Department of Electronic & Telecommunication Engineering

University of Moratuwa

Sri Lanka

EN3551 - Digital Signal Processing

Assignment 2

Application of 2D-DCT for Image Compression

210321X Kumarasinghe R D

18/10/2024

Contents

1 Introduction

In this report, we explore the application of the 2D Discrete Cosine Transform (DCT) to compress digital images. The focus is on dividing an image into 8×8 blocks, transforming them using DCT, applying quantization, and analyzing the compression results. Three quality levels of compression will be tested, and the results will be compared in terms of the percentage of zeros, Peak Signal-to-Noise Ratio (PSNR), and visual quality.

2 Theory Overview

2.1 2D-DCT for Image Compression

The Discrete Cosine Transform (DCT) is used in image processing to convert spatial domain data into frequency domain data. It allows image compression by concentrating most of the image's energy in a few low-frequency components. This report utilizes the following steps to compress images using DCT:

- Divide the image into 8x8 blocks.
- Subtract 128 from each pixel value in these blocks to shift the pixel values from [0,255] to [-128,127].
- Apply the 2D-DCT to each block to convert the spatial representation into a frequency representation.

2.2 Quantization

Quantization reduces the precision of the transformed coefficients to achieve compression. The matrix of DCT coefficients is divided element-wise by a quantization matrix and then rounded. The JPEG standard offers pre-defined quantization matrices (e.g., Q50 for a quality level of 50) to balance between compression and visual quality.

2.3 Peak Signal-to-Noise Ratio (PSNR)

PSNR is used to measure the quality of the compressed image in relation to the original image. It is defined as:

$$\text{PSNR} = 20 \log_{10} \left(\frac{255}{\sigma_e} \right) \tag{1}$$

where σ_e is the mean squared error (MSE) between the original and reconstructed images.

3 Procedure

3.1 2D-DCT Process

The image compression process involves several steps, which are elaborated below:

3.1.1 Dividing the Image into 8x8 Blocks

The first step in the compression process is to divide the grayscale image into non-overlapping blocks of size 8×8 pixels. For an image of size $M \times N$, the total number of blocks will be:

$$\text{Number of blocks} = \left(\frac{M}{8} \right) \times \left(\frac{N}{8} \right)$$

Each block will be processed independently.

3.1.2 Leveling Off the Block

Each pixel value in the block is adjusted by subtracting 128 to shift the values from the range [0, 255] to the range [-128, 127]. This step is important as it helps to center the pixel values around zero, which improves the DCT's effectiveness. The adjusted pixel value can be expressed as:

$$f'(x,y) = f(x,y) - 128$$

where $f(x,y)$ is the original pixel value at position (x,y).

3.1.3 Applying 2D-DCT

The 2D Discrete Cosine Transform is applied to each 8×8 block using the following equation:

$$F(u,v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f'(x,y)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

Here, $C(u)$ and $C(v)$ are the normalization factors defined as:

$$C(u) = \begin{cases} \sqrt{\frac{1}{2}} & \text{if } u = 0 \\ 1 & \text{if } u \neq 0 \end{cases}$$

The resulting $F(u,v)$ is a frequency representation of the image block.

3.1.4 Quantization

Quantization is a crucial step where the DCT coefficients are divided by a quantization matrix and rounded to reduce their precision. The quantization matrix for quality level Q can be defined as:

$$Q(u,v) = \frac{Q_{default}(u,v)}{Q}$$

where $Q_{default}(u,v)$ is the default quantization matrix for JPEG and Q is a scaling factor that controls the quality level. For instance, the standard quantization matrix for quality level 50 (lower values indicate more aggressive quantization) is as follows:

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 110 & 100 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

The quantized DCT coefficients $QF(u,v)$ are then calculated as:

$$QF(u,v) = \text{round}\left(\frac{F(u,v)}{Q(u,v)}\right)$$

3.1.5 Reconstruction of the Image

To reconstruct the image, the quantized coefficients must be multiplied back by the quantization matrix and then transformed back into the spatial domain using the Inverse DCT (IDCT):

$$f'(x,y) = \sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)QF(u,v)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

After obtaining $f'(x, y)$, we add 128 back to center the pixel values:

$$f(x, y) = f'(x, y) + 128$$

This gives us the reconstructed pixel values of the image.

3.2 Quality Levels and Quantization Matrices

Different quality levels can be achieved by adjusting the quantization matrix $Q(u, v)$ using different scaling factors. The following are the scaling factors corresponding to three quality levels used in this report:

- **Quality Level 70:** $\tau = 0.6$
- **Quality Level 35:** $\tau = 1.428$ (more aggressive quantization)
- **Quality Level 15:** $\tau = 3.33$ (highest compression, lowest quality)

These scaling factors modify the quantization matrix accordingly, resulting in varying degrees of compression. The quantization process leads to a higher percentage of zeros in the quantized matrices, which is beneficial for compression.

4 Results and Observations

The results from the image compression task reveal significant variability in the subjective quality of compressed images, largely influenced by their content and characteristics. For images with intricate details, such as those containing fine textures or sharp contrasts, higher compression ratios resulted in noticeable artifacts, including blocking and blurring, leading to a degradation in visual quality. In contrast, images with larger uniform regions, such as landscapes or simple backgrounds, maintained acceptable quality even at higher compression levels. The presence of noise in certain images further exacerbated the compression challenges, as the algorithms struggled to preserve important details amidst random variations. Additionally, variations in the quantization matrix across different quality settings highlighted the trade-off between file size and visual fidelity; more aggressive quantization resulted in a higher percentage of zeros in the DCT coefficients but also led to a loss of crucial image features. Overall, these observations underscore the necessity of tailoring compression strategies to the specific characteristics of each image to achieve optimal results while minimizing quality loss.

4.1 Image 1: Barbara

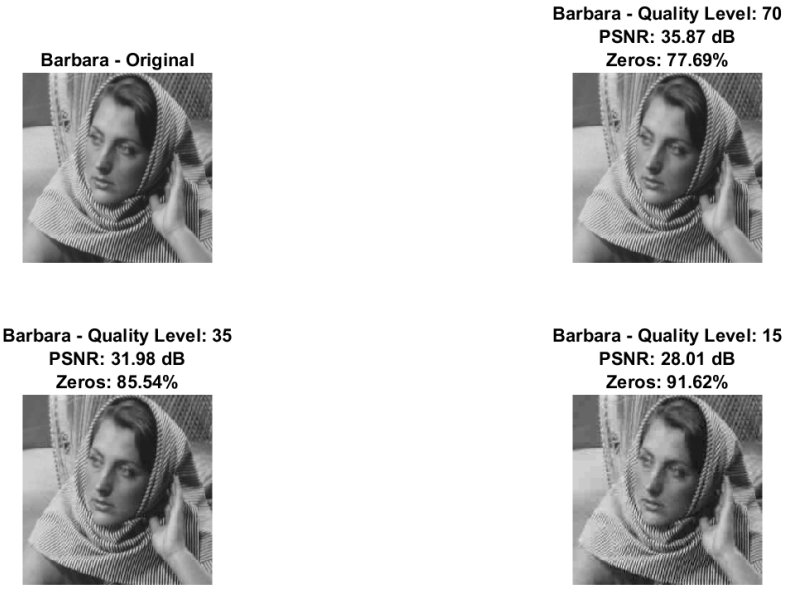

Figure 1: Barbara

Compressed Quality Level	**PSNR (dB)**	**Percentage of Zeros (%)**	**Observations**
70	35.87	77.69	The woman in the frame is visible with clear details and minimal distortion. The edges and textures of the image are preserved, offering a high-quality representation similar to the original.
35	31.98	85.54	The woman in the frame remains visible, but some noticeable blurring and minor artifacts are present. The image still retains acceptable quality.
15	28.01	91.62	The woman in the frame is still recognizable, but significant blurring and distortion have occurred. The image exhibits blockiness and severe loss of detail,resulting in reduced visual quality.

Table 1: Comparison of Compression Results of Barbara

4.2 Image 2: Parrot

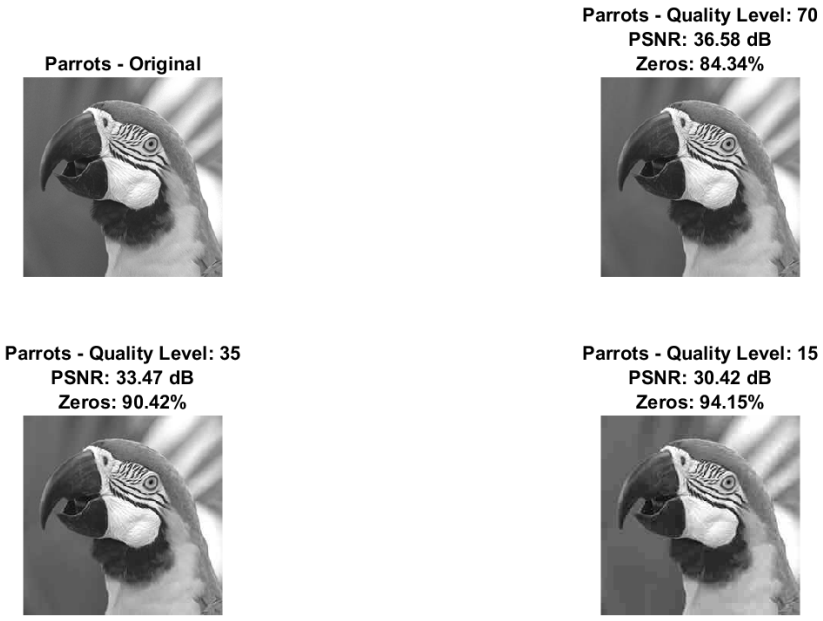

Figure 2: Enter Caption

Compressed Quality Level	**PSNR (dB)**	**Percentage of Zeros (%)**	**Observations**
70	36.58	84.34	The image retains a high level of visual quality, providing a close approximation to the original.. Edges of the pattern in eye are well-preserved.
35	33.47	90.42	The parrot remains distinguishable, but a little loss in clarity is noticeable. There is a slightly visible degradation in fine details compared to the higher quality level.
15	30.42	94.15	The parrot is still visible, but the image has lost much of its fine details. The background has become blurred, and the overall image quality is somewhat degraded, making it harder to distinguish detailed textures.

Table 2: Comparison of Compression Results of Barbara

4.3 Image 3: House

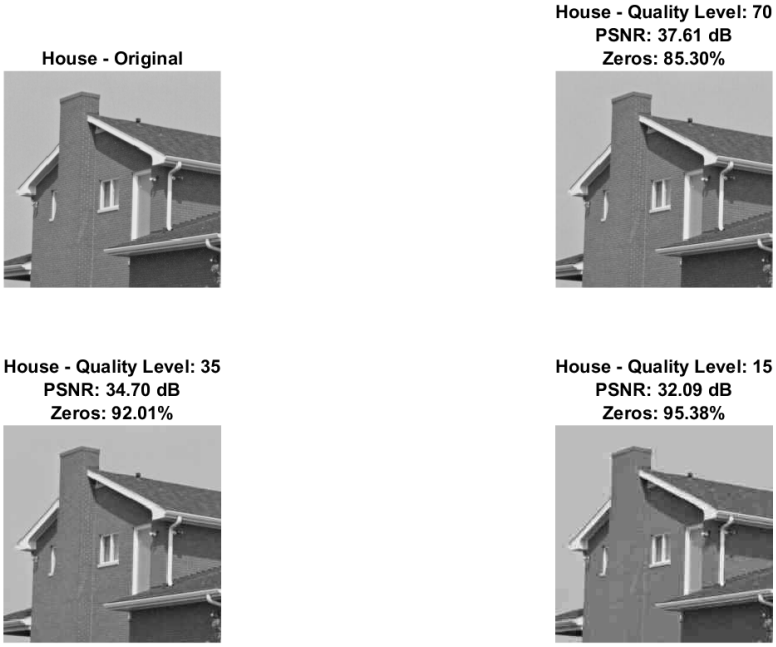

Figure 3: House

Compressed Quality Level	**PSNR (dB)**	**Percentage of Zeros (%)**	**Observations**
70	37.61	85.30	Fine textures, like the roof and window frames, are well-preserved. Only minimal artifacts are noticeable, maintaining a high visual quality close to the original image.
35	34.70	92.01	The house is still visible, but some blurring is visible on the window panes and edges of the roof. The overall sharpness has been slightly reduced, and moderate blocking artifacts are noticeable.
15	32.09	95.38	The house is visible but with significant loss of detail. The edges of the roof and windows are heavily blurred. Results in a heavily degraded image that lacks the original clarity.

Table 3: Comparison of Compression Results of Barbara

4.4 Image 4:Baboon(Chosen Image)

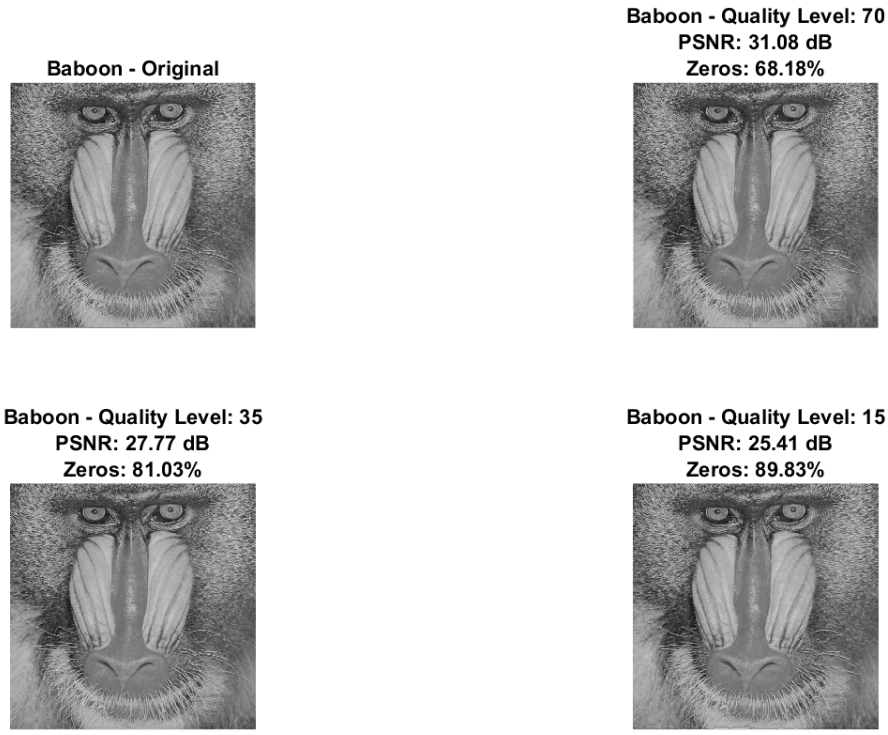

Figure 4: Baboon

Compressed Quality Level	**PSNR (dB)**	**Percentage of Zeros (%)**	**Observations**
70	31.08	68.18	The baboon in the frame is clearly visible, with details well-preserved at this quality level.
35	27.77	81.03	The baboon is still identifiable and maintains a good level of detail, though some minor artifacts may appear.
15	25.41	89.83	The baboon remains visible, but the quality has noticeably decreased, resulting in more compression artifacts.

Table 4: Comparison of Compression Results of Barbara

5 Analysis of Image Compression Difficulty

Different images present varying levels of difficulty when it comes to compression while maintaining visual quality. This analysis examines why certain images are more challenging to compress effectively than others based on their inherent characteristics and compression behavior.

5.1 Image Content Characteristics

5.1.1 High-Frequency Content

Images containing substantial high-frequency components present significant compression challenges:

- Abundance of fine details, textures, and sharp edges
- Require more DCT coefficients for accurate representation
- More susceptible to visible artifacts (blurring, blockiness) during quantization

5.1.2 Smooth Areas

Regions with low-frequency characteristics compress more efficiently:

- Gradual color transitions and minimal detail
- Representable with fewer DCT coefficients
- Better preservation of visual quality under compression

5.2 Analysis of Test Images

5.2.1 Barbara

This image represents one of the most challenging compression scenarios:

- Complex textures present in clothing and tablecloth
- Abundant fine details and patterns
- Significant visual degradation at lower quality levels
- Higher proportion of non-zero DCT coefficients

5.2.2 Parrots

Demonstrates moderate compression difficulty:

- Combination of smooth backgrounds and detailed regions
- Distinct color transitions in feather areas
- Maintains acceptable quality at medium compression levels
- Balanced distribution of frequency components

5.2.3 House

Generally more amenable to compression:

- Extensive regions of uniform or gradually varying color
- Limited fine texture content
- Superior visual quality preservation at higher compression ratios
- Predominantly low-frequency components

5.2.4 Baboon

Presents significant compression challenges:

- Intricate fur texture with high-frequency details
- Complex facial features and fine whiskers
- Marked degradation at higher compression ratios
- Dense high-frequency component distribution

5.3 Visual Quality and Compression Relationship

5.3.1 High Compression Scenarios

At significant compression ratios:

- Barbara and Baboon exhibit pronounced artifacts
- Substantial loss of texture detail
- Notable blocking artifacts in high-frequency regions

5.3.2 Moderate Compression Scenarios

At intermediate compression levels:

- House maintains superior visual integrity
- Parrots demonstrates quality variance between smooth and detailed areas

5.4 Technical Analysis

The DCT-based compression mechanism operates through:

1. Transformation of 8×8 blocks into frequency components
2. Progressive quantization emphasis on higher frequencies
3. Greater information loss in high-frequency content during quantization

This results in:

- Enhanced artifact visibility in complex textures
- Superior preservation of smooth regions
- Block-boundary artifacts at elevated compression ratios

5.5 Correlation with Quality Metrics

Images with predominant high-frequency content typically exhibit:

- Reduced PSNR values at equivalent quality settings
- Lower proportion of zero coefficients post-quantization
- More pronounced visual degradation despite comparable compression ratios

A MATLAB Code for 2D-DCT Image Compression

```
data1= load('C:\Users\risin\OneDrive - University of Moratuwa\Desktop\DSP-Assignment\ImageAssginment\Selected-Images\barbara.mat');
data3= load('C:\Users\risin\OneDrive - University of Moratuwa\Desktop\DSP-Assignment\ImageAssginment\Selected-Images\house.mat');
data2=load('C:\Users\risin\OneDrive - University of Moratuwa\Desktop\DSP-Assignment\ImageAssginment\Selected-Images\Parrots.mat');
extra_image = imread('C:\Users\risin\OneDrive - University of Moratuwa\Desktop\DSP-Assignment\ImageAssginment\Selected-Images\baboon.tif');

save('baboon.mat','extra_image');
data4=load('baboon.mat');
img1 = data1.A;
img2 = data2.B;
img3 = data3.C;
img4= data4.extra_image;

images = {
    'Barbara', img1;
    'Parrots', img2;
    'House', img3;
    'Baboon', img4
};

% Define quality levels
quality_levels = [70, 35, 15];  % Change based on the assigned quality levels

% Loop through each image and apply DCT-based compression
for img_idx = 1:size(images, 1)
    img_name = images{img_idx, 1};
    img_data = images{img_idx, 2};

    % Step 1: Apply DCT to 8x8 blocks
    [M, N] = size(img_data);
    dct_image = zeros(M, N);

    for i = 1:8:M
        for j = 1:8:N
            block = double(img_data(i:min(i+7, M), j:min(j+7, N))) - 128;  % Level off
            dct_block = dct2(block);  % Apply 2-D DCT
            dct_image(i:min(i+7, M), j:min(j+7, N)) = dct_block;
        end
    end

    figure('Position', [100, 100, 1400, 1000]);

    % Display the original image
    subplot(2, 2, 1);
    imshow(uint8(img_data));
    title(sprintf('%s - Original', img_name));

    for ql_idx = 1:length(quality_levels)
        quality_level = quality_levels(ql_idx);

        % Step 2: Quantization
        Q50 = [ 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 ];

        % adjust Q50 matrix for given quality level
```

```
        if quality_level > 50
            scale_factor = (100 - quality_level) / 50;
        else
            scale_factor = 50 / quality_level;
        end
        Q = round(Q50 * scale_factor);

        % Apply quantization
        compressed = zeros(size(dct_image));
        for i = 1:8:M
            for j = 1:8:N
                dct_block = dct_image(i:min(i+7, M), j:min(j+7, N));
                quant_block = round(dct_block ./ Q);  % Quantize
                compressed(i:min(i+7, M), j:min(j+7, N)) = quant_block;
            end
        end

        % Step 4: Decompression
        decompressed = zeros(size(compressed));
        for i = 1:8:M
            for j = 1:8:N
                quant_block = compressed(i:min(i+7, M), j:min(j+7, N));
                dequant_block = quant_block .* Q;  % Dequantize
                decompressed(i:min(i+7, M), j:min(j+7, N)) = idct2(dequant_block) +
                    128;  % Inverse DCT and add level-off
            end
        end

        % Step 5: Calculate PSNR
        mse = mean((double(img_data(:)) - decompressed(:)).^2);  % Mean Squared Error
        max_intensity = 255;
        psnr_val = 20 * log10(max_intensity / sqrt(mse));
        zero_percentage = sum(compressed(:) == 0) / numel(compressed) * 100;

        % Show compressed image
        subplot(2, 2, ql_idx + 1);
        imshow(uint8(decompressed));
        title(sprintf('%s - Quality Level: %d\\nPSNR: %.2f dB\\nZeros: %.2f%%',
            img_name, quality_level, psnr_val, zero_percentage));
    end


    sgtitle(sprintf('%s - Compression Results', img_name));
end
```