Report for EQ2330 Image and Video Processing

EQ2330 Image and Video Processing, Project 1

Hanqi Yang
hanqi@kth.se

Qian Zhou
qianzho@kth.se

November 22, 2021

Summary

This report discusses how images can be compressed and reconstructed by two image processing algorithms, discrete cosine transform (DCT) and fast wavelet transform (FWT). We calculate transform matrix and use matrix multiplication to implement DCT and inverse DCT. For 2-dimensional FWT, we recursively use 1-dimensional FWT with Daubechies 8-tap filter. We quantize the transform coefficients and then do inverse transform to obtain the reconstructed image. To evaluate the quality degradation due to quantization, the MSE between the original image and the reconstructed image is calculated and compared with the MSE between the original and quantized coefficients. To estimate the bit-rate needed for coding of the quantized coefficients, we calculate the entropy and PSNR and obtain the rate-PSNR curve.

1 Introduction

In this project, we discuss two transform-based image compression algorithms which are the discrete cosine transform (DCT) and the fast wavelet transform (FWT). We implement transforms, coefficients quantizer and measure the rate-PSNR curve to evaluate them separately.

2 System Description

2.1 DCT-based Image Compression

2.1.1 Blockwise 8×8 DCT

DCT-Ⅱ is a separable orthonormal transform, so the DCT transform of a signal block of size M×M can be defined by M×M transform matrix A containing elements

$$\alpha_{ik} = \alpha_i \cos\left(\frac{(2k+1)i\pi}{2M}\right) \quad \text{for} \quad i,k = 0,1,\ldots,M-1 \tag{1}$$

with

$$\alpha_0 = \sqrt{\frac{1}{M}}, \quad \alpha_i = \sqrt{\frac{2}{M}} \quad \forall i > 0$$

We can use $y = A^T x A$ and $x = A y A^T$ to implement DCT and inverse DCT. According to the project manual, we assume that M=8.

2.1.2 Uniform Quantizer

Uniform quantization, also known as linear coding, is characterized by the fact that the width of each quantization interval (i.e. the wide order) is the same and all values in a quantization interval are mapped to the midpoint of the interval. Its mathematical expression is

$$y = \lceil \frac{x}{q} \rfloor \times q, \tag{2}$$

where ⌈·⌋ denotes the round operator, x is the input coefficient and q is the step-size.

2.1.3 Distortion and Bit-Rate Estimation

The Peak Signal to Noise Ratio (PSNR) will be used to measure the quality of the reconstructed images. It is defined for 8-bit images as follows:

$$\text{PSNR} = 10\log_{10}(\frac{255^2}{d}) \quad [\text{dB}] \tag{3}$$

where d is the mean squared error between the original and the reconstructed images, which can be expressed by

$$MSE = \frac{1}{m}\frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{n}(f(i,j) - f'(i,j))^2 \tag{4}$$

where $f(i,j)$ can be used to represent a digital image and its output is the gray level at the point (i,j).
To estimate the bit-rate, assume that we use a VLC for each of the 64 coefficients in a block. We need to calculate the entropy of every 64×64 blocks as equation (5) and get a 8×8 entropy matrix. Finally we calculate the mean of the entropy matrix to get bit-rate.

$$\widetilde{H} = -\sum_{k=0}^{L-1} p_r(r_k) \times \log_2(p_r(r_k)) \tag{5}$$

where L is the number of intensity values, r_k is the intensities of a certain pixel and $p_k(r_k)$ is the probability of occurrence of r_k.

2.2 FWT-based Image Compression

2.2.1 The Two-Band Filter Bank

To implement a 1-dimensional two-band analysis (synthesis) filter bank function, we use direct implementation by using *dwt()* and *idwt()* function, which consists of filters, down-sampling and up-sampling operations. The block scheme are shown in Figure 2-1.

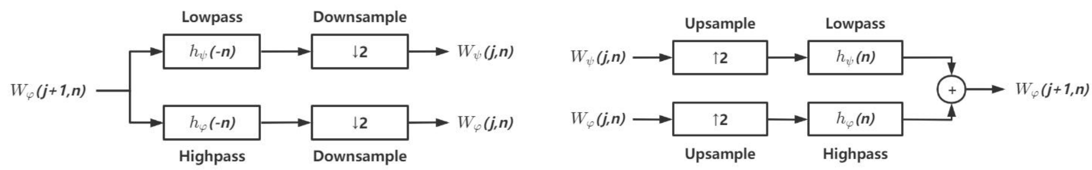

Figure 2-1: 1-dimensional FWT(left) and inverse FWT(right)

2.2.2 2-dimensional FWT and inverse FWT

For the 2-dimensional FWT, we implement Daubechies 8-tap filter. We apply the 1-dimensional FWT to the rows and columns of the image separately to obtain the wavelet coefficients, which are approximate, horizontal, vertical and diagonal coefficients. The block scheme are shown in Figure A-1 in appendix.
We first do the *idwt()* transform to the columns of the approximate and vertical coefficients, then do the *idwt()* transform to the columns of the horizontal and diagonal coefficients to obtain the low and high frequency components, and finally do the *idwt()* transform to the rows of the low and high frequencies to obtain the reconstructed image. The block scheme are shown in Figure A-2 in appendix.

2.2.3 Distortion and Bit-Rate Estimation

We use the same quantizer function as in the DCT transform. We calculate the average distortion d by using equation (4). Similarly, PSNR can be derived by equation (3). When estimating the bit-rate, we measure the entropy of 16 quantized coefficients using equation (5) and calculate their mean as the bit-rate.

3 Results

3.1 DCT-based Image Compression

3.1.1 Blockwise 8 × 8 DCT

Matrix A is shown as below:

$$\mathbf{A}=\begin{pmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{pmatrix}$$

3.1.2 Uniform Quantizer

The uniform mid-tread quantizer with step-size equals to 1 is shown in Figure 3-1.

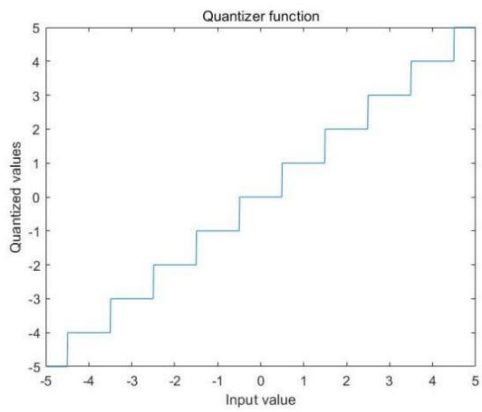

Figure 3-1: Uniform mid-tread quantizer

3.1.3 Distortion and Bit-Rate Estimation

The mean squared error (MSE) between the original and the reconstructed image "*peppers*" is 0.0832, e.g, average distortion *d*. We find that MSE between the original and the quantized DCT coefficients is equal to *d.* This conclusion also applies to the other two images, "*harbour*" and "*boats*", whose *d* are 0.0836 and 0.0833 respectively.

Since DCT is a separable orthonormal transform, it implements a perfect reconstruction. According to Parseval's theorem, the DCT will make the energy of image in spatial domain and frequency domain remain the same. Thus, the two MSE are the same.

Figure A-3 shows the average entropy of all DCT coefficients of three images: boats, harbour and peppers when the step-size equals to 2^6. It can be observed that coefficients of low frequency on the upper left has higher entropy, i.e, carries more information.

The DCT rate-PSNR curve for the total data set is shown in Figure 3-2. We can observe that bit-rate is positively correlated to PSNR. This is because the bit-rate is the sample rate at which an image is compressed according to a particular encoding. Higher bit-rate indicates higher sample rate, which means the original data of the image is preserved better. This leads to a smaller average distortion *d*. Due to the negative correlation between *d* and PSNR, PSNR increases when bit-rate increases.

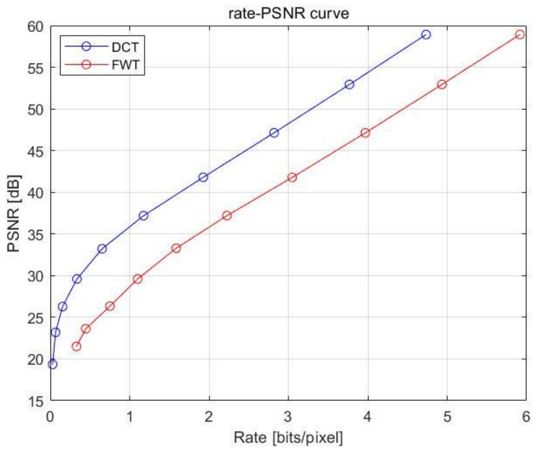

Figure 3-2: rate-PSNR curves

3.2 FWT-based Image Compression

3.2.1 The Two-Band Filter Bank

We create a random one-dimensional array to which we apply the a one-dimensional two-band analysis (synthesis) filter bank function using *dwt()* and *idwt()* function. The signals are shown in Figure A-4. It is observed that there is not much difference between its original and recovered signals.

3.2.2 2-dimensional FWT and inverse FWT

We implement a 2-dimentional FWT by using the 1-dimentional analysis filter bank that we used at 3.2.1. The wavelet coefficients for scale 4 of the image "*harbour*" is

shown in Figure 3-4.
After that we use the uniform mid-tread quantizer to quantize the wavelet coefficients and then implement the inverse FWT transform to reconstruct the image. The comparison is shown in Figure 3-5. We observe that there’s no difference between the original and reconstructed image.

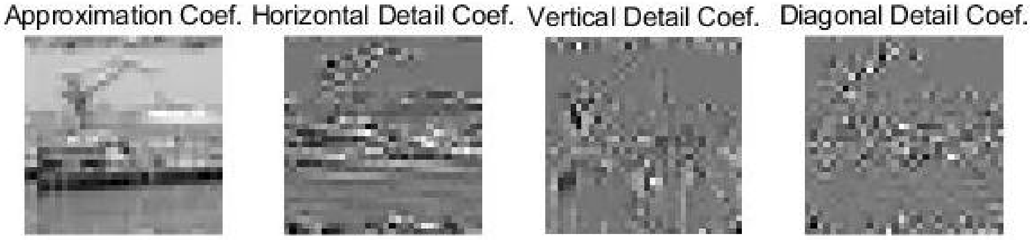

Figure 3-4: Wavelet coefficients of scale 4

Figure 3-5: Original signal and reconstructed images

3.2.3 Distortion and Bit-Rate Estimation

Similarly, we calculate the average distortion d=0.0834 of image “*harbour*” which is approximately equal to the MSE between the original and the quantized wavelet coefficients, roughly 0.08. They are equal as the FWT is also perfectly reconstructed.
When estimating bit-rate, note that each subband can have different size. Thus, we estimate the entropy over each subband (approximation, horizontal, vertical, diagonal details) and average them to get the total entropy of the FWT coefficients. We can observed that the entropy of the approximation coefficients is higher than that of the detail coefficients. For example, when step-size equals to 1, the entropy of approximation coefficients is 9.0714, while the entropy of horizontal, vertical, diagonal details are 5.1210, 5.0932, 4.3793 respectively.
The FWT rate-PSNR curve for the total data set is shown in Figure 3-2. We can observed that bit-rate is positively correlated to PSNR and the gain in performance is of roughly 6dB per added bit/pixel.

4 Conclusions

By using the algorithms of DCT and FWT we designed, we reconstruct the image successfully. We find that DCT and FWT are both perfect reconstruction with the average distortion equaling to the MSE between original and quantized coefficients. When measuring the rate-PSNR curve of the two transforms, it can be observed that bit-rate is positively correlated with PSNR. Comparing the two transformations, we observe that overall DCT performs better than FWT because the DCT's PSNR is higher at the same bit-rate according to the two rate-PSNR curves. But this only shows that DCT performs better than FWT in this comparison condition and can not downplay the usefulness and superior results of FWT.

Reference

[1] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Prentice Hall, 2nd ed., 2002

Appendix

The project was done collaboratively by both authors.

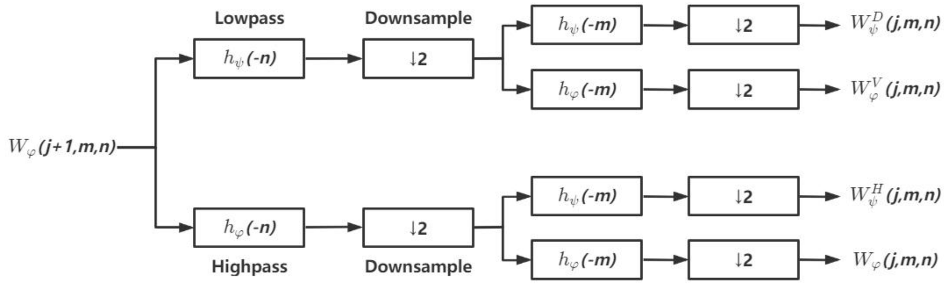

Figure A-1: 2-dimensional FWT

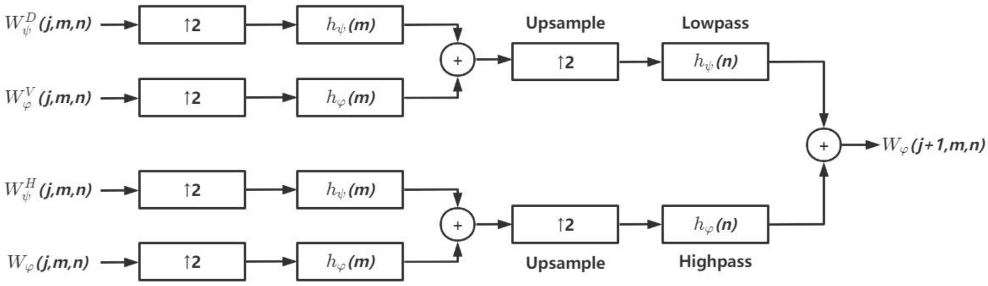

Figure A-2: 2-dimensional inverse FWT

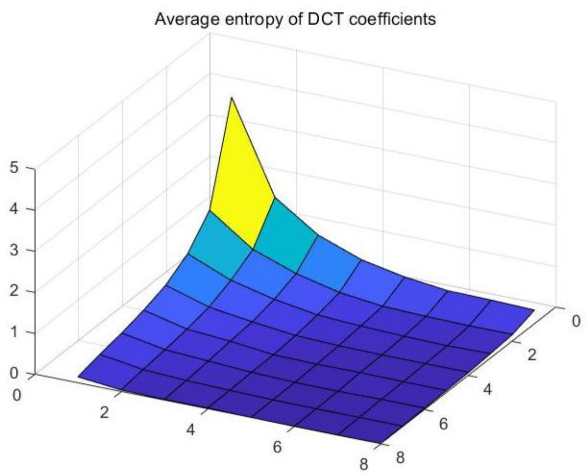

Figure A-3: Average entropy of DCT coefficients of three images, boats, harbour and peppers, at step-size = 2^6

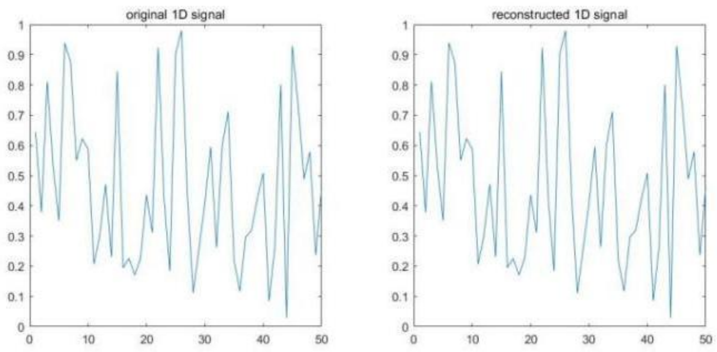

Figure A-4: Original signal and reconstructed signal

Matlab Code

Main program: DCT_quantizer.m, DCT_mse_psnr.m, DCT_rate_psnr_curve.m, FWT_1D.m, FWT_2D_quantizer_mse.m, FWT_rate_psnr_curve.m, DCT_FWT_comparison.m
Function: uniquant.m, plot_matrix.m, mydct.m, mymse.m, mybitrate.m, dwt_analysis.m, dwt_synthesis.m, mymse2.m, mse_coeff.m, mybitrate2.m, myentropy.m

Main program:

- **DCT_quantizer.m**

```
% assignment 2.1&2.2
clc;
clear;

im = double(imread('peppers512x512.tif'));
m = 50;
M = 8;

% assignment 2.1
I = im(m+1:m+M,m+1:m+M);
figure(1);
subplot(1,3,1);
plot_matrix(I);
title('Original');

[D,A] = mydct2(I,M); % DCT
subplot(1,3,2);
plot_matrix(D);
title('DCT');

Dinv = A'*D*A; % inverse DCT
subplot(1,3,3);
plot_matrix(Dinv);
```

```
title('IDCT');

% assignment 2.2
x = [-5:0.01:5];
steplen = 1; %step length
quant = uniquant(x,steplen); % uniform quantizer function
figure(2);
plot(x,quant);
title('Quantizer function');
xlabel('Input value');ylabel('Quantized values');
```

- **DCT_mse_psnr.m**

```
% assignment 2.3.1
% The average distortion d will be the mean squared error between the
original and the
% reconstructed images. Compare d with the mean squared error between
the original and
% the quantized DCT coefficients.
clc;
clear;

im = double(imread('peppers512x512.tif'));
M = 8;
steplen = 1;
size_im = size(im);
size_blk = size_im/M;
mse_oq = 0;
im_recon = zeros(size_im);

for i = 1:size_blk(1)
    for j = 1:size_blk(2)
        row_index = (i-1)*M+1:i*M;
        col_index = (j-1)*M+1:j*M;
        im_blk = im(row_index,col_index);
        % DCT
        [im_dct,A] = mydct2(im_blk,M);
        % quantizer
        im_quant = uniquant(im_dct,steplen);
        % inverse DCT
        im_idct = A'*im_quant*A;
        im_recon(row_index,col_index) = im_idct;
        % MSE between original and quantized DCT coefficient
        mse_oq = mse_oq +
sum((im_dct(:)-im_quant(:)).^2)/length(im_dct(:));
    end
```

```
end
mse_oq = mse_oq/(size_blk(1)*size_blk(2));
% MSE bwtween original and reconstructed image
mse_or = sum((im(:)-im_recon(:)).^2)/length(im(:));

% PSNR
d = mse_or;
psnr = 10*log10(255^2/d);
```

- **DCT_rate_psnr_curve.m**

```
% assignment 2.3.2
% plot rate-PSNR curve
clc;
clear;

im1 = double(imread('boats512x512.tif'));
im2 = double(imread('harbour512x512.tif'));
im3 = double(imread('peppers512x512.tif'));
M = 8;
steplen = 1:10;
bitrate = zeros(1,10);
psnr = zeros(1,10);

for  i = steplen
    step = 2^(i-1);
    mse(1) = mymse(im1,step,M);
    mse(2) = mymse(im2,step,M);
    mse(3) = mymse(im3,step,M);
    d = mean(mse);
    psnr(i) = 10*log10(255^2/d);

    [bitrates(1),entropy1] = mybitrate(im1,step,M);
    [bitrates(2),entropy2] = mybitrate(im2,step,M);
    [bitrates(3),entropy3] = mybitrate(im3,step,M);

    if(i == 7)
        entropy = (entropy1+entropy2+entropy3)/3;
        figure(1);
        surf(entropy);
        title('Average entropy of DCT coefficients');
    end

    bitrate(i) = mean(bitrates);
end
```

- **FWT_1D.m**

```
% assignment 3.1
% 1-D FWT analysis and synthesis
clc;
clear;
load('coeffs.mat');
wname = 'db4';
x = rand(1,50);
[LoD, HiD, LoR, HiR] = wfilters(wname);
[cA,cD] = dwt(x,LoD,HiD,'mode','per');
x_r = idwt(cA,cD,LoR,HiR,'mode','per');

subplot(1,2,1);
plot(x);
title('original 1D signal');
subplot(1,2,2);
plot(x_r);
title('reconstructed 1D signal');
```

- **FWT_2D_quantizer_mse.m**

```
% assignment 3.2&3.3&3.4.1
% 2-D FWT and quantizer
% The average distortion d will be the mean squared error between the
original and the
% reconstructed images. Compare d with the mean squared error between
the original and
% the quantized wavelet coefficients.
clc;
clear;
load('coeffs.mat');
wname = 'db4';
im = double(imread('harbour512x512.tif'));
[m,n] = size(im);
scale = 4;

[cA1,cH1,cV1,cD1] = dwt2_analysis(im,wname);
[cA2,cH2,cV2,cD2] = dwt2_analysis(cA1,wname);
[cA3,cH3,cV3,cD3] = dwt2_analysis(cA2,wname);
[cA4,cH4,cV4,cD4] = dwt2_analysis(cA3,wname);

%plot wavelet coefficients for scale 4
figure(1);
subplot(1,4,1);
plot_matrix(cA4);
title('Approximation Coef.');
```

```
subplot(1,4,2)
plot_matrix(cV4);
title('Horizontal Detail Coef.');
subplot(1,4,3)
plot_matrix(cH4);
title('Vertical Detail Coef.');
subplot(1,4,4)
plot_matrix(cD4);
title('Diagonal Detail Coef.');
sgtitle('Wavelet coefficients of scale 4');

%quantization for step size 1
cA4q = uniquant(cA4,1);
cA3q = uniquant(cA3,1);
cA2q = uniquant(cA2,1);
cA1q = uniquant(cA1,1);

cH4q = uniquant(cH4,1);
cH3q = uniquant(cH3,1);
cH2q = uniquant(cH2,1);
cH1q = uniquant(cH1,1);

cV4q = uniquant(cV4,1);
cV3q = uniquant(cV3,1);
cV2q = uniquant(cV2,1);
cV1q = uniquant(cV1,1);

cD4q = uniquant(cD4,1);
cD3q = uniquant(cD3,1);
cD2q = uniquant(cD2,1);
cD1q = uniquant(cD1,1);

% synthesis
recon3 = dwt2_synthesis(cA4q,cH4q,cV4q,cD4q,wname);
recon2 = dwt2_synthesis(recon3,cH3q,cV3q,cD3q,wname);
recon1 = dwt2_synthesis(recon2,cH2q,cV2q,cD2q,wname);
recon = dwt2_synthesis(recon1,cH1q,cV1q,cD1q,wname);

% plot reconstructed image
figure(2);
subplot(1,2,1);
plot_matrix(im);
title('original image');
subplot(1,2,2);
```

```
plot_matrix(recon);
title('reconstructed image');

% compare d with the mean squared error between the original and the
quantized wavelet coefficients
[mse,mse_co] = mymse2(im,1,wname);
```

- **FWT_rate_psnr_curve.m**

```
% assignment 3.4.2
% plot rate-PSNR curve
clc;
clear;

im1 = double(imread('boats512x512.tif'));
im2 = double(imread('harbour512x512.tif'));
im3 = double(imread('peppers512x512.tif'));
load('coeffs.mat');
wname = 'db4';
% initialize
steplen = 1:10;
bitrate = zeros(1,10);
psnr = zeros(1,10);
entropy = zeros(1,4,10);

for  i = steplen
    step = 2^(i-1);
    [mse(1),~] = mymse2(im1,step,wname);
    [mse(2),~] = mymse2(im2,step,wname);
    [mse(3),~] = mymse2(im3,step,wname);
    d = mean(mse);
    psnr(i) = 10*log10(255^2/d);

    [bitrates(1),entropy1] = mybitrate2(im1,step,wname);
    [bitrates(2),entropy2] = mybitrate2(im2,step,wname);
    [bitrates(3),entropy3] = mybitrate2(im3,step,wname);
    entropy(:,:,i) = (entropy1+entropy2+entropy3)/3;
    bitrate(i) = mean(bitrates);
end
```

- **DCT_FWT_comparison.m**

```
% Comparison of DCT and FWT
clc;
clear;

im1 = double(imread('boats512x512.tif'));
```

```
im2 = double(imread('harbour512x512.tif'));
im3 = double(imread('peppers512x512.tif'));
M = 8;
load('coeffs.mat');
wname = 'db4';
steplen = 1:10;
bitrate_dct = zeros(1,10);
bitrate_fwt = zeros(1,10);
psnr_dct = zeros(1,10);
psnr_fwt = zeros(1,10);

for  i = steplen
    step = 2^(i-1);
    mse_dct(1) = mymse(im1,step,M);
    mse_dct(2) = mymse(im2,step,M);
    mse_dct(3) = mymse(im3,step,M);
    d = mean(mse_dct);
    psnr_dct(i) = 10*log10(255^2/d);

    [bitrates(1),entropy1] = mybitrate(im1,step,M);
    [bitrates(2),entropy2] = mybitrate(im2,step,M);
    [bitrates(3),entropy3] = mybitrate(im3,step,M);

    bitrate_dct(i) = mean(bitrates);
end

for  i = steplen
    step = 2^(i-1);
    [mse_fwt(1),~] = mymse2(im1,step,wname);
    [mse_fwt(2),~] = mymse2(im2,step,wname);
    [mse_fwt(3),~] = mymse2(im3,step,wname);
    d = mean(mse_fwt);
    psnr_fwt(i) = 10*log10(255^2/d);

    [bitrates(1),entropy1] = mybitrate2(im1,step,wname);
    [bitrates(2),entropy2] = mybitrate2(im2,step,wname);
    [bitrates(3),entropy3] = mybitrate2(im3,step,wname);
    bitrate_fwt(i) = mean(bitrates);
end

figure;
plot(bitrate_dct,psnr_dct,'b-o');
hold on;
plot(bitrate_fwt,psnr_fwt,'r-o');
```

```
xlabel('Rate [bits/pixel]');
ylabel('PSNR [dB]');
title('rate-PSNR curve');
legend('DCT','FWT');
grid on;
```

Function:

- **uniquant.m**

```
function quant = uniquant(x,steplen)
% uniform mid-tread quantizer function
quant = round(x/steplen)*steplen;
end
```

- **plot_matrix.m**

```
function plot_matrix(A)
imagesc(A);
axis square;
axis off;
colormap(gray);
end
```

- **mydct2.m**

```
function [D,A] = mydct2(I,M)
A = zeros(M);
for i=0:M-1
    for j=0:M-1
        if i==0
            alpha = sqrt(1/M);
        else
            alpha = sqrt(2/M);
        end
        A(i+1,j+1) = alpha*cos((2*j+1)*i*pi/(2*M));
    end
end
D = A*I*A';
end
```

- **mymse.m**

```
function mse = mymse(im,steplen,M)
% output: mse: MSE between the original and the reconstructed images
size_im = size(im);
[m,n] = size(im);
size_blk = size_im/M;
im_recon = zeros(size_im);

for i = 1:size_blk(1)
```

```
    for j = 1:size_blk(2)
        row_index = (i-1)*M+1:i*M;
        col_index = (j-1)*M+1:j*M;
        im_blk = im(row_index,col_index);
        % DCT
        [im_dct,A] = mydct2(im_blk,M);
        % quantizer
        im_quant = uniquant(im_dct,steplen);
        % inverse DCT
        im_idct = A'*im_quant*A;
        im_recon(row_index,col_index) = im_idct;
    end
end
mse = sum((im(:)-im_recon(:)).^2)/(m*n);
end
```

- **mybitrate.m**

```
function bitrate = mybitrate(im,steplen,M)
size_im = size(im);
size_blk = size_im/M;
im_quant = zeros(size_im);
for i = 1:size_blk(1)
    for j = 1:size_blk(2)
        row_index = (i-1)*M+1:i*M;
        col_index = (j-1)*M+1:j*M;
        im_blk = im(row_index,col_index);
        % DCT
        [im_dct,~] = mydct2(im_blk,M);
        % quantizer
        im_q = uniquant(im_dct,steplen);
        im_quant(row_index,col_index) = im_q;
    end
end

K = zeros(8,8,64*64);
entropy = zeros(M);

for m = 1:M
    for n=1:M
        for index =1
            for p = 0:63
                for q = 0:63
                    K(m,n,index) = im_quant(m+8*p,n+8*q);
                    index = index+ 1;
                end
```

```
            end
        end
    end
end

for m=1:M
    for n=1:M
        value = K(m,n,:);
        bins= min(value):steplen:max(value);
        pr = hist(value(:),bins(:));
        prb = pr/sum(pr);
        etrpy = -sum(prb.*log2(prb+eps));
        entropy(m,n) = etrpy;
    end
end
bitrate = mean2(entropy);
End
```

- **dwt_analysis.m**

```
function [cA,cH,cV,cD] = dwt2_analysis(im,wname)
[m,n] = size(im);
[LoD, HiD, ~, ~] = wfilters(wname);
for i=1:m
    [A,D] = dwt(im(i,:),LoD,HiD,'mode','per');
    im(i,:) = [A,D];
end

for j=1:n
    [A,D] = dwt(im(:,j),LoD,HiD,'mode','per');
    im(:,j) = [A;D];
end

cA=im(1:m/2,1:n/2); % approximation coefficients
cH=im(1:m/2,n/2+1:n); % horizontal detail coefficients
cV=im(m/2+1:m,1:n/2); % vertical detail coefficients
cD=im(m/2+1:m,n/2+1:n); % diagonal detail coefficients

end
```

- **dwt_synthesis.m**

```
function [recon] = dwt2_synthesis(cA,cH,cV,cD,wname)
[m,n] = size(cA);
[~, ~, LoR, HiR] = wfilters(wname);
for j = 1:n
    L(:,j) = idwt(cA(:,j),cV(:,j),LoR,HiR,'mode','per');
```

```
end
for j = 1:n
    H(:,j) = idwt(cH(:,j),cD(:,j),LoR,HiR,'mode','per');
end
m=2*m;
for i = 1:m
    recon(i,:) = idwt(L(i,:),H(i,:),LoR,HiR,'mode','per');
end
end
```

- **mymse2.m**

```
function [mse,mse_co] = mymse2(im,steplen,wname)
% output: mse: MSE between the original and the reconstructed images
%        mse_co: MSE between  the original and the quantized wavelet
coefficients

[cA1,cH1,cV1,cD1] = dwt2_analysis(im,wname);
[cA2,cH2,cV2,cD2] = dwt2_analysis(cA1,wname);
[cA3,cH3,cV3,cD3] = dwt2_analysis(cA2,wname);
[cA4,cH4,cV4,cD4] = dwt2_analysis(cA3,wname);

%quantization for step size 1
cA4q = uniquant(cA4,steplen);
cA3q = uniquant(cA3,steplen);
cA2q = uniquant(cA2,steplen);
cA1q = uniquant(cA1,steplen);

cH4q = uniquant(cH4,steplen);
cH3q = uniquant(cH3,steplen);
cH2q = uniquant(cH2,steplen);
cH1q = uniquant(cH1,steplen);

cV4q = uniquant(cV4,steplen);
cV3q = uniquant(cV3,steplen);
cV2q = uniquant(cV2,steplen);
cV1q = uniquant(cV1,steplen);

cD4q = uniquant(cD4,steplen);
cD3q = uniquant(cD3,steplen);
cD2q = uniquant(cD2,steplen);
cD1q = uniquant(cD1,steplen);

% synthesis
recon3 = dwt2_synthesis(cA4q,cH4q,cV4q,cD4q,wname);
recon2 = dwt2_synthesis(recon3,cH3q,cV3q,cD3q,wname);
```

```
recon1 = dwt2_synthesis(recon2,cH2q,cV2q,cD2q,wname);
recon = dwt2_synthesis(recon1,cH1q,cV1q,cD1q,wname);

% mse between orignal image and reconstructed image
mse = mse_coeff(recon,im);

% mse between original coeff. and quantized coeff.
mse_co = zeros(4,4);
mse_co(1,1) = mse_coeff(cA1q,cA1);
mse_co(1,2) = mse_coeff(cA2q,cA2);
mse_co(1,3) = mse_coeff(cA3q,cA3);
mse_co(1,4) = mse_coeff(cA4q,cA4);
mse_co(2,1) = mse_coeff(cH1q,cH1);
mse_co(2,2) = mse_coeff(cH2q,cH2);
mse_co(2,3) = mse_coeff(cH3q,cH3);
mse_co(2,4) = mse_coeff(cH4q,cH4);
mse_co(3,1) = mse_coeff(cV1q,cV1);
mse_co(3,2) = mse_coeff(cV2q,cV2);
mse_co(3,3) = mse_coeff(cV3q,cV3);
mse_co(3,4) = mse_coeff(cV4q,cV4);
mse_co(4,1) = mse_coeff(cD1q,cD1);
mse_co(4,2) = mse_coeff(cD2q,cD2);
mse_co(4,3) = mse_coeff(cD3q,cD3);
mse_co(4,4) = mse_coeff(cD4q,cD4);
end
```

- **mse_coeff.m**

```
function mse = mse_coeff(oc,qc)
% input: oc: original coefficients
%       qc: quantized coefficients
[m,n] = size(oc);
mse = sum((oc-qc).^2,'all')/(m*n);
end
```

- **mybitrate2.m**

```
function [bitrate,entropy] = mybitrate2(im,steplen,wname)
[cA1,cH1,cV1,cD1] = dwt2_analysis(im,wname);
[cA2,cH2,cV2,cD2] = dwt2_analysis(cA1,wname);
[cA3,cH3,cV3,cD3] = dwt2_analysis(cA2,wname);
[cA4,cH4,cV4,cD4] = dwt2_analysis(cA3,wname);

%quantization for step size 1
cA4q = uniquant(cA4,steplen);
cA3q = uniquant(cA3,steplen);
cA2q = uniquant(cA2,steplen);
```

```
cA1q = uniquant(cA1,steplen);

cH4q = uniquant(cH4,steplen);
cH3q = uniquant(cH3,steplen);
cH2q = uniquant(cH2,steplen);
cH1q = uniquant(cH1,steplen);

cV4q = uniquant(cV4,steplen);
cV3q = uniquant(cV3,steplen);
cV2q = uniquant(cV2,steplen);
cV1q = uniquant(cV1,steplen);

cD4q = uniquant(cD4,steplen);
cD3q = uniquant(cD3,steplen);
cD2q = uniquant(cD2,steplen);
cD1q = uniquant(cD1,steplen);

cAq = [cA1q(:);cA2q(:);cA3q(:);cA4q(:)];
cHq = [cH1q(:);cH2q(:);cH3q(:);cH4q(:)];
cVq = [cV1q(:);cV2q(:);cV3q(:);cV4q(:)];
cDq = [cD1q(:);cD2q(:);cD3q(:);cD4q(:)];

entropy = zeros(1,4);
entropy(1) = myentropy(cAq,steplen);
entropy(2) = myentropy(cHq,steplen);
entropy(3) = myentropy(cVq,steplen);
entropy(4) = myentropy(cDq,steplen);

bitrate = mean(entropy);
end
```

- **myentropy.m**

```
function [entropy] = myentropy(value,steplen)
bins = min(value):steplen:max(value);
pr = hist(value(:),bins(:));
prb = pr/sum(pr);
entropy = -sum(prb.*log2(prb+eps));
end
```