UPC-ETSETB

Proyecto final de carrera

A comparison of scalar and vector quantization of wavelet decomposed images

Author : Albane Delos

Adviser: Luis Torres

Table of contents

Table of figures

List of tables

Table of graphics

1. Introduction

Nowadays, images have become very commonly used, will it be on websites, photos or part of a video. However, without any treatment, an image can be very heavy. Storage capacity and bandwith are not unlimited and image compression [1] permits to reduce this size.

Images, if they are coded entirely, that is, each pixel is coded on 8 bits for black and white pictures and on 3*8 bits for color pictures can have a very huge sizes.

For example, with new cameras which can take photos of 10Mpixels, the size of a color image would be without compression of about 30MBytes.

Therefore, in order to save storage space and bandwidth, image compression is used. It aims to reduce redundancy on images by using compression techniques. Techniques have evolved a lot since the start of image compression, mainly thanks to the discovery of new techniques and the increase of processor power which makes it possible [2].

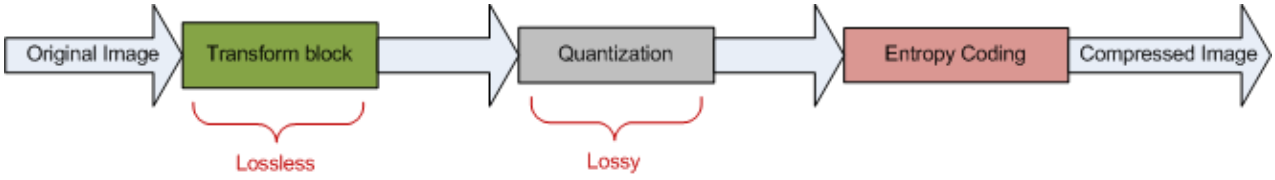


Figure 1 Blocks entering image compression

Image compression can be constituted of the following blocks: see Figure 1.

- Transform block: This is a reversible block, hence it introduces no losses. This block serves to prepare the information on the image in a more adequate working domain.; it is often a transform: Fourier transform, DCT, Wavelet transform for example.
- Quantization block: This block introduces losses, it is said lossy. Quantization aims to reduce the information needed by coding the image with a small number of values which are similar to the range available in the image. Quantization can be of two types: vectorial or scalar. Quantization is not an obligatory block. In the case of lossless compression, it is not used.
- Entropy Coding: The entropy coding block transforms the information of the image into a binary code. It considers the probability of each piece of information, allocating more bits to the low probability pieces and fewer bits to high probability pieces.

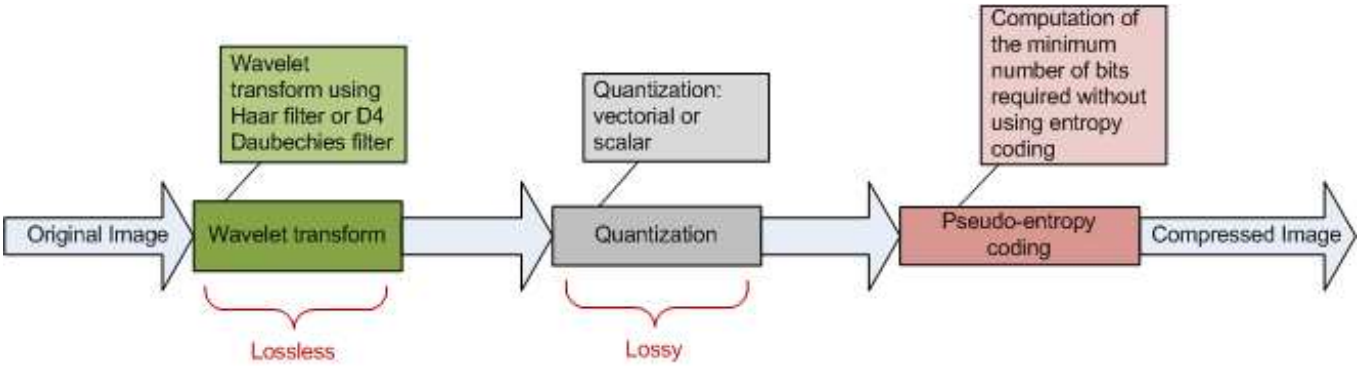


Figure 2 Blocks entering the project

Figure 2 shows the blocks entering the project.

In this project, the transform will be a Wavelet transform and two filters will be compared. Wavelet transform is a potential technique but yet not really used in standard image compression.

Then, quantization will be scalar or vectorial. The scalar quantization will use Max's tables and the vector quantization will use the LBG algorithm [3].

An estimation model will be used in order to obtain the compression rate. It will only consist of the computation of the minimum number of bits required to code the codebook without implementing the decoder.

Finally, a study of PSNR results and compression rate will be conveyed and conclusions will be drawn.

2. Wavelet theoretical background

2.1. The wavelet transform

Mathematical transformations are applied to signals to obtain further information from that signal that is not readily available in the raw signal.

Wavelet transform [4] is one of those transforms. Its principal advantage is that, contrary to Fourier transform [5], wavelet transform is local to both frequency and time. A change in the Fourier transform would affect the entire time signal; however, a change in a wavelet transform would affect only the time domain corresponding to that particular wavelet coefficient [6]. See Figure 3 for the time-frequency plane decomposition, the vertical axis shows the frequency and the horizontal one shows the time.

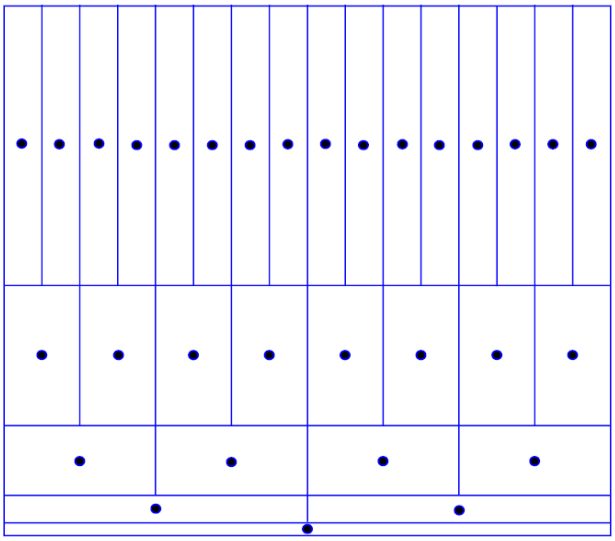


Figure 3 Time frequency plane decomposition

More technically, a wavelet is a mathematical function used to divide a given function or continuous-time signal into different scale components. Usually one can assign a frequency range to each scale component. Each scale component can then be studied with a resolution that matches its scale. A wavelet transform is the representation of a function by wavelets [7].

2.2. Wavelet decomposition

The Wavelet decomposition [8] is a way to decompose a signal in different sub-bands varying the frequency scale.

The mathematical representation of a continuous wavelet transform is:

$$CWT_x^{\psi}(\tau, s) = \Psi_x^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\psi^* \left(\frac{t-\tau}{s}\right) dt \qquad (2.1)$$

With τ the translation parameter, s the scale parameter and Ψ the mother wavelet [9].

Figure 4 shows a decomposition of a 1-D signal. s is the original signal and the sub-bands are decomposing the signal while increasing the frequency. $a5$ shows the average of the signal and on the contrary, $d1$ shows the smallest variation in frequency. Here the decomposition is effectuated until level 5 but the decomposition can continue depending on the frequency resolution wanted.

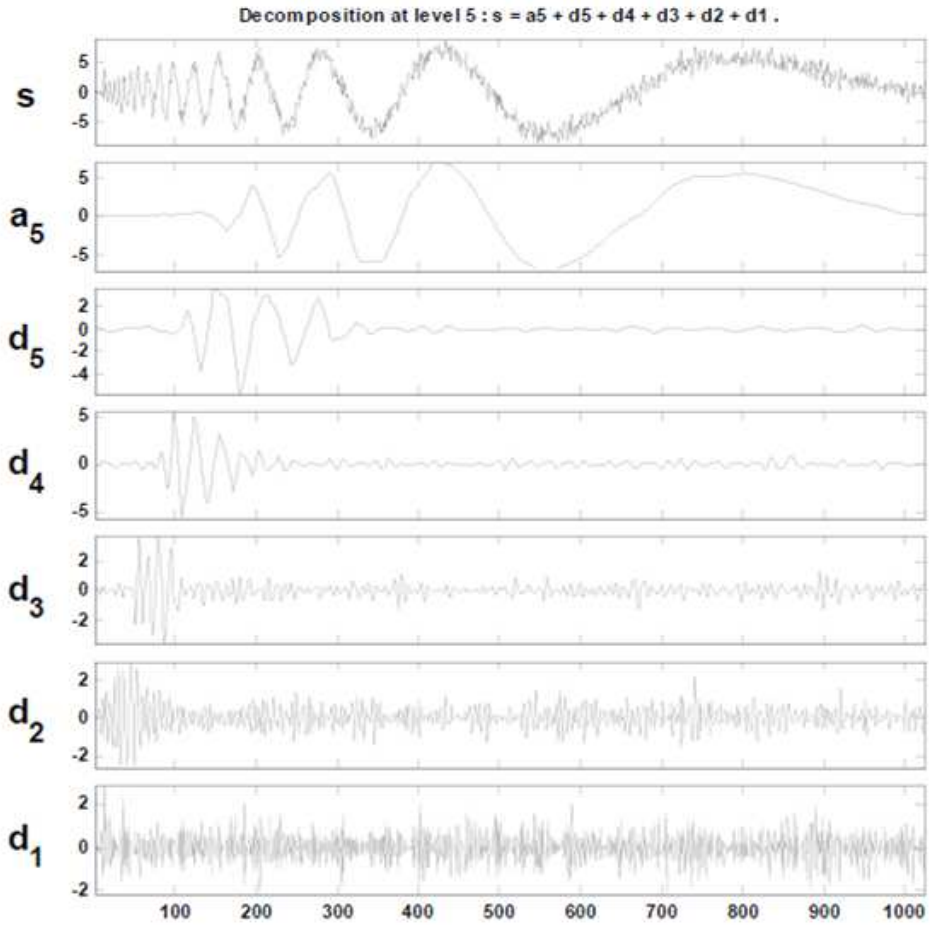


Figure 4 Wavelet decomposition of a 1-D signal, axis x represents the time, axis y represents the amplitude

Applied to 2-D signals, like images, wavelet decomposition permits decomposing the original image in sub-images of inferior resolution and corresponding to different frequency bands. In itself, the wavelet transform is lossless; it is the quantification which introduces losses.

To decompose an image [10], two filters are needed: a low pass filter H and a high pass filter G. The H filter will do an approximation of the image while filter G will retrieve high frequency information.

The image shall have a square size of $2^n * 2^n$. This is required in order for the decomposition to be conducted easily, each decomposition reducing the size of the image by 2 in both directions.

First, on the rows of the matrix, both filters H and G are applied. Two resulting matrices are obtained: fG and fH both of dimension $2^n * 2^{n-1}$. Then, on the columns of matrices fG and fH, filters G and H are applied again. Four resulting matrices fHH, fHG, fGH and fGG of dimension $2^{n-1} * 2^{n-1}$ are obtained, fHH being the average of the original matrix while matrices fHG, fGH and fGG are details.

The decomposition can be repeated from matrix fHH. An image can be decomposed until its resultant filtered matrices have a size of $2 * 2$. However, the decomposition is generally stopped before that, depending on the frequency sensibility wanted.

The resulting matrices are called shapes and the result of one decomposition is called a level. Figure 5 shows the results of two decompositions: level 1 and level 2 and their resulting matrices, shapes 0 to 4.

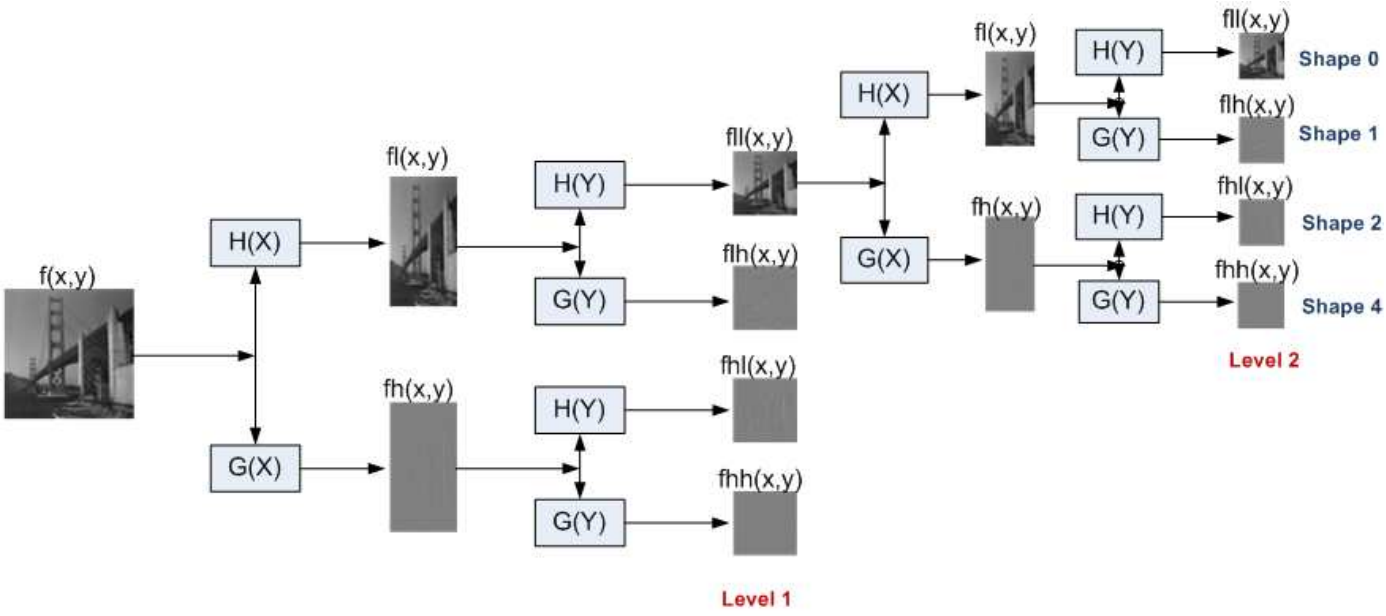


Figure 5 The wavelet decomposition

The original image can then be reconstructed without losses if no quantization is applied. The four shapes of the last level serve to reconstruct shape 0 of level N-1 and so on until the four shapes of level 1 are combined to retrieve the original image. The reconstruction is done applying the inverse of the filters previously applied in the decomposition.

Two filters have been studied in this project: the Haar filter and the D4 Daubechies filter.

Haar filter has been chosen because of its easiness. Daubechies D4 filter has been chosen because it is a well know filter widely used.

2.3. Haar wavelet

The Haar filter [11] was first proposed in 1909 by Alfréd Haar a Hungarian mathematician, a long time before the term wavelet was even known. Haar filter is also a special case of Daubechies wavelet, the D2 Daubechies wavelet [11].

It is the simplest filter which can be used for wavelet decomposition. The Haar wavelet is a step function taking values 1 and -1 on [0, ½) and [½, 1) respectively. See Figure 6 for the graph of Haar wavelet.

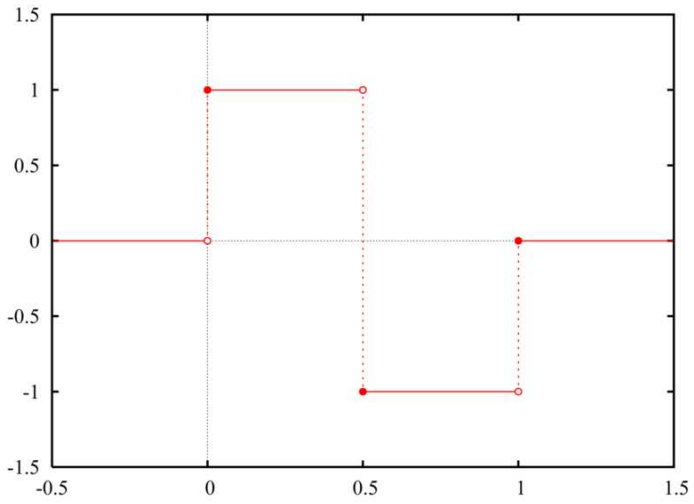


Figure 6 Haar wavelet, x represents time and y amplitude

The Haar wavelet is:

- Low pass filter $$H(X) = \frac{X_n + X_{n+1}}{2} \quad (2.2)$$

  The scaling function serves as a low pass filter and computes the mean of two adjacent pixels

- High pass filter $$G(X) = \frac{X_n - X_{n+1}}{2} \quad (2.3)$$

  The high pass filter computes the difference between two adjacent pixels over 2.

Because there are only two coefficients, it is very easy to implement, it only takes into account two adjacent pixels. The resolution of the resultant shape being divided by 2, we can see that filters do not overlap for two consecutive resultant pixels. The easiness of its implementation comes as well from the value of its coefficients: 0,5 and -0,5.

The reconstruction of the image is done using the following inverse filters which are retrieved from equations (1.1) and (1.2):

$$X_n = H(X) + G(X) \quad (2.4)$$

$$X_{n+1} = H(X) - G(X) \quad (2.5)$$

The Haar filter has a great advantage because of its easiness of implementation for both decomposition and reconstruction.

However, the results given by the filter suffer from block effect and then, the perceived quality of the reconstructed image is lesser. This block effect comes from the fact that the filters do not overlap when applied and hence, losses introduced by quantization will affect only one block. Figure 7 shows the block effect after scalar quantization is applied.

Figure 7 Resulting block effect when using Haar wavelet

2.4. D4 Daubechies wavelet

The Daubechies wavelets are a family of orthogonal wavelets defining a discrete wavelet transform. It is named after Ingrid Daubechies, a Belgian physicist and mathematician [12].

Daubechies wavelets contains a whole set of wavelets, from D2 to D20, taking into account only even index numbers [13]. The index number corresponds to the number of coefficients, D4 having 4 coefficients. The D2 wavelet is the Haar wavelet.

Each wavelet contains a scaling sequence (low pass filter) and a wavelet sequence (high pass filter).

The wavelet coefficients are derived by reversing the order of the scaling coefficients and then reversing the sign of one coefficient over two. We can see it mathematically:

$$b_k = (-1)^k a_{N-1-k} \tag{2.6}$$

where *k* is the coefficient index, *N* is the wavelet index, *b* is a wavelet coefficient and *a* is a scaling coefficient [13].

Hence, Daubechies wavelets can be defined entirely with the scaling filter.

The Daubechies wavelet which will be studied in this project is the D4 wavelet. This choice has been made because of its size; a filter with more taps would not go well with images of small sizes as the ones of the lowest levels of the decomposition. See Figure 8 the representation of D4 scaling and wavelet functions.

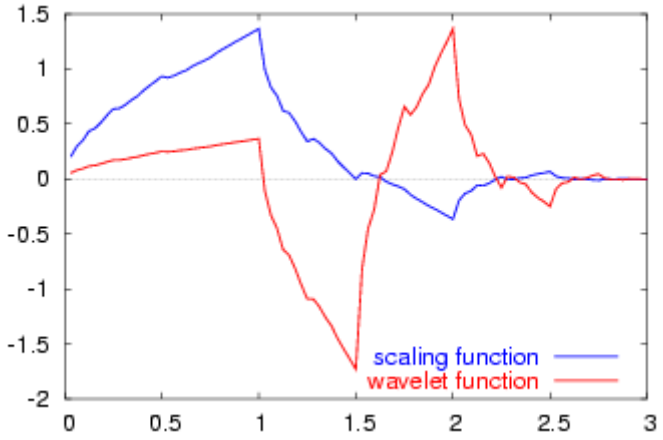


Figure 8 Daubechies 4 tap wavelet

The scaling coefficients are:

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}} \tag{2.7}$$

$$h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}} \tag{2.8}$$

$$h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}} \tag{2.9}$$

$$h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}} \tag{2.10}$$

The wavelet coefficients are retrieved from (2.6), (2.7), (2.8) and (2.9) doing their quadrature mirror filter as explained in (2.5) [8].

The wavelet coefficients are:

$$g_0 = h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}} \tag{2.11}$$

$$g_1 = -h_2 = \frac{-3+\sqrt{3}}{4\sqrt{2}} \tag{2.12}$$

$$g_2 = h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}} \tag{2.13}$$

$$g_3 = -h_0 = \frac{-1-\sqrt{3}}{4\sqrt{2}} \tag{2.14}$$

The scaling and wavelet functions are computed doing the inner product of the coefficients and four data values [14].

Scaling function:

$$y_{low}[i] = h_o s_{2i} + h_1 s_{2i+1} + h_2 s_{2i+2} + h_3 s_{2i+3} \quad (2.15)$$

Wavelet function:

$$y_{high}[i] = g_o s_{2i} + g_1 s_{2i+1} + g_2 s_{2i+2} + g_3 s_{2i+3} \quad (2.16)$$

Each iteration in the wavelet transform step calculates a scaling function value and a wavelet function value. The index *i* is incremented by two for each iteration [14].

For the reconstruction, signals at every level are upsampled by two, passed through the high-pass and low-pass synthesis filters and added. Both analysis filters used for the decomposition and synthesis filters used for the reconstruction are identical except for a time reversal. This is possible because the halfband filters form an orthonormal analysis [8].

This can be resumed by the following formula:

$$x[n] = \sum_{k=-\infty}^{\infty} \left(y_{high}[k].g[-n+2k]\right) + \left(y_{low}[k].h[-n+2k]\right) \quad (2.17)$$

Applying (2.16) to (2.14) and (2.15), we obtain:

$$x[2n] = h_2.y_{low}[n-1] + g_2.y_{high}[n-1] + h_0.y_{low}[n] + g_0.y_{high}[n] \quad (2.18)$$

$$x[2n+1] = h_3.y_{low}[n-1] + g_3.y_{high}[n-1] + h_1.y_{low}[n] + g_1.y_{high}[n] \quad (2.19)$$

D4 Daubechies wavelet presents a problem for computing the value of the edge of the image. Indeed, the index *i* increments by two but the functions use four data values. For the last iteration, there will be a problem because the last two data values do not exist. This case does not occur for the Haar wavelet since there are only two coefficients. For the inverse transform, a similar problem occurs but with the first two data values. Figure 9 shows the edge problem in the case of the decomposition. We can see that there is no data left for the last two filter coefficients.

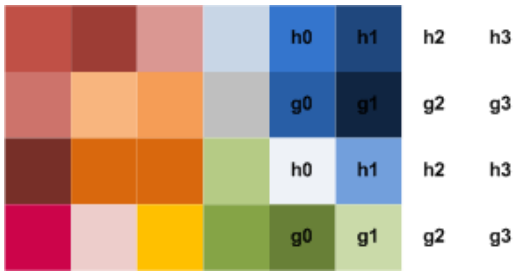


Figure 9 Edge problem with D4 filter, horizontal axis represents the x axis of pixels and vertical axis represents the y axis of pixels

This problem will lead to distorsions on the edges of the reconstructed images. To decrease the distorsion, two of the solutions available are [14]:

- Consider the input image as a periodic signal. This way the first two sets of data will be used for the transform, replacing the missing ones at the end of the image. And in the case of the inverse transform, the last two sets will be used raplacing the missing ones at the beginning of the image. Figure 10 shows the image considered as periodic, the end of the image wrapping the beginning and vice versa.

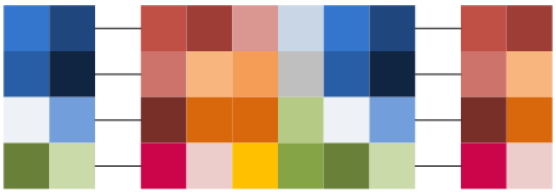


Figure 10 Image considered as periodic, horizontal axis represents the x axis of pixels and vertical axis represents the y axis of pixels

Figure 11 presents a comparison between the original image and the reconstructed image after applying D4 wavelet and using the input image as periodic, distortions are being emphasized with red ellipses.

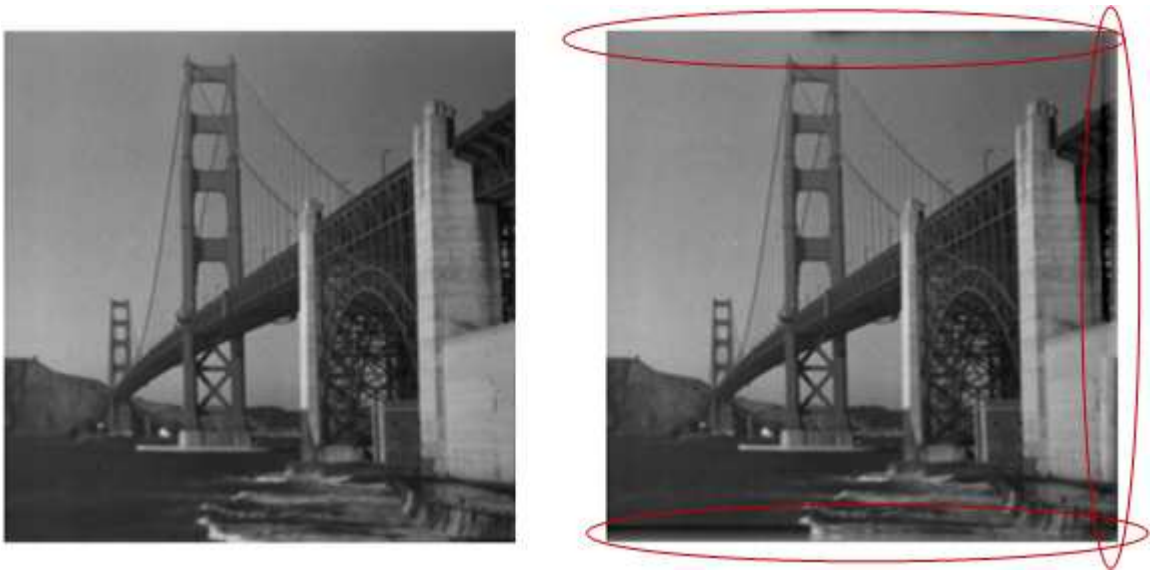


Figure 11 Edge distortions when using image as periodic

- Consider the input image as a mirror. The data is reflected at each end as if a mirror were held up to each end of the data sequence. Figure 12 shows the image considered as mirrored.

Figure 12 Image considered as mirrored, horizontal axis represents the x axis of pixels and the vertical axis represents the y axis of pixels

Figure 13 resents a comparison between the original image and the reconstructed image after applying D4 wavelet and using the input image as mirrored, distortions are being emphasized with red ellipses.

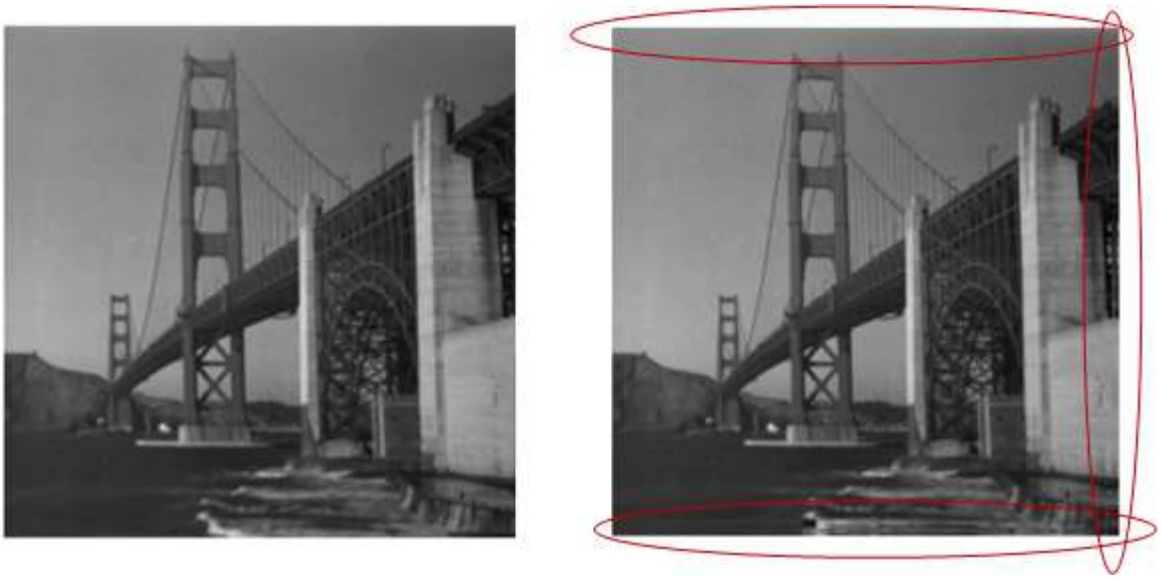


Figure 13 Edge distortions when using image as mirror

We can see rather easily that using the second solution gives better visual results. Therefore, from now on, the input image will be considered as mirrored when D4 wavelet is being used.

2.5. JPEG 2000

JPEG 2000 is a standard applying wavelet transform however not generally used by fault of support of the standard in internet browsers [15].

In JPEG 2000, after a color transformation is applied, the image is divided in tiles or blocks. The wavelet transform is applied on each tile. Two types of wavelet transform are available for use: [15]

- A reversible transform: 5/3 wavelet. It uses only integer coefficients with 5 coefficients for low-pass filter and 3 coefficients for high-pass filter. This wavelet transform is used in the case of lossless coding.
- An irreversible transform: 9/7 wavelet. It uses real coefficients with 9 coefficients for low-pass filter and 7 coefficients for high-pass filter. This wavelet is irreversible because

as it uses real coefficients, it introduces quantization noise, depending on the decoder precision. This wavelet transform achieves a higher compression rate than the previous one but is also more complex.

Because wavelet transform has a natural scalability, JPEG 2000 is very interesting for large images. One of its targets is medical imaging which can use the lossless version of the standard [16].

In higher compression rate, JPEG 2000 tends to perform better than JPEG. JPEG shows 8x8 blocks effects as well as Mach bands [17]. JPEG can provide a compression gain of 20% depending of the image characteristics. However, JPEG 2000 requires encoders and decoders that are complex and computationally demanding.

3. Quantization theoretical background

Quantization [18], involved in image processing, is a technique used in lossy compression achieved by compressing a range of values to a single quantum value.

Quantification can be scalar, it is applied taking into account every pixels. Quantification can also be vectorial. In this case, quantification is not applied on pixels but vectors of pixels.

3.1. Scalar quantization

Scalar quantization [19] consists of a mapping of an input value x into a finite number of output values y.

$$Q: x \rightarrow y \tag{3.1}$$

Scalar quantization induces losses, called quantization noise; this loss can be seen as:

$$\varepsilon = q(x) - x \tag{3.2}$$

With $q(x)$ the output of scalar quantization and x the original value of the pixel

In practice, the scalar quantization consists of two mappings [19]:

- Encoder mappings: It consists of dividing the range of values that the source generates into intervals. Each interval is then associated to a binary word. An Analog/Digital converter is a scalar quantization when the input data is analog.
- Decoder mapping: Given the binary word, the decoder gives a value that best represents all the values in the interval. If the reconstruction is analog, the decoder is often referred to as D/A converter.

Scalar quantization can be of different types.

3.1.1. Uniform quantizer

In uniform quantization [20], the entire data range is divided into L intervals of same length Q. The input data as to be defined in a finite range (f_{min}, f_{max}).

$$Q = (f_{max} - f_{min})/L \tag{3.3}$$

Interval i is mapped to the middle value of its interval. See Figure 14.

Figure 14 Mapping of the intervals in uniform quantizer

Depending on the input values, the output intervals are symmetric or not to 0.

Figure 15 and Figure 16represent the output values, axis y, and the input values, axis x of a scalar quantizer.

If zero is one of the output levels, then L, the number of output intervals, is odd as shown in Figure 15 and the intervals are symmetric to 0. If however zero is not part of the output levels, then L is even and the intervals are not symmetric to 0 as shown in Figure 16.

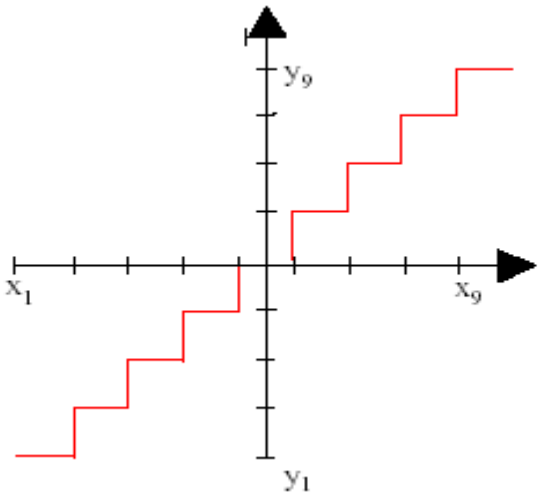


Figure 15 Uniform quantization with an odd interval number

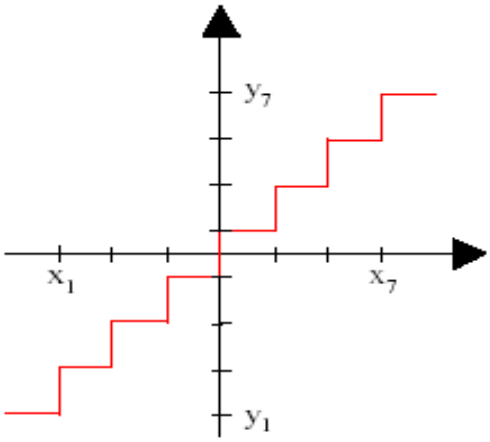


Figure 16 Uniform quantization with an even interval number

However, a uniform quantization is not optimal in the MSE meaning. Indeed, it does not take into account the pdf (probability density function) of the input data. This is what would do an optimal quantizer.

3.1.2. Optimal quantizer in MSE

The optimal quantizer [20] takes into account the probability density function of the input values to define the output levels. It is optimal in the way that it corresponds exactly to the probability density function so more intervals will be present in higher probability zones.

The probability density function [21] identifies the probability of the value falling within a particular interval. If the pdf is uniform, that is, all intervals have the same probability of appearance, then a uniform quantizer would be the optimal choice.

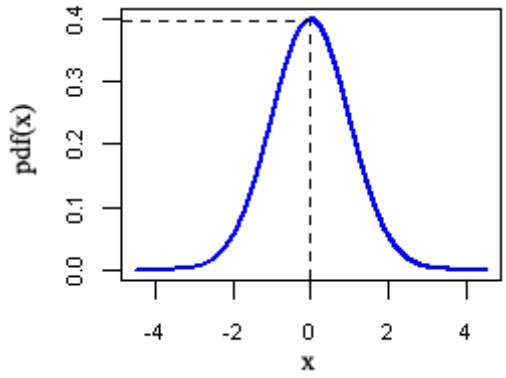


Figure 17 Non-uniform pdf

If though the pdf is not uniform, a uniform quantizer would not represent the input data optimally.

Taking into account the probability density function permits to allocate more intervals in high density zone than in low density zone. For example, with the pdf shown in Figure 17, more intervals would be allocated around zero because this is where the probability is the highest.

And contrary to the uniform quantization, the interval i is not necessary mapped to the middle of the interval. See Figure 18.

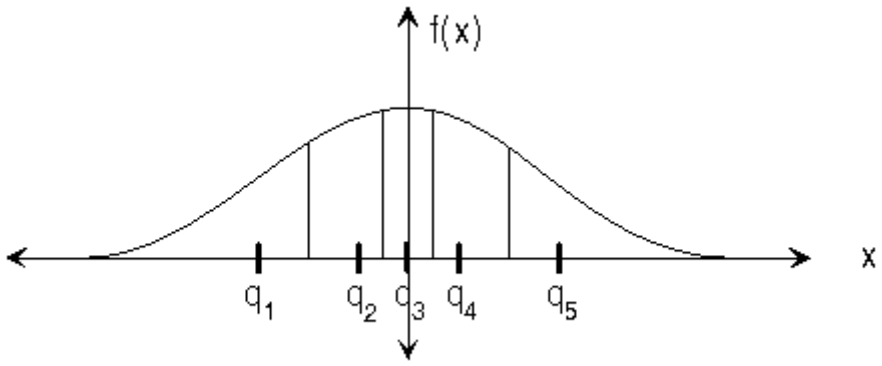


Figure 18 Mapping when pdf is not uniform

3.1.3. Max quantization

Lloyd-Max algorithm is an algorithm developed by Lloyd in 1957 but never published. It was published later in 1960 by Max [22].

This algorithm [23] permits to construct an optimal scalar quantizer which minimizes the distortion, measured by the mean square error (MSE) [24].

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}[I(x,y) - I'(x,y)]^2 \quad (3.4)$$

In [23], the author gives optimum quantization tables for input with a Gaussian pdf Figure 19.

$$Gaussian(x, \sigma_x) = \frac{1}{\sqrt{2\pi\sigma_x^2}}\exp\left(-\frac{x^2}{2\sigma_x^2}\right) \quad (3.5)$$

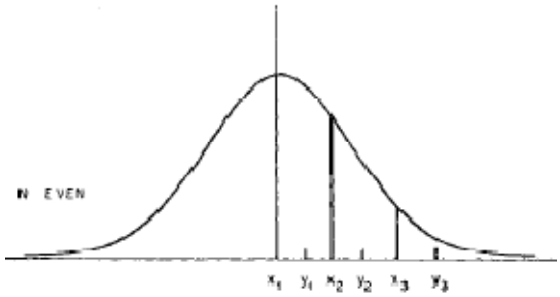


Figure 19 Gaussian pdf quantization with an even number of output levels

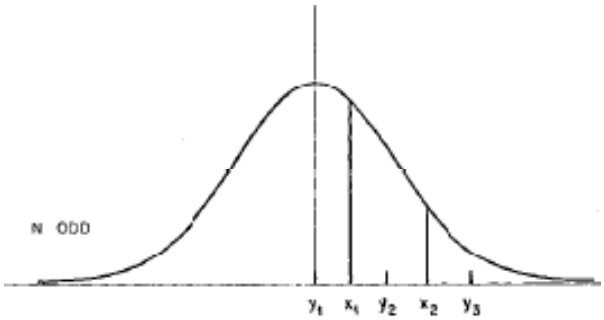


Figure 20 Gaussian pdf quantization with an odd number of output levels

The tables present a correspondence between output levels y_j and input interval end points x_i for a minimum mean-squared error quantization scheme for inputs with a Gaussian amplitude probability density with standard deviation unity and mean zero.

See Figure 21 three tables for N=4, 5 and 6.

For the number of output levels, N, even, x_1 is the first end point of an input range to the right of the origin. An input between x_i and x_{i+1} produces an output y_j Figure 19.

For the number of output levels, N, odd, y_1 is the smallest non-negative output. An input between x_{i-1} and x_i produces an output y_j Figure 20.

	$N = 4$		$N = 5$		$N = 6$	
	x_i	y_i	x_i	y_i	x_i	y_i
$j = 1$	0.0	0.4528	0.3823	0.0	0.0	0.3177
2	0.9816	1.510	1.244	0.7646	0.6589	1.000
3				1.724	1.447	1.894
Error	0.1175		0.07994		0.05798	
Entropy	1.911		2.203		2.443	

Figure 21 Max's tables for N=4, 5 and 6

Max quantizer is symmetric. In this way, as can be seen in Figure 21, it is necessary to define only the positive inputs intervals and the positive output levels. The negative values can be deducted directly as it is simply the opposite values presented in the tables.

Max's tables are created for input values going from -4 to +4 with mean 0.

3.1.4. Wavelet scalar quantization

In this project, all the shapes of all levels are quantized. See section Wavelet decomposition. The lowest frequency shape of last level can remain unquantized or be quantized with a lower compression rate because it is the image with most of the information and so a quantization would generate a lot of losses. See Figure 22.

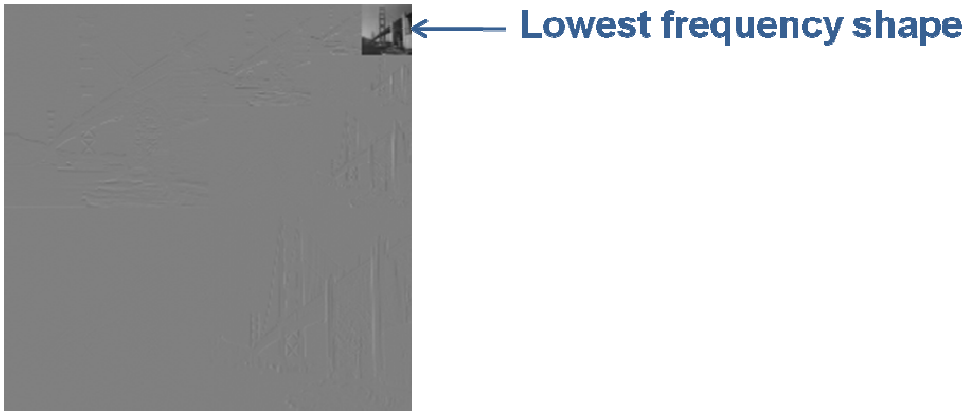


Figure 22 The lowest frequency shape of wavelet decomposition

In [25], we can see that all shapes, have an average more or less equal to zero which corresponds to Max quantization.

However, it is also shown in [25] that the pdf of wavelet coefficients is not Gaussian but Gamma. See Figure 23 for the pdf of level 1 wavelet decomposition.

$$Gamma(x,\sigma_x) = \frac{\sqrt[4]{3}}{\sqrt{8\pi\sigma_x|x|}}\exp\left(-\frac{\sqrt{3}|x|}{2\sigma_x}\right) \tag{3.6}$$

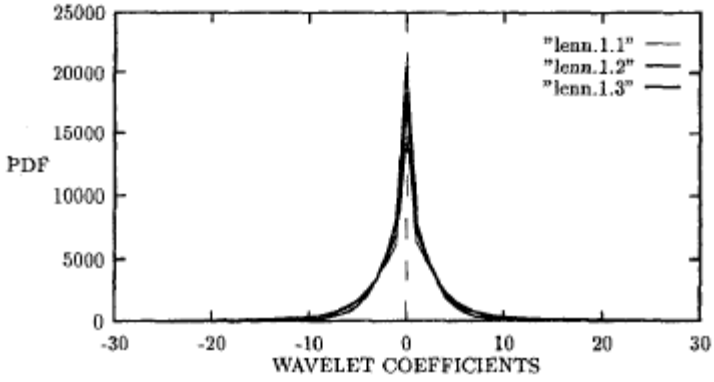


Figure 23 Gamma pdf for the first level of wavelet coefficients

In the project, even though the pdf does not correspond, Max tables have been used for the scalar quantization. The reason for those tables to be used is that they are the tables used in [25] for the scalar quantization. [25] was the paper on which relied the project as part of the Image Compression course.

As we can see in Figure 23, the probability density function is maximum for 0. Therefore, an odd number of output level is used, this way, the first output level is $y_1 = 0$. The input values in Max tables are situated between -4 and +4, as a result, input intervals and output levels are weighted so it takes the whole range of the wavelet coefficients.

3.2. Vector quantization

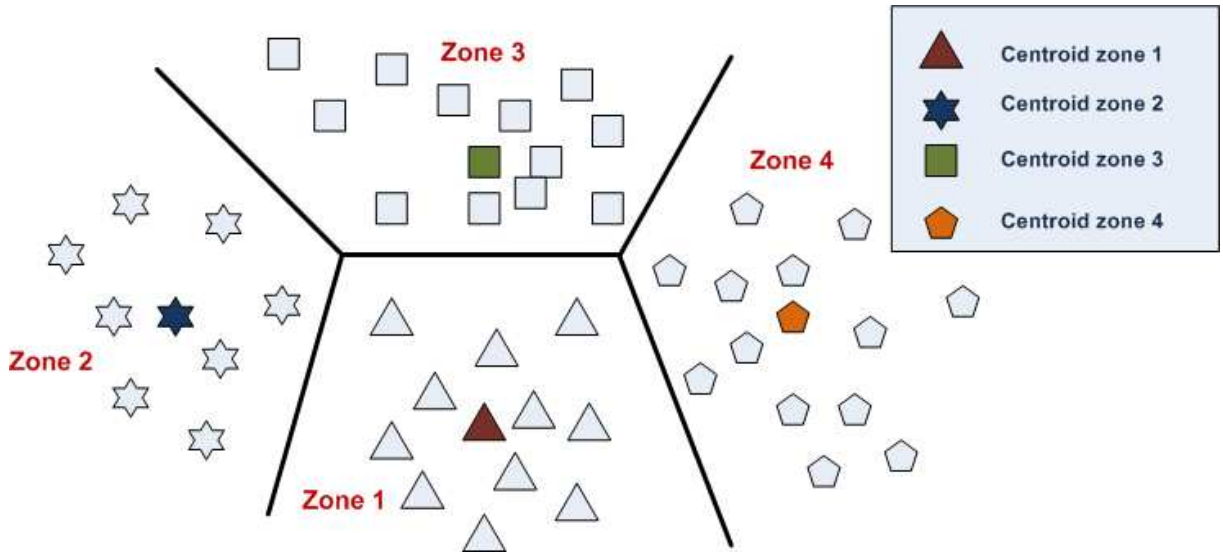


Figure 24 Vector quantization and centroids

The vector quantization is a type of quantization used in lossy compression data. It consists of mapping vectors, blocks, in the case of images, into groups so that all the vectors in one group are similar in values. Those groups are represented by their centroids which is the mean of all the vectors in the same group [26]. Figure 23 shows a vector quantization for 2D objects. They are divided in 4 zones according to their coordinates. Each zone is represented by their centroids which are displayed with colors.

All the centroids constitute the codebook. The codebook is a list of vectors which will be used to code the image altogether reducing the size of data needed to code that image.

Vector quantization matches the density probability of the blocks, hence, the quantization errors are inversely proportional to the probability density of the vectors [26].

Once a codebook matching the probability density of the blocks is created, each blocks of the image are mapped to the codebook and this is those index of mapping which are kept in memory, decreasing the need for space storage.

3.2.1. LBG algorithm

The LBG algorithm stands for Linde-Buzo-Gray algorithm, it has been first presented in 1980 [3].

The purpose of this algorithm is to find the codebook which will be applied for the vector quantization on the wavelet coefficients in this context but it can be applied to all sort of coefficients.

The steps of the training algorithm are: [3]

- Initialization: Given N the number of vectors wanted in the codebook; choose an initial alphabet constituted by N vectors. Those vectors are chosen randomly or in order to be the most dissimilar from each other as possible.
- Mapping: For each block of the given image, find the vector which has the minimum distortion.
- Creation of new codebook: For each vector of the codebook, compute the average of the blocks mapped to that vector and replace the vector by the average.
- Repetition: Repeat steps mapping and creation of new codebook until stop condition is reached.
- Stop: If a given number of repetitions exists, stop when this number is attained. Else, compute the distortion of the whole image and if it is inferior to a given threshold ε, then stop.

Once the training is finished and the codebook is retrieved, each block of the image can be mapped to the vectors of the codebook following the Mapping step from the previous training algorithm.

A scheme representing the LBG algorithm can be seen in Figure 25.

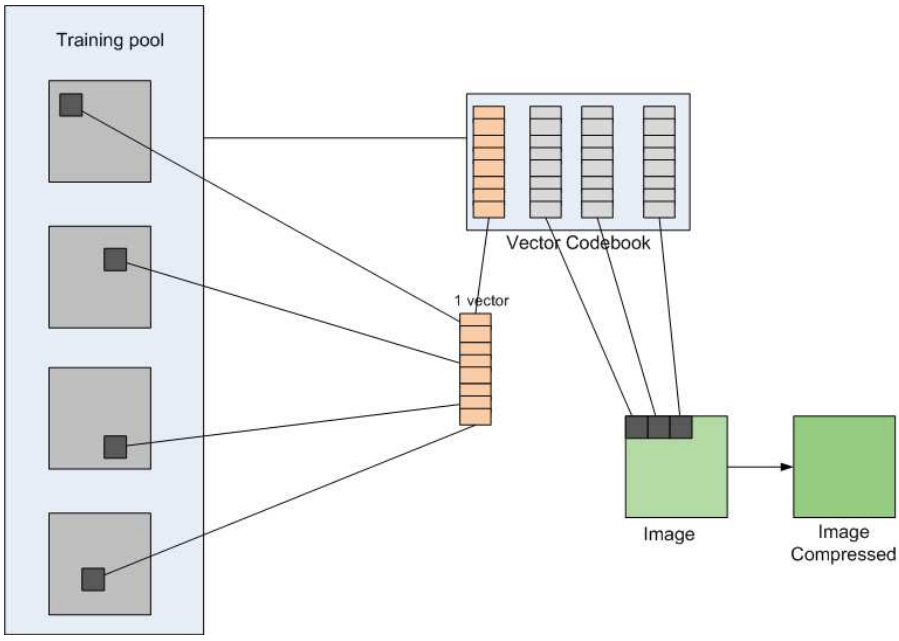


Figure 25 Codebook generation using LGB algorithm

3.2.2. Training set

The training set for the LBG algorithm can be one image or a whole set of images.

If a set of images is used, this means that one codebook can be used to quantize more than one image. And the images which have to be quantized are not necessarily present in the training set of images.

The images present in the training set have to be similar to the one to be quantized. In the case of the project, the images quantized are in reality wavelet coefficients. To have a training set the closest to the wavelet coefficients, training will be made for each shapes of each levels.

Figure 26 shows the images which will be used in the training set. The training set presents well-known images usually used in image compression, as Lena or Camman. The images in the training set represent a wide range of types of images, from very sharp straight edges, as MIT, to people which does not have straight edges at all nor very sharp ones.

All the images would not automatically be present in the training set. For example, in Figure 26, it can be seen that the images with the peppers and the fruits are similar. On the contrary, the image of the MIT is very dissimilar from the image of the people, so they won't be in their respective training set.

Figure 26 Images of the training set

4. Experimental results

In this part, some experimental results will be presented. They analyze every parameter of the wavelet decomposition and quantization involved in the project.

The study will focus more on vector quantization than scalar quantization as vector quantization permits, in theory, a higher compression rate.

4.1. Compression rate

First of all, the computation of the compression rate used either for scalar quantization or vector quantization is detailed.

4.1.1. Scalar quantization

In the case of scalar quantization, the parameters taken into account are:

- The number of levels in the wavelet decomposition
- The number of levels in the scalar quantization for each level
- The quantization or not of the low frequency shape of last level

The steps to compute the compression rate are:

- For each level:
  - Compute the number of bits necessary to code each pixel of the shape. To do that, find the number of bits necessary to code the number of quantization levels for the shape

$$2^x = N_{levels} \quad (4.1)$$

With x the number of bits and N_{levels} the number of levels used for the shape in the quantization.

Number of bits	8	7	6	5	4	3	2	1
Number of coding possibilities	256	128	64	32	16	8	4	2

Table 1 The number of coding possibilities varying with the number of bits

Table 1 represents the results of equation (4.1).

  - Compute the number of bits required for each shape:

$$B_{Shape\ k} = S_S.S_S.x \quad (4.2)$$

With $B_{Shape\ k}$ the number of bits necessary to code shape k and S_S the size of the shape.

- If level is last level:
    - Compute the number of bits required to code shape 0 using previous steps.
- Add all the results from all levels and all shapes
- Compute the number of bits per pixel:

$$b_{pixel} = \frac{T_{bits}}{S_{image}.S_{image}} \quad (4.3)$$

With b_{pixel} the number of bits per pixel, T_{bits} the total amount of bits required to code the whole decomposition and S_{image} the size of the original image.

- To compute the compression rate:

$$R = \frac{8}{b_{pixel}} \quad (4.4)$$

With R the compression rate.

4.1.2. Vector quantization

For vector quantization, the parameters taken into account are:

- The number of levels in the wavelet decomposition
- The size of the vectors for each level
- The number of vectors for each level
- The quantization or not of the low frequency shape of last level

The steps to compute the compression rate are:

- For each level:
    - Compute the number of blocks for each shape except low frequency shape shape0. To do that, take the size of the shape and divide it by the size of the vectors as follow:

$$N_{blocks} = \frac{S_S.S_S}{S_V.S_V} \quad (4.5)$$

With N_{blocks} the number of blocks in the shape, S_S the size of the shape and S_V the size of the vector.

    - Compute the number of bits necessary to code each pixel of the shape. To do that, find the number of bits necessary to code the number of vectors for the shape. The number of vectors being a power of 2, to obtain x, the following equation has to be solved:

$$2^x = N_{vectors} \quad (4.6)$$

With x the number of bits and $N_{vectors}$ the number of vectors used for the shape.

    - Compute the number of bits required for each shape:

$$B_{Shape\ k} = N_{blocks}.x \quad (4.7)$$

With $B_{Shape\ k}$ the number of bits necessary to code shape k.

- If level is last level:
  - Compute the number of bits required to code shape 0 using previous steps.
- Add all the results from all levels and all shapes
- Compute the number of bits per pixel:

$$b_{pixel} = \frac{T_{bits}}{S_{image}.S_{image}} \quad (4.8)$$

With b_{pixel} the number of bits per pixel, T_{bits} the total amount of bits required to code the whole decomposition and S_{image} the size of the original image.

- To compute the compression rate:

$$R = \frac{8}{b_{pixel}} \quad (4.9)$$

With R the compression rate.

4.2. Wavelet decomposition

In order to show the impacts of different parameters in the best conditions, for the following tests, except the ones in the Training set part, the training set will be composed only of the image itself which will be compressed.

4.2.1. Impact of the number of levels

The number of levels in the wavelet decomposition has an influence on both quality and compression rate.

The study has been conducted with the following parameters:

- Vector quantization:

	Size of vectors	Number of vectors
Level 1	4x4	128
Level 2	2x2	64
Level 3	2x2	32
Level 4	2x2	16

Those parameters have been chosen following some tests. They provide average results in term of PSNR, not too low but not too high either. This will permit to study correctly the impacts of other parameters. This remark will stand for all the following tests.

- Daubechies filter
- Shape 0 of last level is not quantized

The variable parameter is:

- The number of decomposition levels

The test has been realized from 1 level to 4 levels. In each case, the quality and the compression rate have been noted. The quality is represented by the PSNR and the compression rate by the number of bit per pixel.

The reconstructed images can be seen in Figure 27, Table 2 shows the results of PSNR and compression and Graphic 1 displays results of Table 2 in a graph.

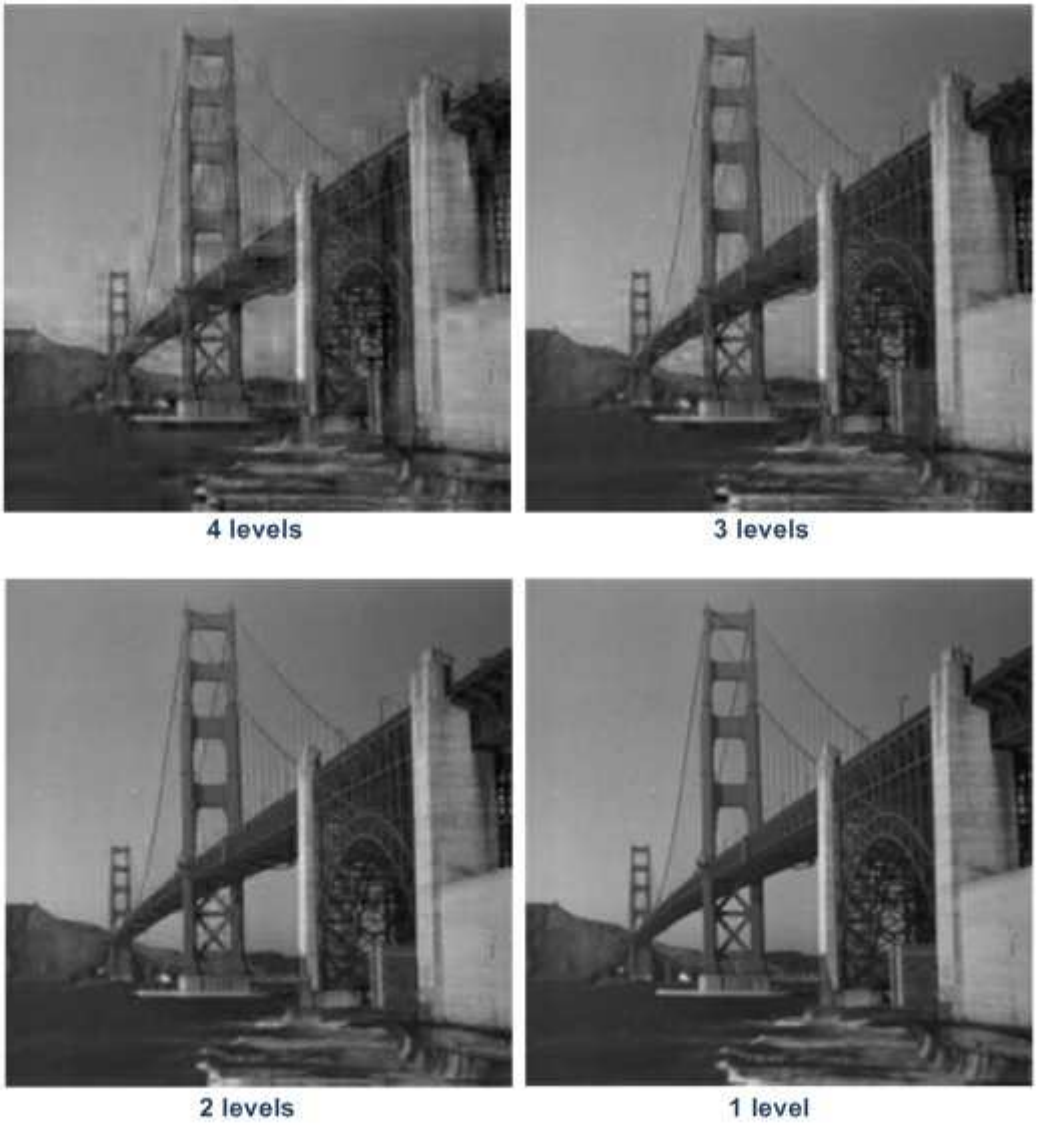
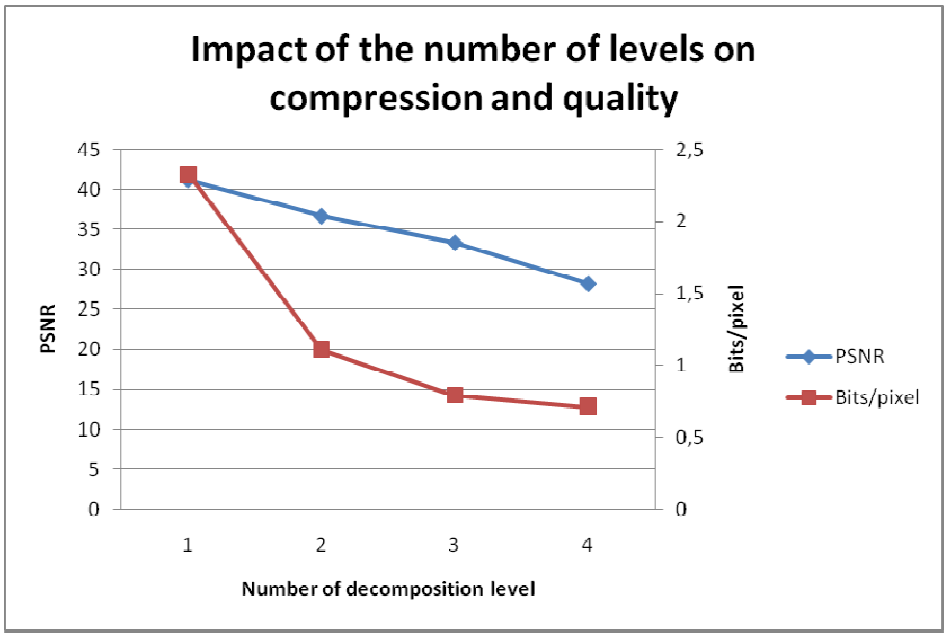


Figure 27 Reconstructed images using 1, 2, 3 and 4 levels

Impact of the number of decomposition on compression and quality		
Number of decomposition	PSNR	Bits/pixel
1	41,1411	2,328125
2	36,6695	1,109375
3	33,3593	0,79296875
4	28,2631	0,7109375

Table 2 Impact of the number of decomposition levels on compression and quality



Graphic 1 Impact of the number of levels on compression and quality

In Graphic 1, on the one hand, we can see that the quality evolves in a quasi linear way. Without surprise, PSNR decreases when the number of decomposition increases. But this can also depend on the fact that a fix number of bits has been assigned to each level. If a variable number of bits was assigned for each vector and did not depend on the level but only on the vector itself, the result could change drastically and even improve.

On the other hand, the number of bits per pixel decreases really fast at the beginning to tend toward stagnation near 0,7 bits/pixel. We can see that after 3 levels, the gain in compression is too small compared to the loss of quality. This can be observed also on the images, the image for 3 levels has a fairly good visual quality but the one with 4 levels has a poorer quality.

To conclude, 3 levels seems to be the value which permits to increase the compression rate to a quite high value without decreasing too much the quality of the image.

4.2.2. Impact of the quantization of last level's low frequency shape

Shape 0 of last level is the most important shape. Indeed, it is the shape with the most information. In order not to lose too much information, this shape can sometimes be left without quantization.

In the following study, the loss in quality and the gain in compression will be analyzed for different level of quantization for shape 0 of level 3. Quantization for all the other shapes will remain unchanged for all the tests.

The study has been conducted with the following parameters:

- Vector quantization:

	Size of vectors	Number of vectors
Level 1	4x4	128
Level 2	2x2	64
Level 3	2x2	32

Table 3 Standard parameters for vector quantization

- Daubechies filter
- 3 levels of decomposition

The variable parameter is:

- The number of vectors for shape 0 of level 3.

The test has been realized with no quantization for shape 0, a quantization with 128 vectors, 64 vectors and 32 vectors. In each case, the quality and the compression rate have been noted. The quality is represented by the PSNR and the compression rate by the number of bit per pixel.

The reconstructed images can be seen on Figure 28. Table 4 shows the results of PSNR and compression and Graphic 2 displays the results of Table 4 in a graph.

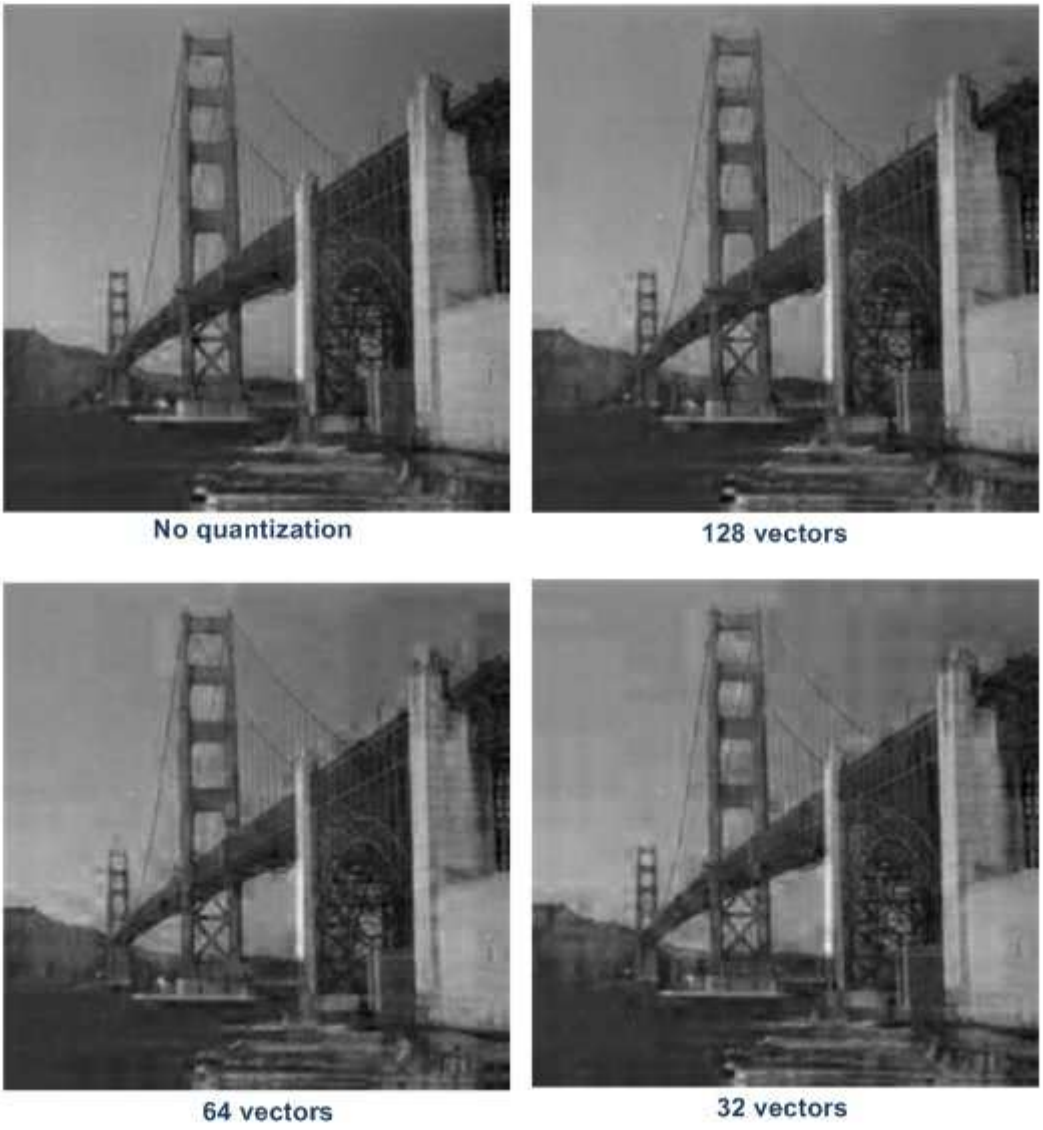
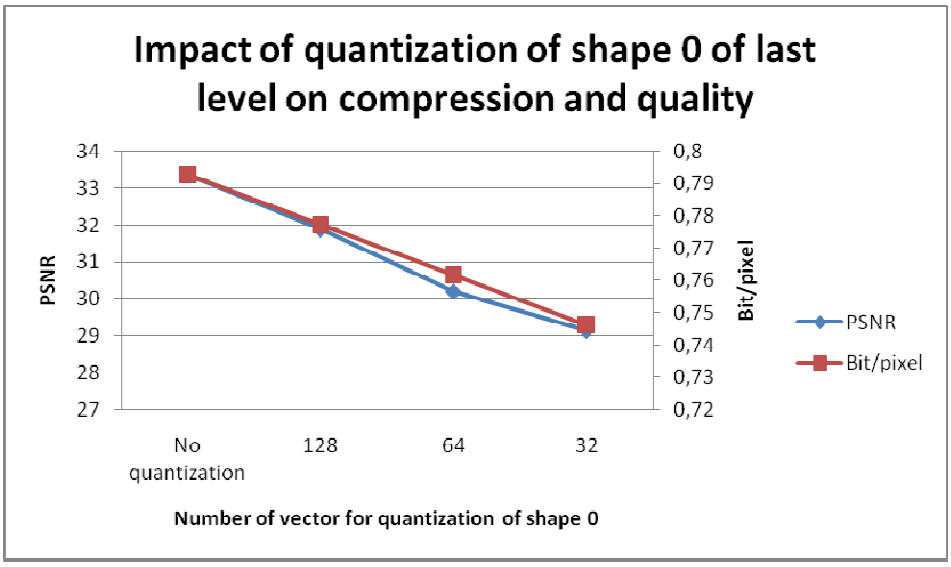


Figure 28 Reconstructed images with different quantization for shape 0 of level 3

Impact of quantization of shape 0 on compression and quality		
Number of vectors	**PSNR**	**Bits/pixel**
No quantization	33,3596	0,79296875
128	31,8983	0,77734375
64	30,2069	0,76171875
32	29,1545	0,74609375

Table 4 Impact of quantization of shape 0



Graphic 2 Impact of quantization of shape 0 of last level on compression and quality

In Graphic 2, it is shown that both PSNR and bit/pixel decrease linearly with the number of vectors used in quantization of shape 0. However, the loss in quality is more important that the gain in compression, indeed, the number of bit/pixel only looses 0,047 whereas PSNR looses more than 4 points. In Figure 28, we can observe that with 64 vectors and less, the visual quality is very bad.

From the above results, we can wonder about the worthiness of quantization of shape 0 of last level when we see the small gain in compression and the loss in quality. For instance, if we choose 128 vectors, the value of bit/pixel only decreases from 0,793 to 0,777 and PSNR passes from 33,4 to 31,9 as well as big difference visually. So maybe, keep shape 0 unquantized is better. But this would depend on the compression rate and the final quality wanted.

4.2.3. Impact of the filter

In the project, two filters have been implemented, the Haar filters and the Daubechies filters. As said above in sections Haar wavelet and D4 Daubechies wavelet, Haar filter is simpler than Daubechies filter but increases block effect in the case of scalar quantization.

In this part, vector and scalar quantization will be used for with both Haar and Daubechies filters and the quality of the images will be analyzed.

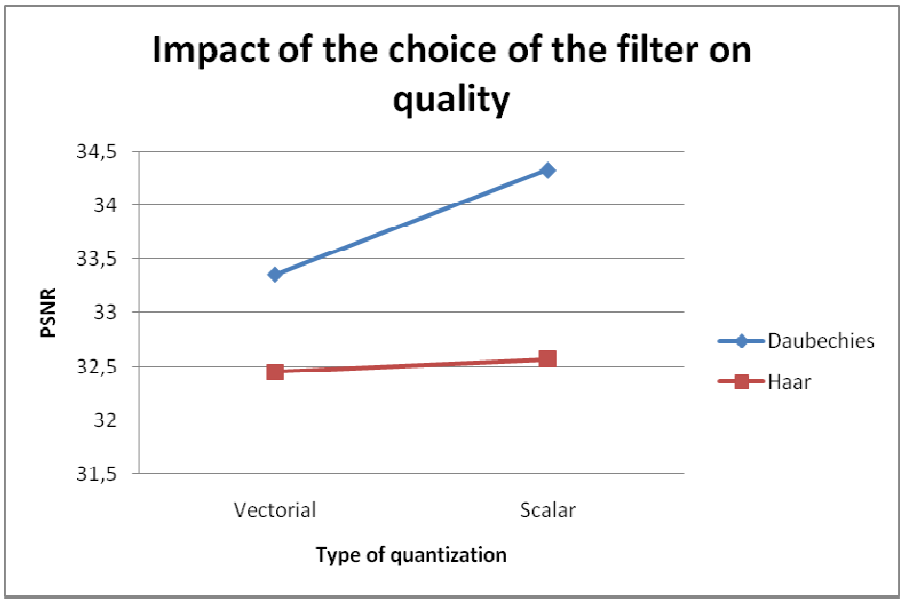
The vector and scalar quantizations applied are standard, with 7 levels for scalar quantization and the parameters described in Table 3 for vector quantization.

Figure 29 shows the visual results whereas Table 5 shows the numeric results and Graphic 3 displays them.

Figure 29 Reconstructed images using Haar and Daubechies filters

Impact of the choice of the filter on quality		
Quantization	**Vectorial**	**Scalar**
Daubechies	33,3596	34,3287
Haar	32,4488	32,5678

Table 5 Impact of the choice of the filter on quality



Graphic 3 Impact of the choice of the filter on quality

We can see at once in Graphic 3 that Daubechies filter gives better results, a difference of almost 1 point in PSNR for vector quantization and more than 1,5 points for scalar quantization.

The variation in the case of scalar quantization is the most obvious also in Figure 29 where we can clearly see more blocks effect when Haar filter is used than when Daubechies filter is used. This is due to the size of Haar filter, as explained in Haar wavelet. With Daubechies filter, we can see Mach bands [16] near the sharp edges.

This is the confirmation of what was said in section D4 Daubechies wavelet that despite its difficult implementation, Daubechies filter was better.

As for scalar and vector quantization, we can see an obvious difference. Scalar quantization shows block effect whereas vector quantization does not show block effect but artifact about the contours in the image. In the case of this test, as we can see in Graphic 3, PSNR is lower in the case of vector quantization but we can't really take those results into account because the compression rate is not given to modulate them.

4.3. Scalar quantization

4.3.1. Impact of the number of levels

As explained in section Scalar quantization, to quantize an image, a certain number of threshold values have to be defined. Only those values will be used after quantization; points lying between thresholds will be affected to one of the thresholds.

The numbers of thresholds or levels have an impact on both quality and compression. On quality because only a small number of values represent all the existing values on the image

which induces losses. On compression rate because each pixel will be coded only with the number of bits required to code the number of levels.

The number of levels has to be an odd number to have 0 as an output as explained in section Uniform quantizer. In the following study, the values have been chosen to use one bit more of information in comparison to last test. For example, first test will use 3 levels, that is, 2 bits of information so second test will use 7 levels, that is, 3 bits of information.

The test has been realized with 3 levels, 7 levels, 15 levels and 31 levels. The filter used is Daubechies filter.

The resultant images are shown in Figure 30, Table 6 gives the values of PSNR and bit/pixel and Graphic 4 displays the results.

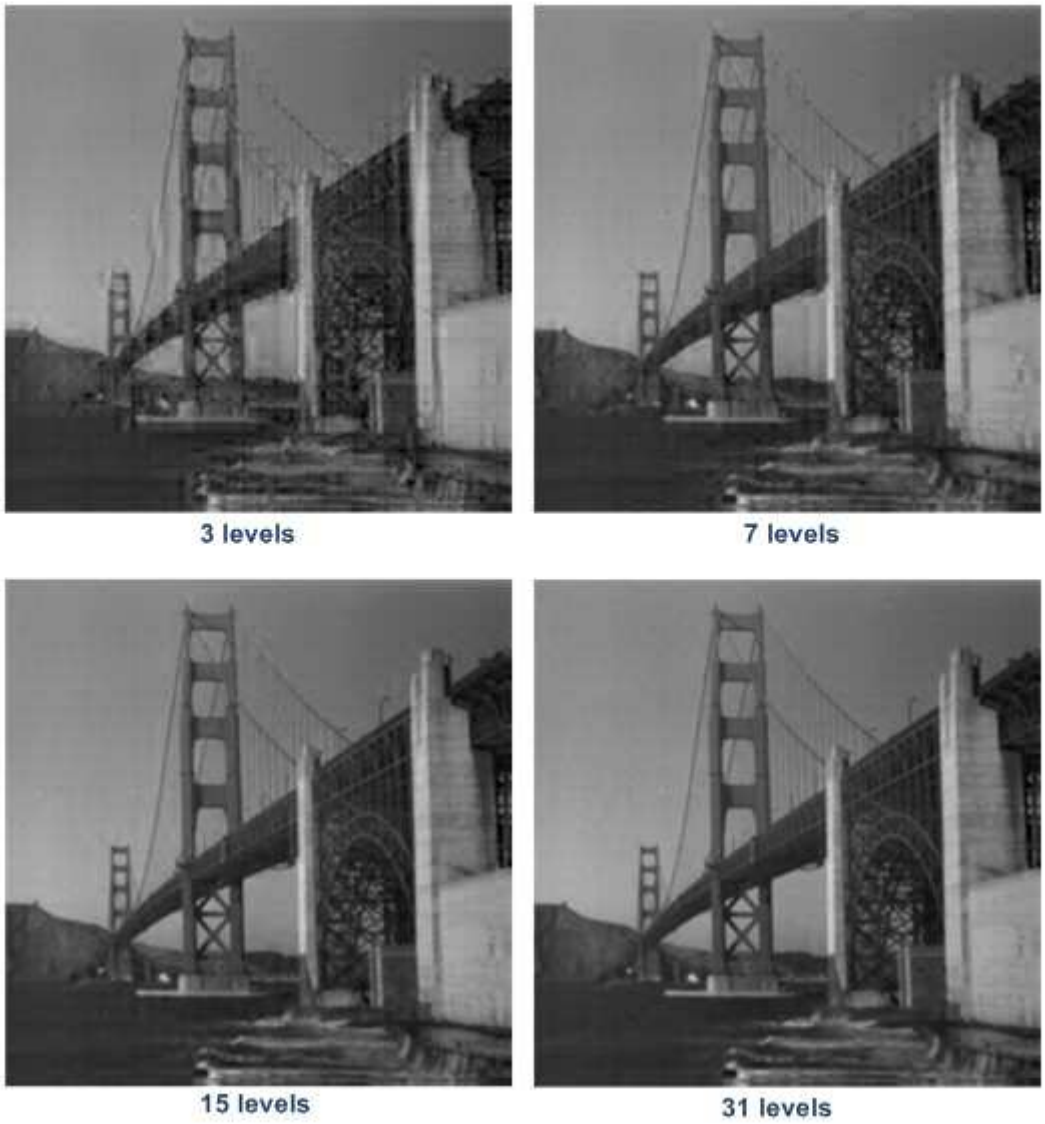
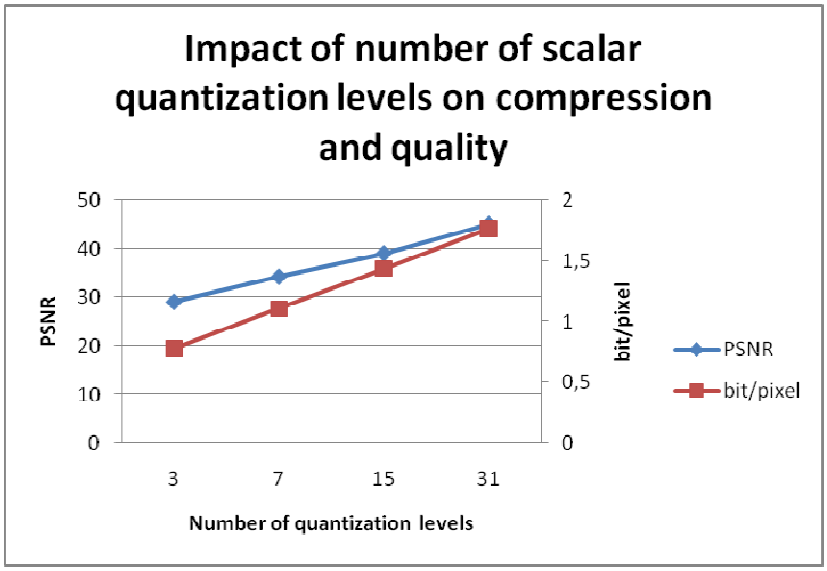


Figure 30 Reconstructed images with different number of levels for scalar quantization

Impact of scalar quantization levels on compression and quality		
Number of levels	PSNR	Bits/pixel
3	29,1545	0,78125
7	34,3287	1,109375
15	39,0999	1,4375
31	45,1205	1,765625

Table 6 Impact of scalar quantization levels on compression and quality



Graphic 4 Impact of the number of scalar quantization levels on compression and quality

We can see that scalar quantization can give a really high PSNR: 45 with 31 levels in this case. However, because in scalar quantization each pixel has to be coded, the compression rate is not very high and almost always superior to 1.

In Figure 30, we can see that with a low number of levels, a block effect appears which disappears from 15 levels. With a low number of levels, we can see that artifacts appear principally on edges; with 3 levels, we can see clearly Mach bands and distortions. Quality is very good with 15 levels and 31 levels but the number of bits per pixel is too high.

4.4. Vector quantization

4.4.1. Impact of the number of iterations

In vector quantization, as explained in section LBG algorithm, to obtain the final codebook, a same sequence of instructions is repeated several times to improve the quality of the codebook.

In the following tests, the impact of the number of iterations in the construction of the codebook is analyzed.

The standard parameters of Table 3 are used with Daubechies filter and shape 0 of level 3 is not quantized.

Figure 31 shows the reconstructed images with 5, 3, 2 and 1 iterations. The image with 4 iterations is not shown because there is almost no difference with the image of 5 iterations. Table 7 gives the values of PSNR for each case and Graphic 5 displays the results on a graph.

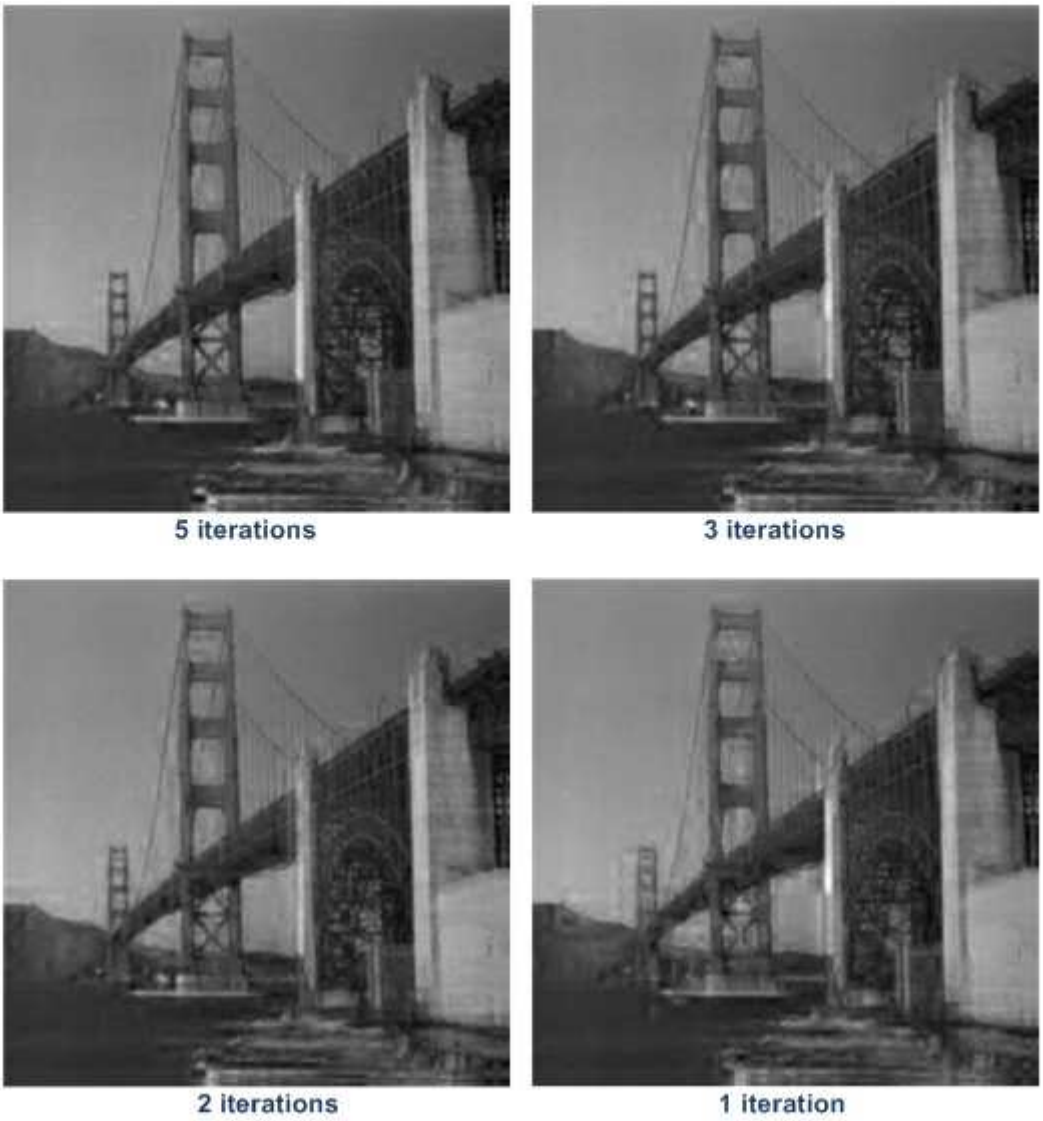
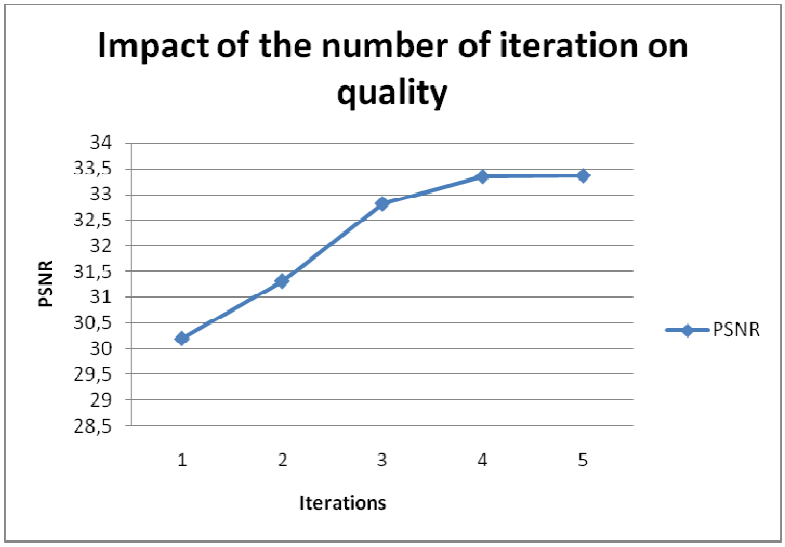


Figure 31 Reconstructed images using different number of iterations for vector quantization

Impact of the number of iterations on quality	
Number of levels	PSNR
1	30,2069
2	31,3184
3	32,816
4	33,3429
5	33,3596

Table 7 Impact of the number of iterations on quality



Graphic 5 Impact of the number of iterations on quality

We can see in Graphic 5 that the impact of the number of iterations is especially influent for 1, 2 and 3 iterations. Results for 4 and 5 iterations are almost the same and if we kept going with the iteration, there would be no change.

Therefore, we can conclude that there is no need to use more than 5 iterations. The convergence to the codebook is completed with 5 iterations.

4.4.2. Impact of the size of codevectors

The size of the codevectors used in the vector quantization has an influence on quality and compression rate.

Indeed, the bigger the codevectors, the less there will be blocks in the image to quantize and hence, the less information will be needed to code the image.

The study has been conducted with the following parameters:

- Wavelet decomposition of 3 levels
- Vector quantization:

	Number of vectors
Level 1	128
Level 2	64
Level 3	32

- Daubechies filter
- Shape 0 of last level is not quantized

The variable parameter is :

- The size of the vectors

Given the number of vectors for each level, there is a maximum value in term of size for those vectors. In this case, it has been calculated, that the maximum size for the first level is 8x8. Indeed, with an image of size 128x128 at level 1, a block size of 8x8 implies 256 blocks in the image to quantize with a number of vector of 128. If however, we choose a size 16x16, it implies 64 blocks in the image to quantify which will have no sense with a number of vectors of 128.

Therefore, the test has been realized with the following values for the size of the vectors:

	Size of vectors		
	Level 1	Level 2	Level 3
Test 1	8	4	2
Test 2	4	2	2
Test 3	2	2	2

For each test, quality and compression rate have been noted.

Figure 32 shows the reconstructed images, Table 8 gives the numeric results of PSNR and compression and Graphic 6 displays the results on a graph.

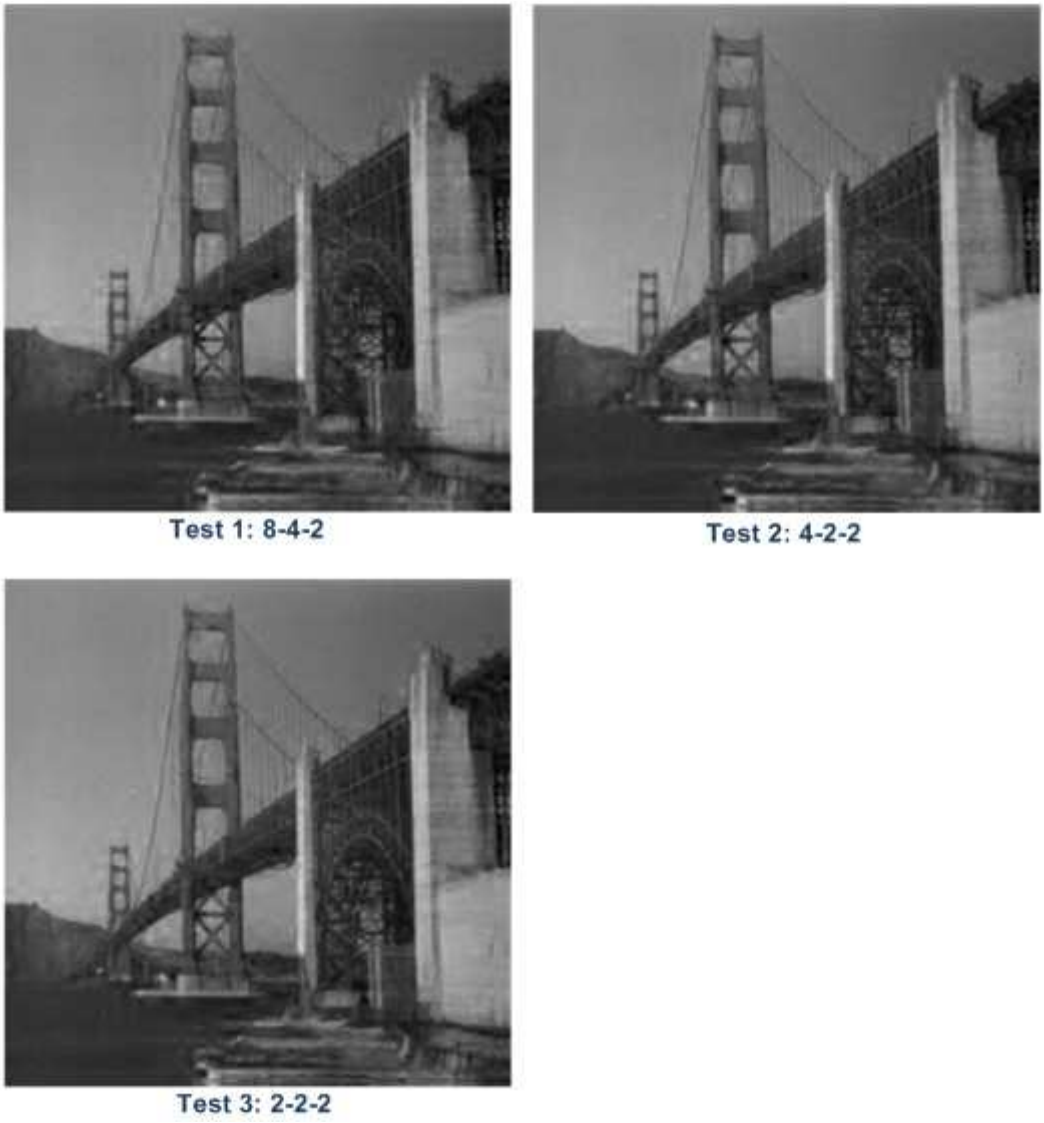
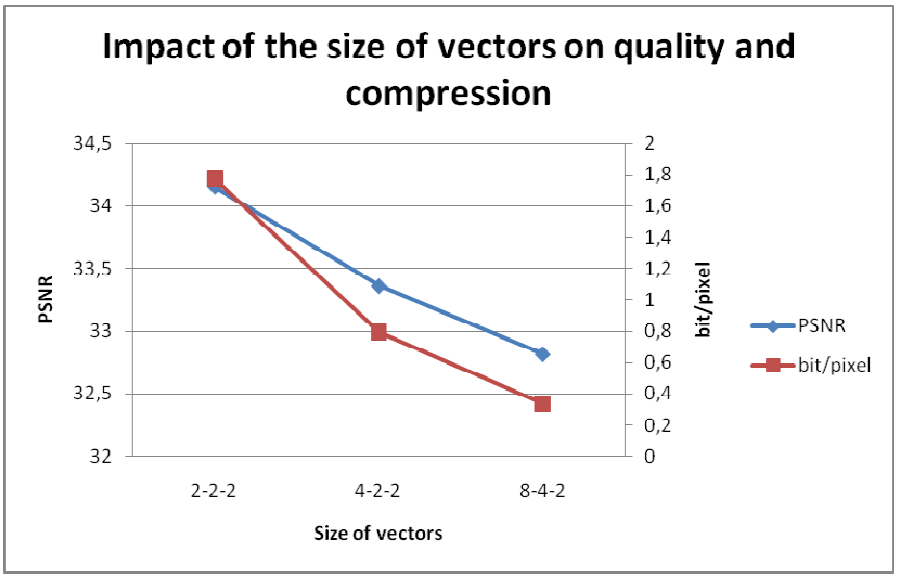


Figure 32 Reconstructed images for different sizes of vector

Impact of the size of the vectors on compression and quality		
Size of vectors (level 1-level 2-level 3)	**PSNR**	**Bits/pixel**
2-2-2	34,1514	1,77734375
4-2-2	33,3596	0,79296875
8-4-2	32,816	0,3359375

Table 8 Impact of the size of the vectors on compression and quality



Graphic 6 Impact of the size of vectors on quality and compression

On Graphic 6, we can see that the quality and the compression rate evolve in the same way. However, although the PSNR values maintain themselves rather high, the lowest value is 32,8 which is good quality, we can see that the variation for the bit/pixel values are more significant. It passes from 1,8 for 2-2-2 to 0,3 for 8-4-2.

Visually, 8-4-2 shows a lot of artifacts at the edges. 4-2-2 and 2-2-2 show a visual quality which is good. So, the choice would be between 4-2-2 and 8-4-2 depending on the quality and the compression rate we want, 8-4-2 permiting to have a high compression rate without lowering the quality too much.

4.4.3. Impact of the number of vectors

As well as the size of the vectors, the number of vectors constituting the codebook is important. The more there are vectors, the more there are possibilities to code the blocks of the images and the more those vectors will match with the codevectors.

This study will analyze the impact of the number of vectors in the codebook on quality and compression.

The study has been conducted with the following parameters:

- Wavelet decomposition of 3 levels
- Vector quantization:

	Size of vectors
Level 1	4
Level 2	2
Level 3	2

- Daubechies filter

- Shape 0 of last level is not quantized

The variable parameter is :

- The number of vectors constituting the codebook.

Six tests with different numbers of vectors have been conducted. The details of all the tests can be seen in Table 9.

	Number of vectors		
	Level 1	Level 2	Level 3
Test 1	256	128	64
Test 2	128	64	32
Test 3	64	32	16
Test 4	32	16	8
Test 5	16	8	4
Test 6	8	4	2

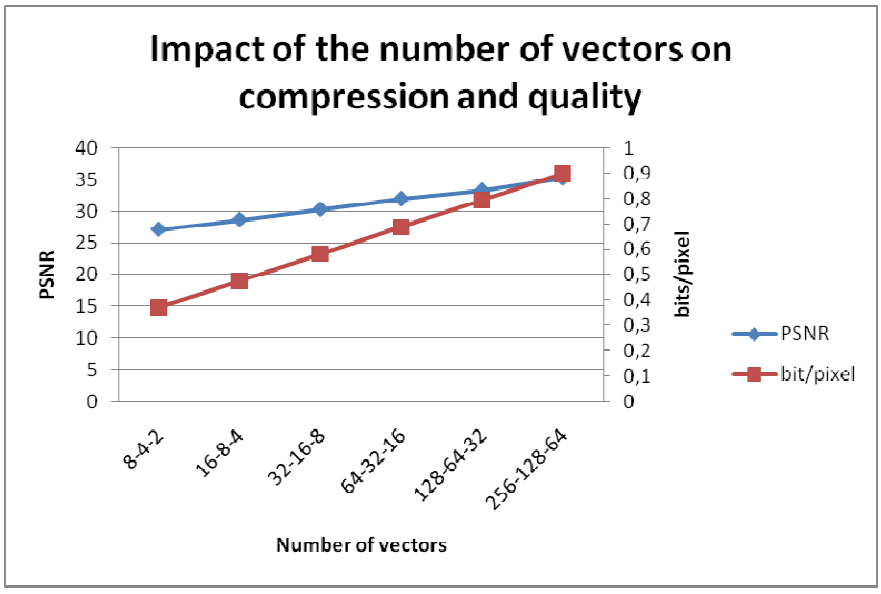
Table 9 Number of vectors for each test

Figure 33 presents the reconstructed images for each test, Table 10 shows the numeric results of PSNR and bits per pixel and Graphic 7 displays the results on a graph.

Figure 33 Reconstructed images with different number of vectors

Impact of the number of vectors on compression and quality		
Number of vectors	PSNR	Bits/pixel
8-4-2	27,1271	0,3710938
16-8-4	28,686	0,4765625
32-16-8	30,3493	0,5820313
64-32-16	32,003	0,6875
128-64-32	33,3596	0,7930
256-128-64	35,3433	0,8984375

Table 10 Impact of the number of vectors on compression and quality



Graphic 7 Impact of the number of vectors on compression and quality

The impact of the number of vectors on both quality and compression is quite important. The number of bits per pixel in the case of the less number of vectors is very low with 0,37 bits/pixel, however, the quality is also very low: a PSNR of 27 and it is visually of a poor quality, very blurred.

From 64-32-16, the quality begins to improve and it permits to keep a good compression rate with a value of 0,69 bit/pixel.

4.5. Training set

In this part, as said before, vector quantization will be performed with a training set composed of various images excluding the one being compressed and the impact of the training set on the reconstructed images will be analyzed.

4.5.1. Impact of the choice of images in the training set

The images constituting the training set have to be chosen in order to be the most similar possible to the image which has to be quantized. Therefore, a same training set will not give the same results with different images.

In this study, a vector quantization is realized with the parameters described in Table 3. The filter used is Daubechies. Shape 0 of level 3 is left unquantized.

The images constituting the training set are: peppers and people. See Figure 26.

The images quantized are: MIT, Trevor, Fruit and Bridge. See Figure 26.

Figure 34 shows the reconstructed images; Table 11 gives the PSNR results for each reconstructed images and Graphic 8 displays the results on a graph.

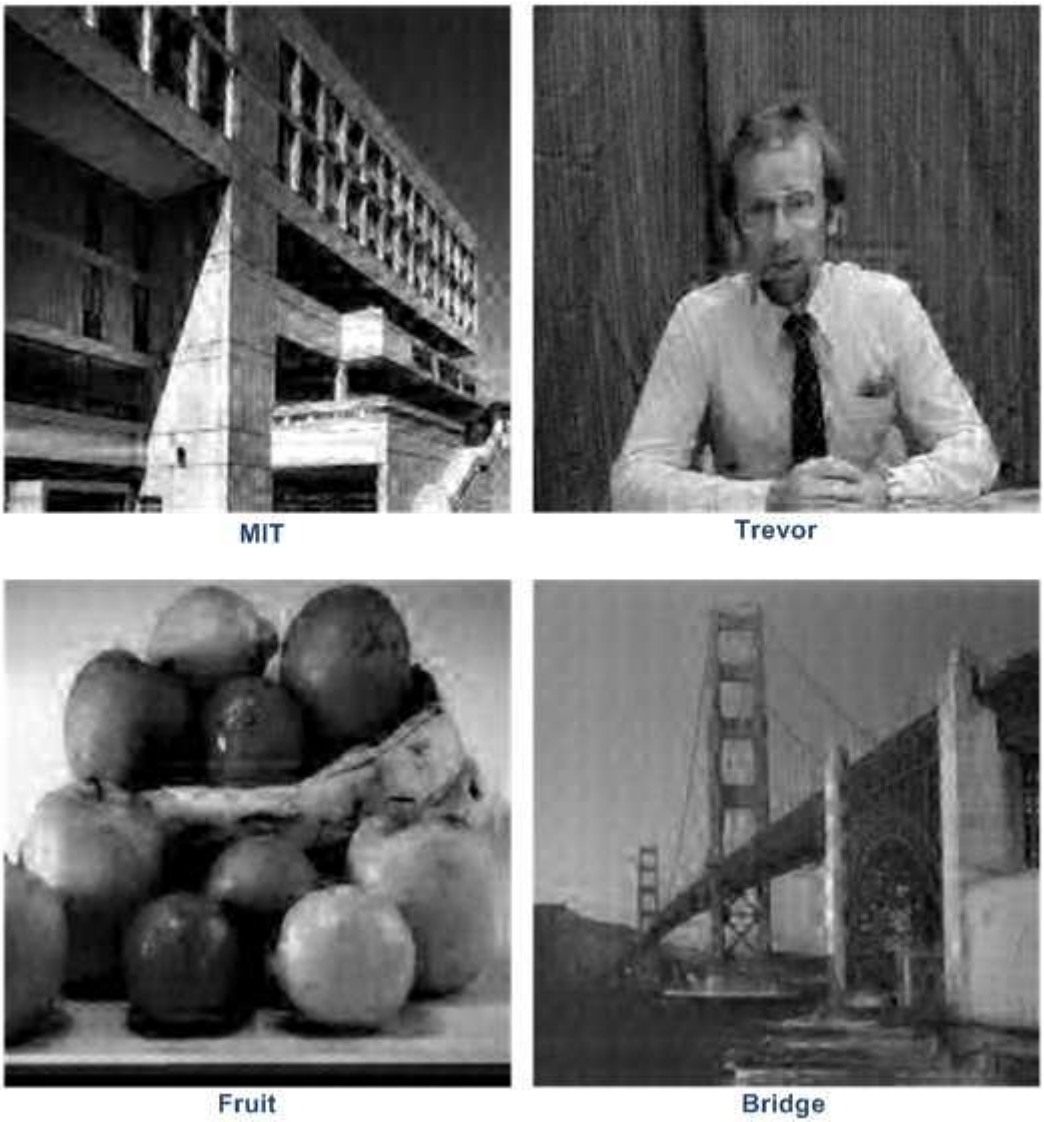
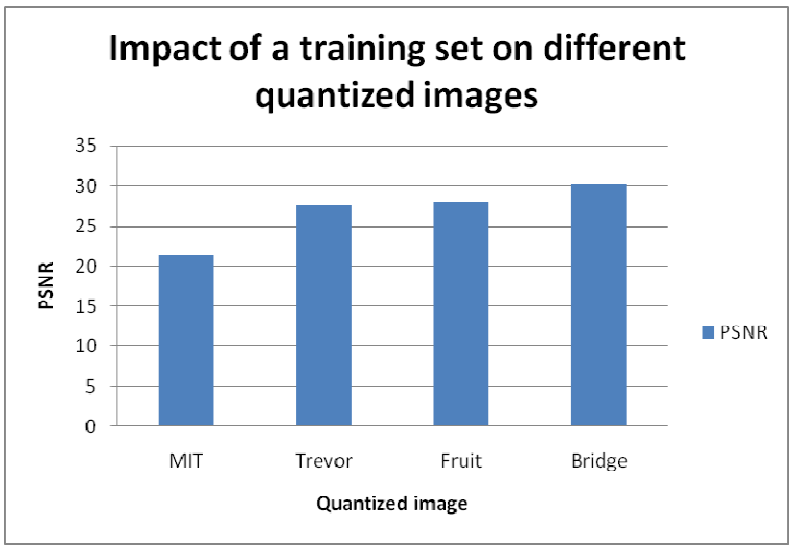


Figure 34 Different reconstructed images using the same training set of images

Impact of the training set on quality	
Image	PSNR
MIT	21,3365
Trevor	27,5618
Fruit	27,9605
Bridge	30,2069

Table 11 Impact of the training set on the quality of different quantized images



Graphic 8 Impact of a training set on different quantized images

We can see that MIT is the image giving the worst result. This is due to the fact that MIT has a lot of straight sharp edges which are not present in the two images of the training set. Bridge is the image giving the best PSNR result even if visually we still see a lot of artifacts.

With those results, we can see that the choice of the images of the training set is very important. And some images with a lot of sharp edges will always be more complicated to quantize.

4.5.2. Impact of the number of images in the training set

In this study, the impact of the number of images in the training set will be analyzed.

The image MIT will be quantized in the same conditions of section Impact of the choice of images in the training set.

The training set will be constituted first by only one image: bridge, then by bridge and camman, then peppers will be added and finally people.

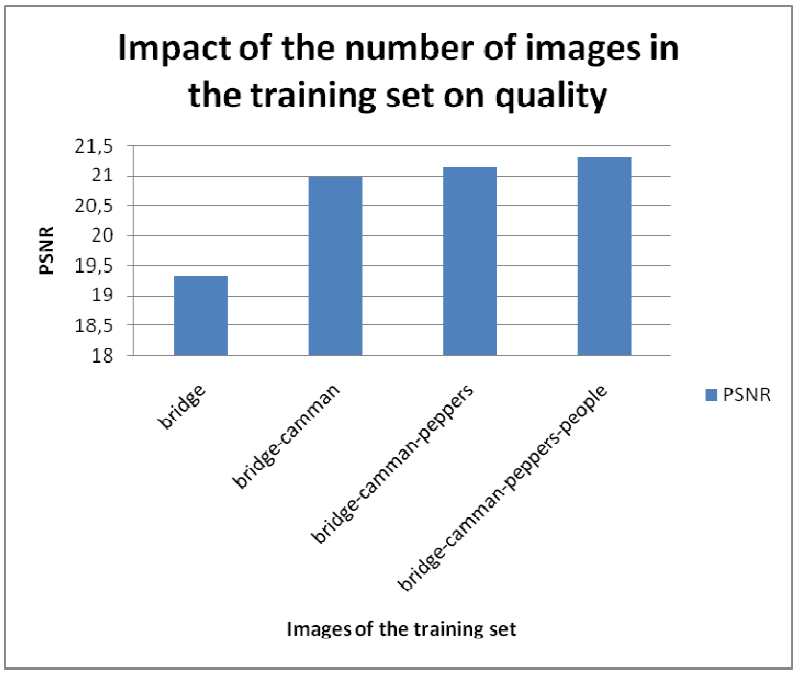
The number of images in the training set has an impact on quality because the image coded is never totally alike the images of the training set so the more we have images in the training set, the more we will be able to find similarity with the original image.

Figure 35 shows the reconstructed images, Table 12 gives the PSNR results and Graphic 9 displays the results.

Figure 35 Reconstructed images changing the number of images in the training set

Impact of the training set on quality	
Training set	**PSNR**
bridge	19,3227
bridge-camman	20,9875
bridge-camman-peppers	21,1498
bridge-camman-peppers-people	21,3094

Table 12 Impact of the images of the training set on quality



Graphic 9 Impact of the number of images in the training set on quality

From Graphic 9 we can see that the quality increases with the number of images in the training set. The most obvious difference is between 1 image and 2 images.

However it is not to be believed that only the number of images is to be taken into account. Indeed, if we add a fifth image which is not at all alike the original image, the PSNR will decrease.

5. Conclusions

The main objective of this project was to study image compression using wavelet transform.

Some background theoretical concepts have first been reviewed.

It has then been shown the difference between two possible filters in the wavelet transform: Haar and D4 Daubechies. It resulted that D4 Daubechies gave better results but was also more complicated. Contrary to Haar filter, D4 Daubechies shows very few block effects when using with scalar quantization. Therefore the PSNR for D4 Daubechies filter is higher: a difference of one point in the case of vector quantization and 3 points in the case of scalar quantization.

Both scalar and vector quantization have been used and without any surprise, vector quantization presented a better compression rate. Indeed, whereas in the case of scalar quantization each pixel has to be coded, vector quantization permits to code a whole block with only one value. Depending on the parameters, vector quantization can present values lower than 0,3bits/pixel while for scalar quantization it is hardly below 1bit/pixel.

In vector quantization, the number of iteration in the convergence of the codebook has a significant impact on PSNR. If we use fewer than 4 iterations, there is a large loss in quality, PSNR losing more than three points when an only iteration is performed.

The quantization of shape 0 of last level can increase the compression rate but with an important decrease of PSNR: 3 points for a decrease of only 0,04bits/pixel. The number of codevector has also a major impact on quality and compression rate. Quality increases with the number of codevector.

Finally, the impact of the images constituting the training set has been analyzed, showing that the image has to be similar with the image compressed and that it is better to have at least 3 to 4 images in the training set in order to match better the image compressed.

6. References

1. **Wikipedia.** Image compression. *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/Image_compression.

2. **Ringler, Christophe et Wentz, Gilles.** La compression d'images numériques. *Techno-Sciences.* [En ligne] 4 August 2004. http://www.techno-science.net/?onglet=articles&article=9.

3. **Linde, Y., Buzo, A. et Gray, R. M.** An algorithm for vector quantizer design. *IEEE Transactions on Communications.* January 1980, Vol. 28, 1, pp. 84-95.

4. **Wikipedia.** Discrete wavelet transform. *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/Discrete_wavelet_transform.

5. —. Fourier transform. *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/Fourier_transform.

6. **Polikar, Robi.** Overview: Why wavelet transform? *The wavelet tutorial.* [En ligne] 12 January 2001. http://users.rowan.edu/~polikar/WAVELETS/WTpart1.html.

7. **Wikipedia.** Wavelet. *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/Wavelet.

8. **Polikar, Robi.** Multiresolution analysis: the discrete wavelet transform. *The wavelet tutorial.* [En ligne] 12 January 2001. http://users.rowan.edu/~polikar/WAVELETS/WTpart4.html.

9. **Wikipedia.** Wavelet. *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/Wavelet#Mother_wavelet.

10. **Vidakovic, Brani et Mueller, Peter.** Wavelets for kids: A tutorial introduction.

11. **Wikipedia.** Haar Wavelet. *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/Haar_wavelet.

12. —. Ingrid Daubechies. *Wikipedia.* [En ligne] http://fr.wikipedia.org/wiki/Ingrid_Daubechies.

13. —. Daubechies wavelet. *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/Daubechies_wavelet.

14. **Kaplan, Ian.** The Daubechies D4 wavelet transform. [En ligne] January 2002. http://www.bearcave.com/misl/misl_tech/wavelets/daubechies/index.html.

15. **Wikipedia.** JPEG 2000. *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/JPEG_2000.

16. **The JPEG committee.** Medical Imaging. *The JPEG committee home page.* [Online] [Cited: December 16, 2009.] http://www.jpeg.org/apps/medical.html.

17. **Wikipedia.** Mach bands. *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/Mach_bands.

18. —. Quantization (image processing). *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/Quantization_%28image_processing%29.

19. **Mukherjee, Amar.** Scalar quantization. [ppt]. www.cs.ucf.edu/courses/cap5015/scalar.ppt.

20. **Wang, Yao.** Quantization. [pdf]. http://eeweb.poly.edu/~yao/EE3414/quantization.pdf.

21. **Wikipedia.** Probability density function. *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/Probability_density_function.

22. —. Algorithme de Lloyd-Max. *Wikipédia.* [En ligne] http://fr.wikipedia.org/wiki/Algorithme_de_Lloyd-Max.

23. **Max, Joe.** Quantizing for minimum distortion. *IRE transaction on Information Theory.* March 1960, Vol. 6, 1, pp. 7-12.

24. **Kumar, Satish.** An introduction to image compression. [En ligne] 22 October 2001. http://www.debugmode.com/imagecmp/.

25. **Averbuch, Amir, Lazar, Danny et Israeli, Moshe.** Image compression using wavelet transform and multiresolution decomposition. *IEEE transactions on image processing.* January 1996, Vol. 5, 1, pp. 4-15.

26. **Wikipedia.** Vector quantization. *Wikipedia, the free encyclopedia.* [En ligne] http://en.wikipedia.org/wiki/Vector_quantization.