B31SE2 Matlab Lab 2 Fourier Theory. Image compression/quantization

Daniel Barmaimon
Advanced Image Analysis - Vibot Master Degree
Heriot Watt University

1 Introduction

The size of the images captures nowadays by cameras is increasing as much as the development of the hardware. For avoiding the problem of the size for storing them and reduce the time for transferring techniques for compress the images are commonly used. In this lab some of this techniques will be studied and analyzed. The quality of quantized and compressed images is going to be compared with the original images.

2 Implementation of Haar Wavelet Transform and Inverse Haar Wavelet Transform

The reduction compression of the image could be implemented by using some linear combinations over the original image. The use of this matrix is based in dividing the image in low frequencies (approximation level) that will be obtained by the average of two consecutive elements of a signal, and the high frequencies (detail level) that will be obtained by the gradient between two consecutive elements of a signal. This step could be repeated iteratively until reach the level of compression desired. The matrix W, for applying over a signal of size N, where N in this case is 8, is the following.

$$\left[\begin{array}{cccccccc} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \hline -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} \end{array}\right] \cdot \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{array}\right] = \frac{1}{2} \left[\begin{array}{c} x_1 + x_2 \\ x_3 + x_4 \\ x_5 + x_6 \\ x_7 + x_8 \\ \hline x_2 - x_1 \\ x_4 - x_3 \\ x_6 - x_5 \\ x_8 - x_7 \end{array}\right]$$

Figure 1: Discrete Haar Wavelet Transformation, W_8

In the same way, it is possible to apply the inverse of this transformation, what is quite easy to calculate because $W_N^{-1} = W_N^T$ The previous theory was applicable over a signal of one dimension. To apply it over an image it will be necessary to apply one transformation for each of the dimensions. This will lead Eq.1.

$$B = W_M A W_N^T \tag{1}$$

The code implemented to get the Haar Wavelet transformation, its inverse, and computation of the matrices needed for its generation are shown below.

```
function [ imOut ] = HaarTransform( im, iter )
[n,m] = size(im);
[Wn,Wm] = generateMatrices( n,m );
% Get the output image for the first iteration
B=Wn*double(im)*Wm';
imOut = B;
% Check if is the last iteration
if (iter == 1), return; end;
% Recursively get the image
imOut(1:n/2,1:m/2) = HaarTransform(imOut(1:n/2,1:m/2), iter-1);
end
```

```
function [ imOut ] = inverseHaarTransform( im, iter )
[n,m] = size(im);
% Get the output image for the first iteration
imOut = im;
% Check if all iterations have been performed
if (iter == 0), return; end;
imOut(1:n/2,1:m/2) = inverseHaarTransform(imOut(1:n/2,1:m/2), iter-1);
% Update the matrices after getting the Inverse Wavelet Tranform
[Wn,Wm] = generateMatrices( n,m );
imOut = Wn'*double(imOut)*Wm;
end

function [ Wn, Wm ] = generateMatrices( n, m )
H = zeros(n/2,n);
G = zeros(n/2,n);
% Repeating the each column to generate H, G
combos = eye(n/2);
for i =0:size(combos,1)-1
H(:,2*i+1) = 1/sqrt(2)*combos(:,i+1);
H(:,2*(i+1)) = H(:,2*i+1);
G(:,2*i+1) = -H(:,2*i+1);
G(:,2*(i+1))= H(:,2*i+1);
end
Wn = double([H;G]);
% Repeating the process for dimension m
H = zeros(m/2,m);
G = zeros(m/2,m);
combos = eye(m/2);
for i =0:size(combos,1)-1
H(:,2*i+1) = 1/sqrt(2)*combos(:,i+1);
H(:,2*(i+1)) = H(:,2*i+1);
G(:,2*i+1) = -H(:,2*i+1);
G(:,2*(i+1))= H(:,2*i+1);
end
Wm = double([H;G]);
end
```

The result of applying consecutively this transformation over an image could be seen in Fig.2

(a) Iteration = 1 (b) Iteration = 2

Figure 2: Wavelet transformation

It could be seen as pyramid, where the size is decrease to the half after each iteration. In the upper left corner is set the original image with it new size (averaged image). Next to it, to the right is located the horizontal gradient. Below the original image is located the vertical gradient. In the lower right corner is set the diagonal gradient vertical gradient (detailed images).

In absence of noise during the compression, the original image can be obtained applying the inverse transformation the same number of iteration that the first one was used.

In terms of implementation, calculation of the matrices H and G to create the matrix W are generated given the size of the original image, the posterior compression and decompression are implemented as recursive function with only two parameters, the image to compress (decompress) and the number of iterations.

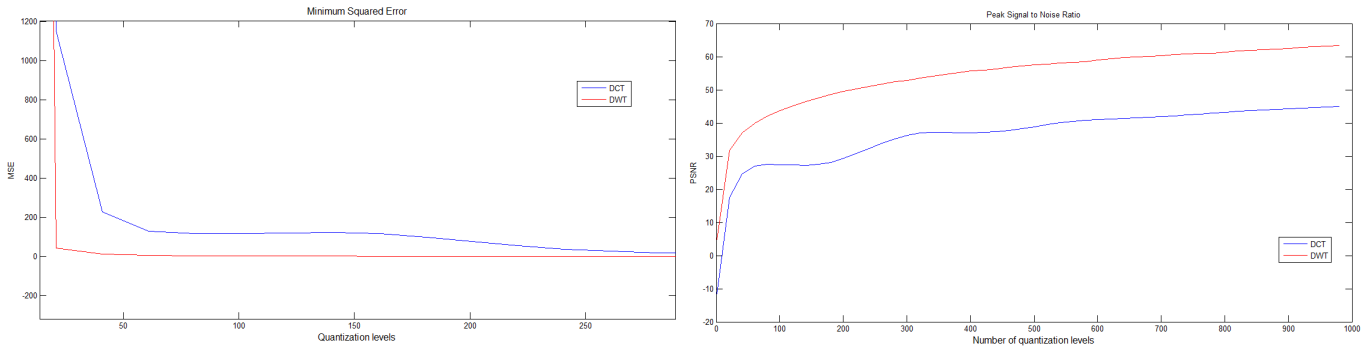
3 Image Compression using DCT and Haar/Daubechies Wavelets

In this section the performance of DCT and Haar/Daubechies wavelets are going to be compared, using as metrics the MSE (minimum squared error) and PSNR (peak signal to noise ratio). In the first case the performance will be compared using the

variation of the number of levels for the quantization to check the evolution of the methods. The second experiment will consist in the comparison between Daubechies with different vanishing moments.

3.1 Compression between DCT and Haar varying the number of quantization levels

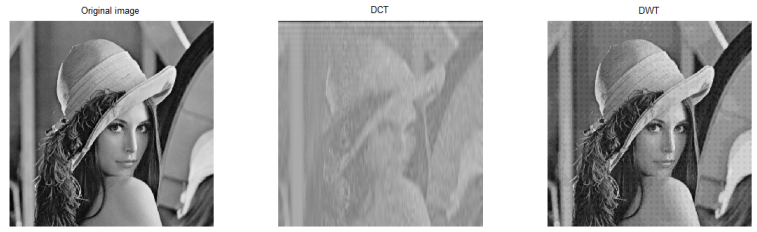
The image of Lena will be compressed using the Discrete Cosine Transform (DCT) and the Haar Transform Wavelet (DWT), and both images will be quantized with the same number of levels for such quantization. The results could be observed quantitatively in Fig. and qualitatively after quantization and decompression in Fig.4.



(a) Minimum Squared Error

(b) Peak Signal to Noise Ratio

Figure 3: Metrics to compare the methods



(a) Number of levels = 21

(b) Number of levels = 61

(c) Number of levels = 201

Figure 4: Comparison for different number of quantization levels

As result it should be said that the performance of the two methods are really close for a great number of quantization levels, but the smaller is this parameter the better is the DWT with respect to DCT to compress the image and quantize the image.

3.2 Comparison of Daubechies with different vanishing moments

In this section different levels of quantization will be applied with Daubechies wavelets with different vanishing moments. As it could be seen in the results, the higher is the number for the vanishing moment, the lower is the MSE and the higher is the PSNR (for high levels of quantization). For low amount of levels the best performance is given by D8 among the rest of the wavelets.

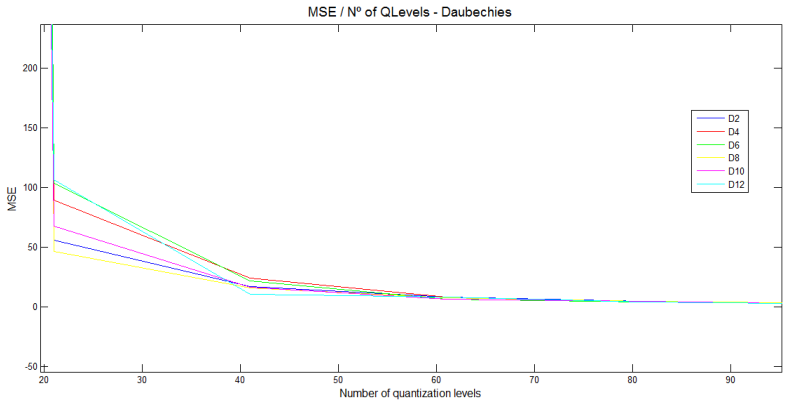
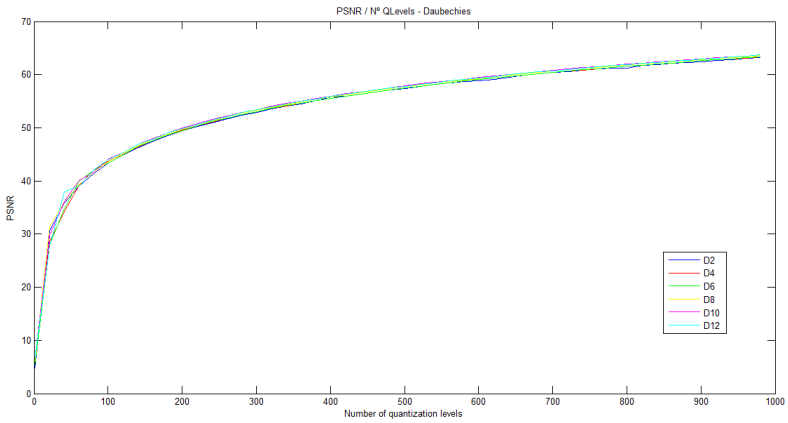
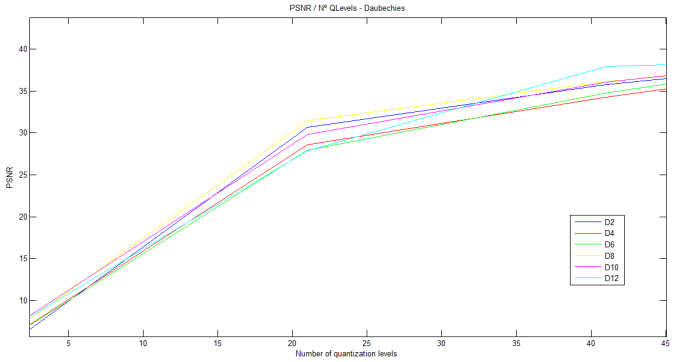


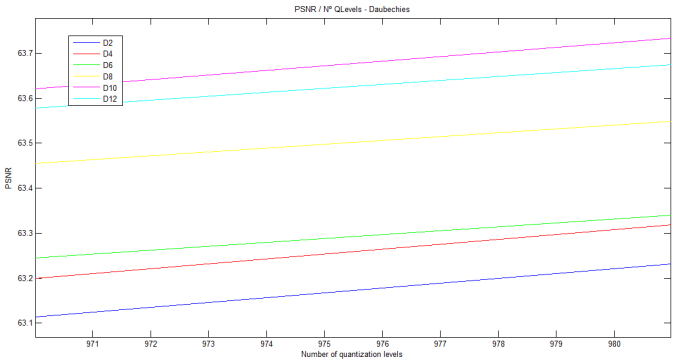
Figure 5: Minimum Squared Error for Daubechies



(a) Peak Signal to Noise Ratio in dB



(b) Detail 1 of PSNR



(c) Detail 2 of PSNR

Figure 6: Comparison for different number of quantization levels

3.3 Possible improvements for quantization

For the quantization two great improvements could be performed.

First one is for the DCT, and the idea is to apply quantization only in the upper right quarter of the image, where the most of the data is saved. Most of the information will be processed and time and space will be reduced quantitatively.

The quantization could be improved using an iterative algorithm that optimize the levels grouping the stretched distribution. This algorithm is called Lloyd's algorithm and will achieve the perfect distribution among the levels because the centroids and borders of them move after each iteration. It will take much longer time to converge if the tolerance is too low.

References

[1] 'Advanced Image Analysis Notes', Mathini Sellathurai, Heriot Watt University, 2014