Class 3: solving the Phase II of the class proyect.

EE7790 Proyect in MSEE
VISUAL SIGNAL PROCESSING
AND COMMUNICATIONS

SP11

Prof.: Dr. Luis M. Vicente

Project

Phase II: Baseline Image Encoding Decoding System using DCT

03/24/2011

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

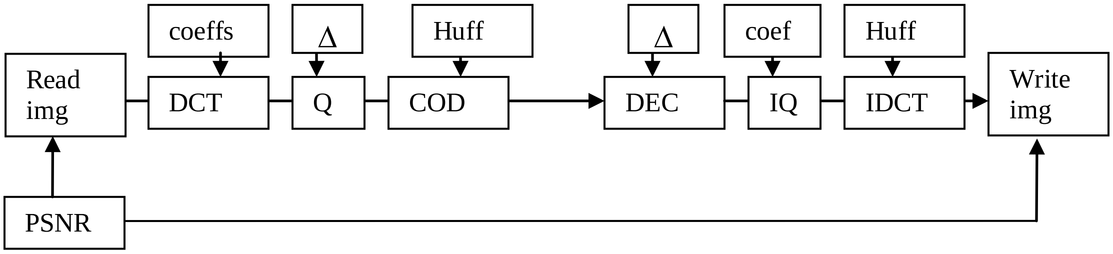
Objective:
Implement a baseline image encoding and decoding system with DCT.

Steps to implement:

1. Write functions to read and write image data. Images are posted on http://www.lmvicente.com/ee7790/images.zip
2. Write a function to measure PSNR between two images. The images are stored in files or in the memory.
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:



We should 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. Read and write blocks.

The Read block is implemented in Matlab in *freadimg.m* function. I will be using *fopen* to open the input file in binary format and then use *fread* storing the data in a custom size square matrix as unsigned character format. Then convert the data in double format to operate with it.

The Write block is implemented using the inverse procedure implemented in *fwriteimg.m*. It uses *fopen* to open the output file in binary format and then use *fwrite* to store the data.

The Matlab code is next

```
%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);
```

We call this function with the following input parameters. Notice the strim must have the filename of the image we want to read

```
imsize = 512;
strim = 'image1.512';
isigd = freadimg(strim,imsize);
```

Yourself should analyze what is inside isigd and plot it.

To write the image we implement:

```
%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);
```

The code to call this function is

```
strom = ['rec_',strim];
rsigd = isigd; %Here we do nothing with the image yet
fwriteimg(strom,rsigd);
```

The student (you) must implement this code, understand it, make appropriate comments and test it with the images provided in the instructions of this project.

<u>This part answers #1</u>

2. PSNR block

The PSNR block is implemented in Matlab in *fpsnr.m* function. The PSNR block will read two matrices representing two images and compare them using the PSNR algorithm [1], that is:

$$d(i,j) = (image\_1(i,j) - image\_2(i,j));$$
$$md(i,j) = mean\,(d(i,j));$$
$$MSE = \frac{1}{512x512}\sum_i\sum_j\left(d(i,j) - md(i,j)\right)^2$$

Finally:

$$PSNR = 10\log_{10}\frac{255x255}{MSE}$$

When the *MSE* is very small, the *PSNR* is limited to a maximum value of 100db. In that case both images are considered without any difference between them.

The matlab code is next

The function is called as:

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

The function implementation is:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%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)])
```

This part answers #2

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.