Exploring the potential of wavelets

In the field of image processing

J.J. Wong

Exploring the potential of wavelets

In the field of image processing

by

J.J. Wong

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Friday July 8, 2022 at 10:00 AM.

Student number:	4555317	
Project duration:	March 1, 2022 – July 15, 2022	
Thesis committee:	Dr. H. Kekkonen,	TU Delft, supervisor
	Dr.ir. W.G.M. Groenevelt,	TU Delft

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

This research consists of two applications of image processing, namely, image compression and image denoising. Image compression aims to reduce the size of an image without losing too many features. This is often used to store a large number of images such as fingerprints. Denoising is a technique for removing noise from an image while preserving as many of the edges and other detailed features as possible.

This research studies the use of different discrete wavelets and the Dual Tree Complex Wavelets in the image compression and denoising process. The wavelet transform decomposes the original image into approximation and detail coefficients, where the approximation coefficients are calculated by averaging and the detail coefficients by taking differences. The wavelet transform is also invertible so that the image can be reconstructed again using the approximation and detail coefficients.

The compression and denoising method consists of three steps: decomposition, thresholding and reconstruction. The difference between compression and denoising lies in the threshold part. For image compression, a percentage of the detail coefficient is chosen as the threshold. The wavelets db6, sym5, coif3, bior4.4 and rbio1.5 are chosen to compress the images. The images are tested with compression rates ranging from 5:1 to 43:1. Based on the Structural Similarity Index Measurement (SSIM), the bior4.4 wavelet performs best. For denoising, the threshold is optimised to obtain a denoised image. The discrete wavelets used are db4, coif3, bior2.8. Of these, the bior2.8 wavelet performs the best on images used in this research. Therefore, the bior2.8 wavelet is compared with the Dual Tree Complex Wavelet (DTCW). Based on the Peak Signal-to-Noise-Ratio (PSNR), the denoised image using the DTCW performs better than the bior2.8 wavelet.

Overal, wavelets are a powerful tool in image processing. The different wavelets each have their own characteristics. The choice of the optimal wavelet depends on the application and cannot be generalised.

Layman abstract

There are many applications of image processing, ranging from detecting cancer to face recognition. One of them is called image compression. Image compression is used to reduce the size of an image. This is for example used when a picture is send over a digital platform, such as Whatsapp or Facebook. It can also be used to store large data, for example, fingerprints. Another application is image denoising. Image denoising is a technique of removing noise from an image. For example, if you take a photo with your older mobile phone, the photo is often a bit noisy. This research explores the potential of wavelet in image compression and denoising. In the case of image denoising, the image is deconstructed using the wavelet, which allows to separate the details of the image from the rest or noise. In the case of image compression, wavelet transformations are applied to the image to decrease the total memory required to store the image, by deconstructing the image using the wavelet. This research report concludes that the application of wavelets in the field of image processing yields a satisfactory result.

Contents

1 Introduction

Digital images are an integral part of today's world. These images can be medical or scientific images obtained through ultrasound, X-rays or gamma rays, but most images are obtained through a digital camera, for example from a mobile phone.

Cameras work with light reflection. Light consists of photons, or in other words particles that move at the speed of light. Light is needed to take a picture of an object. Photons bounce off the object to the camera's light-sensitive sensor. Photons are not moving in a fixed pattern, so if we take two pictures with the same device of the same object, the number of photons captured could still be different. Since one photon's arrival is independent of the other photons, the number of collected photons in a specific interval of time is Poisson distributed. When we define this random variable as X_i, with $i = 1, \ldots, N$ as the number of the drawing, the sample mean $\bar{X}$ of all drawings $X_i, \ldots, X_n$ follows a Gaussian distribution under the Law of Large Numbers.

The noise associated with the arrivals of photons is called the Poisson noise. This type of noise always exists whenever you are capturing light. In the dark parts of the image, fewer photons are hitting the sensor and therefore the noise is more visible.

Moreover, all those images have to be stored or processed. Every day, billions of photos are taken and sent to each other. Sending and storing these photos costs memory. The better the quality, the more memory is required. The Federal Bureau of Investigation has about 200 million fingerprints in storage [4]. Storing them in the same quality as your photos on your mobile phone costs about 2000 terabytes of memory; in other words, it takes much more memory than a conventional computer could store by itself. So there must be a way to store images more efficiently. This is called image compression. Compression can be distinguished into lossless compression and lossy compression. If an image is compressed lossless, the original image can be recovered exactly from the compressed image. Since the human eye can only perceive about 32 shades of grey [5], lossless compression is not always necessary. A good method for lossy compression is the discrete wavelet transform. Both image compression and denoising are part of image processing.

The concept of transformations is not new. In 1822, Joseph Fourier claimed that functions could be expressed into sines and cosines. The Discrete Fourier Transforms were used for many applications ranging from image pattern recognition to image processing. However, the Fourier Transformation has some disadvantages. The main disadvantage is that it only has frequency resolutions and no time resolutions. For signals, it means we know what happens to the signal but not when it happens. For images, it means that the global frequency can be determined, but not the local changes.

In 1909, Alfred Haar proposed the Haar wavelet transform. The Haar transform is the simplest form of Discrete Wavelet Transform (DWT) and intends to preserve the features of the original image while reducing the size of the image significantly. In the meantime, the field of (discrete) wavelet transform has expanded rapidly, and multiple different wavelet families have been introduced by different researchers, such as Daubechies, Symlet, Coiflet, Biorthogonal, Reverse Biorthogonal, and the Dual Tree Complex Wavelets. These wavelet families will be evaluated in this research report.

The structure of this research report will be as follows. First, the image representation in mathematics is discussed. This mathematical image representation is then transformed using Discrete Wavelet Transform. The process of DWT will be elaborated in detail, which will include signal decomposition, reconstruction, and processing. This elaboration is followed by the description of the different wavelet families evaluated in this research report. Then, an enhancement of the DWT is discussed, namely the Dual Tree Complex Wavelets.

Subsequently, the methods used in this research report are discussed. First, the thresholding methods applied to the wavelets are discussed, which is followed up by the image processing applications. These applications consist of image compression and denoising, which will be individually elaborated upon.

Afterwards, the results of this research are discussed, which will be followed up with the conclusion of this research report.

2

Wavelet Transform

Wavelet transform is the technique of applying different wavelets to transform the data. In this research, the data consists of image data, and the goal of the transformation is to compress the image for saving memory and remove noise from the image. In particular, the Discrete Wavelet Transform and Dual Tree Complex Wavelet Transform are evaluated in the aforementioned image processing techniques. This chapter will take a closer look at the differences between the wavelets. The methodology is divided into three parts. The first part introduces how an image can be represented in mathematics, which is followed up with the description of the Discrete Wavelet Transform and the Dual Tree Complex Wavelet Transform.

2.1. Images

Images come in all shapes and sizes. The images used in this article are 512x512 pixel images, retrieved from MATLAB. These images are all in black and white with a greyscale of 255.

2.2. Representing an image in mathematics

The first step is to understand how images are represented in mathematics. An 8-bit image consist of squares that are indexed by 255 grey levels. In figure 2.1, an image of a fingerprint is shown. In the zoomed in version, the different squares can be seen. Every square denotes a pixel, which can be indicated by one of the 255 grey levels, where zero denotes black and 255 denotes white. This means every square has a number. Let every number be an element of a matrix, then the matrix can be used for the 2D wavelet transformation.

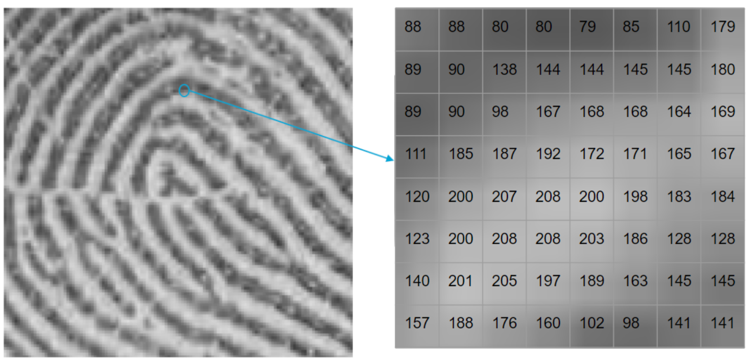
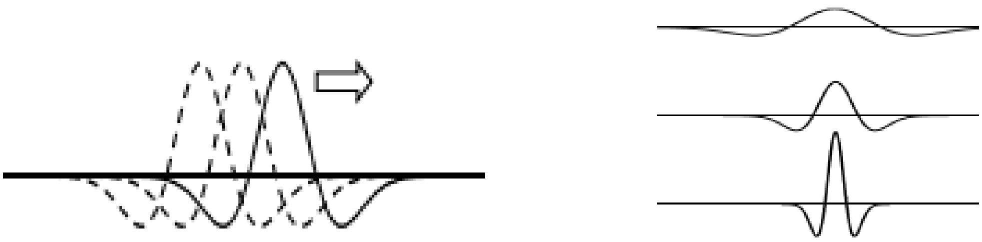


Figure 2.1: Fingerprint [4]

This process can also be carried out for images with colour. In this case, each pixel represents three numbers, namely the Red Green Blue (RGB) values, each ranging from 0 to 255. The transformation is then performed for the Red, Green and Blue values. In this paper, however, only the 255 grey scales are taken into account.

2.3. Discrete Wavelet Transform (DWT)

Wavelets are oscillating functions which begins at zero, increases or decreases on an interval and then returns zero again, in other words, the wavelets are compactly supported. The wavelet transformation turns a function into a set of wavelets coefficients in order to represent information in a more useful way. Wavelets can be adapted by moving the wave to the left or right, called translation, see Figure 2.2a and by changing the amplitude, also called dilation, see Figure 2.2b.



(a) Wavelet can move to the left and right.

(b) Wavelets with different amplitudes.

Figure 2.2: Different ways to manipulate wavelets [4].

In Figure 2.3 some examples are shown of wavelets with different dilation and translation functions. There are multiple wavelet families, all with different dilationa and translation functions, for example, the Daubechies, Symlet of Coiflet wavelets.

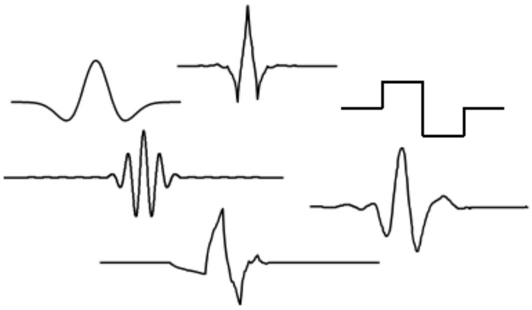


Figure 2.3: Different wavelets [4].

The mathematical representation of a wavelet is given by

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right), \tag{2.1}$$

where a is the scale (dilation) parameter and b the location (translation) parameter. Let $x(t)$ be a signal, then the wavelet coefficient is given by

$$T_{a,b} = \langle x, \psi_{a,b}\rangle = \int x(t)\cdot\psi_{a,b}(t)dt. \tag{2.2}$$

2.3.1. Signal decomposition using the Haar wavelet

Signal decomposition can be carried out using two functions that generates a family of functions. The two functions are the scaling function ϕ and the wavelet function ψ. The simplest form of the discrete wavelet transform is called the Haar wavelet transform or the Daubechies 1 (db1) wavelet transform, which is introduced in 1910 by the Hungarian mathematician Alfred Haar [7]. The Haar scaling function is defined by

$$\psi(x) = \begin{cases} 1, & 0 \le x < 1, \\ 0, & \text{elsewhere} \end{cases}. \tag{2.3}$$

The Haar wavelet function is defined by

$$\phi(x) = \begin{cases} 1, & 0 \le x < \frac{1}{2}, \\ -1, & \frac{1}{2} \le x < 1, \\ 0, & \text{elsewhere} \end{cases} . \tag{2.4}$$

The functions are shown in Figure 2.4.

(a) The scaling function. (b) The wavelet function.

Figure 2.4: The Haar functions.

Let S = [0, 0, 4, 2, 6, 4, 2, 0] be a signal, then the Haar wavelet transform decomposes this into two sub-signal of half its length. One subsignal is obtained by taking the average of the pairs $[0,0],[4,2],[6,4],[2,0]$,. The other sub-signal is obtained by taking half the difference of the same pairs. The original signal and the two sub-signals are shown in Figure 2.5.

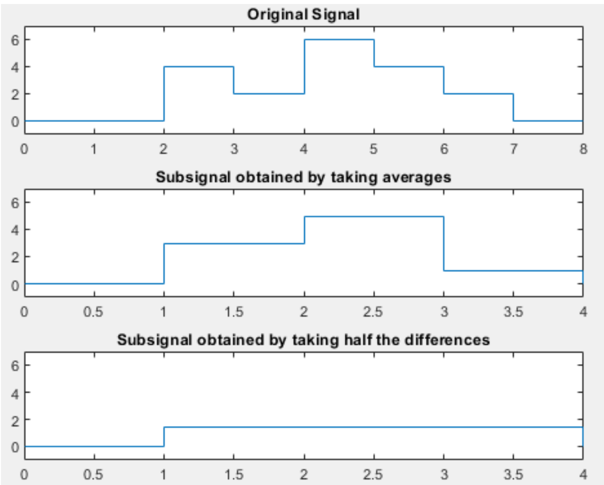


Figure 2.5: Original signal and the two sub-signals.

Let $\mathbf{f}_{j+1} = (a_1^{j+1}, a_2^{j+1}, ..., a_N^{j+1})$ denotes a signal with length N, where N is of the form 2^m, where m is an integer which denotes the maximum number of decomposition. The first step of the decomposition is to split the input signal into two sub-signals. One signal contains the approximation coefficient, denoted by f_j, which is calculated by averaging the elements of the vector in pairs, in other words, the average of the first two elements, the third and fourth element and so on. In order to normalise the vector, the coefficient is multiplied by the $\sqrt{2^j}$. The formula of the approximation coefficient a_n^j is given by

$$a_n^j = \sqrt{2^j}\frac{a_{2n}^{j+1} + a_{2n-1}^{j+1}}{2}, \tag{2.5}$$

for $n = 1...N/2$.
The other sub-signal, denoted by v_j, contains the detail coefficients, which are calculated by taking half of

the difference between two elements and multiplying by $\sqrt{2}^j$. The formula of the detail coefficient d_n^j is given by

$$d_n^j = \sqrt{2^j}\,\frac{a_{2n}^{j+1} - a_{2n-1}^{j+1}}{2}, \tag{2.6}$$

for $n = 1...N/2$.
The output of this transformation equals $(a_1^j, ..., a_{N/2}^j, d_1^j, ..., d_{N/2}^j)$. This process can be repeated by decomposing the sub-signal $f_j = (a_1^j, ..., a_{N/2}^j)$ until j = 1. In figure 2.6, the decomposition is repeated three times. Since there is only one element left, this is the maximum number of times the signal can be decomposed. The left node denotes the sub signal with the approximation coefficients and the right node denotes sub-signal with the detail coefficients. After decompose the signal three times, the output equals $(\sqrt{2}\cdot 2.25, \sqrt{2}\cdot 0.75, \sqrt{2^2}\cdot 1.5, \sqrt{2^2}\cdot -2, \sqrt{2^3}\cdot 0, \sqrt{2^3}\cdot -1, \sqrt{2^3}\cdot -1, \sqrt{2^3}\cdot -1)$.

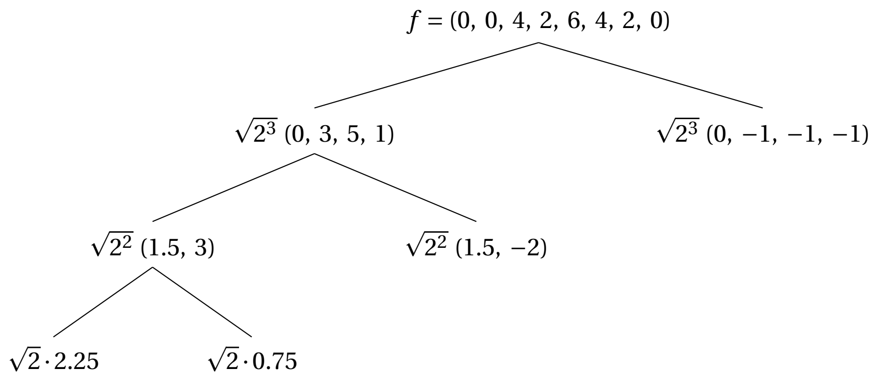


Figure 2.6: Example Haar wavelet decomposition

The detail coefficients are shown in Figure 2.7 together with the level of decomposition. Note that the constant $\sqrt{2^j}$, where j denotes the level of decomposition is left out.

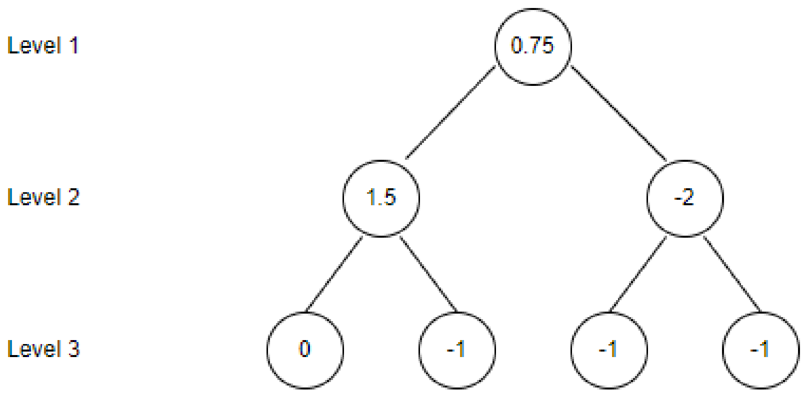


Figure 2.7: The detail coefficients.

2.3.2. Signal reconstruction using Haar wavelet

Consider the output vector of Figure 2.6, $(\sqrt{2}\cdot 2.25, \sqrt{2}\cdot 0.75, \sqrt{2^2}\cdot 1.5, \sqrt{2^2}\cdot -2, \sqrt{2^3}\cdot 0, \sqrt{2^3}\cdot -1, \sqrt{2^3}\cdot -1, \sqrt{2^3}\cdot -1)$, where $\mathbf{f}_1 = \sqrt{2}\cdot 2.25$, $\mathbf{v}_1 = \sqrt{2}\cdot 0.75$, $\mathbf{v}_2 = \sqrt{2^2}\cdot(1.5, -2)$ and $\mathbf{v}_3 = \sqrt{2^3}\cdot(0, -1, -1, -1)$. This vector can be reconstructed using equation 2.5 and 2.6. Rewrite equation 2.5 and 2.6 gives

$$\left.\begin{aligned} \frac{2}{\sqrt{2^{j'+1}}}\cdot a_n^{j'} &= a_{2n}^{j'+1} + a_{2n-1}^{j'+1}, \\ \frac{2}{\sqrt{2^{j'+1}}}\cdot d_n^{j'} &= a_{2n}^{j'+1} - a_{2n-1}^{j'+1}, \end{aligned}\right\} \quad \Longrightarrow \quad \left.\begin{aligned} a_{2n}^{j'+1} &= \frac{1}{\sqrt{2^{j'+1}}}\cdot(a_n^{j'} + d_n^{j'}), \\ a_{2n-1}^{j'+1} &= \frac{1}{\sqrt{2^{j'+1}}}\cdot(a_n^{j'} - d_n^{j'}), \end{aligned}\right\}.$$

The coefficients $a_{2n}^{j'}$ and $a_{2n-1}^{j'}$ are determined recursively for $j' = 2, ...$ unitl $j' = j+1$

2.3.3. Signal Processing using Filter Banks

Generally, in signal and image processing, the decomposition and reconstruction can be expressed in terms of filter banks. The DWT is computed using two filters, the lowpass and the highpass filter. The low-pass filter averages the signal and the high-pass filter takes the differences between the elements of the signal. How the approximation (lowpass) and detail (highpass) coefficients are calculated is dependent on the wavelet chosen. Decomposing the signal into sub-signals is called downsampling. The downsampling process is shown in Figure 2.8. A signal $X[n]$ passes through two filter banks, the H_0, which denotes the highpass filter and, the G_0, which denotes the lowpass filter. The output of the highpass filter is the vector with the detail coefficients. The lowpass filter gives the apporximation coefficients, which can pass the filters again. The filters used for downsampling are also called the analysis filters.

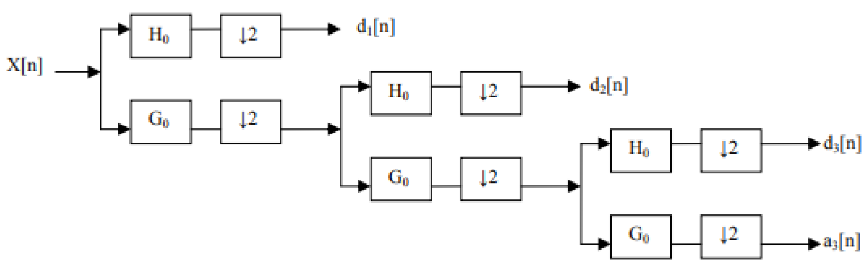


Figure 2.8: Mallat-tree decomposition.

The reconstruction process is basically the reverse process of decomposition. This process is called upsampling and the filters used for upsampling are called the synthesis filters. The diagram of upsampling is shown in Figure 2.9.

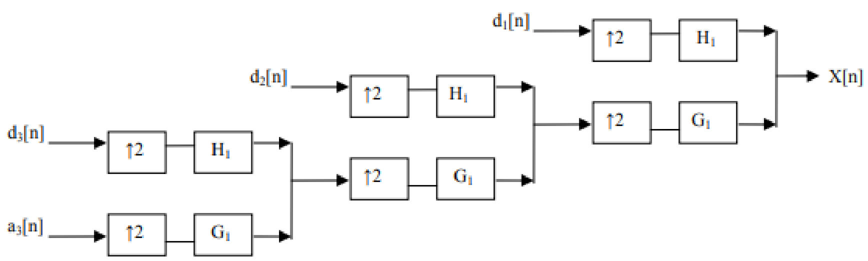


Figure 2.9: Mallat-tree reconstruction.

2.3.4. Image decomposition using the Haar wavelet

The image decomposition (2D decomposition) is similar to the signal decomposition, where the difference is that the input is a nxn matrix and both the rows and columns are decomposed. The low-pass filter averages the image and the high-pass filter takes differences between two elements. To show how the decomposition works, we take the zoomed-in 8x8 image in Figure 2.1 and follow the method in Figure 2.10. In the figure, H denotes the high-frequency bands and L the low-frequency bands. The image is first decomposed row-wise, so the lowpass filter denoted as L in the figure and the highpass filter denoted as H is applied to the rows. After that, the image is decomposed column-wise. In short, LL means both the rows and columns are passed through the lowpass filter, which means the average is taken for every row and every column. HL means the differences is taken row-wise and then the average is taken column-wise. Then LH means the average is taken row-wise and the differences is taken column-wise. Lastly, HH denotes the image where the differences is taken for every row and column,.

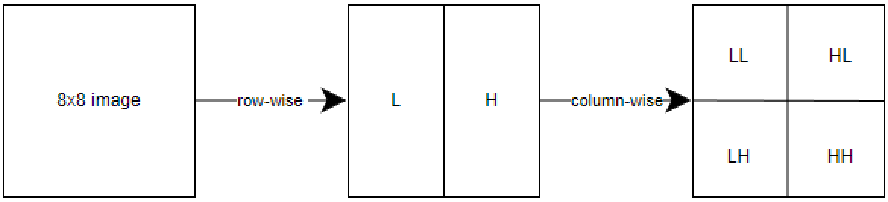


Figure 2.10: Decomposition of an image.

First, every row of the matrix is decomposed using the method described in the previous section. Let

$$A = \begin{pmatrix} 88 & 88 & 80 & 80 & 79 & 85 & 110 & 179 \\ 89 & 90 & 138 & 144 & 144 & 145 & 145 & 180 \\ 89 & 90 & 98 & 167 & 168 & 168 & 164 & 169 \\ 111 & 185 & 187 & 192 & 172 & 171 & 165 & 167 \\ 120 & 200 & 207 & 208 & 200 & 198 & 183 & 184 \\ 123 & 200 & 208 & 208 & 203 & 186 & 128 & 128 \\ 140 & 201 & 205 & 197 & 189 & 163 & 145 & 145 \\ 157 & 188 & 176 & 160 & 102 & 98 & 141 & 141 \end{pmatrix}$$

denote the 8x8 image, then the first row

$$r_1 = \begin{pmatrix} 88 & 88 & 80 & 80 & 79 & 85 & 110 & 179 \end{pmatrix}$$

is decomposed as follows. The average and difference is calculated over the pairs $[88, 88]$, $[80, 80]$, $[79, 85]$, $[110, 179]$. The average is calculated using equation 2.5 and the difference using equation 2.6, where f_{2n} the second element of every pair, f_{2n-1} the first element and j equals 1, since we only decompose the matrix once. The average of the 4 pairs replaces the first 4 entries of r_1 and the difference of the 4 pairs replaces the last 4 elements of r_1. The new row is denoted by

$$r_1 \cdot h_1 = \sqrt{2} \cdot \begin{pmatrix} 88 & 80 & 82 & 144.5 & 0 & 0 & -3 & -34.5 \end{pmatrix}$$

From this equation, we can deduce that the transformation matrix equals

$$h_1 = \sqrt{2} \cdot \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} \end{pmatrix}$$

The matrix h_1 can be multiplied on the right side with the matrix A in order to decompose all rows. Therefore, we have

$$A \cdot h_1 = \sqrt{2} \cdot \begin{pmatrix} \frac{1}{88} & 80 & 82 & \frac{289}{2} & 0 & 0 & -3 & -\frac{69}{2} \\ \frac{179}{2} & 141 & \frac{289}{2} & \frac{325}{2} & -\frac{1}{2} & -3 & -\frac{1}{2} & -\frac{35}{2} \\ \frac{179}{2} & \frac{265}{2} & 168 & \frac{333}{2} & -\frac{1}{2} & -\frac{69}{2} & 0 & -\frac{5}{2} \\ 148 & \frac{379}{2} & \frac{343}{2} & 166 & -37 & -\frac{5}{2} & \frac{1}{2} & -1 \\ 160 & \frac{415}{2} & 199 & \frac{367}{2} & -40 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ \frac{323}{2} & 208 & \frac{389}{2} & 128 & -\frac{77}{2} & 0 & \frac{17}{2} & 0 \\ \frac{241}{2} & 201 & 176 & 145 & -\frac{61}{2} & 4 & 13 & 0 \\ \frac{345}{2} & 168 & 100 & 141 & -\frac{31}{2} & 8 & 2 & 0 \end{pmatrix}$$

This matrix corresponds with the LH image of Figure 2.10. The second step is decompose the image column-wise. This can be done by multiplying A on the left by the transpose of h_1, h_1^T. Hence,

$$h_1^T \cdot A \cdot h_1 = 2 \cdot \begin{pmatrix} \frac{355}{4} & \frac{221}{2} & \frac{453}{4} & \frac{307}{2} & -\frac{1}{4} & -\frac{3}{2} & -\frac{7}{4} & -26 \\ \frac{475}{4} & 161 & \frac{679}{4} & \frac{665}{4} & -\frac{75}{4} & -\frac{37}{2} & \frac{1}{4} & -\frac{7}{4} \\ \frac{643}{4} & \frac{831}{4} & \frac{787}{4} & \frac{623}{4} & -\frac{157}{4} & -\frac{1}{4} & \frac{19}{4} & -\frac{1}{4} \\ \frac{343}{2} & \frac{369}{2} & 138 & 143 & -23 & 6 & \frac{15}{2} & 0 \\ -\frac{3}{4} & -\frac{61}{2} & -\frac{125}{4} & -9 & \frac{1}{4} & \frac{3}{2} & -\frac{5}{4} & -\frac{17}{2} \\ -\frac{117}{4} & -\frac{57}{2} & -\frac{7}{4} & \frac{1}{4} & \frac{73}{4} & -16 & -\frac{1}{4} & -\frac{3}{4} \\ -\frac{3}{4} & -\frac{1}{4} & \frac{9}{4} & \frac{111}{4} & -\frac{3}{4} & -\frac{1}{4} & -\frac{15}{4} & -\frac{1}{4} \\ -1 & \frac{33}{2} & 38 & 2 & -\frac{15}{2} & -2 & \frac{11}{2} & 0 \end{pmatrix},$$

where blue corresponds to LL, green corresponds to HL, red corresponds to LH and orange corresponds to HH, see Figure 2.10.

As mentioned before, zero denotes black and 255 denotes white. The larger the difference between two pixels, the whiter the pixel after decomposition. If the difference is small, it appears darker after decomposition. However, the matrix also contains negative values. Since the difference between black and white are considered, we can take the absolute value. The difference between black and white, 255 minus 0 is similar to the difference between white and black, 0-255. In Figure 2.11, it can be seen that the edges are white and the remaining parts are black. In other words, the LL means that the average is taken for each row and each column, so that the image is reduced by a factor of 2. The HL means that the average is taken for every row and the difference for every column, so the horizontal features appear, see Figure 2.11. Conversely, for LH, the average is taken vertically and the difference horizontally, so the vertical features are visible, see Figure 2.11. The HH means that the difference is taken for every row and every column.

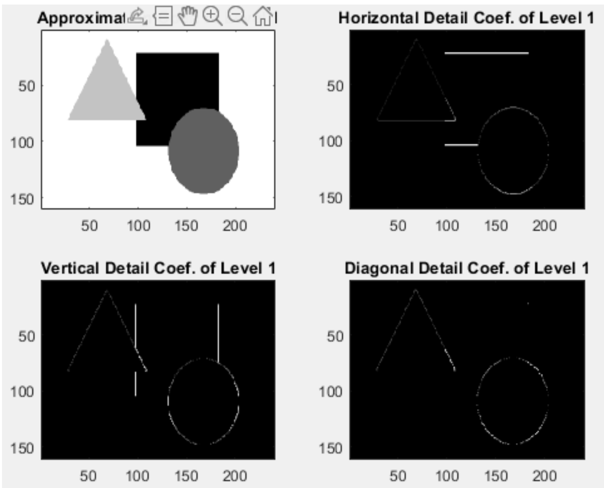


Figure 2.11: Decomposition of an simple image.

An image can also be decomposed more than once, by applying the method to the low-pass component. A N-level decomposition results in 3N+1 different frequency subbands, see Figure 2.12.

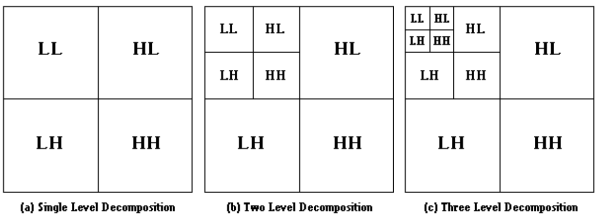


Figure 2.12: Decomposition of an image [6]

The same calculation can be performed on 512 x 512 images, but then the transformation matrix is also 512 x 512.

The Haar functions are the simplest wavelet functions. However, as shown in Figure 2.4, the functions are discontinuous. Therefore, the Haar wavelets are not good at approximating continuous signals. Using the Haar wavelet transform for images can lead to blocking artifacts.

2.3.5. Daubechies Wavelets

The Daubechies wavelets were discovered by Ingrid Daubechies, in 1988 [2]. These wavelets are compactly supported orthonormal wavelets. A function is compactly supported if it is zero outside a compact set. This property ensures that wavelets are localized in time. The Haar wavelet, also called the db1 wavelet is the only discontinuous wavelet of this family. The other Daubechies are compactly supported and continuous. The Daubechies have two naming schemes, i.e. DN, where N denotes the filter length and dbA, where A is the number of vanishing moments. The filter length equals two times the number of vanishing moments, so the wavelet db2 and D4 are the same. A wavelet has N vanishing moments if and only if the scaling and wavelet function can generate polynomials up to degree N-1. The higher the number of vanishing moments, the more complex functions the wavelets can generate. In images, it means that, the higher the vanishing moments, the more details can be extracted. In addition, the smoothness of the functions increases with the vanishing moments. In terms of filter length, the larger the filter, the more pixels are taken into account. For example, if the filter length equals 4, the average and difference is taken over 4 pixels, so in the end the image is smoother than when using a filter with filter length equal to 2. Another advantage is that the window size is overlapping, which means the result reflects all changes between the pixels [2]. However, sometimes the Daubechies wavelets can cause problems at the border. Suppose the input signal has length 8, the filter length is 4 and the overlap window is equal to 2. In this case, the sub-signal should consist of 4 elements, but with a filter length of 4 and an overlapping window 2, it is not possible to get 4 elements. Therefore, either another filter must be used at the borders, or the signal must be extended. By Strang [12], symmetric extension is the best solution. There are two ways to extend the signal symmetrically. The signal can reflect about a line through the endpoint or midway between the end point and the next point. The first type is shown on the left and the other type is shown on the right, see Figure 2.13.

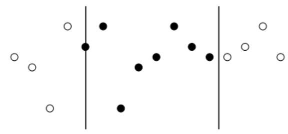


Figure 2.13: Two ways to extend the signal.

2.3.6. Other wavelets

For the Discrete wavelet transform, there are a few basis functions that can be used. Besides Daubechies, you also have Symlet, Coiflet, which are based on the Daubechies. The Symlet wavelet are the Daubechies with increased symmerty. The Coiflet wavelet is also derived from the Daubechies. The function is also symmetric and it uses three overlapping windows, which led to even smoother functions. Beside these orthogonal wavelets, there also exist non-orthogonal wavelets, the Biorthogonal and the Reverse Biorthogonal wavelet. The effect of these wavelets on image compression and denoising will be tested in this research.

2.4. Dual Tree Complex Wavelet Transform

As the concept of the DWT has been introduced and elaborated upon, an expansion to this subject will be introduced, namely the Dual Tree Complex Wavelet Transform (DTCWT). This technique adds additional properties, such as being nearly shift-invariant and directionally selective in dimensions larger and equal than two [11]. Instead of using one tree resulting from the deconstruction of the original signal, two times the deconstruction is applied, resulting in a dual tree construction, as shown in Figure 2.14.

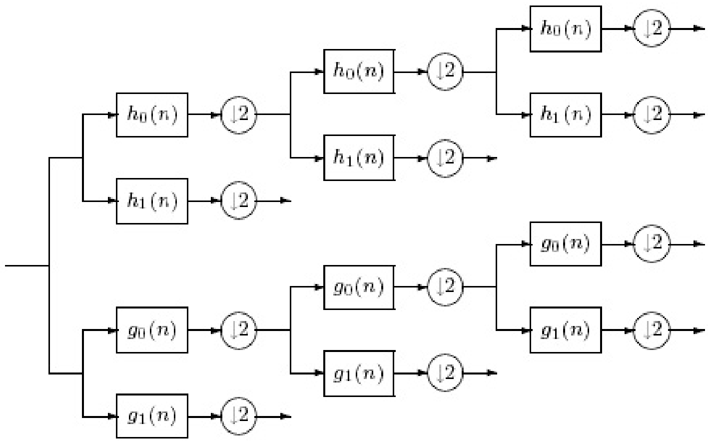


Figure 2.14: Example of a DTCWT deconstruction.

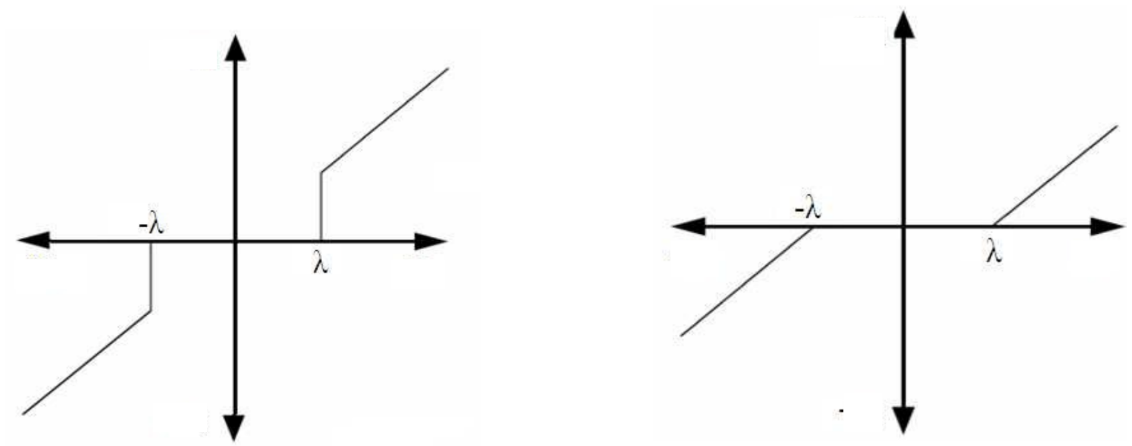
The upper and lower DWT trees are designed in such a way that the upper and lower parts relate to the real and respectively the imaginary parts of the complex wavelet. Note that the the filters itself are real. The two real wavelet transformations uses two different filters. For the first stage, the nearly symmetric Farras filters are used and for the remaining stages the Kingsbury Q-shift filters [11]. The algortihm on how these filters are designed are explained in [10].

3
Image Compression

Image compression is a method to convert the image so that it consumes less space. Compression can be distinguished into lossless compression and lossy compression. If an image is compressed lossless, the original image can be recovered exactly from the compressed image. However, the compression rate is usually higher and therefore it is less efficient. Since the human eye can only perceive about 32 shades of grey [5], lossless compression is not necessary to maintain the quality of the image. Furthermore, since the wavelets are turning many coefficients to zero with a minimal effect on the image, it is a good method for lossy compression. For this reason, lossy compression is considered in this research. This chapter consists of three sections. Section 3.1 describes the different threshold methods and section 3.2 gives a detailed description of the compression method. The last section discusses the results of compressing an image using different wavelets.

3.1. Threshold

There are two threshold methods, soft thresholding and hard thresholding, see Figure 3.1.



(a) Hard Thresholding. (b) Soft Thresholding.

Figure 3.1: Thresholding methods.

Hard thresholding is the process of setting the coefficients to zero if the coefficients are smaller than a chosen number (threshold), see Equation 3.1.

$$T_H(c) = \begin{cases} 0 & \text{if } |c| \leq \lambda \\ c & \text{if } |c| > \lambda \end{cases}, \tag{3.1}$$

where λ is the threshold and c is the value of the wavelet coefficient.

Soft thresholding is the extended version of hard thresholding. First, the coefficients smaller than the threshold are set to zero. And the non-zero coefficients are moved towards zero by adding or subtracting λ, see equation 3.2.

$$T_S(c) = \begin{cases} 0 & \text{if } |c| \leq \lambda \\ c - \lambda & \text{if } c > \lambda \\ c + \lambda & \text{if } c < -\lambda \end{cases}, \tag{3.2}$$

where λ is the threshold and c is the value of the wavelet coefficient.

Soft thresholding produces smoother results compared to hard thresholding, but in hard thresholding, the edges are better preserved, in other words, the edges remain sharper. The superiority of the thresholding method depends on its application. For example, to process (either compress or denoise) Whatsapp images, a smooth image is more desired, so soft thresholding works better. And to process fingerprint images, the edges are the main features you do not want to lose, therefore hard thresholding is a better option.

3.1.1. Threshold

The threshold is crucial in the compression and denoising process. If the threshold is too small, the image is not optimally compressed. If the threshold is too large, the compressed image may contain blocking artefacts and some details will be lost. For image denoising, a small threshold may result in less noise being removed. A threshold that is too large may result in more noise being removed, but also in some details being lost.

The wavelet decomposition of an image divides the image into four parts as shown in Figure 2.11. One part contains the approximation coefficients and the remaining three parts the detail coefficients. First, consider the detail coefficients. The detail coefficients are calculated by taking the differences between the adjacent elements of the matrix. When the difference between two adjacent elements is large, it could imply that it is a pixel of an object's edge in the picture. An example is given in Figure 2.11, where the horizontal, vertical, and diagonal detail coefficients are shown. On the locations of the borders of the elements, white lines or lines close to white are shown. The differences between the borders of the square and the white background result in a white line. The border of the grey circle and triangle with the white background results close to white lines.

The choice of the threshold coefficient determines the degree of filtering of those differences. The higher the threshold, the higher the difference has to be such that it stays in the figure. An example is given in Figure 3.2b and Figure 3.2c, where the triangle is not shown anymore in the decomposition because the difference between the coefficients denoting the edge of the triangle and the background is smaller than the threshold coefficient. In Figure 3.2c, the threshold coefficient is even larger. Therefore, the line denoting the edge of the circle also disappeared.

The image used as an example only contains black, white and two grey shades. In images such as the peppers image shown in Figure 3.3a, there are more grey shades and more objects. A small threshold coefficient will remove small details in the image, in other words, it will remove the parts where the difference in the wavelet coefficients is small. The larger the threshold coefficient, the more features are removed.

(a) Detail coefficients with $\lambda = 0$. (b) Detail coefficients with $\lambda = 40$. (c) Detail coefficients with $\lambda = 200$.

Figure 3.2: Horizontal detail coefficients thresholded by different threshold coefficients.

3.2. Method

Compression involves four steps:

- Decomposition: Choose a wavelet and decompose it to level N, where N = 1..5.

- Choose the percentage of detail coefficients to threshold, for example 90% and apply soft or hard thresholding.
- Reconstruction: Use the approximation and thresholded detail coefficients to reconstruct the image.

The reduction of the size of an image is dependent on the way of coding. Long strings of zeros can be encoded very efficiently using standard entropy coding techniques such as Huffman coding [9]. As shown in Figure 3.2a, the biggest part of the image is black, which means most of the detail coefficients are zero. The length of a string consisting of only zero coefficients cannot be determined. Consequently, the reduction of the image in size can not be found. Therefore, we assume that saving a nonzero coefficient equals 1 unit of memory and saving a zero coefficient costs 0 unit of memory. With this assumption, the Compression Rate can be defined by

$$CR = \frac{Nonzero\ coefficients\ Original\ Image}{Nonzero\ coefficients\ Compressed\ Image}. \tag{3.3}$$

The image will be decomposed to different levels using the previously mentioned wavelets. The difference between the compressed images is sometimes not visible to the human eye. Therefore, we need to choose a quality measurement method. In this study, the Peak Signal-to-Noise Ratio (PSNR) and the Structure Similarity Index Method (SSIM) are taken into account. According to Wang et al. [14], PSNR is good for capturing noise, changes in brightness, contrast and saturation, but it does not work well for capturing different types of distortion and blur. Compared to the PSNR, the Structure Similarity Index Method (SSIM) is good at capturing blur, different types of distortion and noise. However, it is not good at capturing changes in brightness, contrast and saturation. The purpose of image compression is to reduce the size of an image. Distortions such as blocking artefacts are often the result of compression. Because of the visibility of structural distortions, it is better to use SSIM [13].

3.3. Results

In this section, the pepper image shown in Figure 3.3a will be compressed using the hard thresholding method. First, multiple different wavelets are evaluated in the field of image compression. The evaluation of the compression on this figure is done in terms of the wavelet choice, threshold, and level of decomposition, which will be elaborated upon below.

3.3.1. Comparing different discrete wavelets

First, the Daubechies, Symlet, Coiflet, Biorthogonal and Reverse Biorthogonal wavelets are tested. Table 3.1 contains the SSIM values for different compression rates. A CR of 5:1 means that we keep 20% of the nonzero coefficients and set 80% equal to 0. A CR of 9:1, 15:1, 36:1, 43:1 corresponds respectively to removing 89%, 93.3%, 97.3% and 97.7% of the coefficients. The more the image is compressed the lower the SSIM value. The highest SSIM is obtained by using the bior4.4 wavelet.

CR	db6	sym5	coif3	bior4.4	rbio1.5
5:1	0.9947	0.9954	0.9950	0.9954	0.9953
9:1	0.9865	0.9875	0.9871	0.9883	0.9874
15:1	0.9734	0.9749	0.9750	0.9771	0.9751
36:1	0.9255	0.9320	0.9319	0.9377	0.9332
43:1	0.9057	0.9118	0.9126	0.9190	0.9136

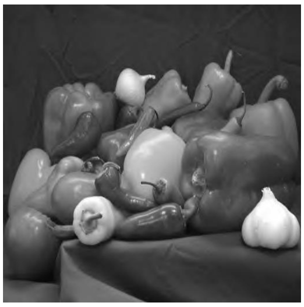
Table 3.1: SSIM values after compression

Figure 3.3 shows the original image and the compressed image with different CR using the bior4.4 wavelet. Compression using the discrete wavelet transform works satisfactorily. Removing 93.3% of the coefficients still gives a sharp image, see Figure 3.3d. Removing even more coefficients gives slightly worse images. As stated before, the CR does not tell how much the image is reduced in size. How much the image is reduced in size depends on the way of coding. The run-length encoding could be an efficient way to use in combination with the wavelet transformation. A reason for this is, that if the same value appears in many consecutive elements, it is stored as a single value and a count. Therefore, storing a matrix with many zero elements could be more efficient.

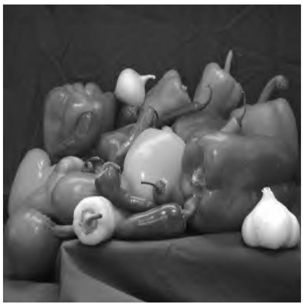
(a) Original image

(b) CR = 5:1

(c) CR = 9:1



(d) CR = 15:1



(e) CR = 36:1

(f) CR = 43:1

Figure 3.3: Compression with different CR using bior4.4.

3.4. Conclusion

The DWT is a useful method to compress images. Turning the small detail coefficients into zero affects the image minimally. Which wavelet and thresholding should be used depends on the application. The soft thresholding method produces smoother images and the hard thresholding method produces more blocking artefacts, but sharper edges. In the case of storing fingerprints, the edges are more important and therefore compression using hard thresholding works better. On the other hand, in the case of storing vacation photos, you probably want smoother images, therefore soft thresholding is a better method. Furthermore, the different wavelets have their advantages and disadvantages. For example, the Haar wavelet is "square" shaped, compressing images using this wavelet causes blocking artefacts. However, the computation speed is higher compared to other wavelets. Thus, it could be used for compressing large data. The Daubechies and Symlet have an overlapping window size of 2 and the Coiflet wavelet even more. Because of this property, using these wavelets gives smoother images. The characteristics of Biorthogonal and Reverse Biorthogonal wavelets can be varied by changing the various properties of the wavelet such as orthogonality and symmetry [8]. These wavelets are more flexible and therefore suitable for image compression. In conclusion, based on the SSIM values, the bior4.4 wavelet yields the best performance in compressing the peppers image.

4 Image Denoising

Denoising is the process of removing the noise of an image. A more detailed explanation of noise and its origins may be found in Chapter 1. To compare the denoising techniques, the original and noisy images are required. For this reason, noise is added to an image instead of using an already noisy image. This chapter consists of three sections. The first section explains what noise in an image mathematically implies, and section 4.2 describes the denoising method and the influence of the threshold and level of decomposition. The last section shows some results of different denoising techniques.

4.1. Noisy images

The fact that an image can be represented as a matrix is already known. Denote the original 512x512 image by a matrix A, then the 512x512 matrix of the noisy image B given by

$$B = A + \epsilon \cdot \sigma, \tag{4.1}$$

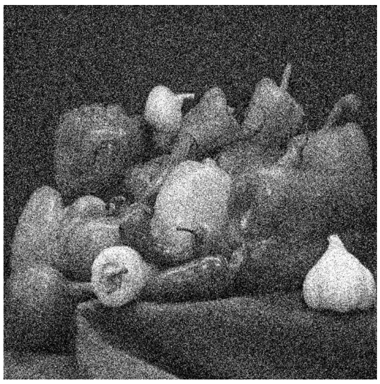
where $\epsilon \sim \mathcal{N}(0, 1)$ denotes the 512x512 noise matrix and σ the standard deviation of the noise. The normal distribution function is given by

$$\phi(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}, \tag{4.2}$$

where μ is the mean and σ the standard deviation. Note that the variance is σ^2. The larger the variance, the higher the probability of adding a large noise term. The effect of different values of σ is shown in Figure 4.3.

(a) Image with $\mu = 0$ and $\sigma = 10$.

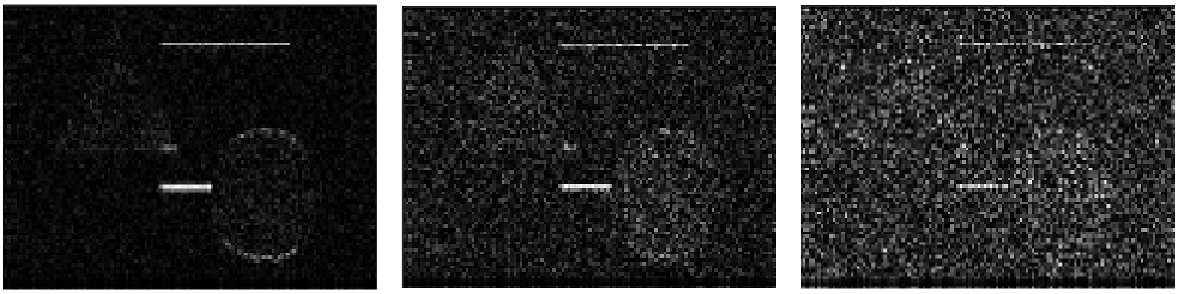
(b) Image with $\mu = 0$ and $\sigma = 25$.



(c) Image with $\mu = 0$ and $\sigma = 50$.

Figure 4.1: Effect of different values of σ.

In Figure 4.2, the horizontal detail coefficients are plotted. The image is decomposed to level 2 using the db4 wavelet. In Figure 4.2a, noise with $\sigma = 25$ is added, the triangle, circle and square are still visible. Adding more noise, the triangle (which can be seen as a small detail) is lost, see Figure 4.2c. Thus we can conclude: that the more noise, the more small details are lost.



(a) $\sigma = 25$ (b) $\sigma = 50$ (c) $\sigma = 100$

Figure 4.2: The horizontal detail coefficients of several standard deviation.

4.2. Denoising method

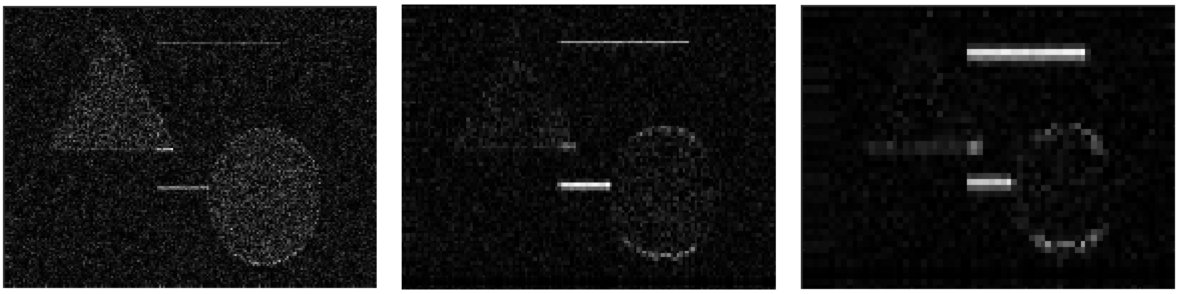
After the noisy image is generated, the denoising process can start. The denoising process consists of three steps:

- Decomposition: Choose a wavelet and decompose it to level N, where N = 1...5.
- Thresholding: Select a threshold method
- Reconstruction: Use the approximation and thresholded detail coefficients to reconstruct the image.

The Daubechies, Symlet, Coiflet, Biorthogonal and Dual-Tree Complex wavelets are considered to decompose and reconstruct images. As stated before, both threshold methods perform adequately, which threshold method works better depends on what you want to achieve. For denoising, the soft threshold method is used.

4.2.1. Level of decomposition

The different levels of decomposition split the image into layers with different levels of detail. The db4 wavelet is used to decompose the image shown in Figure 2.11. Figure 4.3 shows the horizontal detail coefficients on several levels of decomposition. In Figure 4.3a, the noise can be clearly seen. This level contains all the fine details of the image. The higher the level of decomposition, the more noise is averaged out. In Figure 4.3c, most of the noise is averaged out, but some details are also gone. so only the main features are visible.

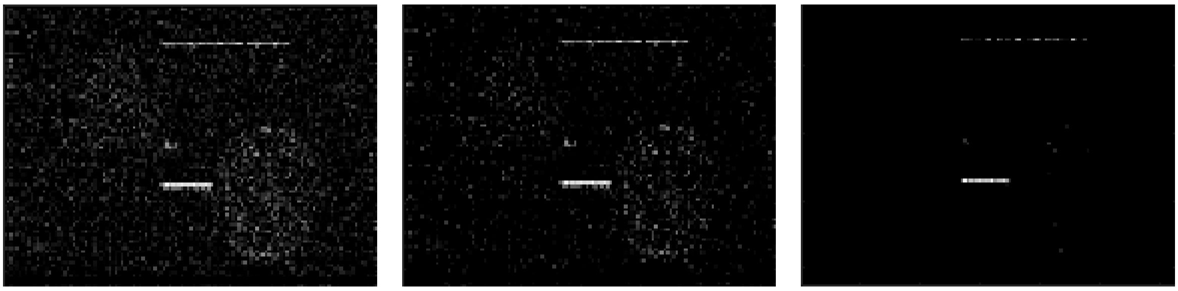


(a) Decomposition of level 1 (b) Decomposition of level 2 (c) Decomposition of level 3

Figure 4.3: The horizontal detail coeficients of a noisy image decomposed using the db4 wavelet.

4.2.2. Threshold

The optimal threshold is dependent on the noise's variance. As explained above, the noise is added to the original image and the higher the noise variance, the higher the probability of adding a large noise term to the image. Therefore, the larger the noise variance, the larger the threshold coefficient has to be in order to remove the noise. However, if the threshold value is too large, the small details are also removed. In other words, in wavelet transformation, a small difference between the adjacent coefficients could mean it is noise, but it could also be a small detail of the image. In Figure 4.2b, a noisy image with $\sigma = 25$ is shown. The image is decomposed to level 2 and the soft thresholding method is applied using different thresholds, shown in Figure 4.4. Thus, the higher the threshold, the more noise is removed, but also the more details are lost.



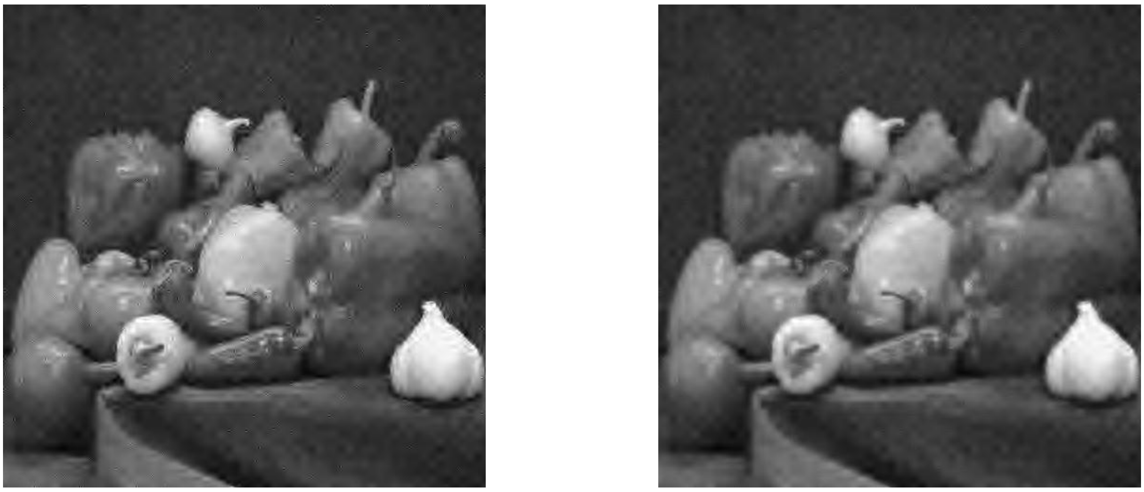
(a) $\lambda = 25$ (b) $\lambda = 50$ (c) $\lambda = 150$

Figure 4.4: The horizontal detail coeficients of a noisy image decomposed using the db4 wavelet.

When using the Dual Tree Complex Wavelet, it is more effective to threshold the magnitude than the real and imaginary parts separately. Thresholding the magnitude results in a nearly shift-invariant denoising process [11].

The threshold methods are discussed in section 3.1. The two different threshold methods are hard and soft thresholding. As mentioned before, hard thresholding is better in preserving edges, but performs less well in denoising, and soft thresholding is better at denoising but often smooths out the edges, such that the resulting edges are less sharp.

Figure 4.5 shows the denoised image using hard and soft thresholding with the same wavelet, db4 and an optimised threshold. The threshold is optimised by minimizing the SSIM value. The optimal threshold for Figure 4.5a is 91 and the optimal threshold Figure 4.5b is 71. Indeed, the image denoised using the hard thresholding method contains more noise, but the edges are sharper and the image denoised using the soft thresholding method is less noisy but some edges are less sharp.



(a) Hard Thresholding (b) Soft Thresholding

Figure 4.5: Denoised image using the db4 wavelet.

4.2.3. Quality Measurement method

Consequently, an optimal threshold coefficient must be found. Since the PSNR and SSIM are both good for capturing noise, the threshold coefficient will be optimized by minimizing the PSNR and the SSIM. The Peak Signal to Noise Ratio (PSNR) is the ratio between the maximum possible pixel value and the MSE. The higher the PSNR, the better the quality of an image. The formula is given by

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right), \tag{4.3}$$

where MAX_I is the maximum possible pixel value. The MSE measures the average squared error between the denoised image and the original image. The formula is given by

$$MSE = \frac{1}{m \cdot n} \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} [O(i,j) - D(i,j)]^2, \tag{4.4}$$

where m and n denotes the dimension of the image, $O(i,j)$ denotes the pixels of the original image, $D(i,j)$ denotes pixels of the denoised or compressed image. A small error implies that the difference between the denoised image and the original image is smaller.

4.3. Results

In order to find a suitable wavelet for denoising, several tests need to be performed. The wavelets we consider are the Daubechies, Symlet, Coiflet, Biorthogonal and the Dual Tree Complex wavelets. All these wavelets could be used in combination with different filter lengths. Other things we should consider: How many times do we decompose the wavelet? And which threshold method do we use? The more changing variables, the more tests need to be performed and the more difficult it is to see the changing effect.

First, the pepper image with Gaussian noise, where the mean equals 0 and the noise standard deviation equals 25, is considered. The Daubechies, Symlet, Coiflet, Biorthogonal and Reverse Biorthogonal wavelets are tested using the built-in MATLAB function wdenoise2(). Both the soft and hard threshold methods are tested. In this function, the threshold is calculated using the formula introduced by Donoho and Johnstone [3]. The formula of the threshold λ is given by

$$\lambda = \sigma * \sqrt{2 * log(T)}, \tag{4.5}$$

where T is the number of pixels and σ is the standard deviation of the noise. The noise standard deviation is estimated by the robust median estimator [1]. The formula can only be used for Gaussian white noise and is given by

$$\hat{\sigma} = \frac{Median(|y_{i,j}|)}{0.6745}, \tag{4.6}$$

where $y_{i,j}$ are the coefficients of the first level subband HH, see Figure 2.10. However, the threshold coefficient calculated using this formula is not optimal. The PSNR and SSIM values of the original image and the denoised image are shown in appendix A. The best results are given by using the db4, coif3 and bior2.8 wavelets. Therefore, the db4, coif3 and bior 2.8 will be evaluated.

4.3.1. Threshold

The influence of the threshold coefficient is shown in Figure 4.6. The db4 wavelet is used and the level three detail coefficients are thresholded by different values. The higher the threshold coefficient, the more detail coefficients are thresholded, and the more noise is removed. Consequently, the higher the threshold coefficient, the fewer edges are preserved. Calculating the threshold using equation 4.5 gives $\lambda = 84.6$. Increasing the threshold initially results in significant changes in the denoised image. However, this only holds for an approximate threshold of up to 60, after which increasing the threshold does not result in a significant different output figure.

(a) Threshold = 20

(b) Threshold = 40

(c) Threshold = 60

(d) Threshold = 80

(e) Threshold = 100

(f) Threshold = 200

Figure 4.6: Denoising using db4 wavelet and hard thresholding.

The threshold is plotted against the PSNR in Figure 4.7. This gives the same result as above. Initially, the higher the threshold, the higher the PSNR, which means that the denoised image is better. At a threshold higher than 60, the function does not increase significantly anymore. This threshold will be used. The threshold used hereafter will be determined in the same way. For the different wavelets, this point, where the PSNR does not increase significantly, differs.

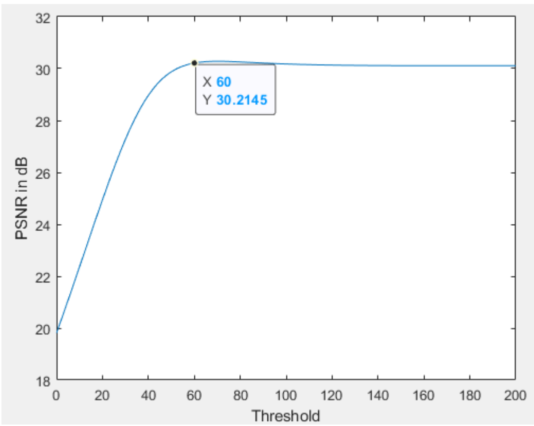


Figure 4.7: The PSNR plotted against the threshold.

4.3.2. Comparing db4, coif3, bior2.8

First, the hard thresholding method is considered. The db4, coif3 and bior2.8 are decomposed to several levels. For each level, the threshold is determined using the method of section 4.3.1. The results are shown in Table B.1 in Appendix B. For all three wavelets, the highest PSNR value is obtained by thresholding the level 3 detail coefficients. For all levels, the bior2.8 wavelet gives the highest PSNR value, followed by the coif3 wavelet. The lowest PSNR values are obtained by using the db4 wavelet. Overall, the highest value is obtained using the bior2.8 wavelet together with the soft thresholding method. The corresponding PSNR value is 30.4074 dB.

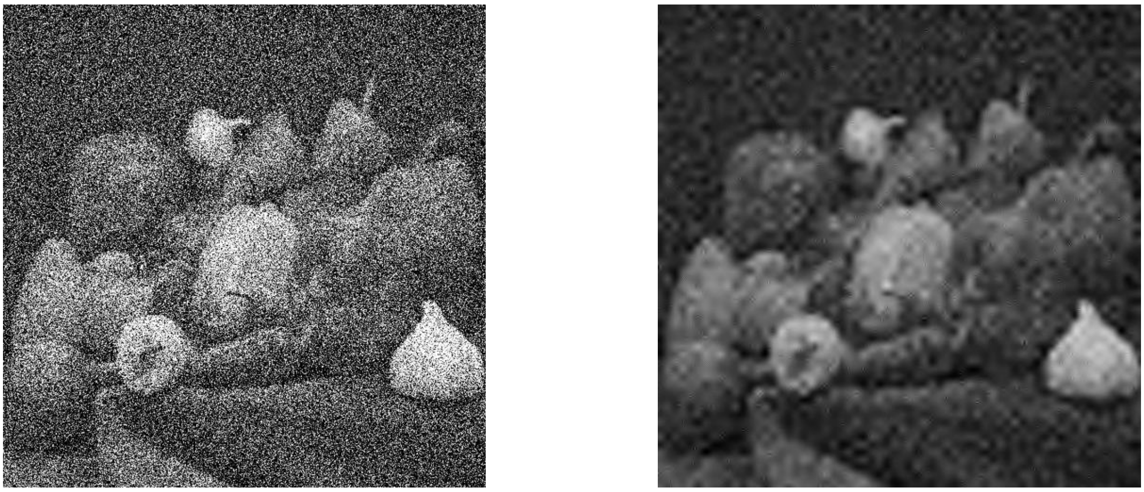
4.3.3. Images with different noise variance

This section discusses the influence of the noise standard deviation on the PSNR, SSIM and threshold. The coif3 wavelet is used to denoise the pepper image with $\sigma = 10, 25, 50, 100, 200$ and the level 4 coefficients are thresholded. The results are shown in table 4.1. It can be concluded from this table that the higher the noise variance, the lower the PSNR and SSIM values, which makes sense. The more noise, the harder it is to denoise the image.

	Soft Thresholding (Coif3)
$\sigma = 10$	34.750 0.917
$\sigma = 25$	29.681 0.856
$\sigma = 50$	26.751 0.818
$\sigma = 100$	23.066 0.783
$\sigma = 200$	18.792 0.746

Table 4.1: PSNR and SSIM values with optimised threshold.

Furthermore, the noise variance is correlated with the optimal threshold in the following way: the more noise, the higher the optimal threshold. The optimised threshold coefficient for the noisy image with $\sigma = 200$ equals 211. The noisy and denoised image is shown in Figure 4.8. The noise added is normally distributed with a mean of 0 and a standard deviation of 200. The noise coefficient is sometimes even larger than the original image coefficients. For this reason, not only the noise is thresholded, but also some important features of the original image.



(a) Noisy image with noise standard deviation equals 200

(b) Denoised image

Figure 4.8: Denoising an image with a lot of noise.

4.3.4. DWT vs DTCWT

In section 4.3.2, the db4, coif3 and bior2.8 are compared. The bior2.8 wavelet outperforms the other two, therefore, the bior2.8 wavelet will be compared to the Dual-Tree Complex Wavelet (DTCW). The noisy image with $\sigma = 25$ and the pepper image denoised using bior2.8 is shown in Figure 4.9a and 4.9, respectively. For the DWT, soft thresholding does not preserve edges that well. Using the DTCWT together with soft thresholding, the edges are preserved and there is no noise anymore. However, there are still some small features removed in the background.

(a) Noisy image (b) Denoised using bior2.8 (c) Denoised using DTCWT

Figure 4.9: Comparing the noisy image with the denoised image using bior2.8 and DTCWT.

4.4. Conclusion

In this chapter, the performance of the DWT and the DTCWT in image denoising is evaluated. The effect of more noise in an image is shown, as well as the effect of the noise in high-frequency sub-bands. The denoising method consists of three steps: decomposition, thresholding and reconstruction. The influence of the choice of the wavelet has already been discussed in section 3.4. The smoother the wavelet and the more overlapping windows, the less blocking artefacts. The db4, coif3 and bior2.8 are the discrete wavelets used to denoise the pepper image. Furthermore, the threshold and level of decomposition are crucial in the denoising process. The wavelet decomposition splits the details into several layers. The higher the level of decomposition, the more noise is averaged out, but also the more details are lost. The optimal level of decomposition is often between levels 3 and 5. The optimal threshold depends on the variance of the noise. The higher the noise variance, the higher the probability of adding large noise terms. Therefore, a larger threshold is needed to remove the noise. However, the larger the noise terms, the more details are lost. Therefore, if the noise variance is too large, it is almost impossible to denoise. To denoise the noisy pepper image with $\sigma = 25$, the threshold is around 60. Taking the threshold and the level of decomposition into account, the bior2.8 wavelet performs the best. Comparing the bior2.8 wavelet with the DTCW, the DTCW outperforms the discrete wavelets. The image denoised using the DTCW is smoother than the one denoised using the bior2.8 wavelet. Due to the near shift-invariance and the directionally selective property, the DTCW is better at edge detection, which led to sharp edges and a smoother image.

5 Conclusion

This research report has laid its emphasis on exploring the potential of wavelets in the field of image processing. In particular, image denoising and compression techniques are chosen as image processing techniques evaluated in this research report. This exploration has been conducted by applying the Discrete Wavelet Transform (DWT) technique and its extensions, such as Dual Tree Complex Wavelet Transform (DTCWT), with the usage of a broad variety of wavelets.

The images used in this research for wavelet transform are retrieved from MATLAB with a size of 512x512 pixels. In contrast to using the general RGB values of the images, this research has limited to using the 256 grey scales of them.

Consequently, these images are used as the basis to apply the image processing techniques on. The original image is deconstructed using DWT or DTCWT, which constructs approximation and detail coefficients. The detail coefficients are determined by taking the differences, and the approximation coefficients are determined by averaging the coefficients. Afterwards, the image can be reconstructed again using the same DWT or DTCWT.

In this report, wavelets are used to compress and denoise images. The compression and denoising methods are similar. These methods consists of three steps, decomposition, thresholding and reconstruction. The difference is in the thresholding step. In image compression, we choose a percentage of the detail coefficient to apply thresholding on. In image denoising, we try to find an optimal threshold in order to remove noise.

The ability to turn the small wavelet coefficients to zero with a minimal effect on the image is very useful in image compression. Compared to db6, sym5, coif3 and rbio1.5, the bior4.4 wavelet performed best. This is true for all compression rates tested. Setting 95% of the detail coefficients to zero still yields sharp images without blocking artefacts. However, how much the image is reduced in size or how much space is saved by saving the compressed image instead of the original image can not be determined. This depends on the way of coding, which has not been investigated in this research.

Furthermore, for image denoising, the db4, coif3, bior2.8 and the Dual Tree Complex Wavelets are compared. Besides the choice of wavelet, the threshold method is also important. Hard thresholding is better for preserving edges but worse for denoising, and soft threshing is better for denoising but worse for edge preservation. Compared to the discrete wavelets, the Dual Tree Complex wavelets are much better at denoising and preserving edges.

Finally, a conclusion can be provided to answer the key question of this research report of exploring the potential of wavelets in the field of image processing. Wavelet transformation is a powerful tool in terms of image processing. The choice of the optimal wavelet can not be generalised for all scenarios, it depends on the application of the image. For example, in the application of image denoising for photos, the usage of soft thresholding and smoother wavelets, such as the discrete wavelet coif3 or the Dual Tree Complex wavelet seems like a better choice. However, in the application of compressing or denoising fingerprint images, hard thresholding could be a better choice, since the method is better for preserving edges. As mentioned in Chapter 1, storing fingerprints costs a lot of memory, which forms a reason to apply compression. Consequently, the more complex the method, the longer it takes before the image has been compressed and stored. This could be a reason to choose a simpler wavelet, for example, the db2 wavelet.

A

Comparing different Threshold Methods for Denoising Gaussian Noise

PSNR values

	db1	db2	db3	db4	db5	db6	db7	db8	db9	db10
lv1	22.825	22.9028	22.8343	22.8746	23.0224	23.1739	23.1452	23.0117	22.9489	22.9398
lv2	23.6417	23.9383	23.8455	23.9516	24.1694	24.267	24.2923	24.1818	24.0853	24.0442
lv3	23.5855	23.9566	23.9076	23.9571	24.2166	24.2709	24.2612	24.159	24.1126	24.0742
lv4	23.5013	23.8723	23.8126	23.8813	24.1128	24.1492	24.1377	24.0543	23.996	23.9515
lv5	23.4814	23.8571	23.8054	23.856	24.0819	24.115	24.1045	24.0258	23.9556	23.9101

Table A.1: Daubechies, Hard Thresholding, Universal Threshold

	db1	db2	db3	db4	db5	db6	db7	db8	db9	db10
lv1	23.0956	23.0047	22.8433	22.9529	23.1367	23.2514	23.2572	23.1713	23.0576	22.981
lv2	23.9961	24.3036	23.8484	24.2476	24.3595	24.2348	24.4147	24.498	24.437	24.3461
lv3	23.7431	24.0921	23.9873	24.1264	24.2864	24.2181	24.2617	24.3411	24.3648	24.2496
lv4	23.3408	23.7127	23.5978	23.8045	23.8611	23.8148	23.8445	23.9412	23.9667	23.8468
lv5	23.1439	23.5026	23.3432	23.5593	23.6505	23.5604	23.6287	23.6846	23.6726	23.5657

Table A.2: Daubechies, Soft Thresholding, Universal Threshold

	sym2	sym3	sym4	sym5	sym6	sym7	sym8
lv1	22.9028	22.8343	22.9501	22.9431	22.9546	22.7599	22.966
lv2	23.9383	23.8455	23.9953	24.0066	24.0161	23.7988	24.0537
lv3	23.9566	23.9076	24.0509	24.0671	24.0677	23.8614	24.1218
lv4	23.8723	23.8126	23.9705	23.9696	23.9758	23.7948	24.0236
lv5	23.8571	23.8054	23.9625	23.9338	23.9543	23.7657	23.9972

Table A.3: Symlet, Hard Thresholding, Universal Threshold

	sym2	sym3	sym4	sym5	sym6	sym7	sym8
lv1	23.0047	22.8433	23.2581	23.1257	23.2522	22.8978	23.2487
lv2	24.3036	23.8484	24.3811	24.3391	24.4626	24.2314	24.4598
lv3	24.0921	23.9873	24.4082	24.3704	24.4516	24.2687	24.4408
lv4	23.7127	23.5978	24.0915	23.987	24.1302	23.9415	24.1153
lv5	23.5026	23.3432	23.8314	23.7292	23.8733	23.7187	23.873

Table A.4: Symlet, Soft Thresholding, Universal Threshold

	coif1	coif2	coif3	coif4	coif5
lv1	22.9532	22.9628	22.9781	22.9957	23.0212
lv2	23.9539	24.0333	24.0824	24.1094	24.1519
lv3	23.9429	24.1088	24.1512	24.1619	24.2151
lv4	23.8743	24.0267	24.0612	24.0824	24.116
lv5	23.8738	23.993	24.0323	24.0584	24.0837

Table A.5: Coiflet, Hard Thresholding, Universal Threshold

	coif1	coif2	coif3	coif4	coif5
lv1	23.2802	23.2647	23.2531	23.2445	23.2373
lv2	24.3888	24.4559	24.4864	24.4853	24.4944
lv3	24.1658	24.4566	24.5196	24.4395	24.5154
lv4	23.844	24.1374	24.1276	24.1232	24.187
lv5	23.6047	23.8636	23.8845	23.8882	23.918

Table A.6: Coiflet, Soft Thresholding, Universal Threshold

	bior1.1	bior1.3	bior1.5	bior2.2	bior2.4	bior2.6	bior2.8	bior3.1	bior3.3	bior3.5	bior3.7	bior3.9	bior4.4	bior5.5	bior6.8
lv1	22.825	22.727	22.632	22.9431	22.9644	22.9491	22.9265	22.1086	22.6874	22.8046	22.8453	22.8633	22.9554	22.8243	23.0119
lv2	23.6417	23.5312	23.4674	23.7892	23.8369	23.8736	23.814	21.2018	22.4031	22.7026	22.7813	22.8614	24.0333	23.9476	24.0823
lv3	23.5855	23.5222	23.5221	23.8364	23.9181	23.958	23.8954	21.0206	22.2554	22.6052	22.696	22.7747	24.0829	23.8499	24.1588
lv4	23.5013	23.4361	23.4309	23.8249	23.9096	23.9544	23.8778	20.994	22.2209	22.5744	22.6667	22.7582	24.0062	23.5446	24.0934
lv5	23.4814	23.4217	23.408	23.8166	23.9059	23.951	23.8798	20.993	22.2193	22.5715	22.664	22.748	23.9651	23.4104	24.0784

Table A.7: Biorthogonal, Hard Thresholding, Universal Threshold

	bior1.1	bior1.3	bior1.5	bior2.2	bior2.4	bior2.6	bior2.8	bior3.1	bior3.3	bior3.5	bior3.7	bior3.9	bior4.4	bior5.5	bior6.8
lv1	23.0956	22.9615	22.8496	23.2744	23.2834	23.2555	23.2267	22.1788	22.7896	22.924	22.97	22.9884	23.2486	22.8529	23.2447
lv2	23.9961	23.796	23.7987	24.4741	24.4556	24.4904	24.4087	22.7769	23.8686	24.0217	24.0733	24.1287	24.4254	24.2673	24.4141
lv3	23.7431	23.7033	23.7468	24.4343	24.5715	24.6113	24.5483	22.7356	23.9582	24.2034	24.2453	24.2838	24.4072	24.0974	24.4243
lv4	23.3408	23.4461	23.503	24.3109	24.44	24.5246	24.4377	22.6389	23.9	24.148	24.1612	24.2643	24.0449	23.4986	24.1323
lv5	23.1439	23.3636	23.4164	24.1765	24.2845	24.3931	24.3506	22.5896	23.8727	24.1136	24.1025	24.206	23.7246	22.9472	23.9301

Table A.8: Biorthogonal, Soft Thresholding, Universal Threshold

	rbio1.1	rbio1.3	rbio1.5	rbio2.2	rbio2.4	rbio2.6	rbio2.8	rbio3.1	rbio3.3	rbio3.5	rbio3.7	rbio3.9	rbio4.4	rbio5.5	rbio6.8
lv1	22.825	22.8854	22.8068	22.8606	22.9507	22.949	22.9359	21.2853	22.7242	22.8724	22.8955	22.8878	22.9274	22.8091	22.9718
lv2	23.6417	23.9776	23.9339	23.5768	23.9374	23.9809	23.9862	18.4054	23.0514	23.5791	23.8264	23.788	23.9456	23.8337	24.0203
lv3	23.5855	24.0104	23.9766	23.0961	23.7304	23.8221	23.8151	14.459	21.1814	22.6442	23.0608	23.1631	23.9781	23.9451	24.0504
lv4	23.5013	23.8963	23.8557	22.6876	23.3706	23.455	23.4789	12.2193	19.2208	21.4786	22.0194	22.2546	23.8728	23.9397	23.9394
lv5	23.4814	23.8579	23.8078	22.5663	23.2411	23.3278	23.3717	11.2726	18.684	21.0475	21.6071	21.9345	23.8523	23.9399	23.9198

Table A.9: Reverse Biorthogonal, Hard Thresholding, Universal Threshold

	rbio1.1	rbio1.3	rbio1.5	rbio2.2	rbio2.4	rbio2.6	rbio2.8	rbio3.1	rbio3.3	rbio3.5	rbio3.7	rbio3.9	rbio4.4	rbio5.5	rbio6.8
lv1	23.0956	23.1506	23.0688	23.1796	23.2589	23.2493	23.2294	21.3238	22.7688	22.9349	22.9779	22.9888	23.2258	22.8298	23.2393
lv2	23.9961	24.3288	24.377	24.0207	24.3834	24.4572	24.4136	18.3928	23.5639	24.0616	24.3098	24.2682	24.3949	24.2262	24.4256
lv3	23.7431	24.2489	24.2867	23.4202	24.1849	24.2716	24.2449	14.5881	22.3373	23.7233	24.0475	24.0757	24.4001	24.3577	24.3924
lv4	23.3408	23.7918	23.8158	22.7843	23.6029	23.7047	23.6737	12.3219	20.4142	22.7431	23.1923	23.3473	24.1037	24.2864	24.0395
lv5	23.1439	23.5322	23.5398	22.3735	23.1245	23.2169	23.2298	11.0491	19.2712	22.0817	22.5439	22.6972	23.8702	24.1846	23.8001

Table A.10: Reverse Biorthogonal, Soft Thresholding, Universal Threshold

SSIM

	db1	db2	db3	db4	db5	db6	db7	db8	db9	db10
lv1	0.49454	0.50205	0.50088	0.49931	0.49976	0.50197	0.50332	0.50272	0.50179	0.50054
lv2	0.72528	0.76093	0.7713	0.77104	0.76965	0.77422	0.77416	0.76911	0.76986	0.77196
lv3	0.76487	0.81711	0.8226	0.82377	0.82113	0.82288	0.81983	0.8178	0.81351	0.81556
lv4	0.76355	0.81425	0.81818	0.81994	0.81677	0.81672	0.81316	0.81107	0.80581	0.80603
lv5	0.76328	0.81348	0.81718	0.81858	0.81537	0.81557	0.81193	0.80954	0.80431	0.80455

Table A.11: Daubechies, Hard Thresholding, Universal Threshold

	db1	db2	db3	db4	db5	db6	db7	db8	db9	db10
lv1	0.49532	0.50246	0.50106	0.49971	0.50037	0.50263	0.50393	0.5033	0.50196	0.50051
lv2	0.72328	0.76176	0.77104	0.77464	0.77245	0.77381	0.77597	0.7748	0.77566	0.77691
lv3	0.74941	0.80644	0.81872	0.8211	0.82366	0.82396	0.8214	0.82422	0.82269	0.82172
lv4	0.7419	0.79751	0.81036	0.81169	0.81469	0.81153	0.81218	0.81269	0.81131	0.80857
lv5	0.74432	0.79619	0.80697	0.80839	0.81137	0.80792	0.80887	0.80808	0.80695	0.80429

Table A.12: Daubechies, Soft Thresholding, Universal Threshold

	sym2	sym3	sym4	sym5	sym6	sym7	sym8
lv1	0.50205	0.50088	0.5032	0.50325	0.50297	0.50045	0.50265
lv2	0.76093	0.7713	0.77388	0.77277	0.77371	0.77283	0.77537
lv3	0.81711	0.8226	0.82962	0.82879	0.82861	0.82746	0.82952
lv4	0.81425	0.81818	0.82541	0.82315	0.82244	0.82263	0.82313
lv5	0.81348	0.81718	0.82445	0.82186	0.82134	0.82117	0.82206

Table A.13: Symlet, Hard Thresholding, Universal Threshold

	sym2	sym3	sym4	sym5	sym6	sym7	sym8
lv1	0.50246	0.50106	0.50411	0.50377	0.50388	0.5009	0.50354
lv2	0.76176	0.77104	0.7749	0.77425	0.77589	0.77536	0.77656
lv3	0.80644	0.81872	0.82621	0.82609	0.82874	0.82746	0.82882
lv4	0.79751	0.81036	0.81698	0.81542	0.8183	0.81862	0.81857
lv5	0.79619	0.80697	0.81303	0.81102	0.81397	0.81415	0.81454

Table A.14: Symlet, Soft Thresholding, Universal Threshold

	coif1	coif2	coif3	coif4	coif5
lv1	0.50186	0.50251	0.50231	0.50194	0.5017
lv2	0.76378	0.774	0.77547	0.77547	0.77521
lv3	0.8183	0.83147	0.83094	0.82933	0.82794
lv4	0.81516	0.82587	0.82579	0.82371	0.82241
lv5	0.81413	0.82456	0.82448	0.82231	0.82102

Table A.15: Coiflet, Hard Thresholding, Universal Threshold

	coif1	coif2	coif3	coif4	coif5
lv1	0.50273	0.50338	0.5031	0.50273	0.50242
lv2	0.76452	0.77535	0.77676	0.77667	0.77653
lv3	0.80681	0.82828	0.82962	0.82836	0.82889
lv4	0.79897	0.81916	0.81969	0.81864	0.81918
lv5	0.79656	0.81517	0.81523	0.81421	0.81449

Table A.16: Coiflet, Soft Thresholding, Universal Threshold

	bior1.1	bior1.3	bior1.5	bior2.2	bior2.4	bior2.6	bior2.8	bior3.1	bior3.3	bior3.5	bior3.7	bior3.9	bior4.4	bior5.5	bior6.8
lv1	0.49454	0.48105	0.46926	0.49664	0.50135	0.4993	0.4963	0.40908	0.4717	0.48598	0.4905	0.49221	0.4989	0.49209	0.50147
lv2	0.72528	0.71154	0.69783	0.71689	0.72966	0.7358	0.72748	0.34905	0.46139	0.50145	0.50959	0.52099	0.77131	0.76904	0.77359
lv3	0.76487	0.75944	0.75227	0.75879	0.773	0.78059	0.76989	0.34045	0.44998	0.49288	0.50212	0.51417	0.83226	0.81708	0.832
lv4	0.76355	0.75714	0.7495	0.7592	0.77392	0.78128	0.7699	0.33985	0.44879	0.49172	0.50128	0.51352	0.82934	0.79657	0.82988
lv5	0.76328	0.75594	0.74792	0.75894	0.7737	0.78119	0.76979	0.33986	0.44881	0.49184	0.50128	0.5133	0.82788	0.79149	0.82883

Table A.17: Biorthogonal, Hard Thresholding, Universal Threshold

	bior1.1	**bior1.3**	**bior1.5**	**bior2.2**	**bior2.4**	**bior2.6**	**bior2.8**	**bior3.1**	**bior3.3**	**bior3.5**	**bior3.7**	**bior3.9**	**bior4.4**	**bior5.5**	**bior6.8**
lv1	0.49532	0.48172	0.46988	0.49805	0.50288	0.50079	0.49776	0.41641	0.47957	0.49399	0.49821	0.49937	0.49979	0.49242	0.50222
lv2	0.72328	0.71034	0.69516	0.76402	0.77355	0.77552	0.77129	0.53322	0.68962	0.72819	0.73427	0.74291	0.77402	0.77059	0.77669
lv3	0.74941	0.74723	0.74271	0.83532	0.84022	0.84302	0.84046	0.56089	0.73437	0.77488	0.78451	0.79123	0.83021	0.81774	0.83186
lv4	0.7419	0.74484	0.74214	0.83733	0.84073	0.84385	0.84132	0.56139	0.734	0.77449	0.78383	0.79082	0.81949	0.79397	0.82454
lv5	0.74432	0.7475	0.74489	0.83544	0.83879	0.84245	0.83993	0.56101	0.73313	0.77315	0.78228	0.78929	0.81455	0.78339	0.82091

Table A.18: Biorthogonal, Soft Thresholding, Universal Threshold

	rbio1.1	**rbio1.3**	**rbio1.5**	**rbio2.2**	**rbio2.4**	**rbio2.6**	**rbio2.8**	**rbio3.1**	**rbio3.3**	**rbio3.5**	**rbio3.7**	**rbio3.9**	**rbio4.4**	**rbio5.5**	**rbio6.8**
lv1	0.49454	0.49338	0.48189	0.49132	0.50191	0.50053	0.49755	0.34496	0.47193	0.49068	0.49524	0.49595	0.49801	0.49188	0.50141
lv2	0.72528	0.76077	0.75361	0.7125	0.76312	0.76905	0.77169	0.26678	0.653	0.73164	0.75143	0.75883	0.76242	0.74859	0.77315
lv3	0.76487	0.82338	0.82226	0.71526	0.79129	0.80349	0.80491	0.17643	0.58793	0.71504	0.74944	0.76083	0.815	0.8046	0.8223
lv4	0.76355	0.81992	0.81798	0.68219	0.76156	0.77247	0.77549	0.13145	0.46118	0.62182	0.66889	0.68456	0.80906	0.80526	0.81426
lv5	0.76328	0.81817	0.81612	0.67149	0.754	0.76714	0.77022	0.075345	0.40467	0.59305	0.64797	0.66857	0.8077	0.80531	0.8128

Table A.19: Reverse Biorthogonal, Hard Thresholding, Universal Threshold

	rbio1.1	**rbio1.3**	**rbio1.5**	**rbio2.2**	**rbio2.4**	**rbio2.6**	**rbio2.8**	**rbio3.1**	**rbio3.3**	**rbio3.5**	**rbio3.7**	**rbio3.9**	**rbio4.4**	**rbio5.5**	**rbio6.8**
lv1	0.49532	0.49414	0.48264	0.49232	0.5029	0.5015	0.49853	0.34733	0.47515	0.49397	0.49866	0.49949	0.4988	0.49228	0.50219
lv2	0.72328	0.76162	0.7554	0.71447	0.76535	0.77241	0.77372	0.26455	0.66242	0.74214	0.76166	0.76898	0.76411	0.7637	0.775
lv3	0.74941	0.81914	0.82027	0.7125	0.80115	0.81406	0.81668	0.16692	0.62076	0.75671	0.79166	0.80132	0.81778	0.83426	0.82521
lv4	0.7419	0.80658	0.80814	0.68072	0.77921	0.79197	0.79466	0.12333	0.5265	0.70914	0.75575	0.76449	0.81117	0.8365	0.81453
lv5	0.74432	0.80204	0.803	0.66884	0.76781	0.78149	0.78461	0.075161	0.45441	0.68316	0.73418	0.74487	0.80758	0.83493	0.81

Table A.20: Reverse Biorthogonal, Soft Thresholding, Universal Threshold

B

Comparing db4, coif3, bior2.8 for denoising

Wavelet	level	Opt. value	PSNR
db4	1	62.4	23.3802
db4	2	97.4	29.8997
db4	3	97.9	30.4615
db4	4	92.9	30.0587
db4	5	92.7	29.9404
coif3	1	63.7	23.5916
coif3	2	97.6	30.0209
coif3	3	97.9	30.6297
coif3	4	93.5	30.2255
coif3	5	92.3	30.1043
bior2.8	1	62.9	23.4974
bior2.8	2	106.5	29.2962
bior2.8	3	119.7	30.6696
bior2.8	4	116.1	30.3861
bior2.8	5	112.8	30.3033

Table B.1: Optimal threshold values for hard thresholding

Wavelet	level	Opt. value	PSNR
db4	1	20.4	22.2678
db4	2	34.2	25.8234
db4	3	44.0	28.5424
db4	4	48.3	29.4881
db4	5	49.0	29.585
coif3	1	20.5	22.2999
coif3	2	34	25.8715
coif3	3	44.4	28.6553
coif3	4	48.6	29.6816
coif3	5	49.4	29.7854
bior2.8	1	20.2	22.4274
bior2.8	2	34.3	25.6637
bior2.8	3	46.8	28.5088
bior2.8	4	54.7	29.9907
bior2.8	5	57.5	30.4074

Table B.2: Optimal threshold values for soft thresholding

Bibliography

[1] S Grace Chang, Bin Yu, and Martin Vetterli. Adaptive wavelet thresholding for image denoising and compression. *IEEE transactions on image processing*, 9(9):1532–1546, 2000.

[2] Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

[3] David L Donoho and Jain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3): 425–455, 1994.

[4] Michael W. Frazier. *An Introduction to Wavelets Through Linear Algebra*. Springer, 1 edition, 1999.

[5] Neil F Johnson and Sushil Jajodia. Exploring steganography: Seeing the unseen. *Computer*, 31(2):26–34, 1998.

[6] Satish Kumar. Classifying image data, 2001. URL `"https://www.debugmode.com/imagecmp/classify.htm"`.

[7] Piotr Porwik and Agnieszka Lisowska. The haar-wavelet transform in digital image processing: its status and achievements. *Machine graphics and vision*, 13(1/2):79–98, 2004.

[8] PMK Prasad and G Umamadhuri. Biorthogonal wavelet-based image compression. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, pages 391–404. Springer, 2018.

[9] Ida Mengyi Pu. *Fundamental data compression*. Butterworth-Heinemann, 2005.

[10] Ivan W Selesnick. The design of approximate hilbert transform pairs of wavelet bases. *IEEE Transactions on Signal Processing*, 50(5):1144–1152, 2002.

[11] Ivan W Selesnick, Richard G Baraniuk, and Nick C Kingsbury. The dual-tree complex wavelet transform. *IEEE signal processing magazine*, 22(6):123–151, 2005.

[12] Truong Nguyen Gilbert Strang. *Wavelets and Filter Banks*. Wellesley College, 2nd edition, 1996. ISBN 0961408871; 9780961408879. URL `libgen.li/file.php?md5=c0b5cf29f1301d5372d056086043ee2e`.

[13] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

[14] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error measurement to structural similarity. *IEEE transactions on image processing*, 13(1), 2004.