**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

**Computer Engineering Department**

LAB Manual

PART A

(PART A : TO BE REFFERED BY STUDENTS)

**Experiment No.10**

**A.1 Aim:**

Write a program to compress the image (your own photograph) using energy compaction concept of discrete cosine transform and calculate RMSE, PSNR and compression ratio.

**A.2 Prerequisite:**

1 MATLAB programming syntax (Refer the MATLAB manual).

2. Knowledge of fundamentals of Image Compression & Properties of DCT.

2. Soft copy of input image.

**A.3 Outcome:**

**After successful completion of this experiment students will be able to**

1. Compress the given input image.
2. Calculate RMSE, PSNR and Compression ratio.
3. Identify other image compression techniques and its applications

**A.4 Theory:**

**A.4.1. Data Compression**

**1. Image Compression**

Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages.

There are several different ways in which image files can be compressed. For Internet use, the two most common compressed graphic image formats are the [JPEG](http://searchsoa.techtarget.com/definition/JPEG) format and the [GIF](http://searchwindevelopment.techtarget.com/definition/GIF) format. The JPEG method is more often used for photographs, while the GIF method is commonly used for line art and other images in which geometric shapes are relatively simple. Mathematical Transformation functions like DCT can be used for image compression.

**1. DCT**

The discrete cosine transform (DCT) helps separate the image into parts (or spectral sub-bands) of differing importance (with respect to the image's visual quality). The DCT is similar to the discrete Fourier transform: it transforms a signal or image from the spatial domain to the frequency domain.

**DCT Encoding:**

The general equation for a 1D (*N* data items) DCT is defined by the following equation:

 (1.1)

and the corresponding ***inverse*** 1D DCT transform is simple *F-1*(*u*), where

 (1.2)

The general equation for a 2D (*N* by *M* image) DCT is defined by the following equation:

 (1.3)

and the corresponding ***inverse*** 2D DCT transform is simple *F-1*(*u*,*v*), where

 (1.4)

The basic operation of the DCT is as follows:

* The input image is N by M;
* f(i,j) is the intensity of the pixel in row i and column j;
* F(u,v) is the DCT coefficient in row k1 and column k2 of the DCT matrix.
* For most images, much of the signal energy lies at low frequencies; these appear in the

upper left corner of the DCT.

Compression is achieved since the lower right values represent higher frequencies and are often small - small enough to be neglected with little visible distortion. This achieves the energy compaction.

**3. Performance Evaluation parameters:**

* 1. **Root mean squared error (RMSE):**

In order to evaluate the performance of the image compression coding, it is necessary to define a measurement that can estimate the difference between the original image and the decoded image. Two commonly used measurements are the **Mean Square Error (MSE)** and the **Peak Signal to Noise Ratio (PSNR)**, which are defined in (1.5) and (1.6), respectively. f(x,y) is the pixel value of the original image, and f’(x,y)is the pixel value of the decoded image. Most image compression systems are designed to minimize the MSE and maximize the PSNR.

 **(1.5)**

* 1. **Peak signal to noise ratio (PSNR):**

 **(1.6)**

* 1. **Compression Ratio (CR):**

CR = Compressed image size/ uncompressed image size **(1.7)**

The ratio should be between 0 and 1 (multiply with 100 to get ratio in percentages)

If CR>1 that means compressed image size is greater than actual uncompressed image size. If CR is just below 1 means bad compression.

**A.5 Procedure/Algorithm:**

**A.5.1:**

**TASK 1:**

1. Read 3 different input images as per following categories.
2. Your photograph
3. Flower image
4. Animal image

2. Apply DCT on the image as per following directions separately for each image.

a. Full DCT

b. 4x4 Block DCT

c. 8x8 Block DCT

3. Eliminate 50% spatial frequencies (take in a matrix of DCT coefficients and set everything beyond the upper left triangle to zero) for each method to compress the image.

4. Reconstruct the compressed image and fill up the following table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | Subjective quality \* | RMSE | PSNR | CR |
| Full DCT |  |  |  |  |
| 4x4 Block DCT |  |  |  |  |
| 8x8 Block DCT |  |  |  |  |

\* Comments should be (no difference/less difference/ maximum difference)

5. Observe/compare all outputs and complete PART B of lab manual.

6. Save and close the file and name it as **EX10\_Task1\_your Roll no.m**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No.: N049 | Name: Tarun Tanmay |
| Class: MBATech CE | Batch: B3 |
| Date of Experiment: 10/09/2020 | Date of Submission: 13/09/2020 |
| Grade: | Time of Submission: |
| Date of Grading: |  |

**B.1 Software Code written by student:**

***(Paste your Matlab code completed during the 2 hours of practical in the lab here)***

clc;

clear all;

img1=imread('/Users/tjrox0825/Desktop/TJ.jpg');

img1=imresize(img1,[256,256],'nearest');

img1=rgb2gray(img1);

% subplot(1,3,1);

% imwrite(img1,'I1.jpg')

% imshow(img1);

% title('Input image 1')

img2=imread('/Users/tjrox0825/Desktop/flower.jpg');

img2=imresize(img2,[256,256],'nearest');

img2=rgb2gray(img2);

% subplot(1,3,2);

% imwrite(img2,'I2.jpg')

% imshow(img2);

% title('Input image 2')

img3=imread(‘/Users/tjrox0825/Desktop/animal.jpg’);

img3=imresize(img3,[256,256],'nearest');

img3=rgb2gray(img3);

% subplot(1,3,3);

% imwrite(img3,'I3.jpg')

% imshow(img3);

% title('Input image 3')

ig1=double(img1);

% ig2=double(img2);

% ig3=double(img3);

figure('Name','FULL DCT');

%DCT

subplot(2,2,1);

imshow(img1);

title('Input image');

h=dctmtx(256);

subplot(2,2,2);

imshow(h);

title('DCT matrix 256\*256');

it=(h\*ig1\*h');

subplot(2,2,3);

imshow(uint8(it));

title('DCT transform');

for i=1:256

for j=1:256

if(i+j>150)

it(i,j)=0;

end

end

end

in=(h'\*it\*h);

subplot(2,2,4);

imwrite(uint8(in),'o13.jpg');

imshow(uint8(in));

title('DCT inverse');

figure('Name','4x4 Block DCT');

% 4\*4 Block DCT

subplot(2,2,1);

imshow(img1);

title('Input image');

h=dctmtx(4);

subplot(2,2,2);

imshow(h);

title('DCT matrix 4\*4');

t=zeros(4,4);

dd=zeros(4,4);

temp=zeros(256,256);

for i=1:4:256

for j=1:4:256

for k=1:4

for l=1:4

t(k,l)=ig1((i+k-1),(j+l-1));

end

end

dd=(h\*t\*h');

for m=1:4

for n=1:4

if(m+n>3)

dd(m,n)=0;

end

end

end

in=(h'\*dd\*h);

for o=1:4

for p=1:4

temp((i+o-1),(j+p-1))=in(o,p);

end

end

end

end

temp=uint8(temp);

imwrite(temp,'o23.jpg');

subplot(2,2,3);

imshow(temp);

title('DCT inverse');

figure('Name','8x8 Block DCT');

% 8\*8 Block DCT

subplot(2,2,1);

imshow(img1);

title('Input image');

h=dctmtx(8);

subplot(2,2,2);

imshow(h);

title('DCT matrix 8\*8');

t=zeros(8,8);

dd=zeros(8,8);

temp=zeros(256,256);

for i=1:8:256

for j=1:8:256

for k=1:8

for l=1:8

t(k,l)=ig1((i+k-1),(j+l-1));

end

end

dd=(h\*t\*h');

for m=1:8

for n=1:8

if(m+n>5)

dd(m,n)=0;

end

end

end

in=(h'\*dd\*h);

for o=1:8

for p=1:8

temp((i+o-1),(j+p-1))=in(o,p);

end

end

end

end

temp=uint8(temp);

imwrite(temp,'o33.jpg');

subplot(2,2,3);

imshow(temp);

title('DCT inverse');

**B.2 Input and Output:**

**Input Images:**

**A person wearing glasses

Description automatically generated**

Figure 1 Photograph

**A close up of a flower

Description automatically generated**

Figure 2 Flower

**An owl looking at the camera

Description automatically generated**

Figure 3 Animal

**Output Images:**

1. **For each edge detection operatoras per the procedure discussed in section A.5.**

**1. Your Photograph**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

1. **Flower**

A screenshot of a cell phone

Description automatically generated

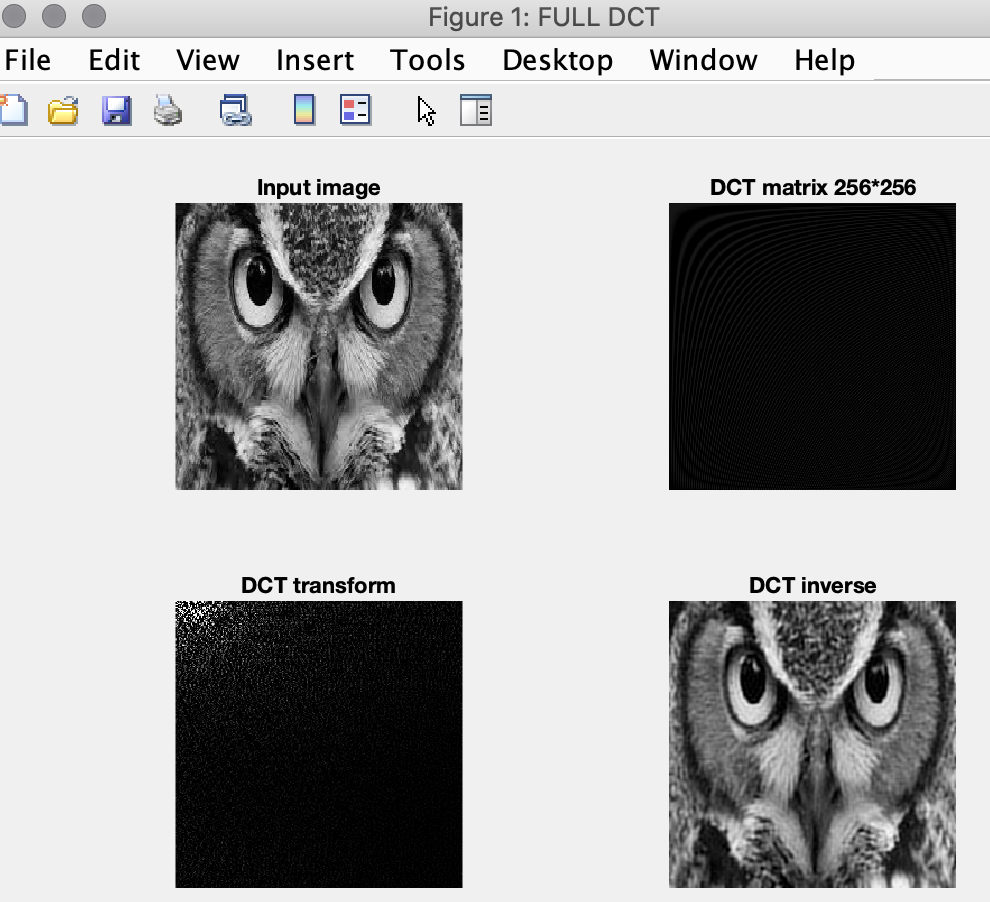
**A screenshot of a cell phone

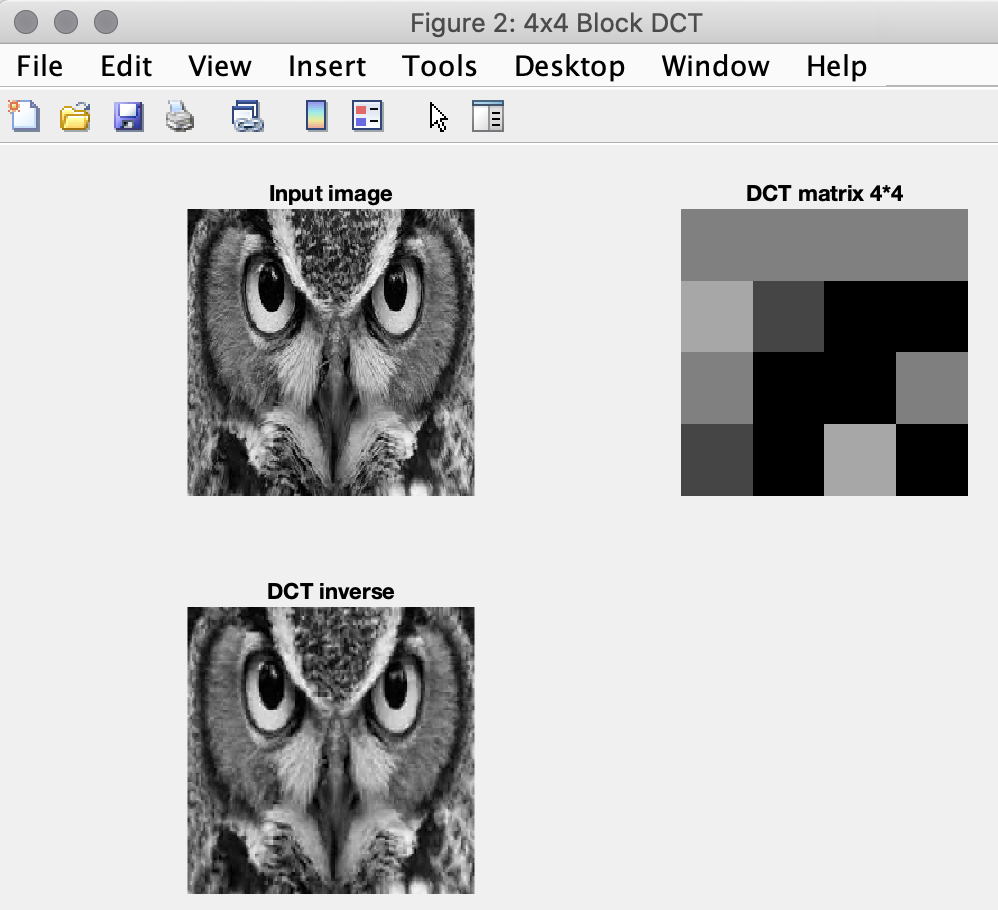
Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

1. **Animal**

****

****

**A screenshot of a social media post

Description automatically generated**

**B.3 Observations and learning:**

* Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level.
* We try to compress the images using DCT transform.
* The process transforms the image from the spatial domain to the frequency domain. Hence, we can analyze the energy values and manipulate them to achieve compression
* For most images, much of the signal energy lies at low frequencies; these appear in the upper left corner of the DCT. Compression is achieved since the lower right values represent higher frequencies and are often small - small enough to be neglected with little visible distortion. This achieves the energy compaction.
* Since the lower right energy values can be neglected, we convert these values to zero, thus when we regenerate the image, it is compressed.
* The magnitude of compression depends on the extent to which we make energy values zero.
* The tradeoff here is the quality, as we compress the image the quality also degrades. Hence it is crucial to determine till what point can the energy values be neglected.
* We also tried performing compression by dividing the image into 4\*4 and 8\*8 blocks. Each block was individually transformed. The lower energy values of each block were neglected and then the image was regenerated. The overall image was reconstructed by combining all the blocks. This led to more compression of the image with respect to the previous method.

**B.4 Conclusion:**

We have successfully implemented a program to compress the image (your own photograph) using energy compaction concept of discrete cosine transform and calculate RMSE, PSNR and compression ratio.

**B.5 Question of Curiosity**

Do you find the steps applied for image compression in this experiment needs further improvement? Justify your answer.

Yes, the distortion in the image when we compress after a particular energy value is high, thus, to further compress the image we need an algorithm that allows us to reduce the image further without causing any distortion.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*