UNIVERSITY OF MORATUWA, SRI LANKA

Faculty of Engineering

Department of Electronic and Telecommunication Engineering

Semester 5 (Intake 2020)

EN3551 - Digital Signal Processing

Assignment 2: Application of 2D-DCT for Image Compression

Pramuditha A. A. H. – 200476P

This report is submitted as a partial fulfillment for module EN3551 - Digital Signal Processing, Department of Electronic and Telecommunication Engineering, University of Moratuwa.

PROCEDURE

This assignment is mainly about the application of 2D-DCT (Discrete Cosine Transform) to compress digital images. As the first step of the assignment, three datasets of images were loaded as struct objects to the MATLAB workspace.

2D-DCT image compression is carried out by following several standard steps.

- The loaded struct object is converted to a matrix which has size $M\ x\ N$, where M and N are multiples of 8 and represented with 8 bits. Then this matrix will be segmented into 8 x 8 blocks (let **B** be a one 8 x 8 block.)

- This **B** matrix is leveled-off by subtracting 128 from each entry and applied 2-D DCT by using **dct2()** function to leveled-off matrix to obtain matrix **C**, which can also be obtained by computing,

$$C = T_8 \tilde{B} T_8^T$$

- Quantization is a critical step in this process. The quantization can be achieved by converting $C = \{c_{i,j}\}$ to matrix $S = \{s_{i,j}\}$ determined by,

$$s_{i,j} = round\left(\frac{c_{i,j}}{q_{i,j}}\right)$$

  Where $Q = \{q_{i,j}\}$ is a quantization matrix that can be selected according to a desired quality level of the compression. If a quantization matrix with a reduced quality level is used, **S** will contain more zeros, thus a higher compression can be achieved at the cost of reduced image quality.

- The quantized matrix S is then converted by an encoder to a stream of binary data by a differential pulse-code modulation (DPCM) scheme under the coding process.

When the compressed image is received by the receiver, it will follow a decompression process to reconstruct the original image.

- First, an image block **R** will be obtained by performing pointwise multiplication between matrix **S** and quantization matrix **Q** such that, $R = Q \odot S$.

- Then, 2-D inverse DCT will be applied to matrix **R** by using **idct2()** function. Alternatively, it con be done using the following formula,

$$E = D_8^T R D_8$$

- Finally, the effect of the “level-off” operation done in transmitter is considered by adding 128 to the entries of the matrix **E**. This yields the reconstructed image block that mimics image block **B**.

- To compute the compression factor of each compression, the total number of zeros in **S** matrices will be counted and presented as a percentage. This will indicate how the image has been compressed. As expected, the use of reduced quality levels leads to a higher percentage of total zeros at the cost of a degraded image.

- Apart from that, PSNR (Peak Signal to Noise Ratio) values will be defined as follows:

$$PSNR = 20log_{10}(\frac{\psi_{max}}{\sigma_e})$$

Where ψ_{max} denotes the maximum light intensity of the original image and the σ_e^2 denotes the mean squared error which can be obtained as follows:

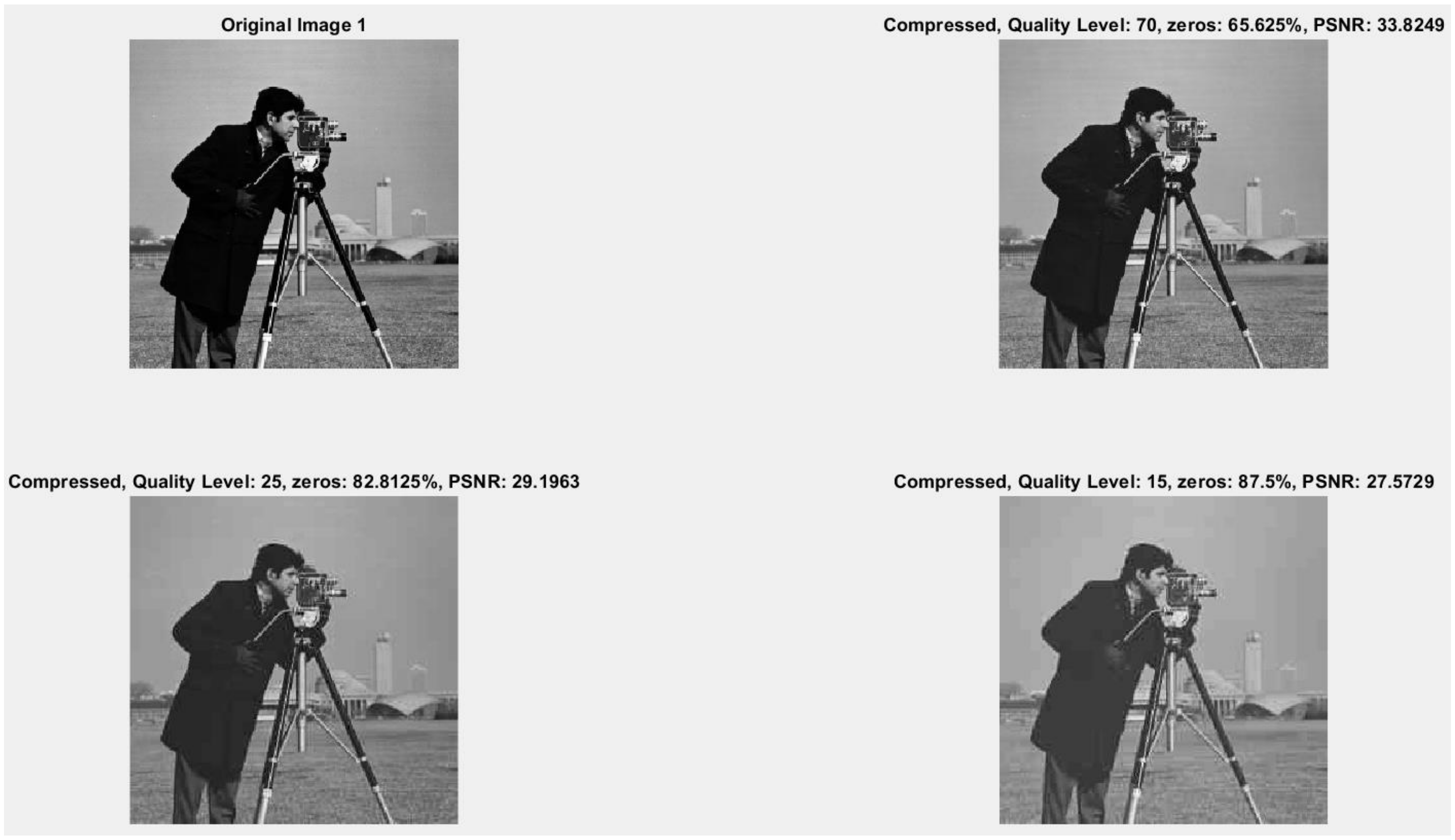
$$mean\ squared\ error = mean(mean((I_{rec} -\ I_{original})^2))$$

For an 8-bit digital image, $\psi_{max} = 255$.

TASKS

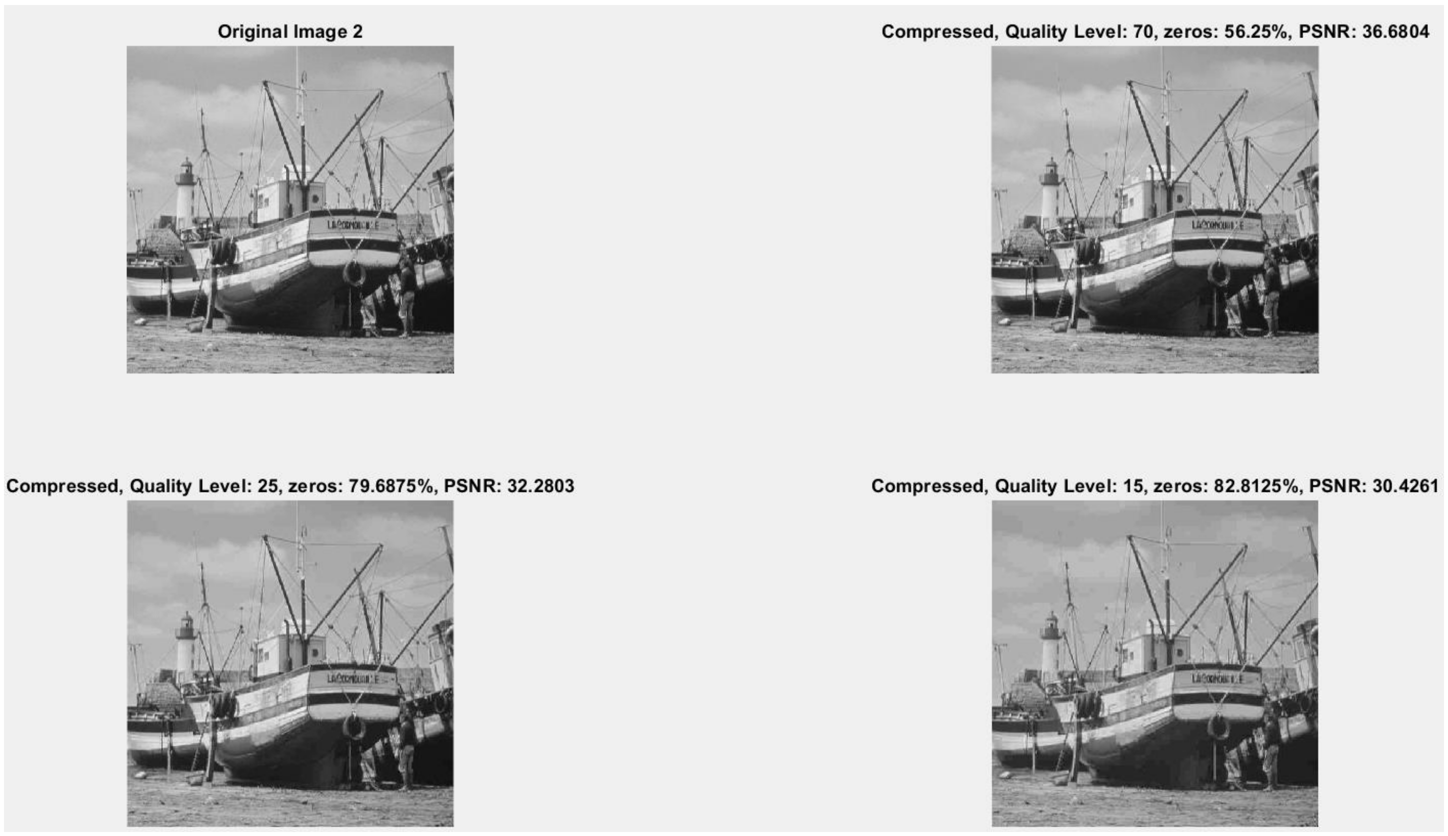
Based on the above process, three images were compressed and reconstructed using MATLAB operations and compression quality is presented as the zero percentage of their **S** matrices and PSNR values. The original and compressed versions of the images can be viewed as follows:

1. Image 1


Original Image 1
Compressed, Quality Level: 70, zeros: 65.625%, PSNR: 33.8249
Compressed, Quality Level: 25, zeros: 82.8125%, PSNR: 29.1963
Compressed, Quality Level: 15, zeros: 87.5%, PSNR: 27.5729

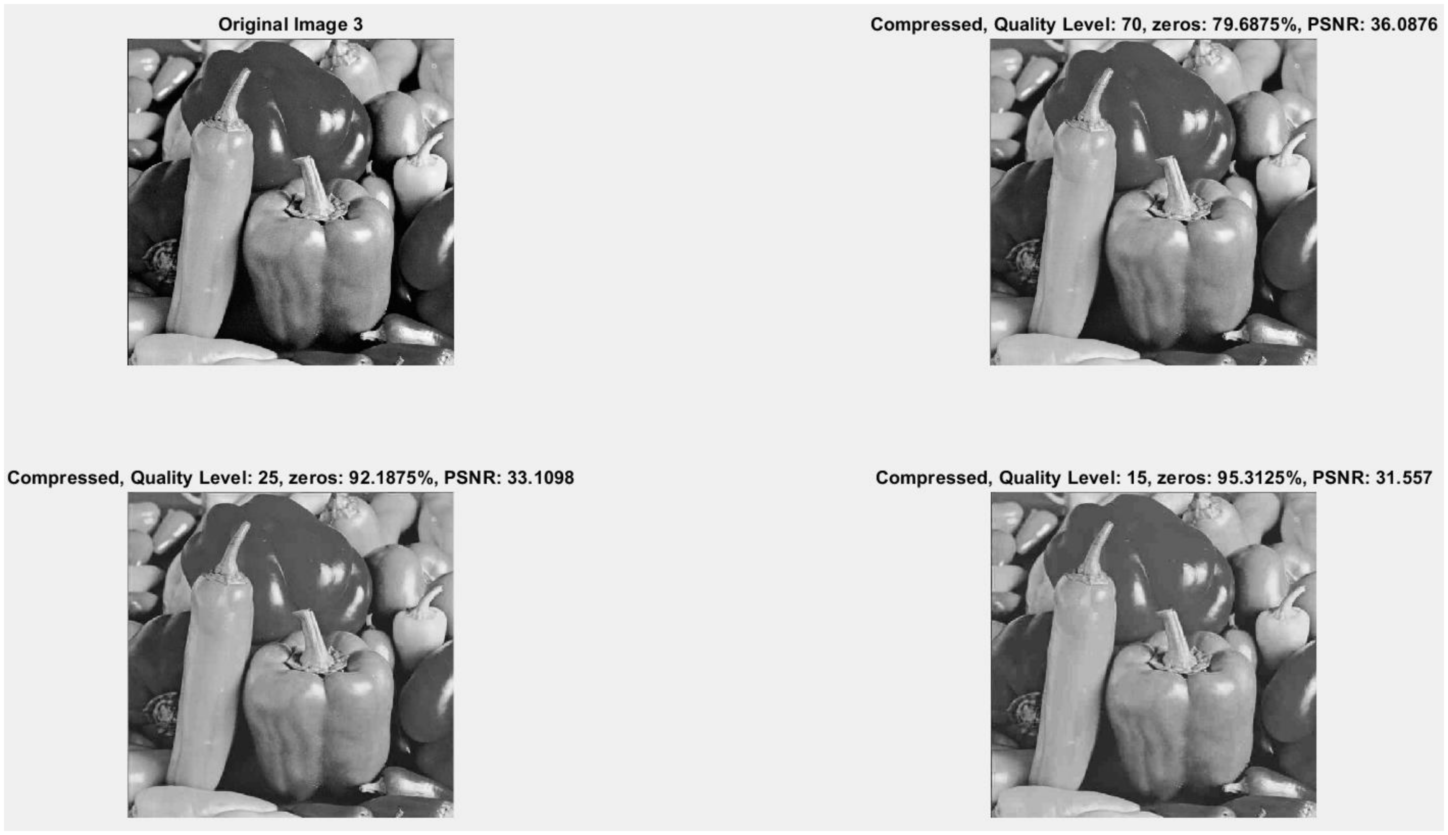
Compressed Quality Level	***Percentage of Zeros (%)***	***PSNR (%)***	***Comments***
70	65.625	33.825	The objects in the frame are visible and at a similar level as the relative quality.
25	82.8125	29.196	The objects in the frame are visible and at a similar level as the relative quality.
15	87.5	27.573	The objects in the frame are visible but the quality of the image has been significantly reduced.

2. Image 2


Original Image 2
Compressed, Quality Level: 70, zeros: 56.25%, PSNR: 36.6804
Compressed, Quality Level: 25, zeros: 79.6875%, PSNR: 32.2803
Compressed, Quality Level: 15, zeros: 82.8125%, PSNR: 30.4261

Compressed Quality Level	***Percentage of Zeros (%)***	***PSNR (%)***	***Comments***
70	56.25	36.68	The objects in the frame are visible and at a similar level as the relative quality.
25	79.687	32.28	The objects in the frame are visible and at a similar level as the relative quality.
15	82.812	30.426	The objects in the frame are visible but the quality of the image has been significantly reduced.

3. Image 3


Original Image 3
Compressed, Quality Level: 70, zeros: 79.6875%, PSNR: 36.0876
Compressed, Quality Level: 25, zeros: 92.1875%, PSNR: 33.1098
Compressed, Quality Level: 15, zeros: 95.3125%, PSNR: 31.557

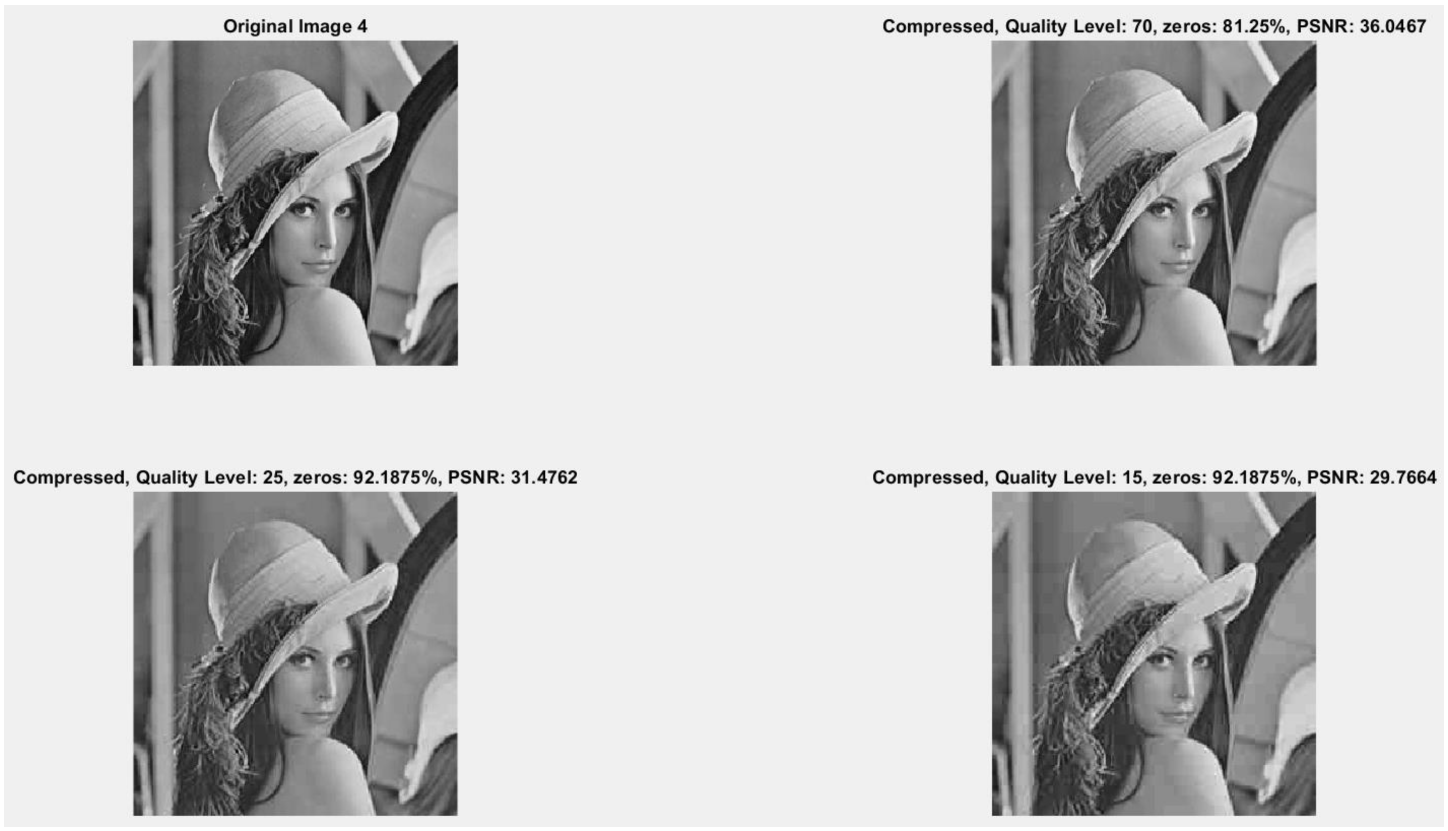
Compressed Quality Level	*Percentage of Zeros (%)*	*PSNR (%)*	*Comments*
70	79.6875	36.087	The objects in the frame are visible and output image quality exceeds the expected quality
25	92.187	33.11	The objects in the frame are visible and output image quality exceeds the expected quality
15	95.312	31.56	The objects in the frame are visible but the quality of the image has been significantly reduced.

Unlike in the previous two image compressions, the zero percentages of this image compression under much lower quality level exists in much more higher values. The main reason for that can be considered as the sparsity of the “S” matrix.

When a significant portion of the frequency components (in DCT matrix) have small values, those values can be easily approximated as zeros. Also, images with more uniform regions and less high-frequency details are more likely to produce highly sparse (with more zero elements) quantized matrices at the end of the compression stage.

The same process was applied to the newly selected image and the differences between each compression levels can be viewed as follows:

4. Image 4


Original Image 4
Compressed, Quality Level: 70, zeros: 81.25%, PSNR: 36.0467
Compressed, Quality Level: 25, zeros: 92.1875%, PSNR: 31.4762
Compressed, Quality Level: 15, zeros: 92.1875%, PSNR: 29.7664

Compressed Quality Level	*Percentage of Zeros (%)*	*PSNR (%)*	*Comments*
70	81.25	36.046	The objects in the frame are visible and at a similar level as the relative quality.
25	92.187	31.47	The objects in the frame are visible and at a similar level as the relative quality.
15	92.19	29.76	The objects in the frame are lightly visible and the quality of the image has been significantly reduced.

In here also, the zero percentages of the "S" matrix for lower quality levels are much higher which indicates the sparsity of the matrix.

DCT image compression is mainly done by exploiting the redundancies present within an image, when compressing the image these redundancies will be eliminated. When DCT is applied, the information below a threshold, that is determined by the compression level, is approximated to zero, which reduces the finer image details effectively. When it comes to images with higher pixel correlation, they can achieve higher compression ratios without a considerable loss in their visual quality. However, images containing features such as edges, textures, depth, shadows, fine lines, and variations in lighting will have poor pixel correlation throughout the image. Compressing such images will cause a significant loss of image details.

As an example:

- In the first image, with the compression, the depth of the far away buildings, the texture of the ground and slight variations of the color in sky will lose their details.
- In the fourth image, with the compression, the depth of the far away background, color texture of the hair and slight variations in the face of the lady will lose their details.

As the compression level rises, the portion of zero values in the image increases, resulting in decrease in PSNR value. This indicates a greater degree of distortion in the image. Images characterized by strong pixel correlation across the entire image, such as image 2 and image 3 (relative to the other two images) were able to maintain a considerably higher level of visual quality even when subjected to compression.

REFERENCES

1. "MATLAB Documentation – MathWorks India" - https://in.mathworks.com/help/matlab/
2. A. V. Oppenheim, R. W. Schafer, and Pearson Education, Discrete-time signal processing. Harlow: Pearson Education Limited, 2014.
3. "Papers with Code – Set11 Dataset" - https://paperswithcode.com/dataset/set11

APPENDIX – MATLAB CODE

```matlab
%% Assignment 2 - Application of 2D-DCT for Image Compression

% Name          : A.A.H. Pramuditha
% Index No      : 200476P

clc;
clear;
close all;

%% 3.1) 2-D DCT - Application to Image Compression

% Load the files containing the image data
image1 = load('camera256.mat');
image2 = load('boat512.mat');
image3 = load('peppers512.mat');

% Load the extra image as a TIFF file and store it as a struct
extra_image = double(imread('lena256.tif'));
image4 = struct('data', extra_image);

% Store all the image data in a single struct object
images = struct();
images(1).Field1 = 'Image 1';
images(1).Field2 = image1;
images(2).Field1 = 'Image 2';
images(2).Field2 = image2;
images(3).Field1 = 'Image 3';
images(3).Field2 = image3;
images(4).Field1 = 'Image 4';
images(4).Field2 = image4;


% Define quality levels
quality_levels = [70, 25, 10];

% Define Standard Quality Matrix (Q_50)
Q_50 = [16, 11, 10, 16, 24, 40, 51, 61;
        12, 12, 14, 19, 16, 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;];

% Define the block size
block_size = 8;

% Image selection and compression procedure
for k = 1:4
```

```
    % Convert the struct object into a matrix
    image = struct2cell(images(k).Field2);
    data_matrix = cell2mat(image);

    % Display the dimensions of the matrix
    [rows, columns] = size(data_matrix);
    fprintf('Image Dimensions: %d rows x %d columns\n', rows, columns);

    % Calculate the number of blocks in each dimension
    x_blocks = size(data_matrix, 2) / block_size;
    y_blocks = size(data_matrix, 1) / block_size;

    figure();
    % Plot the original image
    % subplot(3, 4, 4*k - 3);
    subplot(2, 2, 1);
    imshow(data_matrix, []);
    title(['Original Image ', num2str(k)]);

    % Compress the image for a given quality level
    for n = 1:3

        % Calculate the required quality value
        quality_level = quality_levels(n);
        if (quality_level>50)
            J = (100-quality_level) / 50;
        else
            J = 50 / quality_level;
        end

        % Initialize an empty matrix to store the final image
        final_image = zeros(rows, columns);

        % Extract the blocks and do the operations
        for i = 1:y_blocks
            for j = 1:x_blocks

                % Extract the blocks with required size
                row_start = (i-1)*block_size + 1;
                row_end = i*block_size;
                column_start = (j-1)*block_size + 1;
                column_end = j*block_size;

                B = data_matrix(row_start:row_end,
column_start:column_end);

                % Level-off the matrix by substracting 128 from each entry
                B_tilda = B - 128;

                % Apply 2-D DCT to the leveled-off matrix
                C = dct2(B_tilda);
                % Quantization
                Q = Q_50 * J;
                S = round(C ./ Q);

                % Decompress the S matrix
                R = Q .* S;

                % Apply 2-D inverse DCT  to matrix R
                E = idct2(R);

                % Correct the level-off operation done in compression
stage
                decom_B = E + 128;

                % Reconstruct the total matrix for final image
                final_image(row_start:row_end, column_start:column_end) =
decom_B;
            end
        end

        % Calculate the zero element percentage
        total_elements = numel(S);
        num_zeros = sum(S(:) == 0);
        percentage_zero = (num_zeros / total_elements) * 100;

        % Calculate Peak Signal to Noise Ratio (PSNR)
        squared_error_matrix = (final_image - data_matrix).^2;
        variance_sigma_e = mean(mean(squared_error_matrix));
        PSNR = 20 * log10(255 / (variance_sigma_e^0.5));

        % Plot the compressed versions of the image
        % subplot(3, 4, 4*k - 3 + n);
        subplot(2, 2, n+1);
        imshow(final_image, []);
        title(['Compressed, Quality Level: ', num2str(quality_level), ',
zeros: ', num2str(percentage_zero), '%, PSNR: ', num2str(PSNR)]);
    end
end
```