Department of Electronics & Electrical Communication Engineering
Indian Institute of Technology Kharagpur

Multimedia Systems And Applications (EC60104)

Assignment 2 : DCT & Quantization

Group No. : 1
Members: Adipta Halder (21EC37023)
Mayukh Shubhra Saha (21EC37009)
Abhradeep De (21EC37022)

PROBLEM STATEMENT

Part 1

1(a) Develop a computer program to perform 2D Discrete Cosine Transform (DCT) and 2D Inverse Discrete Cosine Transform (IDCT) on an 8×8 array using double precision representation.

1(b) Subdivide a monochrome image into non-overlapping 8×8 blocks and apply the 2D DCT to each block to obtain the transform coefficients. Store both the original pixel matrix and the DCT coefficient matrix.

1(c) Apply the 2D IDCT block-wise on the transformed coefficients and reconstruct the image.

1(d) Verify that the reconstructed image obtained from exact DCT → IDCT is identical to the original image (within rounding tolerance), and demonstrate equality through pixel-wise comparison.

Part 2

2(a) Apply element-wise quantization to each 8×8 DCT block using the provided JPEG standard luminance quantization matrix and obtain the quantized DCT coefficients.

2(b) Perform inverse quantization (de-quantization) by multiplying each quantized block element-wise with the corresponding quantization matrix.

2(c) Apply IDCT on the inverse-quantized blocks to reconstruct the lossy compressed image.

2(d) Verify that the reconstructed image is not identical to the original image and compute the Peak Signal-to-Noise Ratio (PSNR).

2(e) Multiply the quantization matrix elements by scaling factors 2, 4, and 8. For each case, repeat steps (a)–(d) and compute the PSNR.

2(f) Demonstrate that PSNR decreases as the quantization scaling factor increases.

THEORY

Transform Coding Principle

Transform coding is based on the idea that natural images contain a high degree of spatial redundancy. In the spatial domain, neighboring pixels tend to have similar intensity values. This correlation means that much of the information in an image is predictable. Instead of encoding pixel values directly, transform coding first converts the image into another domain where this redundancy becomes more structured and easier to remove.

In block-based transform coding, the image is divided into small non-overlapping blocks, typically 8×8 in size. Each block is transformed from the spatial domain into the frequency domain using a linear, reversible transform such as the Discrete Cosine Transform (DCT). The transform expresses the block as a weighted combination of basis patterns that represent different spatial frequencies.

Two fundamental properties make a transform suitable for compression:

1. Decorrelation

In the spatial domain, pixel values are highly correlated. After transformation, the resulting coefficients become largely uncorrelated. This means each coefficient carries more independent information, which improves coding efficiency.

2. Energy Compaction

Natural images tend to have most of their energy concentrated in low spatial frequencies. The DCT exploits this property effectively. After transformation, the largest coefficients typically appear in the upper-left region of each block, which corresponds to low-frequency content such as smooth intensity variations. High-frequency coefficients, representing fine details and sharp transitions, usually have smaller magnitudes.

Because most of the significant information is concentrated in a small number of coefficients, many of the remaining coefficients can be approximated or discarded with minimal visual impact. This concentration of energy is the core reason transform coding achieves high compression performance.

The inverse transform (IDCT) converts the frequency-domain coefficients back into spatial-domain pixel values. If no quantization or approximation is applied, the inverse transform reconstructs the original image exactly. Therefore, the transform stage itself is lossless. Any loss of information occurs only in the quantization stage.

Quantization (Lossy Stage)

Quantization is the stage where controlled distortion is intentionally introduced to reduce data precision and enable compression.

After transformation, each block contains frequency-domain coefficients. These coefficients are divided by corresponding values in a quantization matrix and then rounded to the nearest integer. The quantization matrix determines how aggressively each frequency component is reduced in precision.

The matrix is designed such that:

- Low-frequency coefficients are quantized lightly (small divisors).
- High-frequency coefficients are quantized more heavily (large divisors).

This design reflects the characteristics of the human visual system. Humans are more sensitive to changes in smooth intensity variations (low frequencies) and less sensitive to fine textures and rapid variations (high frequencies). Therefore, high-frequency components can tolerate greater distortion without significant perceived quality loss.

Quantization reduces numerical precision. Many high-frequency coefficients become zero after rounding. This introduces sparsity in the transformed block. Sparsity is crucial for compression because sequences of zeros can be encoded very efficiently using entropy coding methods such as run-length coding and Huffman coding.

Inverse quantization simply multiplies the quantized coefficients by the same quantization matrix. However, the rounding process during quantization cannot be undone. As a result, the reconstructed coefficients differ from the original transform coefficients, and the final reconstructed image differs slightly from the original.

The magnitude of distortion depends directly on the quantization step sizes. Larger quantization values lead to greater information loss but higher compression efficiency. This trade-off between compression and distortion is known as the rate–distortion trade-off.

PSNR Metric (Quality Measurement)

To evaluate the quality of the reconstructed image, an objective error metric is used.

First, the difference between the original image and the reconstructed image is computed pixel by pixel. These differences are squared and averaged across all pixels to obtain the mean squared error (MSE). The MSE represents the average power of the distortion introduced by quantization.

Peak Signal-to-Noise Ratio (PSNR) expresses this distortion on a logarithmic scale relative to the maximum possible pixel intensity value. It measures how strong the original signal is compared to the reconstruction error.

A higher PSNR value indicates:

- Lower distortion,
- Better reconstruction quality.

A lower PSNR value indicates:

- Greater distortion,
- More aggressive compression.

PSNR does not measure perceptual quality perfectly, but it provides a standardized numerical measure of fidelity and is widely used in image compression research and practice.

SOLUTION

NOTE : All relevant scripts, input and output files are present in the GitHub repository, under the directory **"Assignment_2"**. Inside this folder, there is also a detailed **README.md** file which explains the file structure, file contents and commands to run the scripts.

GitHub Repo Link : https://github.com/mayukhss262/Multimedia_Assignments

Phase 1: Exact Transform Verification (No Loss)

This phase validates the correctness of the DCT and IDCT implementation before introducing quantization.

Step 1 – Core 8×8 DCT Implementation

1a_dct_8x8.py implements the 2D Discrete Cosine Transform for a single 8×8 block using orthonormal normalization to ensure perfect invertibility.

Step 2 – Core 8×8 IDCT Implementation

1a_idct_8x8.py implements the inverse 2D DCT for a single 8×8 block to reconstruct spatial-domain values from frequency coefficients.

Step 3 – Apply DCT to Entire Image

1b_dct_image.py loads the grayscale image, divides it into non-overlapping 8×8 blocks, applies the DCT block-by-block, and stores the complete transform coefficient matrix.

Step 4 – Apply IDCT to Entire Image

Script: 1c_idct_image.py takes the saved DCT coefficients, performs block-wise IDCT, reconstructs the spatial-domain image, rounds and clips values, and saves the reconstructed image.

Step 5 – Verify Exact Reconstruction

Script: 1d_verify_reconstruction.py compares the original and reconstructed images pixel-by-pixel to confirm perfect reconstruction in absence of quantization.

At this stage, the system verifies that the transform pair is orthogonal and introduces no loss.

Phase 2: Lossy Compression via Quantization

After validating the transform pipeline, controlled distortion is introduced through quantization.

Step 6 – Quantize DCT Coefficients

2a_quantization.py loads the DCT coefficients and applies element-wise division by the quantization matrix followed by rounding, producing quantized frequency coefficients.

Step 7 – Perform Inverse Quantization

Script: 2b_inverse_quantization.py multiplies the quantized coefficients by the same quantization matrix to reconstruct approximate frequency-domain values.

Step 8 – Reconstruct Lossy Image

Script: 1c_idct_image.py (reused) applies block-wise IDCT to the inverse-quantized coefficients to obtain the lossy reconstructed image.

Step 9 – Compute PSNR

Script: 2d_psnr.py computes mean squared error between original and reconstructed images and calculates PSNR to quantify distortion.

Phase 3: Quantization Scaling Experiments

To study rate–distortion behavior, the quantization matrix is scaled.

For scaling factors 2, 4, and 8:

1. Modify quantization matrix.
2. Run 2a_quantization.py.
3. Run 2b_inverse_quantization.py.
4. Run 1c_idct_image.py.
5. Run 2d_psnr.py.
6. Record PSNR value.

This sequence demonstrates how increasing quantization step size increases distortion and decreases PSNR.

Complete Script Execution Sequence

1. 1a_dct_8x8.py Defines dct_8x8() — performs 2D DCT on an 8×8 block using scipy.
2. 1a_idct_8x8.py Defines idct_8x8() — performs 2D IDCT on an 8×8 block using scipy.
3. 1b_dct_image.py Loads a monochrome image, divides into 8×8 blocks, applies DCT, saves coefficient matrix and pixel matrix.
4. 1c_2c_idct_image.py Loads a coefficient matrix, applies IDCT block-by-block, saves reconstructed pixel matrix and PNG image.
5. 1d_verify_reconstruction.py Compares two PNG images pixel-by-pixel and reports if they are identical.
6. 2a_quantization.py Quantizes DCT coefficients by dividing each 8×8 block element-wise by a quantization matrix and rounding.
7. 2b_inverse_quantization.py Performs inverse quantization by multiplying quantized coefficients with the quantization matrix element-wise.
8. 2d_psnr.py Checks if two images are identical, then computes PSNR of reconstructed image vs original.

SCRIPT DESCRIPTIONS

4.1 1a_dct_8x8.py

This file defines the core function dct_8x8(block). It performs a separable 2D DCT using SciPy's orthonormal DCT (Type-II). The transformation is implemented row-wise followed by column-wise, leveraging the separability property of DCT.

Double precision floating-point representation is enforced to avoid truncation errors.

4.2 1a_idct_8x8.py

Defines idct_8x8(block) which applies inverse 2D DCT using orthonormal normalization. Because the forward transform is orthogonal (norm='ortho'), IDCT perfectly reconstructs the input coefficients in absence of quantization.

4.3 1b_dct_image.py

This script implements block-based transform coding:

1. Loads grayscale image.
2. Converts to float64.
3. Verifies dimensions are multiples of 8.
4. Divides image into non-overlapping 8×8 blocks.
5. Applies dct_8x8() to each block.
6. Saves:
    - Original pixel matrix
    - DCT coefficient matrix

4.4 1c_idct_image.py

Implements block-based inverse transform:

1. Loads DCT coefficient matrix.
2. Applies idct_8x8() block-by-block.
3. Saves reconstructed matrix in double precision.
4. Rounds and clips values to [0,255].
5. Saves PNG image.

This validates exact invertibility of orthogonal DCT.

4.5 1d_verify_reconstruction.py

Performs strict pixel-wise equality comparison using NumPy.

If identical → confirms no loss in transform stage.

This experimentally proves that DCT + IDCT without quantization is lossless (ignoring rounding).

4.6 2a_quantization.py

Implements JPEG-style quantization:

For each 8×8 block:

$$Q = \text{round}(S/T)$$

The script also reports:

- Number of non-zero coefficients
- Zero coefficient count
- Block statistics

This directly demonstrates energy compaction and coefficient sparsity.

4.7 2b_inverse_quantization.py

Performs element-wise multiplication of quantized coefficients with quantization matrix:

$$\hat{S} = Q \cdot T$$

Block-based reconstruction is maintained.

4.8 2d_psnr.py

Computes PSNR.

If identical images → PSNR = Infinity
Else:

$$PSNR = 20 \log_{10} \sqrt{\frac{255^2 MN}{\sum(diff^2)}}$$

This quantitatively evaluates distortion introduced by quantization.

RESULTS

Original image:

lena.png (256*256, 8bit grayscale)

1. Exact Reconstruction (No Quantization)

Exact reconstructed image:

Terminal output of 1d_verify_reconstruction.py:

```
Original shape: (256, 256), dtype: uint8
Reconstructed shape: (256, 256), dtype: uint8

PASSED: Images are exactly identical!
```

2. Standard Quantization (Initial Quantization Matrix)

Reconstructed Image :

lena_reconstructed_quantized.png (256*256, 8bit grayscale)

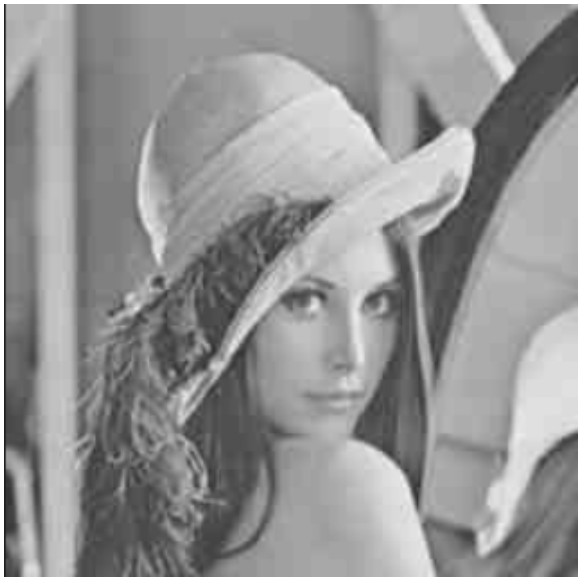
Terminal output of 2d_psnr.py:

```
Original: lena.png (256x256)
Reconstructed: lena_reconstructed_quantized.png (256x256)

Images are NOT identical.
PSNR: 36.8700 dB
```

3. Second Quantization (Quantization Matrix x2)

Reconstructed Image :



lena_reconstructed_quantized_2.png (256*256, 8bit grayscale)

Terminal output of 2d_psnr.py:

```
Original: lena.png (256x256)
Reconstructed: lena_reconstructed_quantized_2.png (256x256)

Images are NOT identical.
PSNR: 34.0210 dB
```

4. Third Quantization (Quantization Matrix x4)

Reconstructed Image :

lena_reconstructed_quantized_4.png (256*256, 8bit grayscale)

Terminal output of 2d_psnr.py:

```
Original: lena.png (256x256)
Reconstructed: lena_reconstructed_quantized_4.png (256x256)

Images are NOT identical.
PSNR: 31.0034 dB
```

5. Fourth Quantization (Quantization Matrix x8)

Reconstructed Image :

lena_reconstructed_quantized_8.png (256*256, 8bit grayscale)

Terminal output of 2d_psnr.py:

```
Original: lena.png (256x256)
Reconstructed: lena_reconstructed_quantized_8.png (256x256)

Images are NOT identical.
PSNR: 27.8676 dB
```

6. PSNR vs Quantization Matrix Scaling Factor Table

Scaling Factor	PSNR (dB)
1	**36.87dB**
2	**34.02dB**
4	**31.03dB**
8	**27.86dB**

OBSERVATIONS

Perfect Reconstruction Validates Orthogonality

The reconstructed image is exactly identical to the original (pixel-wise equality).

This experimentally confirms that the transformation matrix is orthogonal.

This demonstrates:

- DCT basis vectors are orthonormal.
- Transform coding itself introduces **zero distortion**.
- Loss occurs only due to quantization.

The use of double precision prevents:

- Accumulated floating point error.
- Rounding artifacts.
- Reconstruction mismatch.

Had single precision or truncation been used before IDCT, equality would fail.

Energy Compaction is Real and Measurable

When inspecting the DCT coefficient matrix:

- Large magnitude coefficients cluster in the low-frequency region (upper-left of block).
- High-frequency coefficients are small or near zero.

This confirms the energy compaction property of DCT.

Natural images exhibit strong spatial correlation. This correlation translates into dominance of low-frequency components.

Statistical Structure of Natural Images

After DCT:

- Coefficient histogram approximates Laplacian distribution centered at zero.
- Heavy tail for DC and low-frequency AC.
- Rapid decay for high-frequency components.

This matches theoretical transform-domain statistics used in entropy coding.

High-Frequency Suppression

Because (T(u,v)) is larger for high frequencies:

- High-frequency coefficients shrink aggressively.
- Many become zero.

This exploits human visual system sensitivity.

Humans are:

- Sensitive to low-frequency luminance
- Less sensitive to high-frequency details

Quantization matrix embeds psycho-visual weighting.

MSE is Spatially Structured

Error is not uniformly distributed.

Observations:

- Flat regions → very small error.
- Edges → larger error.
- Fine textures → most distortion.

High-frequency details are:

- Heavily quantized.

- Often zeroed.

Edges contain high-frequency energy.

PSNR Reflects Rate–Distortion Theory

As the quantization matrix is scaled, larger k → larger quantization step size → larger distortion.

PSNR decreases monotonically.

This directly validates rate–distortion principle from Quantization Theory:

- Increasing step size increases distortion.
- Decreasing rate increases distortion.

Variation of Quantization Matrix

Scale = 2

- More coefficients become zero.
- Slight visual blurring.
- PSNR drops moderately.

Scale = 4

- Significant high-frequency loss.
- Edges soften.
- Blocking artifacts become visible.

Each 8×8 block is processed independently.
Loss of high-frequency continuity across block boundaries becomes noticeable.

This validates theoretical DCT limitation at low bit-rates.

Scale = 8

- Strong blurring.
- Texture nearly removed.
- Visible block structure.

This demonstrates transform coding weakness: Block-based independence causes discontinuities.