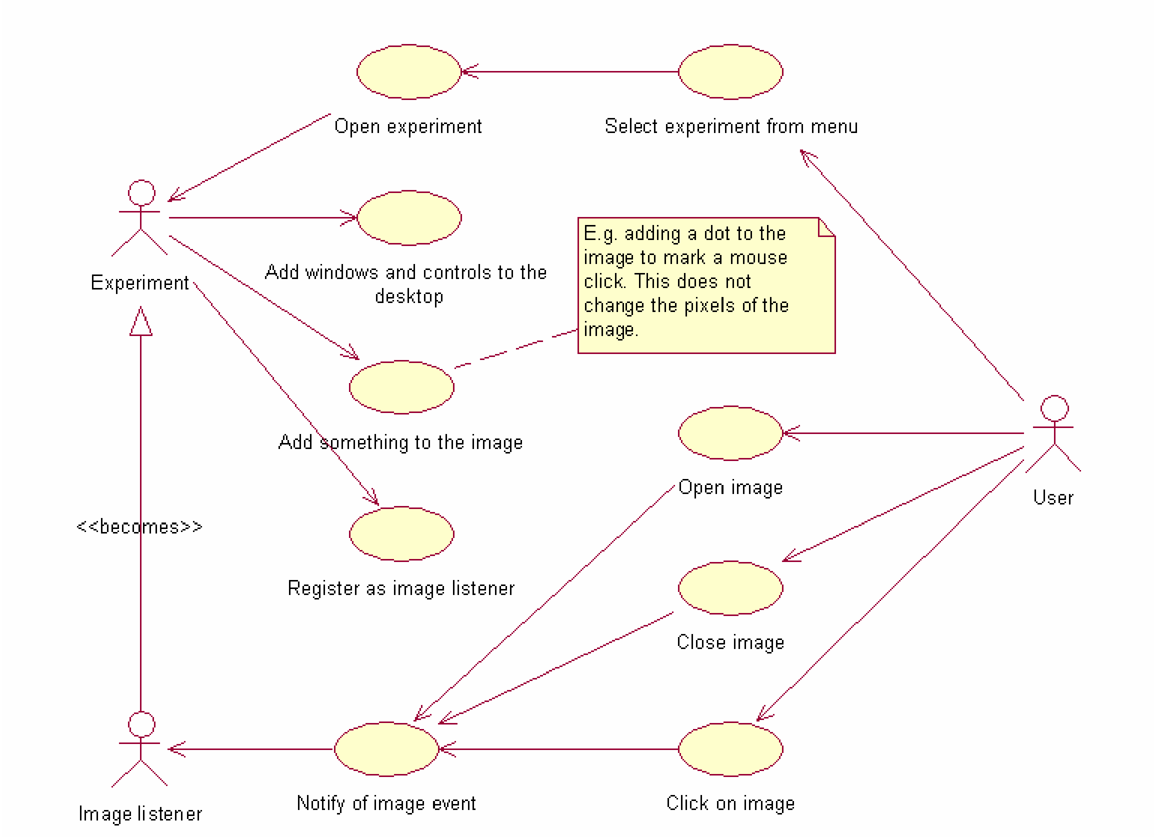

Open experiment
Select experiment from menu
Experiment
Add windows and controls to the desktop
E.g. adding a dot to the image to mark a mouse click. This does not change the pixels of the image.
Add something to the image
Open image
User
<<becomes>>
Register as image listener
Close image
Image listener
Notify of image event
Click on image

Fig. 5.5: The Functionality required by the experiments

In fig. 5.5 the basic needs of an experiment are boiled down to only a few points. The experiment needs to be notified when the user has opened, closed and clicked on the image. When the image is being clicked, both the image itself and the point where the click occurred are passed to the experiment as event information. The experiment may also want to paint on the image without corrupting the image data, e.g. to mark the spots on the image where the image was clicked.

From the experiment’s perspective the system is a desktop application containing windows. The experiment might need to add its own windows to the desktop allowing the user to control the experiment. The user might for example be required to select a wavelet to be used as base function for a wavelet transform. It is up to the experiment to decide how the user can interact with the experiment and how the experiment can be closed.

The class diagram in fig. 5.6 presents a view of the EdgeX application from the specification perspective [12].

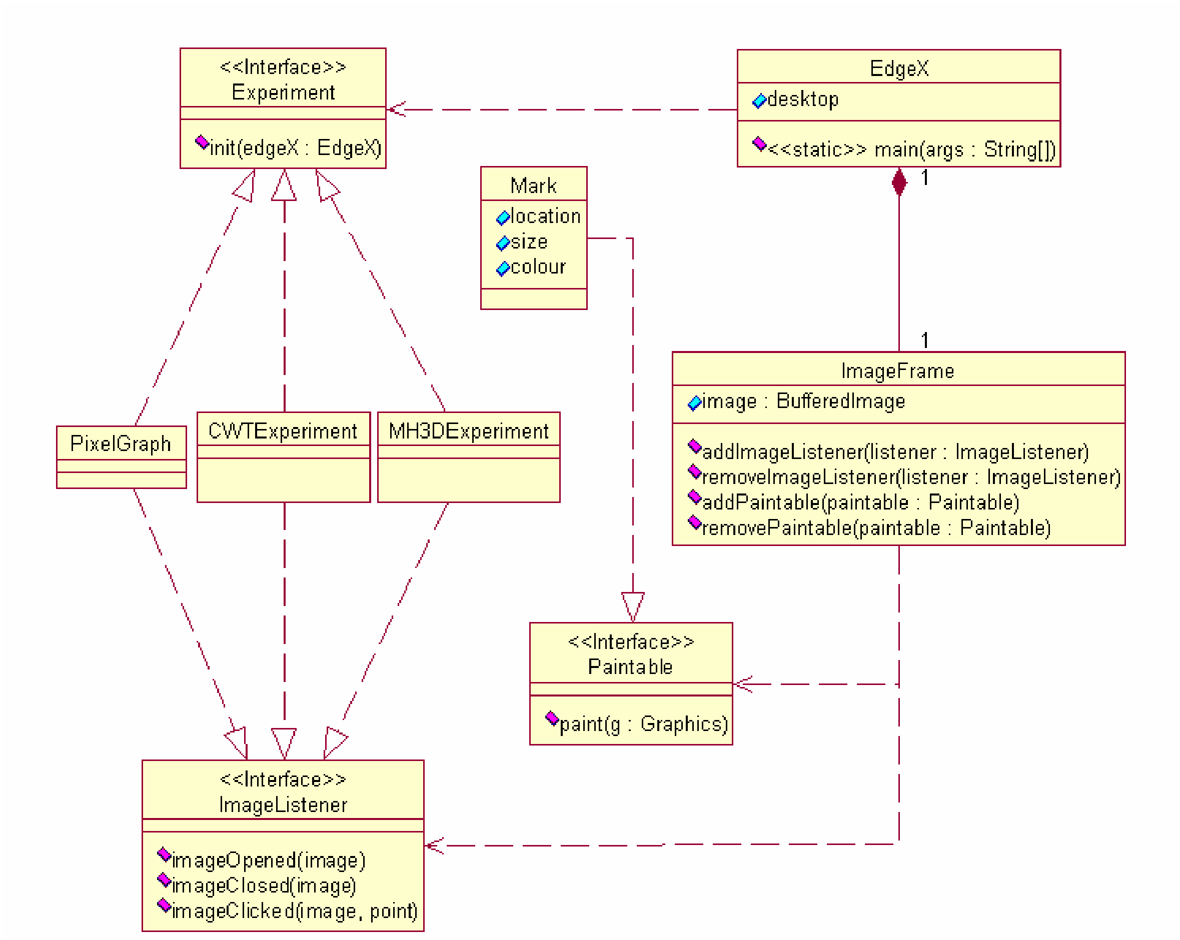


Fig. 5.6: Class diagram of the EdgeX application

The main purpose of the diagram in fig. 5.6 is to capture the essence of EdgeX's design by showing the various interfaces and the dependencies of them. The EdgeX class is shown to have a desktop. This desktop represents the MDI of the EdgeX application, and other classes interface with the desktop in order to add and remove windows to the main frame. The static main function in the EdgeX class is necessary in order to start the EdgeX program (EdgeX is, in other words, the "start up" class).

Fig. 5.6 shows that there are three implementations of the Experiment interface. The first one, PixelGraph, has already been introduced. The next section, Implementation, will give a more thorough description of the other two experiments.

5.4 Implementation

5.4.1 Introduction

In order to act as an experiment framework the EdgeX application utilizes a very convenient feature of the Java programming language, namely the ability to load classes at runtime. The EdgeX prototype was specifically written to accommodate extension with more experiments without necessarily having to edit and recompile source code (although this is also possible). In order to make an experiment one simply needs to supply a class that implements the no.hia.edgex.Experiment interface. This interface allows the EdgeX program to dynamically load and initialize the experiment. In this way the EdgeX framework is similar to a Servlet container (except for Edgex's simplicity), only that it runs Experiment objects instead of Servlet objects. The Experiment interface is shown in listing 5.1 below.

Listing 5.1: The no.hia.edgex.Experiment interface

```
package no.hia.edgex;

public interface Experiment extends java.io.Serializable
{
     public abstract void init(EdgeX edgeX);
}
```

The Experiment interface has only one method, init. This method is invoked when the experiment is loaded, and it gives the experiment access to the EdgeX class. The most important methods of the EdgeX class are:

? getDesktop()
? getImageFrame()
? showMessage()

The first method returns a javax.swing.JDesktopPane that allows the experiment to add its own frames to the desktop. The third method, showMessage(), is used to display modal dialogs with messages (e.g. error messages and instructions). The second method, getImageFrame(), returns an instance of no.hia.edgex.ImageFrame. This is the frame that lets the user load and view images from file. The ImageFrame object allows the experiment to access the currently loaded image, to place marks on the image and to register itself as listener to image events. The events that might be of interest to the experiment is the opening and closing of images, and when the user clicks on the image.

The EdgeX prototype contains, in addition to PixelGraph (described earlier), two experiments that implement image processing techniques. The first one, CWTExperiment, tests how CWT can be used on a line of pixels to detect edges. The idea is to use this technique on lines that cut through blood vessels in ultrasound images, and to try to make the edges of these blood vessels sharper. The other technique that was implemented is similar to CWTExperiment in that it computes a CWT of pixel values. However, this experiment uses a three dimensional Mexican hat as mother wavelet for the CWT, so the CWT will be of an area of pixels rather than a line.

The next two sections describe in more detail the implementation of these two slightly different techniques.

5.4.2 The CWT experiment

The first edge detection scheme that was tested is implemented by an experiment called CWTExperiment. This experiment computes a CWT of a line of pixel and plots the graph of the CWT for selected scales. Fig. 5.7 below shows how this experiment works.

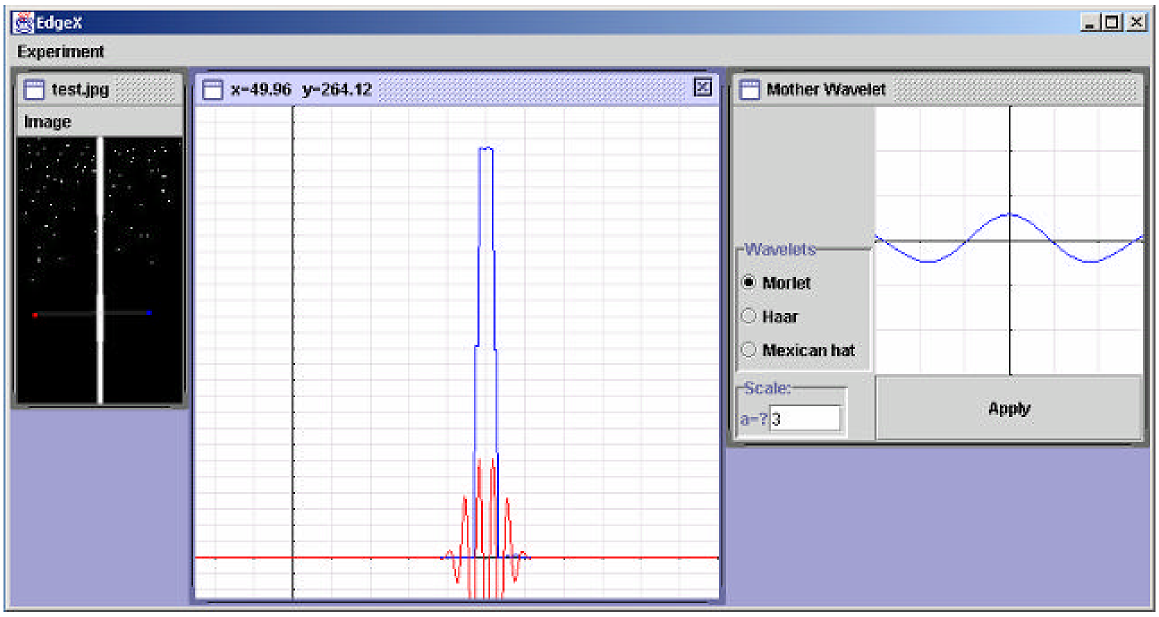


Fig. 5.7: The no.hia.edgex.exps.cwt.CWTExperiment

The leftmost window contains the image used to mark the pixel line of the CWT. Two dots (one red and one blue) and a transparent line show where the user has clicked on this image. The resulting line of pixels is plotted in the coordinate system to the right of the image, together with its CWT. The tall blue column in the graph is the white edge intersecting the line in the image. The red oscillating graph is the CWT of the pixel line. The rightmost window allows the user to select a mother wavelet to be used as base function for the CWT. This window also contains a coordinate system that plots the graph of the selected wavelet. An input box at the left bottom of the window allows the user to specify the scale of the wavelet. In fig. 5.7 the scale is set to 3. With its scale parameter fixed to a predefined value the CWT becomes a function of one variable, namely the variable *b* – the translation parameter. Hence, it can be plotted in a 2D coordinate system along with the pixel graph itself.

In order to compute a CWT of the pixel line CWTExperiment uses a contiguous function representation of both the pixel line and the wavelet. An interface called

MathFunction provides the necessary protocol for doing so. This interface is shown in listing 5.2 below.

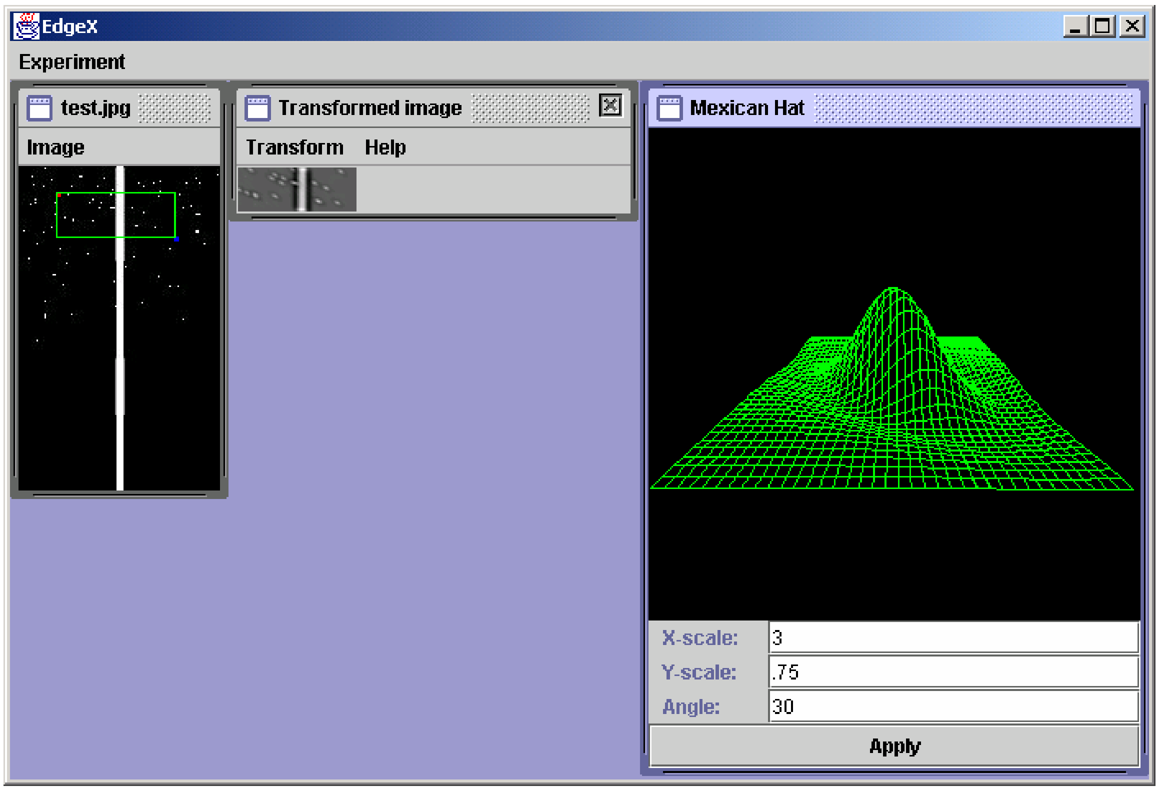
Listing 5.2: The no.hia.edgex.util.MathFunction interface
```
package no.hia.edgex.util;

public interface MathFunction extends
java.io.Serializable {
     public int numberOfVariables();
     public double getValue(double[] params);
}
```

This interface is pretty straight forward. The first method, numberOfVariables, returns the function's number of independent variables. The second method, getValue, accepts an array of independent variables and returns the dependent variable.

When the user selects two points on the image, this will trigger an update of the pixel graph and the CWT graph. When this happens a MathFunction representation of the pixel line will be obtained, and an object of a class called CWT will be created using the pixel line and the currently selected wavelet. The CWT class is a MathFunction that has two independent variables, scale and translation. When the getValue method of the CWT class is called, a new MathFunction is created to represent the product of the wavelet and the pixel line. A definite integral is then evaluated numerically for this product. This integral is approximated using the Simpson's rule [15]. The Simpson's rule divides the integration interval into a certain number of subintervals and calculates the area between the graph and the horizontal axis for each subinterval. The integral is then approximated by a weighted sum of all the areas. The number of subintervals used in the approximation determines both the speed and the accuracy of the computation; the more subintervals, the higher the accuracy and slower the computation. The CWT class divides the interval covered by the pixel line into 1000 subintervals, and based on experience we will say this is a good compromise between speed and accuracy.

5.4.3 The 3D Mexican hat experiment

By the time CWTExperiment was finished we had been requested to try out a similar technique using 3D Mexican hat as mother wavelet. Fig. 5.8 below shows the no.hia.edgex.exps.mexhat3d.MH3DExperiment. Following is a brief description of how this experiment works.



X-scale: 3
Y-scale: .75
Angle: 30

Fig. 5.8: The no.hia.edgex.exps.mexhat3d.MH3DExperiment

The MH3DExperiment is operated in a similar way to the CWTExperiment. The user clicks on an image in two places to trigger a new CWT computation. However, in this experiment the clicks are used to specify an area instead of a line. The rightmost window in fig. 5.8 allows the user to control the mother wavelet used in the transform. In this experiment there is (at the time of writing) only one choice of mother wavelet, the 3D Mexican hat. Three input fields allow the user to enter the scale in x- and y direction, plus the rotation of the Mexican hat. In fig. 5.8 the x-scale is set to 3, y-scale to 0.75 and the rotation to 30 degrees. The transform itself is displayed in the window in the middle.

There are many similarities between the CWTExperiment and the MH3DExperiment. Both experiments present the transform visually by fixing the scale parameter(s) to a predefined value. In the MH3DExperiment, however, there are two translation parameters. So if a graph were to be plotted of the transform it would be 3D (like in the rightmost window in fig. 5.8) instead of 2D. Instead of plotting a graph of the transform, MH3DExperiment uses the rectangle selected by the user to create a new image that represents the transform. This image is shown in the window in the middle in fig. 5.8, and it has the same dimensions as the selected area.

Just like the CWTExperiment, the MH3DExperiment uses a special class implementing the MathFunction interface to represent the transform. This class, MH3DTransform, is a function of 4 variables; the x-scale, y-scale, x-translation and y-translation (the rotation of the Mexican hat is specified in the constructor). These parameters will in the following be denoted by *ax*, *ay*, *bx* and *by*, respectively. For a given *ax*, *ay*, *bx* and *by* the MH3DTransform uses two classes to calculate the value of the transform. One of these two classes is a MathFunction representation of the selected portion of the image. This works the same way as CWTExperiment's pixel line; two variables is needed to specify a position in the rectangle (with origo at the left bottom), and the value at this point is a number between 0 and 255 depending on the shade/colour of the pixel. MH3DExperiments's equivalent to CWTExperiment's wavelets is a class called MH3D. This is a MathFunction representing the 3D Mexican hat. It is given the *ax*, *ay*, *bx*, *by* and rotation parameters at construction time and its two independent variables specify a position in the horizontal plane.

There is one significant difference between the MH3DTransform class and the CWT class. When the getValue method of the CWT class is called, a single integral of the product between the pixel line and the wavelet is computed. The getValue method of the MH3DTransform class needs to compute a double integral because both the wavelet (Mexican hat) and the pixel function are in 3D. This double integral is estimated by sampling the integrand at every pixel of the selected area, and then adding the samples together. The new transformed image (the middle window in fig. 5.8) is obtained by fixing the *ax* and *ay* parameters of the MH3DTransform and sampling the transform at every pixel of the selected area (i.e. varying the *bx* and *by* parameters for every pixel in the area). During this sampling the minimum and maximum sampled values are retrieved. The shades of the pixels in the transformed image are then calculated according to the following formula:

Shade at pixel (x, y) = (MH3DTransform.getValue(x, y) – min) * 255 / (max – min)

where MH3DTransform.getValue(x, y) is the value of the transform at (bx = x, by = y) and min and max are the minimum and maximum of the sampled values. The resulting shade is a number in the interval [0, 255]. This number is a mapping from the set of real numbers in the interval [min, max] to a pixel value.

We will close this section by issuing a fair warning: The computation of the CWT using 3D Mexican hat is extremely slow. It is recommended that the areas selected to be transformed are no larger than 650 square pixels. But if you should use it for larger areas, you might want to get a newspaper and a cup of coffee.

6 Test results

6.1 Overview

The objective of this project was to test techniques that can be used to sharpen edges in ultrasound images of blood vessels, in particular using wavelet transforms. The strategy was to implement the transforms in Java in order to test the techniques on various self-made images containing edges. The resulting Java program, EdgeX, was described in chapter 5 along with its CWT implementations. Now that the EdgeX program has been made, anyone can use it to experiment with wavelets and image processing. There are countless tests that can be performed simply by selecting various parameters for the mother wavelets. The application was also made with the intension in mind that it should be extended and improved in the future, perhaps to test new radical techniques.

We are sorry to say that the making of EdgeX took longer than initially planned, so there was little time to test the methods that were implemented. We will have to place most of the responsibility for this testing onto others. However, the next section presents the results of a few preliminary tests obtained by use of the EdgeX application. These tests are chosen to make a few points that will be discussed in chapter 7. Section 6.3 makes a few observations about the results that will be relevant in the discussion. It should be noted, however, that our efforts so far have resulted in a program that can be used as a tool for further testing. The results revealed in this chapter should be taken as a curiosity rather than a basis for drawing conclusions about the methods being tested.

6.2 The results

Fig. 6.1 shows the image that was used in the tests that produced the results presented in this chapter.

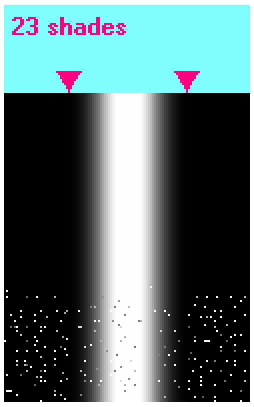


Fig. 6.1: The test image

The vertical white line in the image is supposed to be the edge of a blood vessel in an ultrasound image. The image is divided into two main sections. The lower section contains lots of white dots. These dots simulate noise in an ultrasound image. The upper section is noise free. The edge is symmetrical on each side and has a total width of 50 pixels. Four columns in the middle have the brightest pixels with a shade of 255 (pure white). The transition from black to white uses 23 shades of grey, exclusive pure black and white. Two rectangles at the top mark the exact start of the transition on each side; here the shade of the pixels is one.

The first tests were made using CWTExperiment described in section 5.4.2. Fig. 6.2 below shows the pixel line used to compute the CWTs in table 6.1 – 6.4.

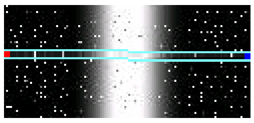


Fig. 6.2: The pixel line used to compute the CWTs in table 6.1 – 6.4

Table 6.1: CWTs with tall peaks

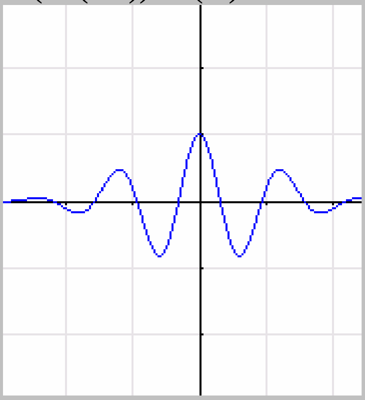
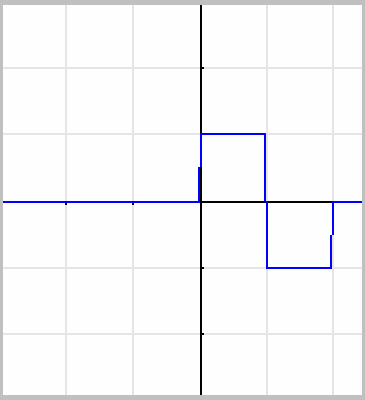
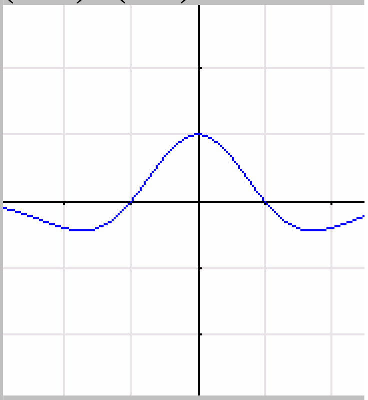
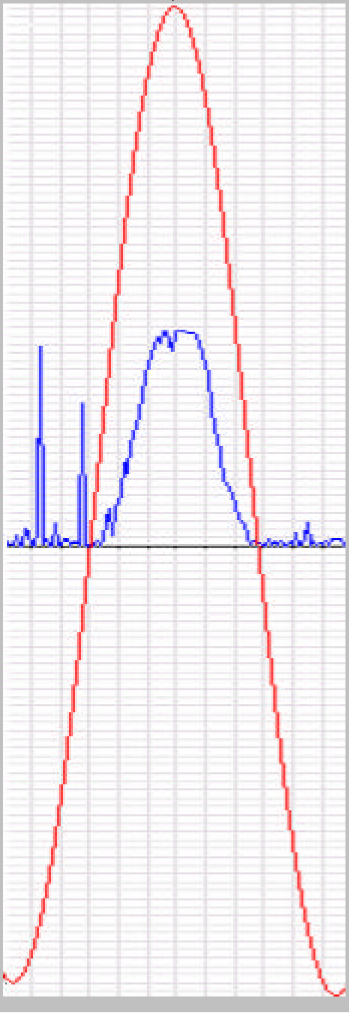
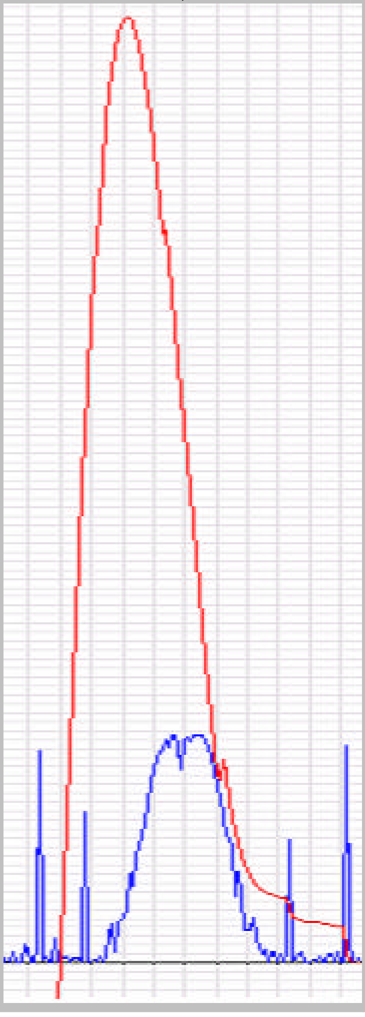
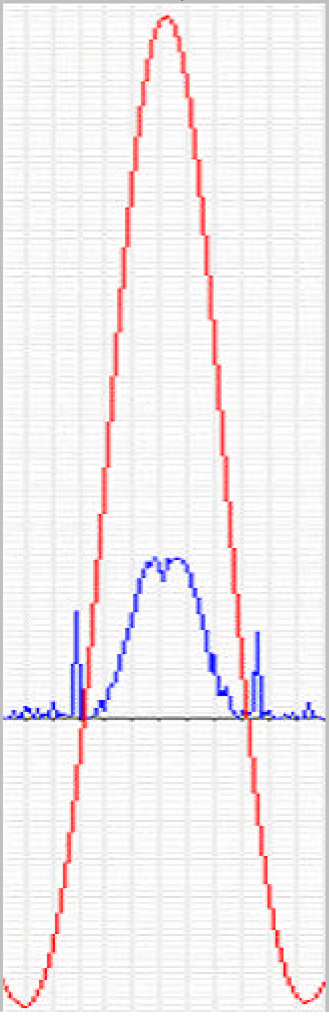
Morlet:	**Haar:**	**Mexican hat:**
e^(-.5(t^2))cos(5t):		(1 - t^2)e^(-.5t^2):
		
Transform t1, scale = 90:	Ttransform t2, scale = 40:	Ttransform t3, scale = 27:
		

Table 6.1 has three columns, each showing the result of a test. The mother wavelet used in a test is given in the header of the column. Beneath the header follows the mathematical definition of the mother wavelet and its graph. The bottom of a column shows the scale parameter that was applied to the mother wavelet followed by the graph of the CWT. The blue curves are the graphs of the pixel line. The red tall

curves are the CWTs. The scale parameters were chosen to give about the highest peaks possible for the transforms.

Table 6.2 – 6.4 show three other CWTs using the same wavelets as in table 6.1 with about half the scales applied to them.

Table 6.2: Transform t4

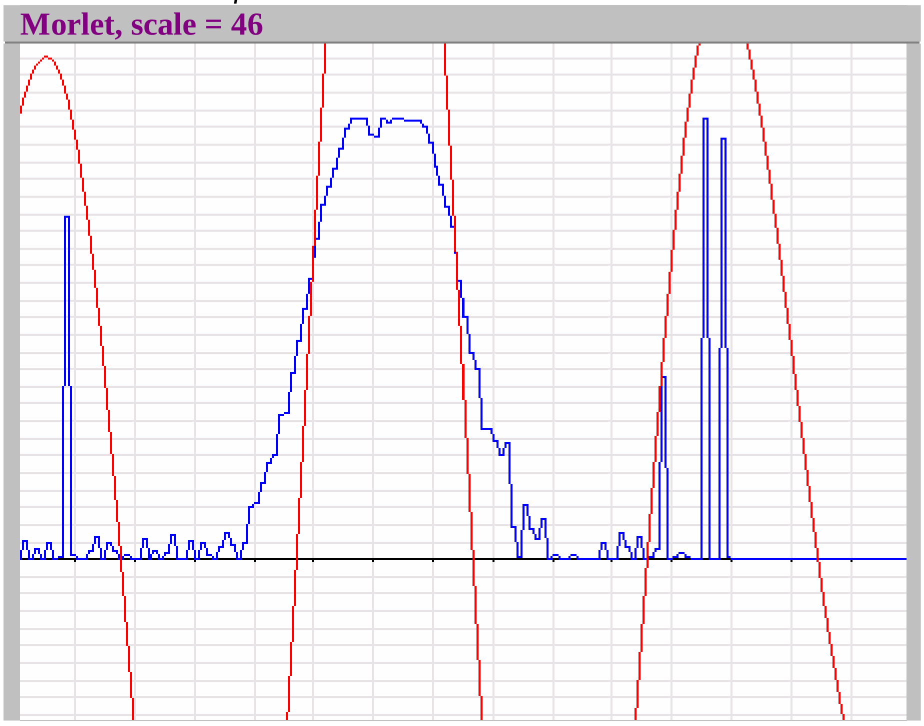


Table 6.3: Transform t5

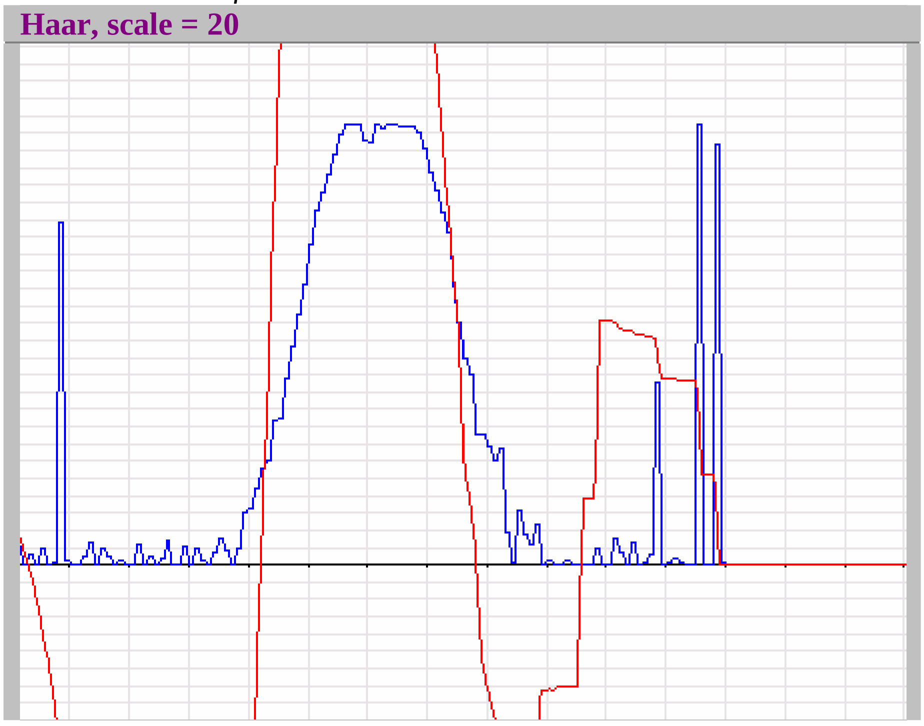
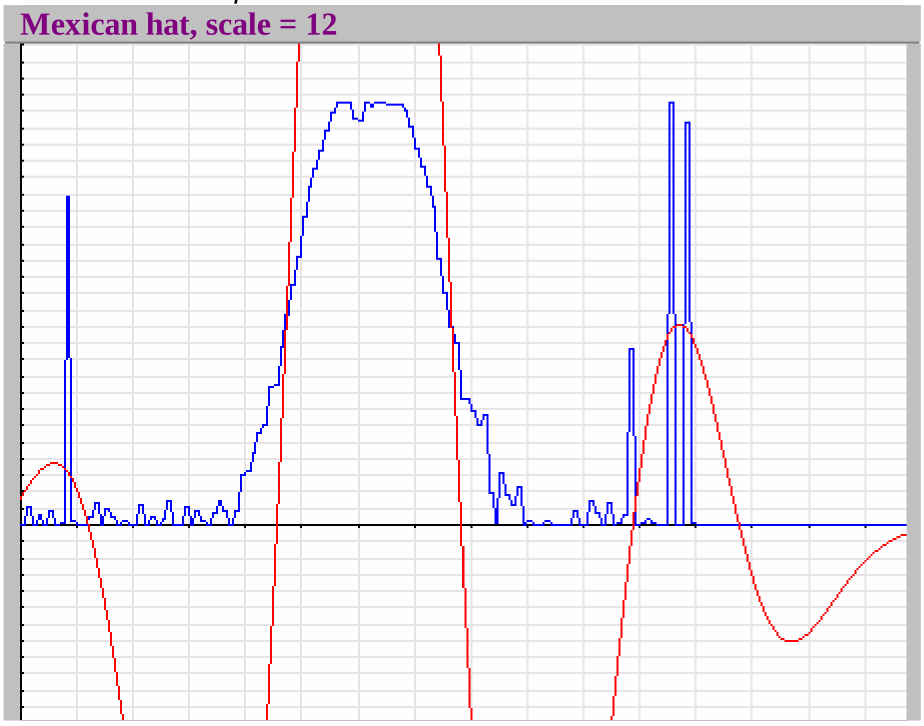


Table 6.4: Transform t6



The coordinate systems in table 6.2 – 6.4 have been zoomed in on the edge, so the top of the CWT graph is not visible.

The final tests were performed using MH3DExperiment described in section 5.4.3. These tests used the 3D Mexican hat wavelet defined in table 6.5.

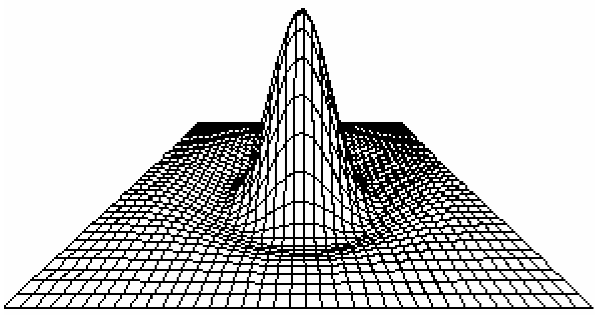
Table 6.5: The 3D Mexican hat mother wavelet

? $_{\mathbf{a},\mathbf{b}}$ = ([2p-((**r**-**b**)^T)P(**r**-**b**)]e^[½((**r**-**b**)^T)P(**r**-**b**)])/v(2|a$_x$a$_y$|)

r = (x, y), **a** = (a$_x$, a$_y$), **b** = (b$_x$, b$_y$),
R = [(cos ang, sin ang), (cos ang, -sin ang)], A = [(1/a$_x$, 0), (0, 1/a$_y$)] and P = (R^T)AR.

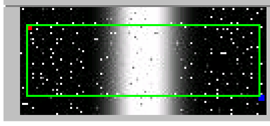
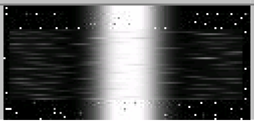
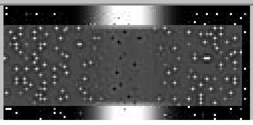
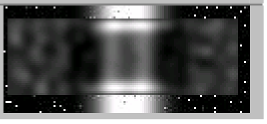
Notes:

- Letters in bold face are vectors.
- ^T means transpose.
- (a, b) is a vector with the entries a and b.
- [(a, b), (c, d)] is a matrix with two columns: the vectors (a, b) and (c, d).



Three tests were made using the 3D Mexican hat. Each of them had a different set of parameters to shape the Mexican hat. Table 6.6 below shows the resulting transforms and the parameters used to make them.

Table 6.6: CWTs using 3D Mexican hat

Original image:	**Transform t7:** **X-scale = 27** **Y-scale = 0.01**	**Transform t8:** **Y-scale = 0.75** **X-scale = 0.75**	**Transform t9:** **Y-scale = 12** **X-scale = 12**
			

The leftmost column in table 6.6 shows the original image. The green rectangle highlights the area of the image that was transformed. The next three columns show the same image after the area was transformed. The scales that were used are shown in the headers above the transformed images. None of the transforms used a rotation of Mexican hat.

6.3 A few notes on the results

The results given in this chapter are presented as 9 CWTs named t1 – t9. A few things should be noticed when looking at these CWTs.

The blue curves in table 6.1 – 6.4 are the graphs of the pixel line in fig. 6.2. The edge in the image in fig. 6.1 can be seen as a hill, or bump, in the middle of the blue graphs. This hill should be a lot smoother than it is, but artificial noise causes it to be somewhat irregular. On each side of the hill there are a few tall thin columns caused by this noise. The graphs in table 6.2 – 6.4 were zoomed in specifically to get a better view of how this noise affects the CWTs.

The CWTs in table 6.1 were made using scales that gave about the tallest peaks possible. Transforms t1 – t3 show that Morlet and Mexican hat produced CWTs with tall peaks right above the centre of the edge, while the peak of t2 failed to hit the centre. Both t1 and t3 appears to be relatively unaffected by the noise. t4 and t6 on the other hand, seems to be more influenced by the noise. The scales used by t4 – t6 are about half of the scales used by t1 – t3. We'll assert that the peaks of t4 – t6 are significantly smaller than the ones in table 6.1, but that t4 and t6 still find the centre of the edge.

The results in table 6.1 to 6.4 seem to indicate that, among those wavelets tested, Mexican hat gives the CWT that is least sensitive to noise and with the tallest peak above the centre of the edge. At this point there is little to be said about the results in table 6.6; the images speak for them selves. A further discussion of these results will be the topic of section 7.3.3.

7 Discussion

7.1 Overview

During the process of specifying the thesis definition for this project we had to choose between two main alternatives. One alternative was to focus mostly on the mathematical theory behind the methods that were going to be studied. The other alternative, the one we chose, was to make a prototype application to be used as a testing tool. The main goal of this project was to make a program that computed CWTs of pixel values in order to see if this could be used to sharpen edges in ultrasound images. Unfortunately the development of the EdgeX application took longer than anticipated, so by the time EdgeX was (more or less) finished we had almost no time left for testing. One of the most contributing factors to the overshoot of our time budget was an unexpected request to implement CWT using the 3D Mexican hat wavelet. It would have been awkward to modify the already existing

CWTExperiment because of the way it was implemented (using Simpson’s rule). We therefore decided to write MH3DExperiment from scratch.

The EdgeX application has now two CWT implementations ready to be used, one using wavelets of a single variable and one using a Mexican hat of 2 variables. We will leave it up to the experts to draw final conclusions about these methods and their usability. In this section we will try to get a first impression of the methods by looking at the results in chapter 6 in light of some of the theory in chapter 3. This will hopefully give us some clue as to whether these methods can be used for edge sharpening or not.

7.2 Preliminary discussion

When planning the work of this project a few limitations were made. First and foremost it was decided that speed was not an issue when implementing the transforms. The purpose of the EdgeX application is to show the effects of transforms rather than to use the transforms in practical situations. The methods tested by EdgeX should be optimized in the future after they have been perfected. In particular the 3D Mexican hat implementation requires a great deal of patients to use. One way to improve its performance would have been to use samples of the Mexican hat instead of frequently having to calculate its complex expression. However, we would then impose restrictions on the choice of parameters used to shape the Mexican hat.

Another limitation worth mentioning is that no ultrasound images were used in this project, either authentic or synthetic. These kinds of images were requested on a few occasions, but we were told to use simple home made images instead. This is not a big deal since testing the methods on real ultrasound images is simply a matter of using EdgeX to experiment with these images. However, the results described in chapter 6 have to be taken with a pinch of salt. For example, the thickness and brightness of an edge compared to the noise may have a significant impact on the CWT. The only way of testing the methods reliably is to use them on the images they are intended for.

7.3 Discussion of the results in chapter 6

7.3.1 Introduction

Chapter 6 displayed the CWTs of an edge using 2D Morlet, Haar and Mexican hat, plus 3D Mexican hat. In order to deduce anything from these results we have to define what the desired characteristics of a CWT should be. For the purpose of this discussion we'll name the following properties. The CWT should have a large negative value on each side of the edge, and the value should increase towards the centre of the edge, where it reaches its maximum. The graph of the CWT should intersect the horizontal axis close to each side of the edge. We also want noise in the image to have as little effect as possible on the transform. If the CWT of an ultrasound image fulfils these criteria for a given scale we may (or may not) be able to sharpen the edge of a blood vessel.

7.3.2 The results in table 6.1 – 6.4

Based on the criteria given above it is clear that Morlet and Mexican hat have given very promising results. The CWTs in table 6.1 are the results of attempts to get the peaks of the CWTs about as high as possible. We were quite successful in these attempts, but only the Morlet and Mexican hat found the centre of the edge. We will therefore leave Haar out of the following discussion because of its inability to locate the centre. Instead we will focus on the Mexican hat wavelet because it seems to be the best candidate given the criteria mentioned above.

The Morlet and Mexican hat wavelets look very similar to each other. The Morlet has a more distributed energy than the Mexican hat, but both wavelets have peaks with maximums in the origin. The positive part of this peak will from now on be called the *focus* of the wavelet. We will try to explain the results in chapter 6 by using some of the theory in chapter 3.

In the section on Fourier transform it was explained that multiplying a signal with a cosine with frequency f and integrating the product over all time, gives a new function with information about the correlation between the signal and the cosine. If the signal does not contain much of the frequency f, then the oscillating nature of the cosine will cause the integral to be close to zero. This concept is important when selecting the right scale for the mother wavelet. For example, if we want to locate noise in an ultrasound image in order to filter it out, we will typically use a relatively small scale to locate the high frequency components of the image. A different scale will have to be applied in order to locate the edge of a blood vessel. As a rule of thumb it is usually a scale that makes the mother wavelet look most like what we are trying to find that gives the best results.

The CWTs in table 6.1 are good demonstrations of this rule. In order to elaborate we will use the Mexican hat as an example. The Mexican hat has its focus between -1 and 1. The scale used to get the tallest CWT possible with Mexican hat was 27. With this scale the focus of the Mexican hat will have about the same width as the edge. The fact that this is not a coincidence is made probable in fig. 7.1 below.

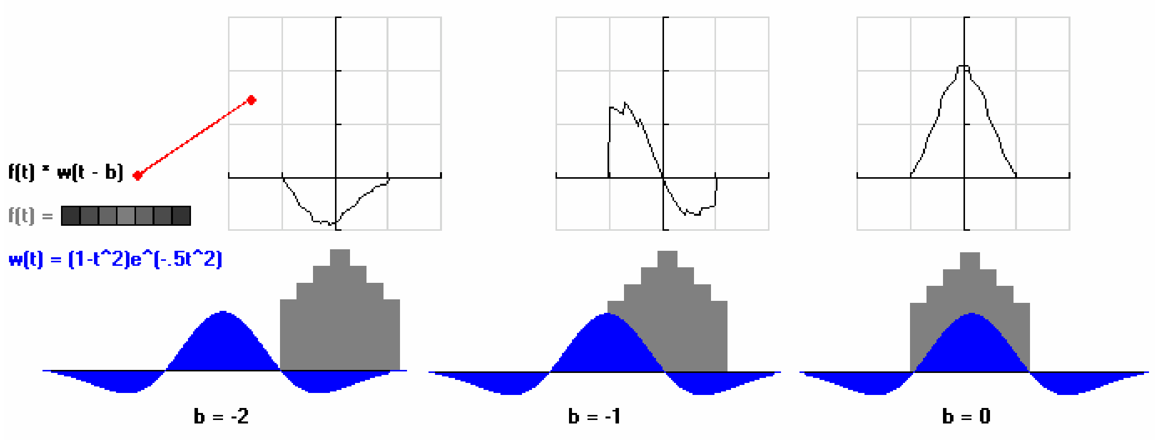


Fig. 7.1: A Mexican hat closing in on and edge.

Fig. 7.1 shows a Mexican hat (blue) as it sweeps over an edge (grey). For each of the three translations (-2, -1 and 0), the graph of the product of the blue and grey functions is shown at the top. The integral of this product gives the value of the CWT for the given translation. In fig. 7.1 the CWT reaches its minimum value at about b = -2. The value then increases and becomes zero at bout b = -1, the position of the left side of the edge. The maximum value of the CWT is at b = 0, right in the middle of the edge. It can be verified by looking at table 6.1 that the CWT using Morlet follows a similar pattern.

The CWTs in table 6.2 – 6.4 were produced by using about half the scales that were used in table 6.1. The result is that the peaks of the CWTs are both narrower and smaller than the ones in table 6.1. At the same time, the noise has had a greater effect on the CWT than in table 6.1. The reason for this is that the wavelets are now more similar to the noise and less similar to the edge than they were in table 6.1. The effect this has on the CWTs is according to the rule of thumb mentioned earlier.

7.3.3 The results in table 6.6

The principle behind the CWTs using 3D Mexican hat is almost identical to the ones using 2D wavelets. The main difference is that the mother wavelet can be stretched and translated in both x- and y direction in addition to being rotated. Another difference is that while the CWTs using 2D wavelets are plotted as graphs, the CWTs using 3D Mexican hat are shown as images.

The results in table 6.6 may not look too impressive at first sight, at least not compared to the graphs in table 6.1. Before drawing any conclusions based on these results, a few notes should be taken into consideration. The image used to test the 3D Mexican hat contains an edge where the centre is very bright. Edges of blood vessels in ultrasound images may be much dimmer, making them hard to see. Based on the results in table 6.1 it is reasonable to assume that the 3D Mexican hat is good at brightening these edges and making them stick out from the background. We say again what we said in section 7.2: the methods that were tested in this project should be tried out on real ultrasound images before we can say anything for sure.

The implementation of the 3D Mexican hat CWT produces a new image where the original pixel values are replaced with new values taken from the CWT. In order to calculate the new pixel value, the minimum and maximum values sampled from the CWT are used to map a given CWT sample to a shade between 0 and 255. The effect of this is shown in fig. 7.2 below.

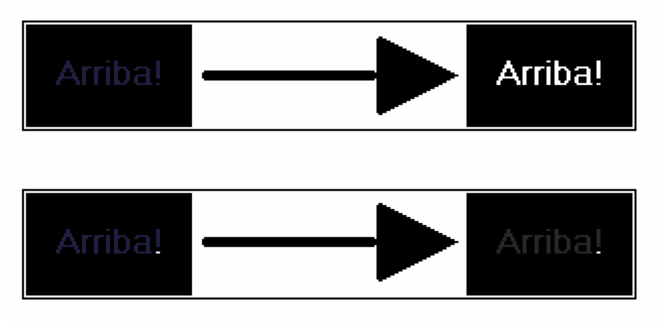


Fig. 7.2: Transforms of text using 3D Mexican hat

The images to the left in Fig. 7.2 have some text with a slightly brighter colour than the black backgrounds. The arrows to the right of the two images point to the transformed versions of the images. It is obvious that the topmost transform has had a dramatic effect on the image, while the other transform does not look much different from the original. The only difference between the two leftmost images is that the one at the bottom has a white dot in the exclamation mark.

The question then is: what if the edge of a blood vessel is very dark and the ultrasound image has some very bright speckles of noise. This question has actually been answered in section 7.3.2. By selecting scale parameters that make the Mexican hat look similar to what we are trying find, we can tune in on specific components of the image, whether it is the noise or the edge of a blood vessel. The results shown in table 6.6 are very good demonstrations of that; transform t8 is tuned to exaggerate the noise, while t7 does the exact opposite.

We are not going into detail about transform t9. This transform is the 3D Mexican hat equivalent to t4 – t6. It shows how the edge can be made thinner by using a smaller scale, but the noise will then have a greater impact on the transform.

7.3.4 General discussion

In the discussion above we have used a wavelet’s ability to suppress noise and amplify the centre of an edge as a measurement of how well suited it is for edge sharpening. There are basically two ways of sharpening edges in ultrasound images: one way is to make them slimmer and the other is to make them brighter. The transforms t4 – t6 and t7 (in table 6.2 – 6.4 and 6.6) demonstrate that it is possible to put an edge on a diet by selecting a scale smaller than what it takes to get the tallest peak. However, this makes the CWT more sensitive to noise and it makes the peak of the CWT smaller.

At this point we cannot see how this choice of scale would have an advantage over the scales used in table 6.1. With a wisely selected scale we should easily be able to locate the (approximate) start points of the edge’s two sides, and one would think that this information is more valuable than what we get when reducing the scale. However, the start points of an edge in an ultrasound image may not necessarily be the start points of an edge of a blood vessel. Blood vessels tend to have dross that causes the edges in ultrasound images to be wider than the edges of the real blood vessels. The research done in this project does not tell us whether CWT (or any other method for that matter) can be used to determine where the actual edge of a blood vessel starts. Perhaps the best we can do is to make a qualified guess based on statistical analysis.

7.4 Future work

The mission of this project was two fold; one goal was to study wavelet-based methods for use in medical imagery and the other goal was to make a tool for testing these methods. The second goal was also the means of achieving the first goal. Unfortunately the development of the EdgeX application consumed so much of our time that we got little time left to test the methods we had implemented. Even though we have presented a few tests and trivial interpretations of their results, this master thesis is by no means a comprehensive study of these methods.

The EdgeX application has great potential for improvement. At the time of writing there are only three 2D wavelets available in EdgeX for use in transforms. These are Morlet, Haar and Mexican hat. We did not expect Haar to be very useful, but it was added as a curiosity. We did have plans to test splines [11] as mother wavelet, but we simply did not make this in time.

Parallel to this project other students have had similar projects were they have analysed wavelet methods for use in digital image processing. One such project, for instance, was aimed at finding micro calcifications in mammograms. What many of these projects have in common with our project is that they have had an implementation-oriented approach to a solution. Most of the efforts put into these projects have resulted in applications intended for use by the people who will take over from here.

It is important that what we have accomplished in our projects is not duplicated in the future. There are currently plans to combine all our works into one application that can be used for further studies of the methods that were implemented. Our successors will then have a platform to start from in the continued investigation of wavelets and image processing. This will hopefully ensure the progress of the research on this field so that our efforts will not have been futile.

8 Conclusion

This thesis is a study of wavelet based image processing techniques for use in medical applications. The purpose has been to test methods for sharpening edges of blood vessels in ultrasound images. The focus has been on techniques using contiguous wavelet transform (CWT). In order to test these techniques an application was developed to be used as a testing tool. This program was named EdgeX and is written in the Java programming language.

The techniques implemented by EdgeX are organized as separate modules, called experiments, that can be loaded and instantiated at runtime. Two such modules were made in this project. The first one, CWTExperiment, allows the user to select a line in an image in order to compute a CWT of the pixel values. The user can choose between three different mother wavelets to be used as base function for the CWT. These wavelets are Morlet, Haar and Mexican hat, all of them of one independent variable. In addition, the user will have to select the scale used to dilate the wavelet. The resulting CWT is plotted in a 2D Cartesian coordinate system together with the graph of the pixel line.

The other module that was made is called MH3DExperiment. This module also computes a CWT of pixel values, but it uses a Mexican hat of two variables as mother wavelet. The user must therefore select an area to be transformed instead of a line. The user must enter three parameters for the 3D Mexican hat. These are the scales in x- and y direction, plus the rotation of the Mexican hat. The resulting CWT is presented as an image where its x- and y translation parameters are varied across the selected pixel area. The shades of the pixels in the produced image are mappings from CWT samples to integers between 0 and 255.

More effort than planned went into the development of EdgeX, and by the time it was finished there was little time left to do any testing. However, a few tests were made to get a first impression of the methods implemented in this project. These tests indicate that the Mexican hat wavelet is able to locate the two sides and centre of an edge in an image, in addition to brightening the edge. If the scale used to get the maximum light amplification of the edge is halved, it will have the effect that the transformed edge is narrower at the same time as the CWT's ability to reduce noise and amplify the brightness of the edge is reduced. The results of this thesis are inconclusive as to whether this will have an advantage over achieving the maximum brightening; once we have located the two sides of the edge in the image we can manually make it narrower if desired.

The application developed in this project was intended for use in projects to come as well as in this project. Future work should be to use this program extensively to test its image processing methods. The application should also be extended with more methods if needed. Plans exist to combine several prototypes developed in wavelet related projects, including this one, to form a richly featured testing tool to be used as a platform for further studies in this field of research.

9 References

Web references:

All links pointing to references were accessed during May 2003.

[1] Engineer's ultimate guide to wavelets, Rowan University, New Jersey.
http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html

[2] Earlier project on edge detection.
http://fag.grm.hia.no/fagstoff/perhh/htm/fag/matem/datwwww/wavelet.htm

[3] Lectures 2003, Hogstad.
http://fag.grm.hia.no/fagstoff/perhh/htm/fag/matem/datwwww/wavelet.htm

[4] Amara's Wavelet page
http://www.amara.com/current/wavelet.html

[5] College of CSS engineering, University of Iowa.
http://www.icaen.uiowa.edu/~dip/LECTURE/lecture.html

[6] Khoral.com.
http://www.khoral.com/contrib/contrib/dip2001/

[7] NTNU, Norwegian University of Science and Technology.
http://www.math.ntnu.no/%7eyura/h02/DIF5913/wavelets1.html

[8] SINTEF Unimed.
http://www.sintef.no/static/UM/UL/cx/cx.html

Litterature:

[9] Daubechies, Ingrid.
Ten lectures on wavelets.
Capital City Press, 1992.

[10] Edvards, C. Hendry. Penny, David E.
Differential equations and boundary value problems, Computing and modelling, second edition.
Prentice Hall International, Inc. 1996.

[11] Foley, James D. Dam, Andries van. Feiner Steven K. Hughes, John F. Phillips, Richard L
Introduction to computer graphics.
Addison-Wesley Publishing Company, Inc. 1994.

[12] Fowler, Martin with Scott, Kendall
UML Distilled. Applying the standars Object Modeling Language.
Addison-Wesly Longman, Inc. 1997.

[13] Hernández, Eugenio, Weiss, Guido
A first course on wavelets
CRC Press 1996.

[14] Stollnitz, Eric J, DeRose, Tony D, Salesin, David H
Wavelets for computer graphics, theory and applications.
Morgan Kaufmann Publishers, Inc. 1996.

[15] Thomas Jr., George B., Finney, Ross L.
Calculus
Addison-Wesley Publishing Company, Inc. 1990.

[16] Tønseth, Svein
Kirurgen ser gjennom deg
In Gemini no. 2 March 2003. Page 19-21