Class 4: solving the Phase II-b of the class proyect.

EE7790 Special Topics
VISUAL SIGNAL PROCESSING AND COMMUNICATIONS

SP11

Prof.: Dr. Luis M. Vicente

Project

Phase II: Baseline Image Encoding Decoding System using DCT

04/06/2011

Your name here
Student number: xxxxxxx
E-mail: your@email.here

Your name here# xxxxxxx 4/8/2011
your@emailhere.com

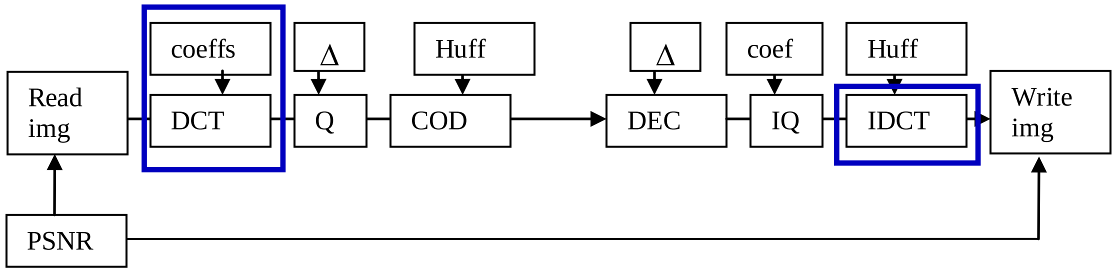
Objective:
Implement a baseline image encoding and decoding system with DCT.

Steps to implement:

1. Write functions to read and write image data (done in class3).
2. Write a function to measure PSNR between two images. The images are stored in files or in the memory (done in class3).
3. **Write functions for DCT and IDCT at block and image levels.**
4. Write functions for Quantization and inverse quantization at block and image levels.
5. Training:
   a. DC: 10-bit binary representation. (No prediction!)
   b. AC: (run, size) + magnitude representation. Run: [0 15], size: [0 10].
   c. Collect statistics on (run, size), and design a Huffman code table
6. Encoding: look up the Huffman table; count the number of bits of encoding.
7. Plot the rate-distortion curve by varying the quantization step size.

Methodology

The complete system diagram implemented in this project is the following:



I will work the system diagram by parts:
1. Read and write blocks.
2. PSNR block
3. **DCT/IDCT blocks**
4. Quantization/Inverse Quantization blocks
5. Image Coding/Image Decoding blocks

1. DCT/IDCT blocks

The DCT and IDCT blocks are implemented in three parts each one.

First, is implemented a function *fdct.m* that implements the Discrete Cosine Transform for an eight element vector. The DCT/IDCT implemented satisfy the following equations [1][2]:

$$DCT(k) = \sqrt{\frac{2}{N}}C(k)\sum_{n=0}^{N-1} x(n)\cos\frac{(2n+1)k\pi}{2N} \qquad k = 0,....,N-1$$

$$x(n) = \sqrt{\frac{2}{N}}\sum_{k=0}^{N-1} DCT(k)C(k)\cos\frac{(2n+1)k\pi}{2N} \qquad n = 0,....,N-1$$

Where

$$C(k) = \begin{cases} 2^{\frac{-1}{2}} & for\ k = 0 \\ 1 & else \end{cases}$$

In order to save processing time the summation at both equations were done using vector multiplication and a look up table where the DCT basis functions were stored (using *mcreatedctcoeff.m* and *mcreatedctcoeff.m* Matlab script files).

Secondly, is implemented a function *fblockdct.m* that performs the DCT to an 8x8 matrix by calling *fdct.m* for each row, and then applying the same function to the result for each column.

Third, is implemented a function *fimagdct.m* that performs the DCT of a 512x512 matrix by calling *fblockdct.m* for each 8x8 sublocks.

The dual functions *fdict.m*, *fblockidct.m*, and *fimagidct.m* were implemented to do the inverse operation at their respective level.

Each function was tested independently. First, the functions *fdct/fidct* were tested with the *mtestdct_idct.m* Matlab script implemented for that purpose, and compared the results with the Matlab *dct* function. The results were identical.

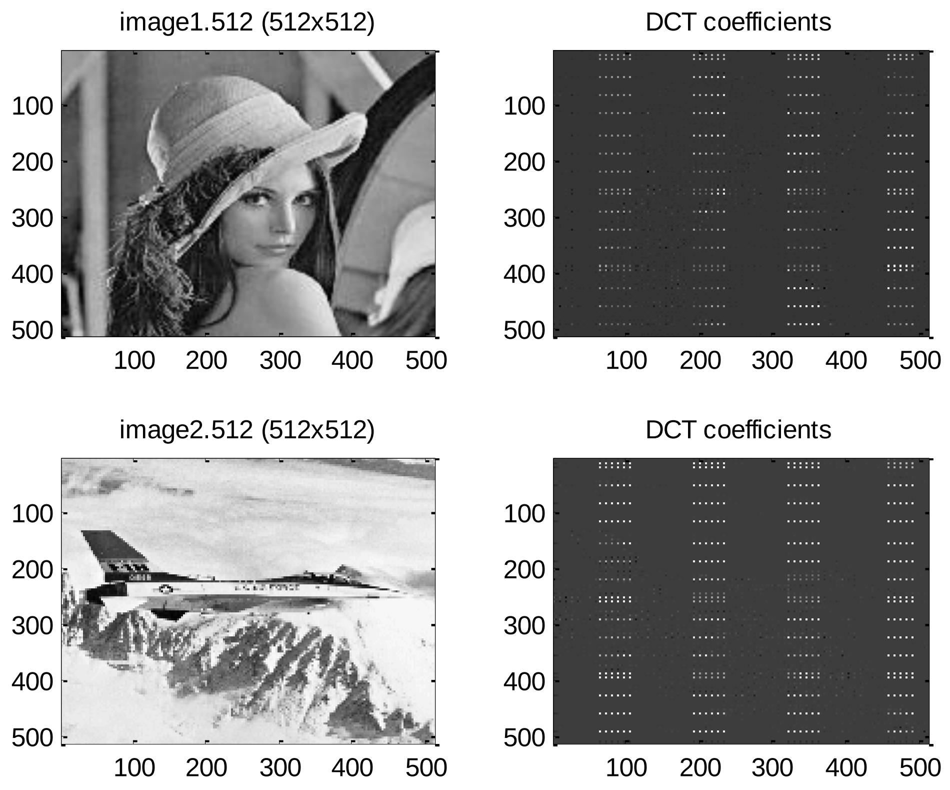
Secondly, the functions *fblockdct/ fblockidct* were tested with the *mtestblockdct_idct.m* Matlab script implemented for that purpose, and verify that the block after DCT and IDCT processes was perfectly reconstructed using the *fpsnr.m* function.

Third, the functions *fimagdct / fimagidct* were tested with the *mprojII.m* Matlab script implemented for that purpose, and verify that the image after DCT and IDCT processes was perfectly reconstructed using the *fpsnr.m* function.

Up to this moment, all processing was lossless; therefore, I verified that the input and output images were exactly the same to verify the implementation was the appropriate over all the process.

<u>This part answer Step to implement #3</u>

FIGURES



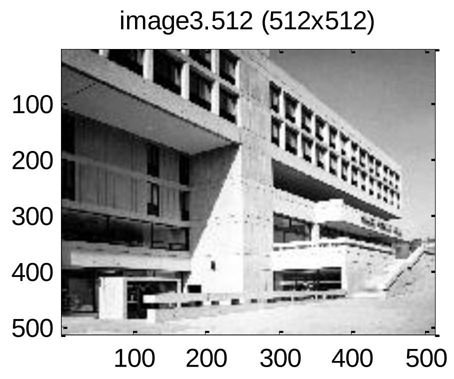
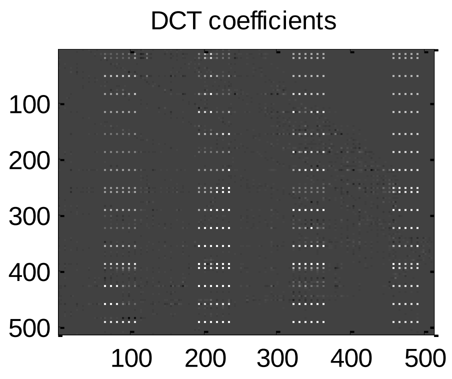


image3.512 (512x512)
100
200
300
400
500


DCT coefficients
100
200
300
400
500

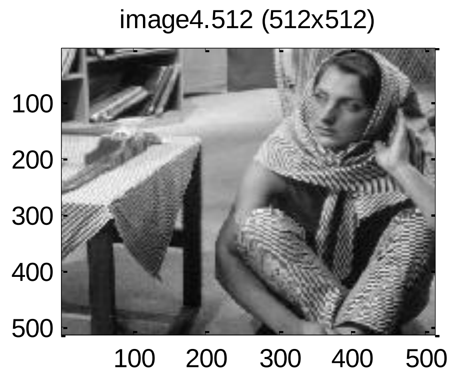
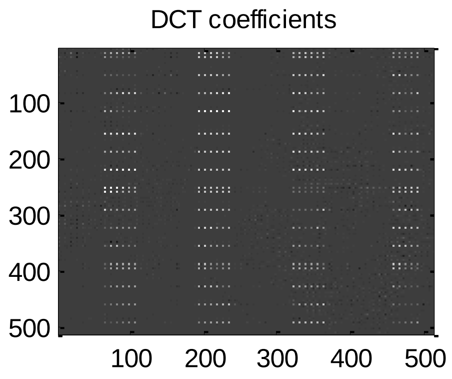


image4.512 (512x512)
100
200
300
400
500


DCT coefficients
100
200
300
400
500

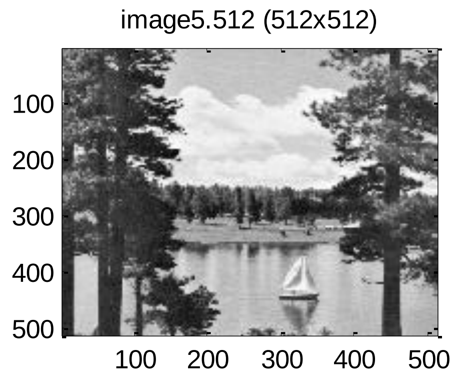
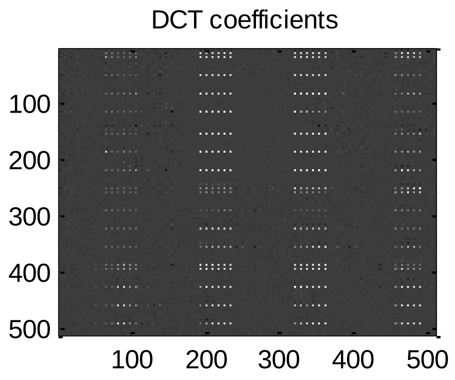


image5.512 (512x512)
100
200
300
400
500


DCT coefficients
100
200
300
400
500

APPENDIX 1 (Source Code)

The main script is called *mprojII.m*. This script read an image, does the DCT. Then do the inverse process to reconstruct the image. The code writes out the PSNR as a measure of the quality of reconstruction. It also plots the original and reconstructed image, as well as the DCT values.

It calls the functions *freadimg fimagdct fimagidct fpsnr fwriteimg*. These functions also call several functions which operate down to the block and vector level.

Before run the script (by typing at the Matlab working directory *mprojII*), open the script with the Matlab editor and select the image to read.

The source code for the main script is the following:

```
%mprojII
%Luis M Vicente 945 995 started 4/2/2005
%script to read files for phase II
%calls custom functions:  freadimg fimagdct fimagidct fpsnr fwriteimg
%calls data: image5.512

clear all, close all, clc
disp(['/////////////////////Project II Main Program/////////////////////'])
%Name of the image to analyze and the reconstructed image to write to
strim = 'image1.512';
strom = ['rec_',strim];
imsize = 512;
subi = imsize;
isigd = freadimg(strim,imsize);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%create a sub image to lesser the processing time COMMENT OUT IF NOT USED
%This one is good for image5.512 to see the boat
% subi = 160;
% xi=170;
% yi=190;
% isigd = isigd(xi:xi+subi-1,yi:yi+subi-1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%create a sub image to lesser the processing time COMMENT OUT IF NOT USED
%This one is good for image1.512 to see the face
% subi = 80;
% xi=260;
% yi=260;
% isigd = isigd(xi:xi+subi-1,yi:yi+subi-1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Do the DCT of the whole image
tic;
DCTsigd = fimagdct(isigd);
disp(['Progress message: DCT done']);
toc;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%Here it will go the code for next classes: Quantizer, encoder and decoder Unquantizer
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Do the IDCT of the whole image
rsigd = fimagidct(DCTsigd);
disp(['Progress message: IDCT done']);
toc;

%compare the images
PSNR = fpsnr(isigd, rsigd);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Write the image in a new file for later comparisons
fwriteimg(strom,rsigd);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('If you want to see the images, wait one second');
pause(1)
%PLOT SECTION
%Plot input image
figure
subplot(2,2,1)
imagesc(isigd);
colormap('gray');zoom on;
title([strim, ' (', num2str(subi),'x',num2str(subi),')']);

%Plot DCT image
subplot(2,2,2)
imagesc(DCTsigd);
colormap('gray');zoom on;
title(['DCT coefficients'])

%Plot reconstructed image
subplot(2,2,4)
imagesc(rsigd);
colormap('gray');zoom on;
title(['Reconstr. ',strim, ' (', num2str(subi),'x',...
    num2str(subi),')'])
xlabel(['PSNR: ',num2str(PSNR)])

disp(['////////////////////Project II End of Main Program////////////////////'])
```

Next, the functions used in this script:

```
%freadimg
%Function to read an image

function [isigd]=freadimg(strim,imsize)
fid = fopen(strim,'rb');
isigc = fread(fid,[imsize,imsize],'uchar');
isigd = double(isigc)';
fclose(fid);
```

```
%fimagdct
%Function to create the dct of a 512x512 image
%calls custom functions: fblockdct

function [oimage]=fimagdct(iimage)

[rowb,colb] = size(iimage);

%load the dct coefficients stored in mat file
load dctcoeff

%I have to divide the image in 8x8 blocks
for rowblock=1:8:rowb
    for colblock=1:8:colb
        %disp(['row: ', num2str(rowblock),' col: ', num2str(colblock)]);
        %select a block
        iblock=iimage(rowblock:rowblock+7,colblock:colblock+7);
        %performthe dct
        oblock=fblockdct(iblock,lut);
        %store in the output DCT image
        oimage(rowblock:rowblock+7,colblock:colblock+7)=oblock;
    end
end
```

```
%fblockdct
%Function to create the dct of a 8*8 block
%calls custom functions: fdct

function [oblock]=fblockdct(iblock,lut)

[rowb,colb] = size(iblock);

%For each row make the dct
for ir = 1:rowb
    xsignal=iblock(ir,:);
    Stemp(ir,:)=fdct(xsignal,lut);
end

%For each column make the dct
for ic = 1:colb
    %transpose to create a row vector
    xsignal=Stemp(:,ic)';
    %transpose the output of fdct to store in a column
    oblock(:,ic)=fdct(xsignal,lut)';
end
```

```
%fdct
%Function to calculate the DCT of a 8x8 vector %calls data: dctcoeff.mat

function [S]=fdct(xsignal,lut)

% %load the dct coefficients stored in mat file
slut = size(lut);
n = slut(1,1);

%remember in Matlab indexes start at 1
C = ones(1,n);
C(1) = 2^(-1/2);

%Find the DCT of an input signal
for u=0:n-1
    uin=u+1;
    %matrix multiplication of the signal with the DCT coefficients
    mmsdct = xsignal*lut(uin,:)';
    %DCT coefficient
    S(uin)=sqrt(2/n)*C(uin)*mmsdct;
end
```

```
%fimagidct
%Function to create the Idct of a 512x512 image
%calls custom functions: fblockidct

function [oimage]=fimagidct(iimage)

[rowb,colb] = size(iimage);

%load the idct coefficients stored in mat file
load idct8

%I have to divide the image in 8x8 blocks
for rowblock=1:8:rowb
    for colblock=1:8:colb
        %disp(['row: ', num2str(rowblock),' col: ', num2str(colblock)]);
        %select a block
        iblock=iimage(rowblock:rowblock+7,colblock:colblock+7);
        %performthe dct
        oblock=fblockidct(iblock,liut);
        %store in the output DCT image
        oimage(rowblock:rowblock+7,colblock:colblock+7)=oblock;
    end
end
```

```
%fblockidct
%Function to create the dct of a 8*8 block
%calls custom functions: fidct

function [oblock]=fblockidct(iblock,liut)

[rowb,colb] = size(iblock);

%For each row make the dct
for ir = 1:rowb
    Ssignal=iblock(ir,:);
    xtemp(ir,:)=fidct(Ssignal,liut);
end

%For each column make the dct
for ic = 1:colb
    %transpose to create a row vector
    Ssignal=xtemp(:,ic)';
    %transpose the output of fdct to store in a column
    oblock(:,ic)=fidct(Ssignal,liut)';
end
```

```
%fidct
%Function to calculate the IDCT of a 8x8 vector
%calls data: dctcoeff.mat
function [xsignal]=fidct(S,liut)

%load the idct coefficients stored in mat file
% load idct8
sliut = size(liut);
n = sliut(1,1);

%Find the IDCT of an input signal
for u=0:n-1
    uin=u+1;

    %IDCT coefficient
    xsignal(uin) = S*liut(uin,:)';

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%function PSNR process

function [PSNR]=fpsnr(image1, image2)

%substract one image from another
%dij = abs(image1 - image2);
dij = (image1 - image2);
%get the mean
% mdij = mean(mean(dij))
mdij = mean(dij(:));
%mdij2 = sum(sum(dij))/prod(size(dij))

%square without the mean
teee =(dij-mdij).^2;

%find the mean square error
MSE = abs(sum(sum(teee))/prod(size(dij)));


%when MSE is small.
if(MSE < 6.5025e-006)
    MSE = 6.5025e-006;
    disp(['Achieved higher Limit of PSNR. The images are the same'])
end
%find the inverse multiplied by the peak value of pixel
IMSE = 255^2/MSE;

%find the PSNR
PSNR = 10*log10(IMSE);
% disp(['MSE: ',num2str(MSE)])
disp(['PSNR: ',num2str(PSNR)])
```

```
%fwriteimg
%Function to write an image

function []=fwriteimg(strim,rsigd)
rsigd = round(rsigd');
rsigc = char(rsigd);
fid = fopen(strim,'wb');
isigc = fwrite(fid,rsigc,'uchar');
fclose(fid);
```

Screenshot

```
////////////////////Project II Main Program////////////////////
Progress message: DCT done
Elapsed time is 16.504000 seconds.
Progress message: IDCT done
Elapsed time is 508.201000 seconds.
PSNR: 35.805
If you want to see the images, wait one second
////////////////////Project II End of Main Program////////////////////
```

REFERENCES

[1] Dr. Zhihai (Henry) He, *"Visual Signal Processing and Communitation Class Notes"*. JPEG Image Compression Standard. Missouri University at Columbia. 2005.

[2] Yao Wang, Jörn Ostermann, Ya-Qin Zhang., *"Video Processing and Communications"*, Prentice Hall, Inc 2002.

[3] Matlab, *"Special Topics, Signal Processing ToolBox"*. The MathWorks Inc. 2004.