University of Victoria
Department of Electrical and Computer Engineering
ELEC 534 Digital Signal Processing

Laboratory Report

Experiment No: 3

Title: Application of 2-D DCT to Image Compression

Report Submitted on: 26/02/2010

To: Dr. W.-S Lu

Name: Chamira Edussooriya (V00716349)

Contents

1 Objective **4**

2 Introduction **4**

2.1 Application of 2-D DCT to Image Compression 4

3 Results **5**

4 Discussion **8**

5 Conclusions **9**

A MATLAB Code **9**

A.1 Main Function 9

A.2 Function *imcomp* 12

List of Figures

1 Camere256 (a) original image (b) quality level = 10 (c) quality level = 40 (d) quality level = 90 6

2 Boat512 (a) original image (b) quality level = 10 (c) quality level = 40 (d) quality level = 90 6

3 Goldhill (a) original image (b) quality level = 10 (c) quality level = 40 (d) quality level = 90 7

4 Peppers (a) original image (b) quality level = 10 (c) quality level = 40 (d) quality level = 90 7

List of Tables

1 Percentage of zeros resulted in the quantization step 8

2 PSNR of compressed images (in dB) 8

1 Objective

The objective of this experiment is to apply 2-D DCT to compress digital images.

2 Introduction

For a 2-D signal $x[m,n]$, where $m,n = 0,1,\ldots,N-1$, 2-D DCT is defined by

$$C(i,k) = \frac{2\alpha(i)\alpha(k)}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x[m,n] \cos\left(\frac{(2m+1)i\pi}{2N}\right) \cos\left(\frac{(2n+1)k\pi}{2N}\right) \tag{1}$$

where $i,k = 0,1,\ldots,N-1$, and $\alpha(\cdot)$ is defined by

$$\alpha(k) = \begin{cases} 1/\sqrt{2} & \text{for } \; k = 0 \\ 1 & \text{for } \; 1 \leq k \leq N-1 \end{cases} \tag{2}$$

The 2-D inverse DCT (IDCT) is defined by

$$x[m,n] = \frac{2}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{N-1} \alpha(i)\alpha(k) C(i,k) \cos\left(\frac{(2m+1)i\pi}{2N}\right) \cos\left(\frac{(2n+1)k\pi}{2N}\right) \tag{3}$$

where $m,n = 0,1,\ldots,N-1$. When 2-D DCT is applied to an 8×8 data block ($N = 8$), such as in JPEG process, the Eq. (1) becomes

$$C(i,k) = \frac{\alpha(i)\alpha(k)}{4} \sum_{m=0}^{7} \sum_{n=0}^{7} x[m,n] \cos\left(\frac{(2m+1)i\pi}{16}\right) \cos\left(\frac{(2n+1)k\pi}{16}\right) \tag{4}$$

where $i,k = 0,1,\ldots,7$. Among the 64 coefficients defined by Eq. (4), $C(0,0)$ is called the *DC coefficient* since it is proportional to the average value of the signal and, the rest 63 coefficients, $C(i,k)$ with $(i,k) \neq (0,0)$, are called *AC coefficients*.

The 2-D DCT exhibits several important properties. First, it is an *orthogonal* transform since the basis matrices are orthogonal. Also, it preserves the energy and can be evaluated as two 1-D DCTs since it is separable. The de-correlation ability and the energy compaction ability of 2-D DCT are the most important properties which lead to be used it for image compression. In fact, the process of image compression is one that removes the redundancy from the image.

2.1 Application of 2-D DCT to Image Compression

Let the image to be compressed is of size $M \times N$ with both M and N being a multiple of 8. First the image is divided into 8×8 blocks. Then, the steps described below are applied to the each of these blocks. Let $\mathbf{B}$ represent an 8×8 data block.

step 1: Level-off and 2-D DCT - the pixel values of $\mathbf{B}$ ranging from 0 to 255 are leveled off by subtracting 128 from each entry since DCT works better on pixel values from -128

to 127. And, let the leveled off date block be $\widetilde{\mathbf{B}}$. Then, 2-D DCT is applied to $\widetilde{\mathbf{B}}$ and let the resulting matrix be $\mathbf{C}$.

step 2: Quantization - the most important step in the compression process is the quantization which determines the quality of the reconstructed image. The quantization is achieved by converting matrix $\mathbf{C} = \{c_{i,j}\}$ to matrix $\mathbf{S} = \{s_{i,j}\}$ with $s_{i,j}$ determined by

$$s_{i,j} = \text{round}\left(\frac{c_{i,j}}{q_{i,j}}\right) \tag{5}$$

where $\mathbf{Q} = \{q_{i,j}\}$ is a quantization matrix that determines the desired quality level of the compression. In the JPEG standard, the matrix for the quality level 50 ($\mathbf{Q}_{50}$) has been specified and the matrices for other quality levels can be obtained by simply multiplying the $\mathbf{Q}_{50}$ by a scaling factor τ given as

$$\tau = \begin{cases} (100 - \text{quality level})/50 & \text{for quality level} > 50 \\ 50/\text{quality level} & \text{for quality level} < 50 \end{cases} \tag{6}$$

and, rounding and clipping to integer values between 0 and 255.

step 3: Coding - the quantized matrix is then coded. The DC coefficients of 8×8 blocks are encoded by differential pulse-code modulation and the AC coefficients of each block are scanned in a zig-zag pattern and ordered into (run,level) pairs, which are then entropy coded.

step 4: Decompression - this consists of three operations. First, matrix $\mathbf{S}$ is pointwise multiplied with $\mathbf{Q}$

$$\mathbf{R} = \mathbf{Q} \odot \mathbf{S} \tag{7}$$

Then, the 2-D IDCT is applied to $\mathbf{R}$ and 128 is added to each entry for the compensation of level off. The resulting matrix approximately resembles the original data block $\mathbf{B}$ quite well.

3 Results

Four test images: *camera256, boat512, goldhill512* and *peppers512* are compressed using the method described in the preceding section with three quality levels: 10, 40 and 90. Figures 1 to 4 illustrate the resulted images and the original images. For all four images, it can be observed that the compressed images with the quality levels 90 and 40 are almost equal to the original ones whereas the compressed images with the quality level 10 are distorted and exhibit the *blocking effect.*

The percentages of zeros resulted in the quantization step for each image with the three quality levels are presented in the Table 1. As expected, the percentage of zeros increases with the decreasing quality level. It can be observed that more than 60% of zeros for all four images even at the quality level 90. And, for the quality level 10, about 95% of zeros can be observed for all images.

The quality of the images are further evaluated using the peak signal-to-noise ratio (PSNR),

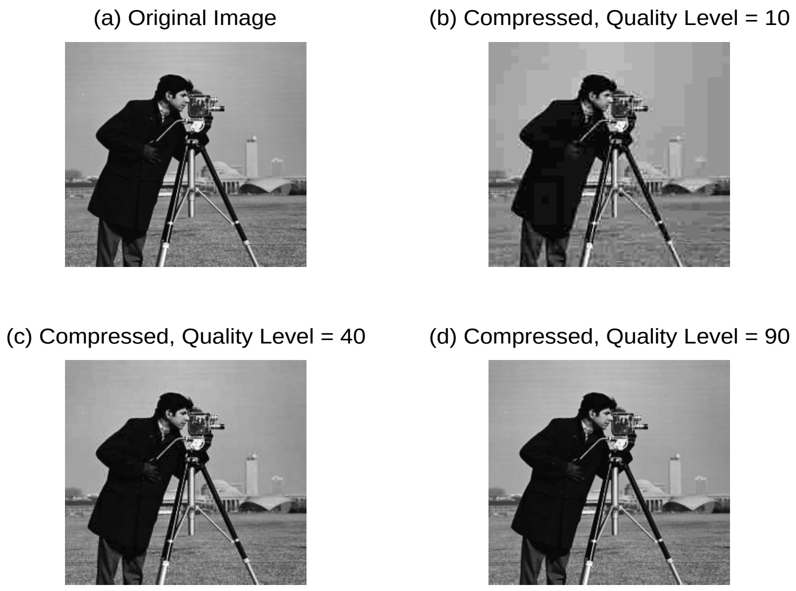
(a) Original Image
(b) Compressed, Quality Level = 10
(c) Compressed, Quality Level = 40
(d) Compressed, Quality Level = 90

Figure 1: Camere256 (a) original image (b) quality level = 10 (c) quality level = 40 (d) quality level = 90

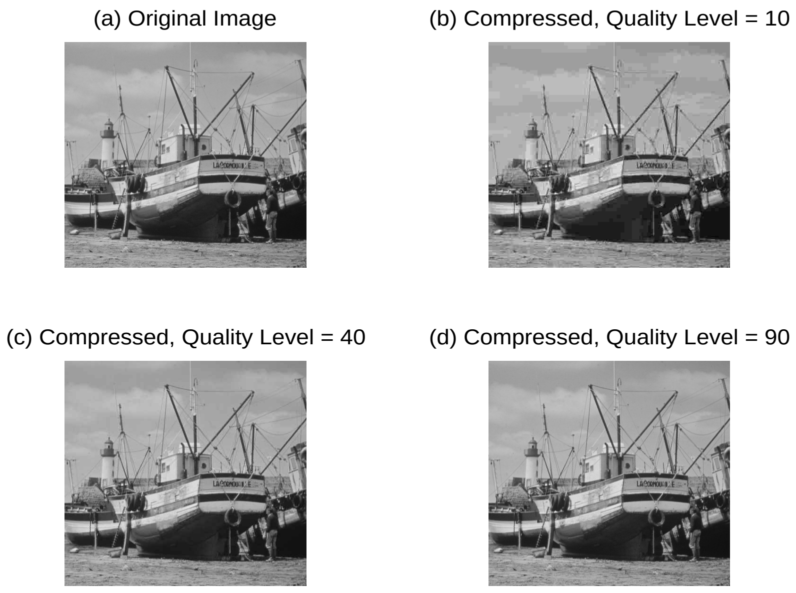
(a) Original Image
(b) Compressed, Quality Level = 10
(c) Compressed, Quality Level = 40
(d) Compressed, Quality Level = 90

Figure 2: Boat512 (a) original image (b) quality level = 10 (c) quality level = 40 (d) quality level = 90

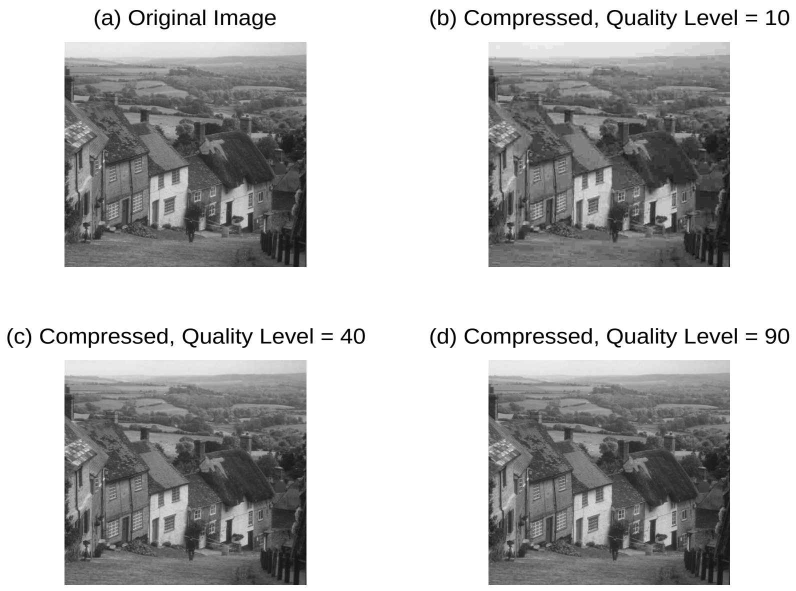
(a) Original Image
(b) Compressed, Quality Level = 10
(c) Compressed, Quality Level = 40
(d) Compressed, Quality Level = 90

Figure 3: Goldhill (a) original image (b) quality level = 10 (c) quality level = 40 (d) quality level = 90

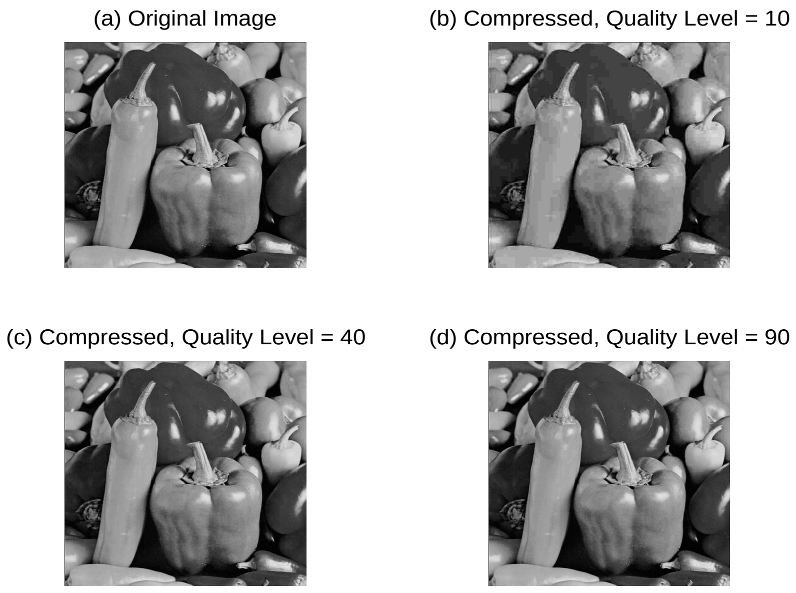
(a) Original Image
(b) Compressed, Quality Level = 10
(c) Compressed, Quality Level = 40
(d) Compressed, Quality Level = 90

Figure 4: Peppers (a) original image (b) quality level = 10 (c) quality level = 40 (d) quality level = 90

Table 1: Percentage of zeros resulted in the quantization step

Image	Quality Level		
	10	40	90
camera256	95.01	87.25	63.87
boat512	95.36	88.49	68.78
goldhill512	95.64	86.86	61.76
peppers512	96.00	90.61	67.27

which is defined as

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{\psi_{\max}^2}{\sigma_e^2}\right) \tag{8}$$

where $\psi_{\max}$ is the maximum light intensity of the original image; for an 8 bit digital image $\psi_{\max} = 255$, and σ_e^2 is the mean squared error. The results obtained for the four images are shown in the Table 2. For the quality level 90, the resulted PSNR is about 40 dB and, even for

Table 2: PSNR of compressed images (in dB)

Image	Quality Level		
	10	40	90
camera256	26.24	30.77	39.97
boat512	28.86	33.88	41.03
goldhill512	28.65	32.91	39.35
peppers512	30.12	34.27	39.17

the quality level 10, an acceptable PSNR, more than 26 dB, can be observed for all four images.

4 Discussion

Four test images were compressed using 2-D DCT method with three quality levels and, the results were compared in terms of visual quality, percentage of zeros, and PSNR. The compressed images with the quality levels 90 and 40 are visually almost equal to the original images and almost no visual loss presents in the compressed images. However, the compressed images with the quality level 10 are distorted and it can be clearly observed the blocking effect. And, the percentage of zeros increases with the decreased quality level. It can be observed that roughly 65%, 88% and 95% of zeros for all images with quality levels 90, 40 and 10, respectively. Also, as the quality level increases, as expected, PSNR increases and, PSNR of more than 26 dB, about 32 dB and 40 dB resulted for the quality levels 10, 40 and 90, respectively. Overall, it can be seen that at lower quality levels, the quality goes down considerably while the compression does not increase very much.

Further, it can be observed that all four images do not maintain the same visual quality for a given compression ratio due to the different contrast and the frequency content of the images. And, generally, high contrast images or images with a considerable high frequency content cannot be compressed to maintain the same visual quality that resulted by smooth and low frequency images.

5 Conclusions

Four test images were compressed using 2-D DCT method with three quality levels. For the quality levels 40 and 90, the reconstructed images are almost equal to the originals and for the quality level 10, reconstructed images are distorted by the blocking effect. And, at lower quality levels, the quality goes down considerably while the compression does not increase very much. Also, the same visual quality cannot be maintained for different images and it depends on the contrast and the frequency content of the image. And, high contrast images or images with a considerable high frequency content are more difficult to compress than the smooth and low frequency images.

A MATLAB Code

A.1 Main Function

```
% ELEC 534 - Experiment 3
% Application of 2-D DCT to Image Compression

clear all; clc;
close all;

% Load images

load camera256.mat
load boat512.mat
load goldhill512.mat
load peppers512.mat

% Compression of camera256

[cam10,zpcam10,psnrcam10] = imcomp(camera256,10);   % quality level = 10
[cam40,zpcam40,psnrcam40] = imcomp(camera256,40);   % quality level = 40
[cam90,zpcam90,psnrcam90] = imcomp(camera256,90);   % quality level = 90

subplot(2,2,1);
imshow(camera256,[0 255]);
title('(a) Original Image');

subplot(2,2,2);
imshow(cam10,[0 255]);
title('(b) Compressed, Quality Level = 10');
```

```
subplot(2,2,3);
imshow(cam40,[0 255]);
title('(c) Compressed, Quality Level = 40');

subplot(2,2,4);
imshow(cam90,[0 255]);
title('(d) Compressed, Quality Level = 90');

fid = fopen('Ex3.txt','wt');
fprintf(fid,'Percentage of zeros for camera256\n');
fprintf(fid,'Quality level[10 40 90] = [%.2f %.2f %.2f]\n',zpcam10,...
        zpcam40,zpcam90);
fprintf(fid,'PSNR for camera256\n');
fprintf(fid,'Quality level[10 40 90] = [%.2f %.2f %.2f]\n',psnrcam10,...
        psnrcam40,psnrcam90);

% Compression of boat512

[boat10,zpboat10,psnrboat10] = imcomp(boat512,10);  % quality level = 10
[boat40,zpboat40,psnrboat40] = imcomp(boat512,40);  % quality level = 40
[boat90,zpboat90,psnrboat90] = imcomp(boat512,90);  % quality level = 90

figure;
subplot(2,2,1);
imshow(boat512,[0 255]);
title('(a) Original Image');

subplot(2,2,2);
imshow(boat10,[0 255]);
title('(b) Compressed, Quality Level = 10');

subplot(2,2,3);
imshow(boat40,[0 255]);
title('(c) Compressed, Quality Level = 40');

subplot(2,2,4);
imshow(boat90,[0 255]);
title('(d) Compressed, Quality Level = 90');

fprintf(fid,'\nPercentage of zeros for boat512\n');
fprintf(fid,'Quality level[10 40 90] = [%.2f %.2f %.2f]\n',zpboat10,...
        zpboat40,zpboat90);
fprintf(fid,'PSNR for boat512\n');
fprintf(fid,'Quality level[10 40 90] = [%.2f %.2f %.2f]\n',psnrboat10,...
        psnrboat40,psnrboat90);

% Compression of goldhill512

[gh10,zpgh10,psnrgh10] = imcomp(goldhill512,10);    % quality level = 10
[gh40,zpgh40,psnrgh40] = imcomp(goldhill512,40);    % quality level = 40
[gh90,zpgh90,psnrgh90] = imcomp(goldhill512,90);    % quality level = 90

figure;
subplot(2,2,1);
imshow(goldhill512,[0 255]);
title('(a) Original Image');

subplot(2,2,2);
imshow(gh10,[0 255]);
title('(b) Compressed, Quality Level = 10');

subplot(2,2,3);
imshow(gh40,[0 255]);
title('(c) Compressed, Quality Level = 40');

subplot(2,2,4);
imshow(gh90,[0 255]);
title('(d) Compressed, Quality Level = 90');

fprintf(fid,'\nPercentage of zeros for goldhill512\n');
fprintf(fid,'Quality level[10 40 90] = [%.2f %.2f %.2f]\n',zpgh10,...
        zpgh40,zpgh90);
fprintf(fid,'PSNR for goldhill512\n');
fprintf(fid,'Quality level[10 40 90] = [%.2f %.2f %.2f]\n',psnrgh10,...
        psnrgh40,psnrgh90);

% Compression of peppers512

[pepp10,zppepp10,psnrpepp10] = imcomp(peppers512,10);   % quality level = 10
[pepp40,zppepp40,psnrpepp40] = imcomp(peppers512,40);   % quality level = 40
[pepp90,zppepp90,psnrpepp90] = imcomp(peppers512,90);   % quality level = 90

figure;
subplot(2,2,1);
imshow(peppers512,[0 255]);
title('(a) Original Image');

subplot(2,2,2);
imshow(pepp10,[0 255]);
```

```
title('(b) Compressed, Quality Level = 10');

subplot(2,2,3);
imshow(pepp40,[0 255]);
title('(c) Compressed, Quality Level = 40');

subplot(2,2,4);
imshow(pepp90,[0 255]);
title('(d) Compressed, Quality Level = 90');

fprintf(fid,'\nPercentage of zeros for peppers512\n');
fprintf(fid,'Quality level[10 40 90] = [%.2f %.2f %.2f]\n',zppepp10,...
        zppepp40,zppepp90);
fprintf(fid,'PSNR for peppers512\n');
fprintf(fid,'Quality level[10 40 90] = [%.2f %.2f %.2f]\n',psnrpepp10,...
        psnrpepp40,psnrpepp90);

fclose(fid);
```

A.2 Function *imcomp*

```
function [R,zeroper,psnr] = imcomp(image,qualev)

% IMCOMP compresses the images using 2-D DCT
% inputs
%   image - image to be compressed
%   qualev - quality level
% outputs
%   R - reconstructed image
%   zeroper - percentage of zero coefficients
%   psnr - peak signal-to-noise ratio

% Divide image into 8x8 blocks

[M,N] = size(image);
m = M/8;    % number of blocks along dimension 1
n = N/8;    % number of blocks along dimension 2

% Determine the quantization matrix

Q50 = [16,11,10,16,24,40,51,61;
       12,12,14,19,26,58,60,55;
       14,13,16,24,40,57,69,56;
       14,17,22,29,51,87,80,62;
       18,22,37,56,68,109,103,77;
       24,35,55,64,81,104,113,92;
       49,64,78,87,103,121,120,101;
       72,92,95,98,112,100,103,99];     % matrix for quality level =50

if (qualev > 50 && qualev <= 100)
    tau = (100-qualev)/50;
    Q = round(tau*Q50);
    for i = 1:8
        for j = 1:8
            if (Q(i,j)>255)
                Q(i,j) = 255;   % clip the values grater than 255
            end
        end
    end
elseif (qualev < 50 && qualev >= 0)
    tau = 50/qualev;
    Q = round(tau*Q50);
elseif (qualev == 50)
    Q = Q50;
else
    fprintf('\nError: quality level should be between 0 and 100.\n');
end

totzero = 0;    % initialization of total number of zeros

% Image compression and reconstruction

ini1 = 1;   % initial position of the block (dim 1)

for i = 1:m
    ini2 = 1;   % initial position of the block (dim 2)
    for j = 1:n
        B = image(ini1:ini1+7,ini2:ini2+7); % extract the 8x8 block
        Blo = B - 128;      % level off by subtracting 128
        C = dct2(Blo);      % apply 2-D DCT
        S = round(C./Q);    % quantization

        D = Q.*S;           % decompression
        E = idct2(D);       % apply 2-D IDCT
        R(ini1:ini1+7,ini2:ini2+7) = E + 128;   % compensation for level-off

        for p = 1:8         % count number of zero coefficients
            for q = 1:8
                if (D(p,q) == 0)
                    totzero = totzero + 1;  % increase by one
                end
            end
        end

        ini2 = ini2 + 8;    % update ini2
    end
    ini1 = ini1 + 8;    % update ini1
end

zeroper = totzero*100/(M*N);    % percentage of zero coefficients

% calculate PSNR

psimax = 255;       % maximum light intensity
sigmasq = mean(mean((R-image).^2));     % mean square error
psnr = 10*log10(psimax^2/sigmasq);
```