Department of Electronic and Telecommunication Engineering

University of Moratuwa

EN3551- Digital Signal Processing

Assignment 2

Ranasingha A.S.N. 200507N

The given images are of mat file type. Therefore, first we need to load the mat files and extract the images from them. Given below is the MATLAB implementation to do that.

```
%% load images
pepper_data = load('peppers512.mat');
camera_data = load('camera256.mat');
boat_data = load('boat512.mat');

%Extract images
pepper_im = pepper_data.peppers512;
camera_im = camera_data.camera256;
boat_im = boat_data.boat512;
```

Some functions were defined to reduce the code length and make coding easier.

The function **convert2gray** takes an image input and checks whether it is in grayscale or in RGB format (color image). If the image is a color image, it is converted to grayscale and if the image is already in grayscale, it outputs the same image.

```
%Convert to grascale
function result = convert2gray(image)
    if ndims(image) == 3
        result = rgb2gray(image);   %color image to grayscale
    else
        result = image;             %originally grayscale image
    end
end
```

The function **genQuantization_mat** generates the quantization matrix corresponding to a given quality level keeping the standard quantization matrix as Q_{50}.

```
%Define the quantization matrix for a given quality level
function  Q_matrix = genQuantization_mat(q_level)
    %Standard quantization matrix = Q_50
    standard_Q_Matrix = [
        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
        ];

    if q_level > 50
        scale = (100 - q_level)/50; %scaling factor for quality level > standard quality level
    elseif q_level < 50
        scale = 50/q_level;         %scaling factor for quality level < standard quality level
    else
        scale = 1;                  %scaling factor for quality level = standard quality level
    end

    Q_matrix = scale * standard_Q_Matrix;   %Find the quantization matrix
end
```

The function **calculatePSNR** takes an original image and its decompressed image as inputs and calculate the peak signal to noise ratio between them and outputs the result.

```
%Calculate PSNR
function PSNR = calculatePSNR(gray_Im, recostructed_Im)
    mean_squaredError = mean(mean((gray_Im - recostructed_Im).^2));  %Mean squared error over 2 dimensions
    phi_max = 255;                                                   %Maximum light intensity of 8bit digital image
    PSNR = 20*log10(phi_max/sqrt(mean_squaredError));
end
```

By setting the variable “original” to the compressing image and adjusting the quality level appropriately, the images are compressed and decompressed as follows.

Compressing the image

```
%% Compress Image

compressImage_mat = zeros(rows,columns);
for row = 1 : block_size : rows
    for col = 1 : block_size : columns
       cur_block = gray_image(row : row + block_size - 1 , col : col + block_size - 1);                %Extract block by block from the image
         levelOff_mat = cur_block - 128;                                                               %Subtract 128 from each pixel value
         DCT_mat = dct2(levelOff_mat);                                                                 %Get 2D DCT
         qunatizedIm_mat = round (DCT_mat ./ Q_matrix);                                                %Perform quantization
         compressImage_mat(row : row + block_size - 1 , col : col + block_size - 1) = qunatizedIm_mat; %Update compressed image matrix
    end
end
```

Decompressing the compressed image

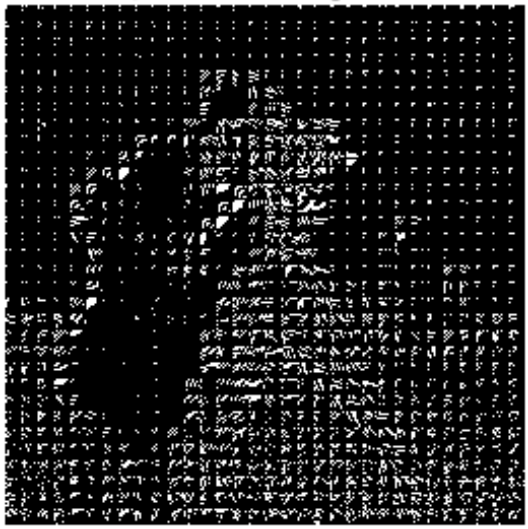
```
%% Reconstruct Image

reconstructed_Image = zeros(rows,columns);
for row = 1 : block_size : rows
    for col = 1 : block_size : columns
        curBlock_Rec = compressImage_mat(row : row + block_size - 1 , col : col + block_size - 1);  %Extract block by block from the compressed image
        R_mat = Q_matrix .* curBlock_Rec;
        IDCT_mat = idct2(R_mat);                                                                    %Get 2D Inverse DCT
        levelup_mat = IDCT_mat + 128;                                                               %Add 128 to each pixel value
        reconstructed_Image(row : row + block_size - 1 , col : col + block_size - 1) = levelup_mat; %Update reconstructed image matrix
    end
end
```

Then PSNR value and zero percentage are calculated and printed. The original image, compressed matrix and the decompresses images are displayed.

- **camera256 Image**

Outputs:

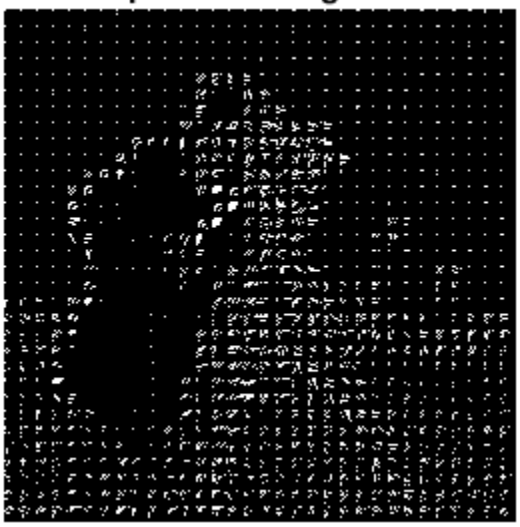
a. Quality level = 70




```
The PSNR of decompressed image: 33.811 dB
The zero percentage of the compressed image: 80.09%
```

b. Quality level = 35

Compressed Image Matrix

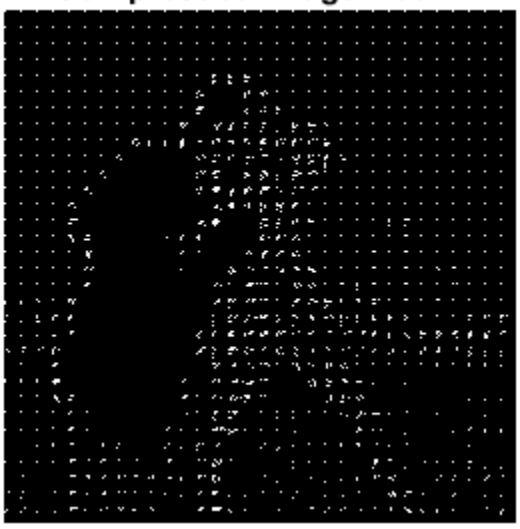


Decompressed Image

```
The PSNR of decompressed image: 30.308 dB
The zero percentage of the compressed image: 88.28%
```

c. Quality level = 10

Compressed Image Matrix

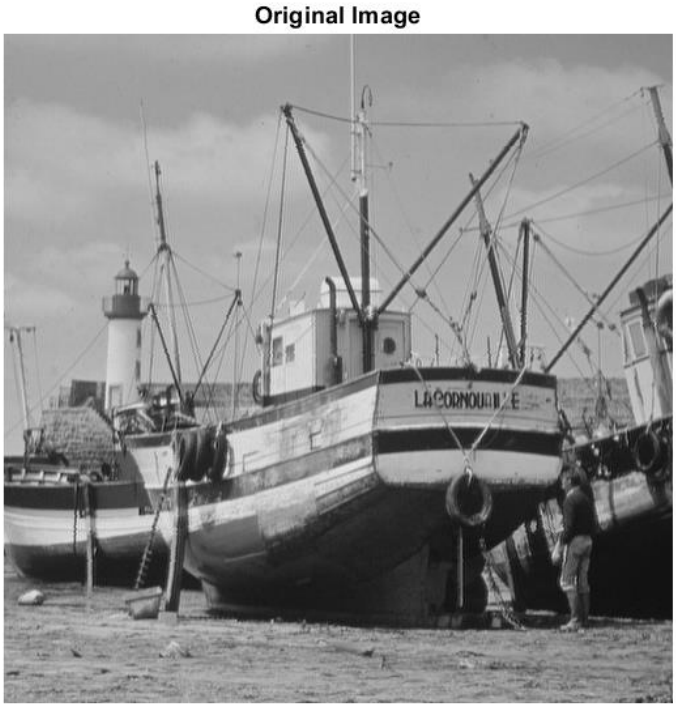


Decompressed Image

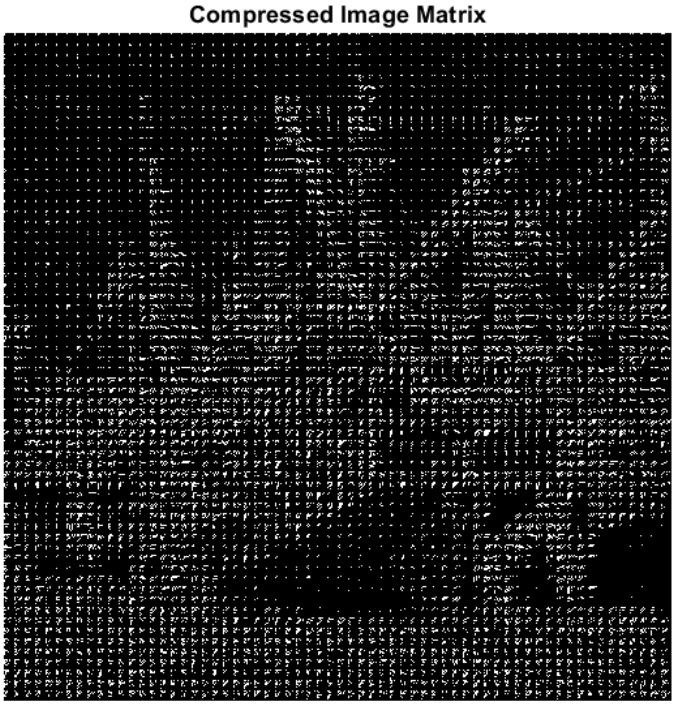
```
The PSNR of decompressed image: 26.243 dB
The zero percentage of the compressed image: 95.01%
```

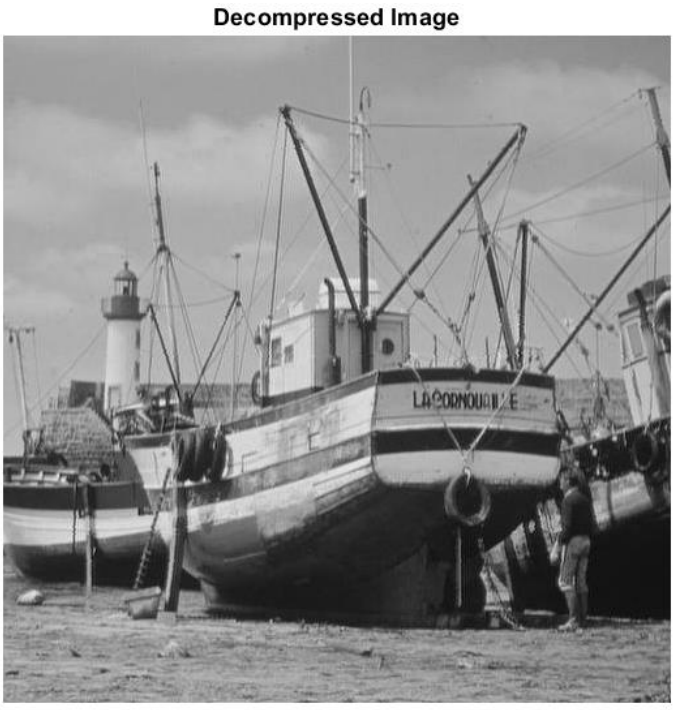
- **boat512 Image**

Outputs



a. Quality level = 70



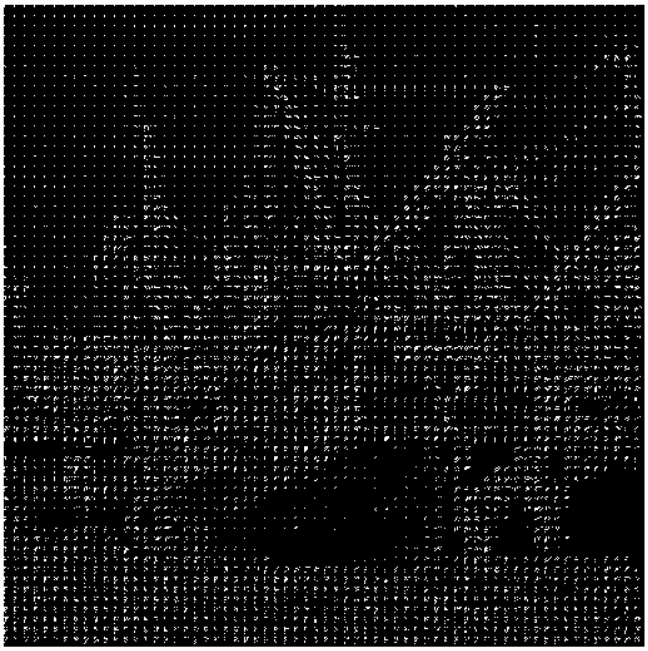


quality level = 70, zeros : 82.54%

```
The PSNR of decompressed image: 36.645 dB
The zero percentage of the compressed image: 82.54%
```

b. Quality level = 35

Compressed Image Matrix



Decompressed Image

quality level = 35, zeros : 89.33%

```
The PSNR of decompressed image: 33.409 dB
The zero percentage of the compressed image: 89.33%
```

c. Quality level = 10

Compressed Image Matrix

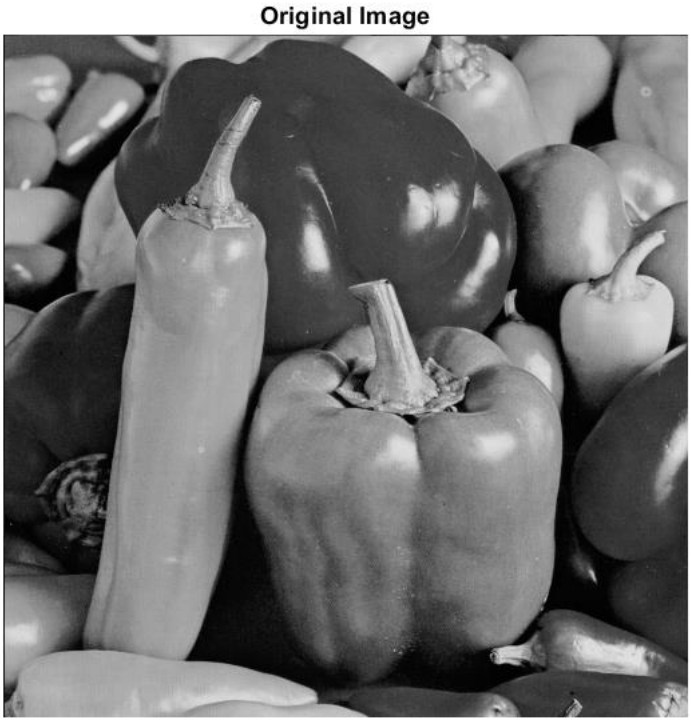
Decompressed Image

quality level = 10, zeros : 95.36%

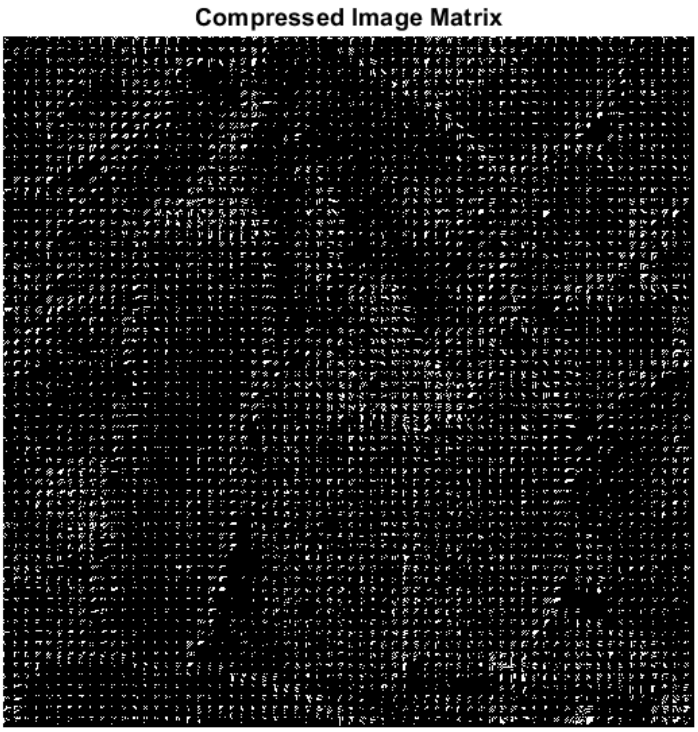
```
The PSNR of decompressed image: 28.860 dB
The zero percentage of the compressed image: 95.36%
```

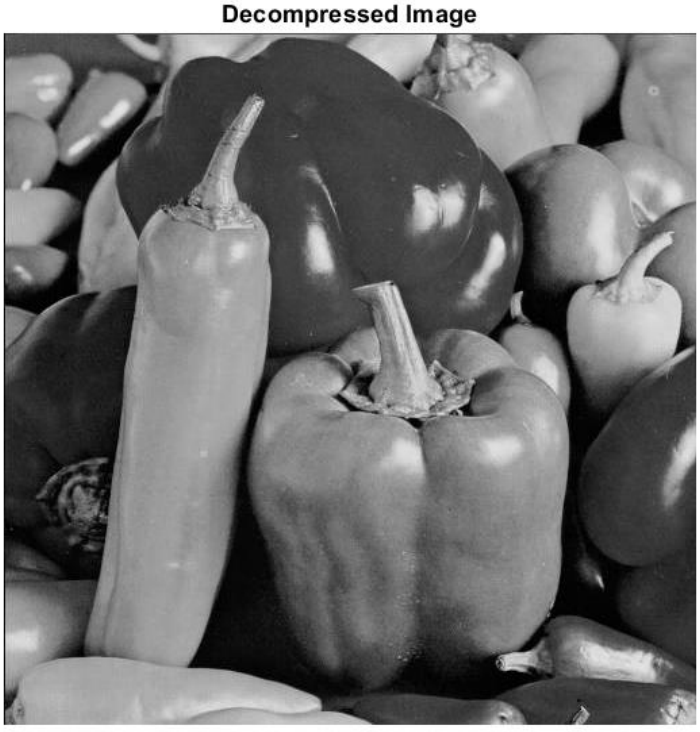
- **peppers512 Image**

Outputs



a. Quality level = 70



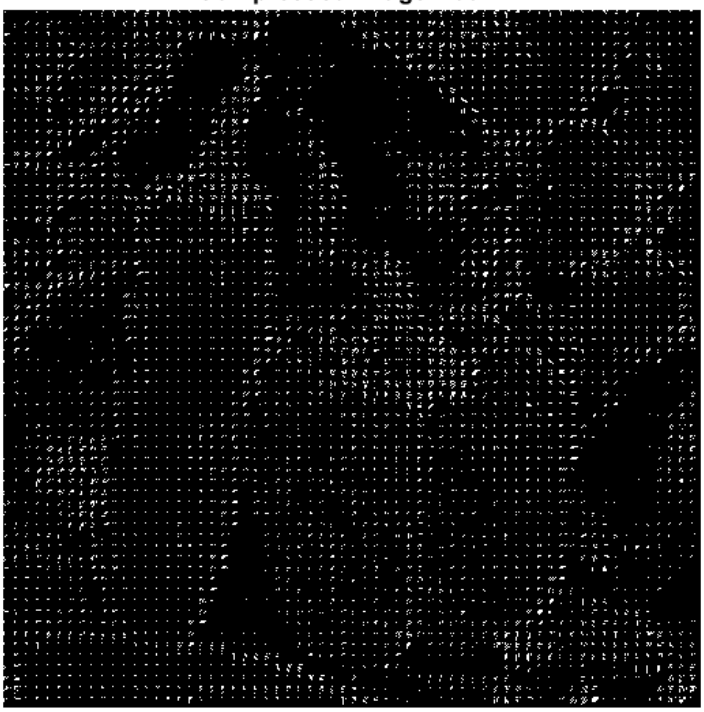


quality level = 70, zeros : 84.25%

```
The PSNR of decompressed image: 36.058 dB
The zero percentage of the compressed image: 84.25%
```

b. Quality level = 35

Compressed Image Matrix



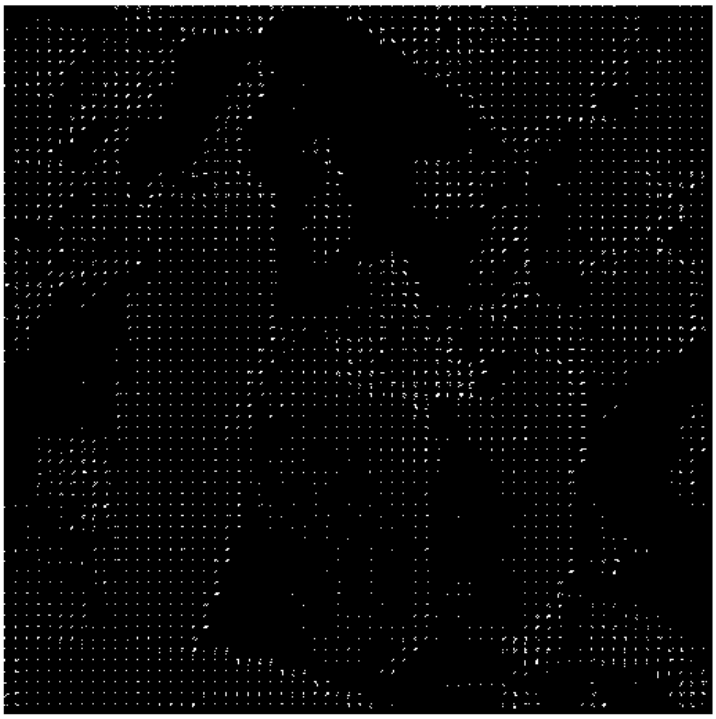
Decompressed Image

quality level = 35, zeros : 91.36%

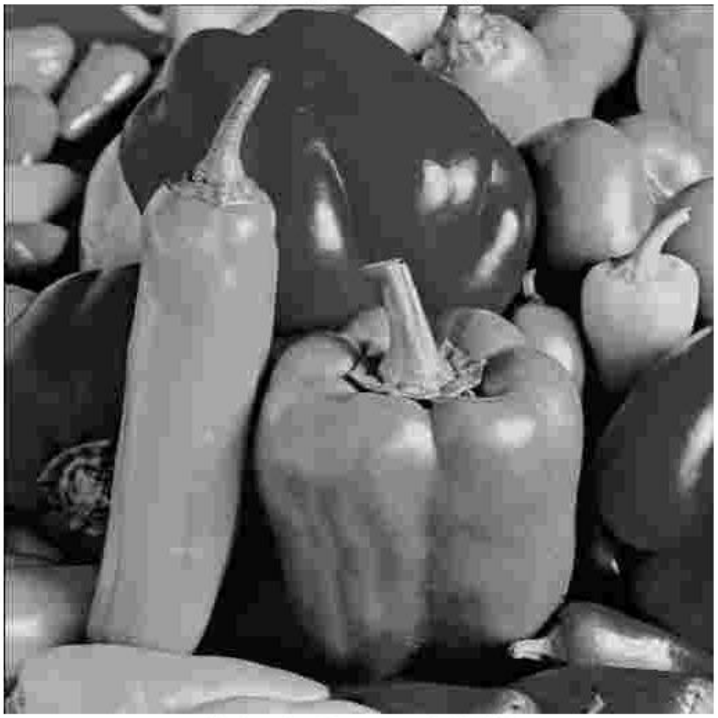
```
The PSNR of decompressed image: 33.959 dB
The zero percentage of the compressed image: 91.36%
```

c. Quality level = 10

Compressed Image Matrix



Decompressed Image



quality level = 10, zeros : 96.00%

```
The PSNR of decompressed image: 30.115 dB
The zero percentage of the compressed image: 96.00%
```

- **Apple Image**

Outputs

a. Quality Level = 70

b. Quality Level = 35

quality level = 35, zeros : 94.39%

c. Quality Level = 10

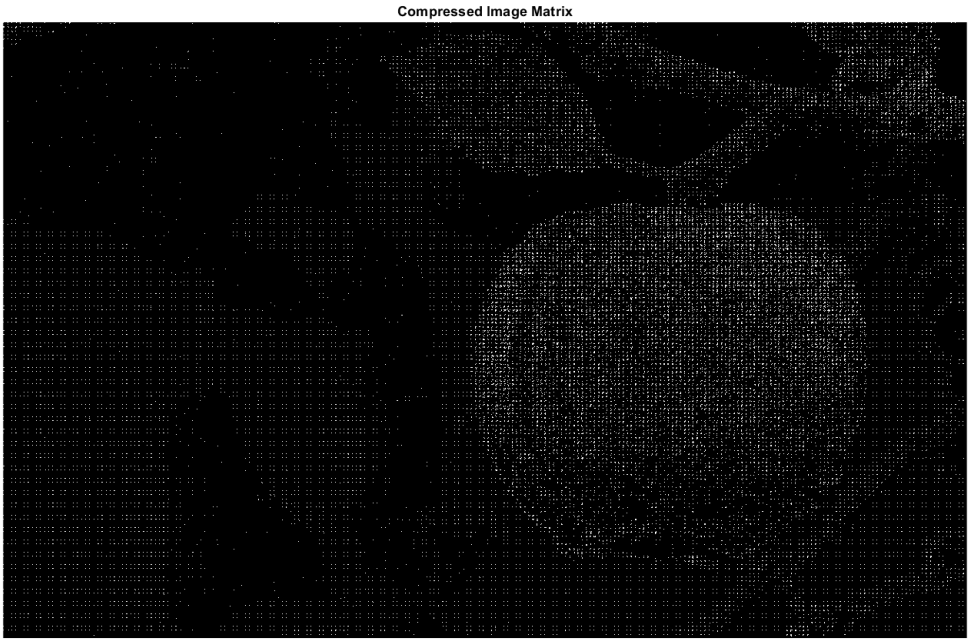
quality level = 10, zeros : 97.40%

Compressed Images and PSNR values

Quality Level = 70

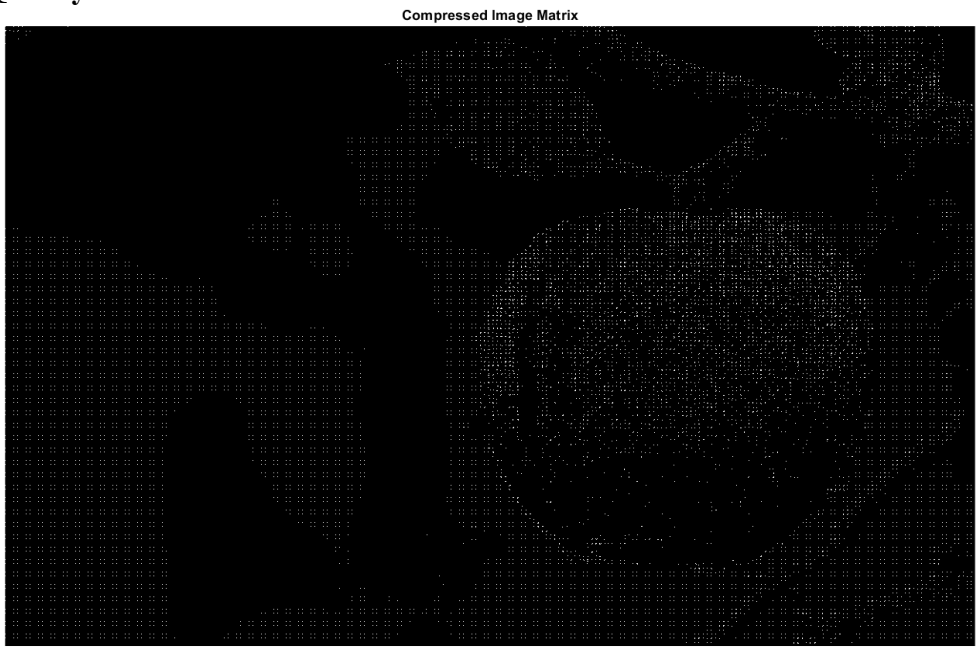

```
The PSNR of decompressed image: 48.454 dB
The zero percentage of the compressed image: 91.06%
```

Quality Level = 35




```
The PSNR of decompressed image: 39.519 dB
The zero percentage of the compressed image: 94.39%
```

Quality Level = 10




```
The PSNR of decompressed image: 33.014 dB
The zero percentage of the compressed image: 97.40%
```

From the above results we can conclude that when quality level of quantization decreases,

- Percentage of zeros increases.
- PSNR decreases.
- Quality of decompressed image reduces.

Question 04

- When compressing an image what we actually do is reducing the file size by simplifying and approximating the image data.
- Some images have lot of intricate patterns and fine details, varying colors, or intricate textures. When we compress such an image, we are trying to fit a lot of details into a small space. It is difficult to simplify such a complex image without losing quality.
- The effect on visual quality of such decompressed image is very significant when the resolution of the original image is relatively low.
- On the other hand, simpler images with fewer details can be compressed with the same compression ratio more effectively while still maintaining a better visual quality.
- Therefore, the original image resolution, complexity, and amount of detail in the image affect how easily it can be compressed while maintaining its visual quality.

Appendix

Codes

```
%% load images

pepper_data = load('peppers512.mat');
camera_data = load('camera256.mat');
boat_data = load('boat512.mat');
apple_data = load('apple.mat');

%Extract images
pepper_im = pepper_data.peppers512;
camera_im = camera_data.camera256;
boat_im = boat_data.boat512;
apple_im = apple_data.M;
```

```
%% Define variables and find quantization matrix

original = apple_im;                          %Change to pepper_im , camera_im or boat_im as required
gray_image =convert2gray(original);            %Convert image to grayscale for processing
image(gray_image);
[rows, columns,~] = size(gray_image);          %size of garyscale image = rows x columns
block_size = 8;                                %Define the extracting block size

quality_level = 70;                            %Change accordingly : 70, 35, 10
Q_matrix = genQuantization_mat(quality_level); %Generate the corresponding quantization matrix
```

```
%% Compress Image

compressImage_mat = zeros(rows,columns);
for row = 1 : block_size : rows
    for col = 1 : block_size : columns
        cur_block = gray_image(row : row + block_size - 1 , col : col + block_size - 1);                %Extract block by block from the image
        levelOff_mat = cur_block - 128;                                                                 %Subtract 128 from each pixel value
        DCT_mat = dct2(levelOff_mat);                                                                   %Get 2D DCT
        qunatizedIm_mat = round (DCT_mat ./ Q_matrix);                                                  %Perform quantization
        compressImage_mat(row : row + block_size - 1 , col : col + block_size - 1) = qunatizedIm_mat;   %Update compressed image matrix
    end
end
```

```
%% Reconstruct Image

reconstructed_Image = zeros(rows,columns);
for row = 1 : block_size : rows
    for col = 1 : block_size : columns
        curBlock_Rec = compressImage_mat(row : row + block_size - 1 , col : col + block_size - 1);  %Extract block by block from the compressed image
        R_mat = Q_matrix .* curBlock_Rec;
        IDCT_mat = idct2(R_mat);                                                                    %Get 2D Inverse DCT
        levelup_mat = IDCT_mat + 128;                                                               %Add 128 to each pixel value
        reconstructed_Image(row : row + block_size - 1 , col : col + block_size - 1) = levelup_mat; %Update reconstructed image matrix
    end
end
```

```
%% Find PSNR and zero percentage

% Calculate Peak Signal-to-Noise Ratio, PSNR
PSNR = calculatePSNR(gray_image, reconstructed_Image);
fprintf('\nThe PSNR of decompressed image: %1.3f dB\n', PSNR);

%Calculate the number of zeros as a percentage
num_zeros = sum(compressImage_mat(:) == 0);           %Get the number of zeros in compressed image
zero_percentage =  (num_zeros / (rows*columns))*100;  %Calculate the percentage
fprintf('The zero percentage of the compressed image: %1.2f%%\n', zero_percentage);
```

```
%% Display the original, compressed and decompressed image

figure (1)
imshow(uint8(original)), title('Original Image'),axis off,axis equal;
figure (2)
imshow(compressImage_mat), title('Compressed Image Matrix'),axis off,axis equal;
figure (3)
imshow(uint8(reconstructed_Image)), title('Decompressed Image'),axis off,axis equal;
text('String', sprintf('quality level = %d, zeros : %1.2f%%', quality_level, zero_percentage), 'Position', [650 1225], 'FontSize', 10, 'Color', 'black'); %Adjust position
```

```
%% Define useful functions

%Convert to grascale
function result = convert2gray(image)
    if ndims(image) == 3
        result = rgb2gray(image);   %color image to grayscale
    else
        result = image;             %originally grayscale image
    end
end

%Define the quantization matrix for a given quality level
function  Q_matrix = genQuantization_mat(q_level)
    %Standard quantization matrix = Q_50
    standard_Q_Matrix = [
        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
        ];

    if q_level > 50
        scale = (100 - q_level)/50; %scaling factor for quality level > standard quality level
    elseif q_level < 50
        scale = 50/q_level;         %scaling factor for quality level < standard quality level
    else
        scale = 1;                  %scaling factor for quality level = standard quality level
    end
    Q_matrix = scale * standard_Q_Matrix;   %Find the quantization matrix
end

%Calculate PSNR
function PSNR = calculatePSNR(gray_Im, recostructed_Im)
    mean_squaredError = mean(mean((gray_Im - recostructed_Im).^2));  %Mean squared error over 2 dimensions
    phi_max = 255;                                                   %Maximum light intensity of 8bit digital image
    PSNR = 20*log10(phi_max/sqrt(mean_squaredError));
end
```