**SVKM’s NMIMS**

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

**Computer Engineering Department**

Program: B.Tech. Sem VI

**Course: Image Processing**

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 common 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

|  |  |
| --- | --- |
| Roll No : B032 | Name: Naman Garg |
| Class : B.Tech CS-B | Batch : B2 |
| Date of Experiment: 23-9-2020 | Date of Submission |
| Grade : | Time of Submission: |
| Date of Grading: |  |

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

img=imread('anml.jpg');

img=rgb2gray(img);

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

imwrite(img , 'originalimage.jpg')

dct=dctmtx(256);

dct\_t=dct.';

F=dct\*double(img)\*dct\_t;

dct\_4=dctmtx(4);

dct\_4t=dct\_4.';

F4=zeros(256);

for i=1:4:256

for j=1:4:256

F4(i:i+3,j:j+3)=dct\_4\*double(img(i:i+3,j:j+3))\*dct\_4t;

end

end

dct\_8=dctmtx(8);

dct\_8t=dct\_8.';

F8=zeros(256);

for i=1:8:256

for j=1:8:256

F8(i:i+7,j:j+7)=dct\_8\*double(img(i:i+7,j:j+7))\*dct\_8t;

end

end

F4\_30=compression\_30\_4x4(F4);

F4\_50=compression\_50\_4x4(F4);

F4\_70=compression\_70\_4x4(F4);

f4\_30=decompressed\_image(dct\_4,dct\_4t,F4\_30);

f4\_50=decompressed\_image(dct\_4,dct\_4t,F4\_50);

f4\_70=decompressed\_image(dct\_4,dct\_4t,F4\_70);

F8\_30=compression\_30\_8x8(F8);

F8\_50=compression\_50\_8x8(F8);

F8\_70=compression\_70\_8x8(F8);

f8\_30=decompressed\_image\_8x8(dct\_8,dct\_8t,F8\_30);

f8\_50=decompressed\_image\_8x8(dct\_8,dct\_8t,F8\_50);

f8\_70=decompressed\_image\_8x8(dct\_8,dct\_8t,F8\_70);

f1=compression\_50(dct,dct\_t,F);

f2=compression\_30(dct,dct\_t,F);

f3=compression\_70(dct,dct\_t,F);

subplot(3,3,1),imshow(uint8(f2)),title('70%');

subplot(3,3,2),imshow(uint8(f1)),title('50%');

subplot(3,3,3),imshow(uint8(f3)),title('30%');

subplot(3,3,4),imshow(uint8(f4\_30)),title('4x4');

subplot(3,3,5),imshow(uint8(f4\_50)),title('4x4');

subplot(3,3,6),imshow(uint8(f4\_70)),title('4x4');

subplot(3,3,7),imshow(uint8(f8\_30)),title('8x8');

subplot(3,3,8),imshow(uint8(f8\_50)),title('8x8');

subplot(3,3,9),imshow(uint8(f8\_70)),title('8x8');

imwrite(uint8(f1),'img\_50.jpg');

imwrite(uint8(f2),'img\_30.jpg');

imwrite(uint8(f3),'img\_70.jpg');

imwrite(uint8(f4\_30),'img\_30\_4.jpg');

imwrite(uint8(f4\_50),'img\_50\_4.jpg');

imwrite(uint8(f4\_70),'img\_70\_4.jpg');

imwrite(uint8(f8\_30),'img\_30\_8.jpg');

imwrite(uint8(f8\_50),'img\_50\_8.jpg');

imwrite(uint8(f8\_70),'img\_70\_8.jpg');

fprintf('RMSE Values:\n');

fprintf('30 percent compression %f\n',rmse\_value(img,f2));

fprintf('50 percent compression %f\n',rmse\_value(img,f1));

fprintf('70 percent compression %f\n',rmse\_value(img,f3));

fprintf('\n4x4 block\n');

fprintf('30 percent compression %f\n',rmse\_value(img,f4\_30));

fprintf('50 percent compression %f\n',rmse\_value(img,f4\_50));

fprintf('70 percent compression %f\n',rmse\_value(img,f4\_70));

fprintf('\n8x8 block\n');

fprintf('30 percent compression %f\n',rmse\_value(img,f8\_30));

fprintf('50 percent compression %f\n',rmse\_value(img,f8\_50));

fprintf('70 percent compression %f\n',rmse\_value(img,f8\_70));

fprintf('\nPSNR Values:\n');

fprintf('30 percent compression %f\n',20\*log10(255/rmse\_value(img,f2)));

fprintf('50 percent compression %f\n',20\*log10(255/rmse\_value(img,f1)));

fprintf('70 percent compression %f\n',20\*log10(255/rmse\_value(img,f3)));

fprintf('\n4x4 block\n');

fprintf('30 percent compression %f\n',20\*log10(255/rmse\_value(img,f4\_30)));

fprintf('50 percent compression %f\n',20\*log10(255/rmse\_value(img,f4\_50)));

fprintf('70 percent compression %f\n',20\*log10(255/rmse\_value(img,f4\_70)));

fprintf('\n8x8 block\n');

fprintf('30 percent compression %f\n',20\*log10(255/rmse\_value(img,f8\_30)));

fprintf('50 percent compression %f\n',20\*log10(255/rmse\_value(img,f8\_50)));

fprintf('70 percent compression %f\n',20\*log10(255/rmse\_value(img,f8\_70)));

fprintf('\nCR Values:\n');

fprintf('30 percent compression %f\n',imfinfo('originalimage.jpg').FileSize/imfinfo('img\_30.jpg').FileSize);

fprintf('50 percent compression %f\n',imfinfo('originalimage.jpg').FileSize/imfinfo('img\_50.jpg').FileSize);

fprintf('70 percent compression %f\n',imfinfo('originalimage.jpg').FileSize/imfinfo('img\_70.jpg').FileSize);

fprintf('\n4x4 block\n');

fprintf('30 percent compression %f\n',imfinfo('originalimage.jpg').FileSize/imfinfo('img\_30\_4.jpg').FileSize);

fprintf('50 percent compression %f\n',imfinfo('originalimage.jpg').FileSize/imfinfo('img\_50\_4.jpg').FileSize);

fprintf('70 percent compression %f\n',imfinfo('originalimage.jpg').FileSize/imfinfo('img\_70\_4.jpg').FileSize);

fprintf('\n8x8 block\n');

fprintf('30 percent compression %f\n',imfinfo('originalimage.jpg').FileSize/imfinfo('img\_30\_8.jpg').FileSize);

fprintf('50 percent compression %f\n',imfinfo('originalimage.jpg').FileSize/imfinfo('img\_50\_8.jpg').FileSize);

fprintf('70 percent compression %f\n',imfinfo('originalimage.jpg').FileSize/imfinfo('img\_70\_8.jpg').FileSize);

function f4=decompressed\_image(dct\_4,dct\_4t,F4)

f4=zeros(256);

for i=1:4:256

for j=1:4:256

f4(i:i+3,j:j+3)=dct\_4t\*double(F4(i:i+3,j:j+3))\*dct\_4;

end

end

end

function f4=decompressed\_image\_8x8(dct\_4,dct\_4t,F4)

f4=zeros(256);

for i=1:8:256

for j=1:8:256

f4(i:i+7,j:j+7)=dct\_4t\*double(F4(i:i+7,j:j+7))\*dct\_4;

end

end

end

function F2=compression\_30\_4x4(F)

F2=zeros(256,256);

for i=1:4:256

for j=1:4:256

for k=i:i+1

for l=j:j+1-k+i

F2(k,l)=F(k,l);

end

end

end

end

end

function F1=compression\_50\_4x4(F1)

for i=1:4:256

for j=1:4:256

for k=i:i+3

for l=j+3-k+i:j+3

F1(k,l)=0;

end

end

end

end

end

function F=compression\_70\_4x4(F)

for i=1:4:256

for j=1:4:256

for k=i+2:i+3

for l=j+5-k+i:j+3

F(k,l)=0;

end

end

end

end

end

function F2=compression\_30\_8x8(F)

F2=zeros(256,256);

for i=1:8:256

for j=1:8:256

for k=i:i+3

for l=j:j+3-k+i

F2(k,l)=F(k,l);

end

end

end

end

end

function F1=compression\_50\_8x8(F1)

for i=1:8:256

for j=1:8:256

for k=i:i+7

for l=j+7-k+i:j+7

F1(k,l)=0;

end

end

end

end

end

function F=compression\_70\_8x8(F)

for i=1:8:256

for j=1:8:256

for k=i+5:i+7

for l=j+12-k+i:j+7

F(k,l)=0;

end

end

end

end

end

function f2=compression\_30(dct,dct\_t,F)

F2=zeros(256,256);

for i=1:90

for j=1:91-i

F2(i,j)=F(i,j);

end

end

f2=dct\_t\*F2\*dct;

end

function f1=compression\_50(dct,dct\_t,F1)

for i=1:256

for j=256-i+1:256

F1(i,j)=0;

end

end

f1=dct\_t\*F1\*dct;

end

function f3=compression\_70(dct,dct\_t,F)

F3=F;

for i=180:256

for j=436-i:256

F3(i,j)=0;

end

end

f3=dct\_t\*F3\*dct;

end

function rmse=rmse\_value(img,f)

rmse=0.0;

for i=1:256

for j=1:256

rmse=rmse+ (double(img(i,j))-double(f(i,j)))\*(double(img(i,j))-double(f(i,j)));

end

end

rmse=rmse/(256\*256);

rmse=sqrt(rmse);

end

**B.2 Input and Output:**

***(Paste your program input and output in following format, If there is error then paste the specific error in the output part. In case of error with due permission of the faculty extension can be given to submit the error free code with output in due course of time. Students will be graded accordingly.)***

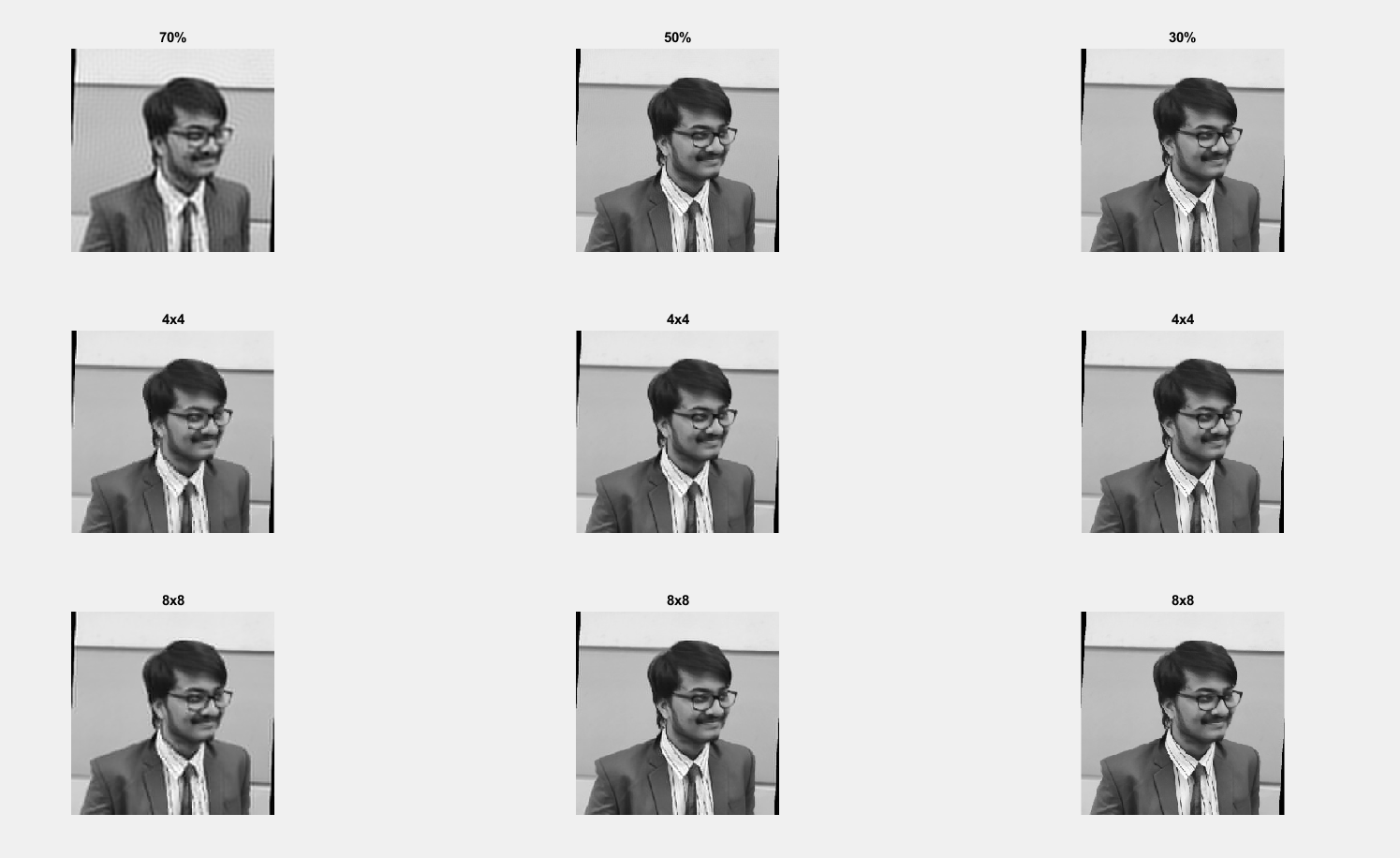
**Input Images:**

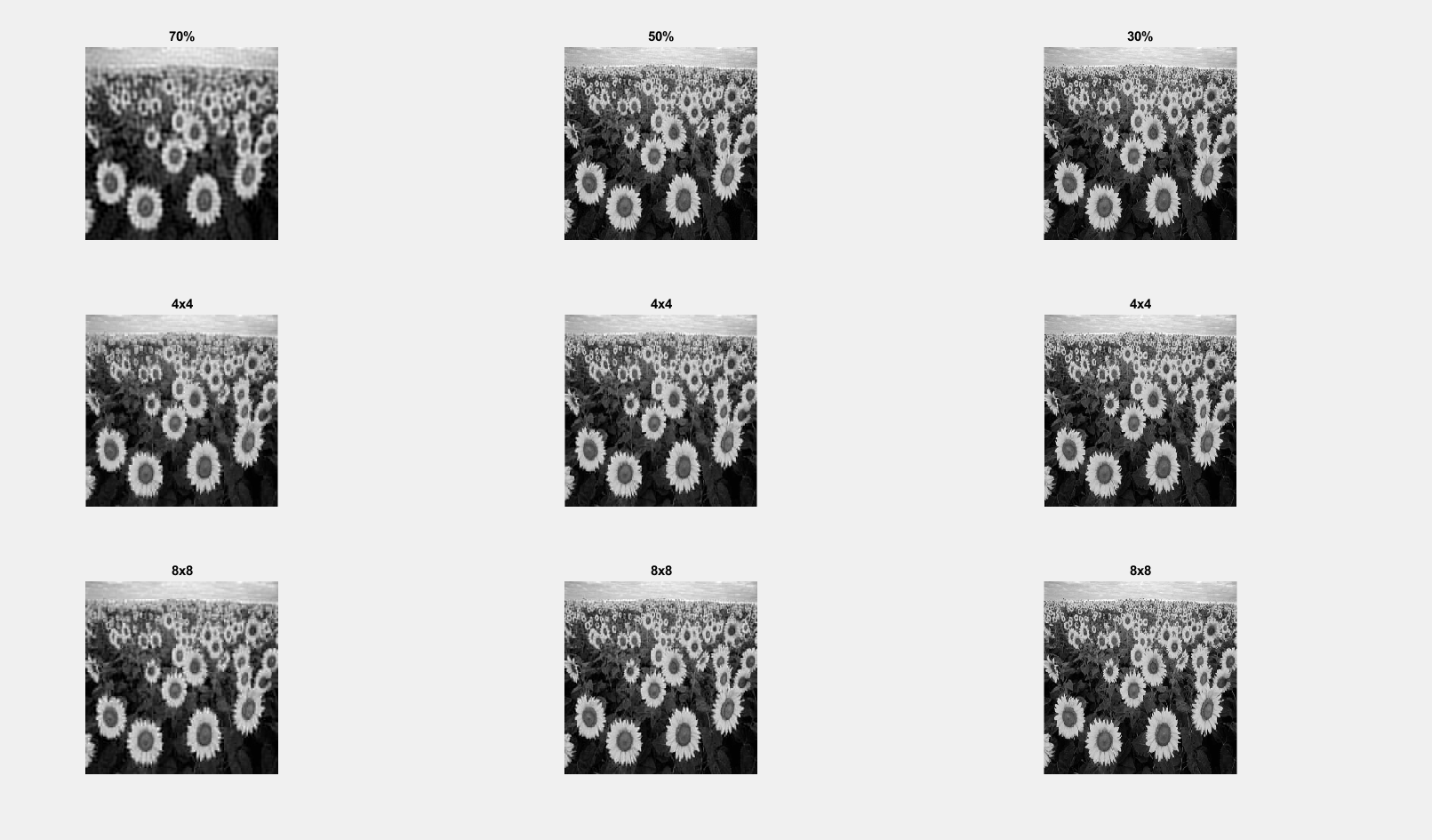
****

**Output Images:**

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

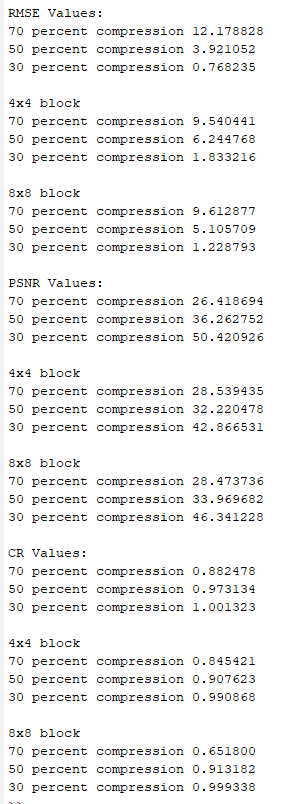
**NAMAN’s Image**

****

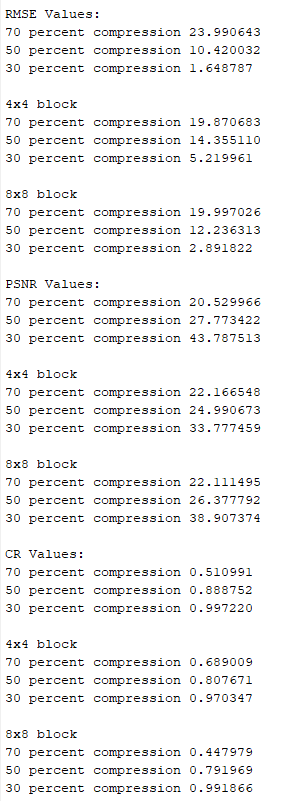
**Flower Image**

**Animal Image**

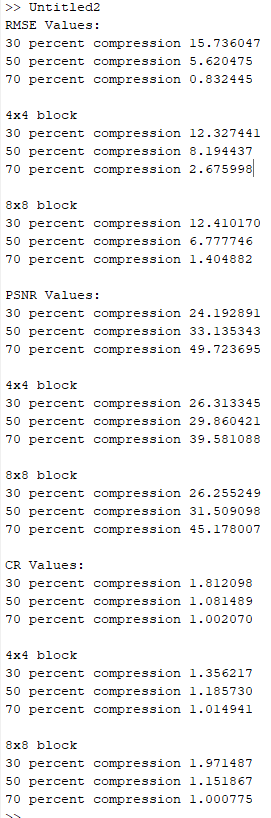
My image::



Flower::



Animal:



**B.3 Observations and learning:**

**My Image:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | Subjective quality \* | RMSE | PSNR | CR |
| **Full DCT** – 30% Retention | blur | 12.178828 | 26.418694 | 0.882478 |
| 50% Retention | Quite blur | 3.921052 | 36.262752 | 0.973134 |
| 70% Retention | Not very blur | 0.768235 | 50.420926 | 01 |
| **4x4 Block DCT –**  30% Retention | blur | 9.540441 | 28.539435 | 0.845421 |
| 50% Retention | Quite blur | 6.244768 | 32.220478 | 0.907623 |
| 70% Retention | Not very blur | 1.833216 | 42.866531 | 0.990868 |
| **8x8 Block DCT-**  30%  Retention | Blur | 9.612877 | 28.4733736 | 0.651800 |
| 50% Retention | Quite blur | 5.105709 | 33.969682 | 0.913182 |
| 70% Retention | Not very blur | 1.228793 | 46.341228 | 0.999338 |

**Flower Image:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | Subjective quality \* | RMSE | PSNR | CR |
| **Full DCT** – 30% Retention | blur | 23.990643 | 20.529966 | 0.868273 |
| 50% Retention | Quite blur | 10.420032 | 27.773422 | 0.959750 |
| 70% Retention | Not very blur | 1.64878 | 43.787513 | 0.998933 |
| **4x4 Block DCT –**  30% Retention | blur | 19.870683 | 22.166548 | 0.857600 |
| 50% Retention | Quite blur | 14.355110 | 24.990673 | 0.910047 |
| 70% Retention | Not very blur | 5.219961 | 33.777459 | 0.991310 |
| **8x8 Block DCT-**  30%  Retention | Blur | 19.997026 | 22.111495 | 0.698125 |
| 50% Retention | Quite blur | 12.236313 | 26.377792 | 0.931849 |
| 70% Retention | Not very blur | 2.891822 | 38.907374 | 0.997561 |

**Animal Image:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method | Subjective quality \* | RMSE | PSNR | CR |
| **Full DCT** – 30% Retention | blur | 22.241996 | 21.187328 | 0.525765 |
| 50% Retention | Quite blur | 10.230533 | 27.932838 | 0.887283 |
| 70% Retention | Not very blur | 2.390306 | 40.561732 | 0.993246 |
| **4x4 Block DCT –**  30% Retention | blur | 18.373095 | 22.847157 | 0.675761 |
| 50% Retention | Quite blur | 13.691138 | 25.402013 | 0.811238 |
| 70% Retention | Not very blur | 5.653888 | 33.083859 | 0.968123 |
| **8x8 Block DCT-**  30%  Retention | Blur | 18.755930 | 22.668032 | 0.455190 |
| 50% Retention | Quite blur | 11.778149 | 26.709262 | 0.796785 |
| 70% Retention | Not very blur | 3.376468 | 37.561551 | 0.988114 |

So overall, when we go for DCT transform, the full DCT with only 30% info retained shows some blurring and can suffer in the minute details part example : white spots on my shirt, edges of the petals of the flower, stripes on the tiger. Full DCT with 50% and 70% information retained are similar with the only difference being the level of compression which also doesn’t have a significant difference. The 4x4 and 8x8 blocks have similar outputs but 8x8 blocks have lesser CR value (better compression) for 30% information retention. And when it comes to best method for compression, we can choose to go for 8x8 block with 30% information retention as it is more detailed than a full DCT with 30% retention and has the highest amount of compression indicated by the least CR value.

**B.4 Conclusion:**

*(****Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.3)***

I deleted information below the diagonal lines and the diagonal line themselves were shifted both towards the top left corner and the bottom right corner.

This created a variation in the amount of compression achieved.

More the information is dropped, less accurate is the compressed image wrt the original image and more is the compression in size.

I dropped information in blocks as well which made the dropping of information uniform across the transformed matrix and not just below the main diagonal in the entire matrix.

This will lead to different values of the metrics. The metrics implemented were rmse, psnr and CR. I understood the applications of the data compression techniques.

**B.5 Question of Curiosity**

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

Depends upon the function that we are using the image for if the images used for something that requires high clarity or high resolution we should compromise on the compression and bare the large file size however since the 30% retained information also result in a recognizable image I feel that we can compress this image even more The information remains the same and as long as the image is recognizable the amount of compression can be tweaked accordingly.

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