EN3551 - Digital Signal Processing

Assignment 02

Application of 2D-DCT for Image Compression

Name: Samaramanna M. A.

Index Number: 220551K

Date: 26 October 2025

1 Introduction

This experiment explores the implementation of the two-dimensional Discrete Cosine Transform (2D-DCT) for the purpose of digital image compression. The 2D-DCT is an essential transformation in multimedia systems such as JPEG, owing to its strong energy compaction and decorrelation properties.

In this study, a set of images was compressed and reconstructed using 2D-DCT at three distinct quality levels (90, 35, and 5). The performance of each compression level was assessed in terms of the percentage of zero coefficients and the Peak Signal-to-Noise Ratio (PSNR).

Additionally, one custom image was selected to further analyze the effect of DCT-based compression.

2 Analysis for Monarch Image

Original Image Quality Level 90 Quality Level 35 Quality Level 5

Figure 1: Original and reconstructed Monarch images at different quality levels.

Quantitative Results

Quality Level	Percentage of Zeros (%)	PSNR (dB)
90	61.35	40.03
35	84.6	31.05
5	95.42	23.76

Table 1: Compression statistics for Monarch image.

Qualitative Analysis

At higher quality levels, the Monarch image retained most fine details with very limited degradation. When the quality level dropped to 35, noticeable blurring appeared on textured regions such as the patterned cloth. At level 5, artifacts such as blockiness and loss of edge sharpness became visible, illustrating the impact of stronger quantization.

Remarks

The Monarch image, which includes fine textures and repetitive patterns, demonstrates that high-frequency content is more sensitive to compression. As quantization increases, subtle details are lost while overall brightness and average tones remain preserved.

3 Analysis for Cameraman Image

Original Image Quality Level 90 Quality Level 35 Quality Level 5

Figure 2: Original and reconstructed Cameraman images at different quality levels.

Quantitative Results

Quality Level	Percentage of Zeros (%)	PSNR (dB)
90	63.87	39.97
35	88.27	30.31
5	96.02	24.22

Table 2: Compression statistics for Cameraman image.

Qualitative Analysis

At quality level 90, the house image remained almost identical to the original. As the quality decreased to 35, a reduction in texture visibility and minor edge blurring were noticed. At the lowest level 5, the image became distinctly blocky, and the fine texture of bricks and edges lost clarity.

Remarks

This image exhibits sharp contrasts and geometric structures. Even under heavy compression, major features were preserved, but texture details faded. The PSNR drop validates the visible deterioration caused by stronger quantization.

4 Analysis for Parrot Image

Original Image Quality Level 90 Quality Level 35 Quality Level 5

Figure 3: Original and reconstructed Parrots images at different quality levels.

Quantitative Results

Quality Level	Percentage of Zeros (%)	PSNR (dB)
90	71.28	41.42
35	90.42	33.47
5	97.03	26.04

Table 3: Compression statistics for Parrot image.

Qualitative Analysis

At high quality levels, the Parrot image preserved good color transitions and edge definition. At lower quality levels, fine details on feathers became indistinct, and the smooth transitions between color regions degraded into patches.

Remarks

The Parrot image, containing strong color variations and sharp boundaries, shows DCT's efficiency in retaining low-frequency structure while reducing high-frequency detail. This behavior highlights DCT's nature of concentrating most of the energy in a few low-frequency coefficients.

5 Analysis of Pepper Image (My Choise of Grayscale Image)

Original Image | Quality Level 90 | Quality Level 35 | Quality Level 5

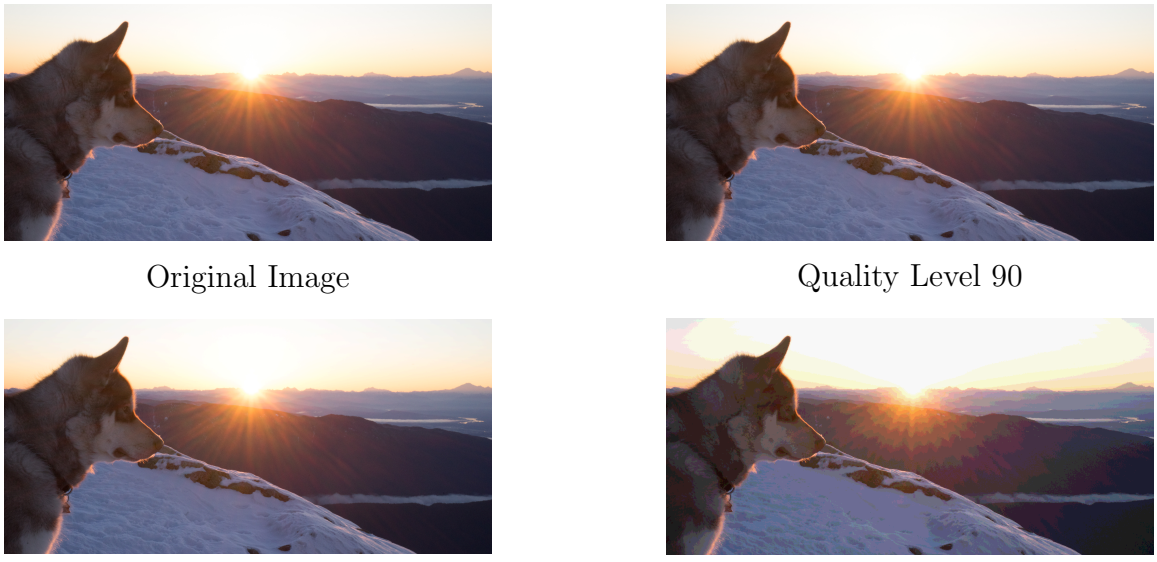
Figure 4: Original and reconstructed Parrots images at different quality levels.

Similar discussion can be done to this image as the previous three images.

6 Analysis for Custom Image (sunrise.bmp)

The custom image chosen for this task was `sunrise.bmp`, a high-resolution **color image**. Due to its large dimensions, the image was divided into 8×8 blocks, and **zero-padding** was applied to ensure complete block coverage before compression.

Each color channel (Red, Green, Blue) was individually processed using 2D-DCT, quantized, and reconstructed through the inverse DCT. The three channels were then combined to form the final decompressed RGB image.



Original Image | Quality Level 90

Quality Level 35 | Quality Level 5

Figure 5: Original and reconstructed Color Sunrise images at different quality levels.

Discussion

At higher quality levels, the image retains most of its natural color transitions and edge clarity. As the quality level decreases, a reduction in contrast and color fidelity is observed. At Q=10, color blocking artifacts and banding appear, especially around the sun and sky gradient.

Because the image is high-resolution, padding slightly increases the total data size before compression, but it does not significantly impact the overall PSNR or visual perception. The test highlights that DCT compression handles color images effectively when applied independently to each channel.

7 Overall Discussion and Conclusion

From the experiments, it is clear that reducing the quantization quality increases the proportion of zero-valued coefficients, improving compression ratio but lowering reconstruction quality.

Images with simple textures maintained acceptable visual quality at lower qualities, while those with complex details degraded more rapidly. The high-resolution color image (`sunrise.bmp`) exhibited a smooth degradation and showed how padding ensures complete block processing for non-multiple-of-eight image sizes.

The relationship between quality level and PSNR is approximately logarithmic—PSNR decreases as quantization strength increases. Despite the losses, the 2D-DCT proves efficient for energy compaction and is suitable for image compression applications balancing visual quality and file size.

Overall, DCT-based compression provides an effective balance between data reduction and image quality, confirming its suitability for standard image compression systems.

Appendix: MATLAB Codes

A.1 Image Compression using 2D-DCT

Listing 1: DCT-based Image Compression in MATLAB for grayscale Images

```
images = [ images/Monarch.mat ,  images/cameraman.mat ,  images/Parrots
    .mat , ...
      images/peppers.mat ];

image_names = [ monarch ,  cameraman ,  parrots ,  peppers ];

for im = 1: length(images)
    % Load and read the image
    image_file = load(images(im));
    image_var = fieldnames(image_file);
```

```
    image = image_file.(image_var{1});

    % Define the block size
    block_size = 8;
    % Image height and width
    [M, N] = size(image);

    % Display the size of the image
    disp([M, N]);

    % Show the original image and save it as png
    figure;
    imshow(image, []);
    title(sprintf('Original Image %d: %s', im, image_names(im)));
    imwrite(uint8(mat2gray(image) * 255), ...
        sprintf('%s_original.png', image_names(im)));

    % Divide the image into 8x8 blocks

    % Create row and column vectors for mat2cell
    row_blocks = repmat(block_size, 1, M/block_size);
    col_blocks = repmat(block_size, 1, N/block_size);

    % Split image into 8x8 blocks
    blocks = mat2cell(image, row_blocks, col_blocks);

    dct_blocks = cell(size(blocks));  % same size as blocks

    % Apply DCT for each block
    for i = 1:size(blocks,1)
        for j = 1:size(blocks,2)
            % Convert block to double and shift values to [-128,127]
            block = double(blocks{i,j}) - 128;
            % Apply 2D DCT
            dct_blocks{i,j} = dct2(block);
        end
    end

    % Quantization matrix eoth quality level 50
    Q_standard = [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];

```

```
    % Quality levels
    Q_levels = [90, 35, 5];
    for q=1:length(Q_levels)
        % Create Quantization matrix for desired quality level
        Q_scaled = scale_q_matrix(Q_standard, Q_levels(q));

        % Quantization
        quant_blocks = cell(size(dct_blocks));
        for i = 1:size(dct_blocks,1)
            for j = 1:size(dct_blocks,2)
                quant_blocks{i,j} = round(dct_blocks{i,j} ./ Q_scaled);
            end
        end

        % Flatten all quantized blocks into one array
        all_coeffs = cell2mat(quant_blocks(:));

        % Count zeros
        num_zeros = sum(all_coeffs(:) == 0);
        total_coeffs = numel(all_coeffs);

        % Percentage of zeros
        percent_zeros = (num_zeros / total_coeffs) * 100;
        fprintf('Percentage of zeros for %s at Q:%d: %.2f%%\n',
           image_names(im), Q_levels(q) , percent_zeros);

        % Reconstruction of compressed image
        recon_img = zeros(M,N);

        for i = 1:size(quant_blocks,1)
            for j = 1:size(quant_blocks,2)
                % Dequantization
                dequant_block = quant_blocks{i,j} .* Q_scaled;
                % Inverse DCT
                recon_block = idct2(dequant_block);
                % Recombine into final image
                row_start = (i-1)*block_size + 1;
                col_start = (j-1)*block_size + 1;
                recon_img(row_start:row_start+block_size-1, ...
                    col_start:col_start+block_size-1) = recon_block +
                       128;
            end
        end

        %  Peak Signal to Noise Ratio(PSNR)
        sigma_squared = mean(mean((double(recon_img(:)) - double(image
           (:))).^2));
        psnr_value = 20 * log10(255 / sqrt(sigma_squared));
```

```
        fprintf('PSNR for for %s at quality %d: %.2f dB\n', image_names
            (im) ,Q_levels(q), psnr_value);

        % Show the reconstructed image and save it
        figure;
        imshow(recon_img, []);
        title(sprintf('Reconstructed Image %s - Q %d', ...
                image_names(im), Q_levels(q)));
        % Create output folder if missing
        imwrite(uint8(mat2gray(recon_img) * 255), ...
                sprintf('reconstructed_%s_Q%d.png', image_names(im),
                    Q_levels(q)));
    end
end

%% Function for calculating quantization matrix for specific quality
function Q_scaled = scale_q_matrix(Q_standard, quality)
    if quality < 50
        scale = 50 / quality;
    else
        scale = (100 - quality) / 50;
    end
    % rounding and clipping the values above 255
    Q_scaled = round(scale * Q_standard);
    Q_scaled(Q_scaled > 255) = 255;
end
```

Listing 2: DCT-based Image Compression in MATLAB for Color Image

```
 % Load color image
img = imread( Sunrise.bmp ) ;
img = im2double(img);  % Convert to [0,1]
[M, N, C] = size(img);

block_size = 8;
Q_standard = [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];

Q_levels = [90, 35, 5]; % Example quality

% Pad image so dimensions are multiples of 8
pad_M = mod(8 - mod(M,8), 8);
pad_N = mod(8 - mod(N,8), 8);
```

```
21
22  img_padded = padarray(img, [pad_M, pad_N], 'replicate', 'post');
23  [M_pad, N_pad, ~] = size(img_padded);
24
25  % Display
26  figure;
27  imshow(uint8(img*255));
28  title('Original Color Image');
29
30
31  for q=1:length(Q_levels)
32      Q_level = Q_levels(q);
33      Q_scaled = scale_q_matrix(Q_standard, Q_level);
34
35      recon_img = zeros(M_pad, N_pad, C);
36
37      for c = 1:C
38          channel = img_padded(:,:,c) * 255;  % scale to 0  255
39
40          row_blocks = repmat(block_size, 1, M_pad/block_size);
41          col_blocks = repmat(block_size, 1, N_pad/block_size);
42          blocks = mat2cell(channel, row_blocks, col_blocks);
43
44          dct_blocks = cell(size(blocks));
45          quant_blocks = cell(size(blocks));
46
47          % DCT + Quantization
48          for i = 1:size(blocks,1)
49              for j = 1:size(blocks,2)
50                  block = double(blocks{i,j}) - 128;
51                  dct_block = dct2(block);
52                  quant_blocks{i,j} = round(dct_block ./ Q_scaled);
53              end
54          end
55          disp('done...')
56
57          % Dequantization + IDCT
58          for i = 1:size(quant_blocks,1)
59              for j = 1:size(quant_blocks,2)
60                  dequant_block = quant_blocks{i,j} .* Q_scaled;
61                  recon_block = idct2(dequant_block) + 128;
62                  row_start = (i-1)*block_size + 1;
63                  col_start = (j-1)*block_size + 1;
64                  recon_img(row_start:row_start+block_size-1, ...
65                            col_start:col_start+block_size-1, c) =
                                recon_block;
66              end
67          end
```

```
        disp('done...')
    end

    % Remove padding
    recon_img = recon_img(1:M, 1:N, :);

    % Clip to 0-255 and convert to uint8
    recon_img = uint8(max(min(recon_img,255),0));

    figure;
    imshow(recon_img);
    title(sprintf('Reconstructed (Q=%d)', Q_level));

    % Save result
    imwrite(recon_img, sprintf('compressed_color_Q%d.png', Q_level));
end
```

A.2 Observed Results for Percentage of zeros and PSNR

```
>> Assignment02_220551K
   256   256

Percentage of zeros for monarch at Q:90: 61.35%
PSNR for for monarch at quality 90: 40.03 dB
Percentage of zeros for monarch at Q:35: 84.60%
PSNR for for monarch at quality 35: 31.05 dB
Percentage of zeros for monarch at Q:5: 95.42%
PSNR for for monarch at quality 5: 23.76 dB
   256   256

Percentage of zeros for cameraman at Q:90: 63.87%
PSNR for for cameraman at quality 90: 39.97 dB
Percentage of zeros for cameraman at Q:35: 88.27%
PSNR for for cameraman at quality 35: 30.31 dB
Percentage of zeros for cameraman at Q:5: 96.92%
PSNR for for cameraman at quality 5: 24.22 dB
   256   256

Percentage of zeros for parrots at Q:90: 71.28%
PSNR for for parrots at quality 90: 41.42 dB
Percentage of zeros for parrots at Q:35: 90.42%
PSNR for for parrots at quality 35: 33.47 dB
Percentage of zeros for parrots at Q:5: 97.03%
```

```
PSNR for for parrots at quality 5: 26.04 dB
   256   256

Percentage of zeros for peppers at Q:90: 63.82%
PSNR for for peppers at quality 90: 39.79 dB
Percentage of zeros for peppers at Q:35: 87.57%
PSNR for for peppers at quality 35: 32.09 dB
Percentage of zeros for peppers at Q:5: 96.30%
PSNR for for peppers at quality 5: 24.97 dB
```