Practical Work 10
Applying Compression Algorithms to Two-Dimensional Signals (Images)

1. Objectives of this Laboratory Work

- Understand the concept and importance of image compression.
- Learn about lossless and lossy compression techniques.
- Apply common compression algorithms (JPEG, PNG, and Run-Length Encoding).
- Evaluate the effect of compression on image quality and file size.
- Compare results of different compression ratios and formats.

2. Theoretical Background

Image compression is a fundamental process in digital image processing aimed at reducing the size of image files without significantly degrading their visual quality. Since images contain large amounts of redundant or irrelevant data, compression techniques help to optimize storage space and transmission bandwidth, which is crucial for modern multimedia, web, and mobile applications.

A **two-dimensional signal** such as an image can be represented as a matrix of intensity values f(x,y)f(x, y)f(x,y). Compression aims to represent this matrix more efficiently by minimizing data redundancy and irrelevancy. There are two main types of compression: **lossless** and **lossy**.

Lossless Compression

Lossless compression techniques reduce file size without losing any image information. When decompressed, the original image can be perfectly reconstructed. Common lossless methods include:

- **Run-Length Encoding (RLE):** Replaces sequences of identical pixels with a single value and a count.
- **Huffman Coding:** Assigns shorter binary codes to frequent pixel values.

- **LZW (Lempel–Ziv–Welch):** Used in formats like GIF and PNG, builds a dictionary of repeating patterns.

Lossless compression is useful for applications that require exact reproduction, such as medical imaging, technical drawings, and text documents.

Lossy Compression

Lossy compression reduces file size by removing information that is less perceptible to the human eye. This type of compression achieves much higher compression ratios but may cause slight quality degradation. Common lossy algorithms include:

- **JPEG (Joint Photographic Experts Group):** Uses the Discrete Cosine Transform (DCT) to represent image data in the frequency domain and quantizes it for storage.
- **JPEG2000:** Based on wavelet transform, providing better quality at high compression ratios.
- **WebP:** A modern format combining predictive coding and transform compression for the web.

Lossy compression is commonly used for photographs, videos, and web applications where minor loss of detail is acceptable.

Image Compression Workflow

The general steps in image compression are:

1. **Transformation:** Convert the image into another domain (spatial → frequency).
2. **Quantization:** Reduce precision of less significant coefficients.
3. **Encoding:** Use entropy coding (Huffman, arithmetic) to represent data efficiently.
4. **Decoding:** Reverse the process to reconstruct the image.

Compression efficiency is usually evaluated by metrics such as:

- **Compression Ratio (CR):**

$$CR = \frac{\text{Uncompressed Size}}{\text{Compressed Size}}$$

- **Peak Signal-to-Noise Ratio (PSNR):** Measures reconstruction quality.
- **Mean Squared Error (MSE):** Quantifies pixel-level differences between original and decompressed images.

Comparison of Compression Techniques

Compression Type	Method	Reversibility	Typical Use	Advantage	Limitation
Lossless	RLE, PNG	Perfect reconstruction	Text, diagrams	No quality loss	Lower compression ratio
Lossy	JPEG, WebP	Approximate reconstruction	Photos, videos	High compression ratio	Quality loss possible
Transform-based	DCT, DWT	Depends on quantization	Multimedia	Efficient frequency representation	Requires more computation

Applications of Image Compression

- **Medical Imaging:** Reduces storage for large scans.
- **Web Optimization:** Accelerates page loading times.
- **Satellite and Remote Sensing:** Efficient transmission of high-resolution images.
- **Surveillance Systems:** Stores continuous camera feeds efficiently.
- **Machine Learning:** Reduces dataset storage without major loss in visual features.

3. Practical Section: Applying Image Compression in Python

Task 1: Comparing JPEG and PNG Compression

Objective:

Observe file size differences between lossless (PNG) and lossy (JPEG) compression formats.

Instructions:

1. Load a color image.
2. Save it in both JPEG and PNG formats.
3. Compare their file sizes and visual quality.

Code Example:

```
import cv2
import os
import matplotlib.pyplot as plt

# Load image
img = cv2.imread('/content/sample_data/lena.png')
cv2.imwrite('image_png.png', img, [cv2.IMWRITE_PNG_COMPRESSION, 3])
cv2.imwrite('image_jpeg.jpg', img, [cv2.IMWRITE_JPEG_QUALITY, 50])

# Get file sizes
png_size = os.path.getsize('image_png.png') / 1024
jpeg_size = os.path.getsize('image_jpeg.jpg') / 1024

print(f"PNG File Size: {png_size:.2f} KB")
print(f"JPEG File Size: {jpeg_size:.2f} KB")

# Display images
plt.figure(figsize=(10,4))
plt.subplot(1,2,1), plt.imshow(cv2.cvtColor(cv2.imread('image_png.png'),
cv2.COLOR_BGR2RGB)), plt.title("PNG (Lossless)")
```

```
plt.subplot(1,2,2), plt.imshow(cv2.cvtColor(cv2.imread('image_jpeg.jpg'),
cv2.COLOR_BGR2RGB)), plt.title("JPEG (Lossy)")
plt.show()
```

Expected Outcome:

JPEG file will be significantly smaller, but with slight loss in image quality.

Task 2: Implementing Run-Length Encoding (RLE) for Grayscale Image

Objective:

Apply basic lossless compression using RLE.

Instructions:

1. Convert the image to grayscale.
2. Flatten it into a one-dimensional array.
3. Compress using RLE and then reconstruct it.

Code Example:

```
import numpy as np
import cv2

def rle_encode(data):
    encoding = []
    prev = data[0]
    count = 1
    for pixel in data[1:]:
        if pixel == prev:
            count += 1
        else:
            encoding.append((prev, count))
            prev = pixel
            count = 1
    encoding.append((prev, count))
    return encoding
```

```
def rle_decode(encoding):
    decoded = []
    for value, count in encoding:
        decoded.extend([value] * count)
    return np.array(decoded, dtype=np.uint8)

# Load grayscale image
img = cv2.imread('/content/sample_data/lena.png', cv2.IMREAD_GRAYSCALE)
flat = img.flatten()

encoded = rle_encode(flat)
decoded = rle_decode(encoded).reshape(img.shape)

print(f"Original size: {flat.size}")
print(f"Compressed size: {len(encoded)}")

cv2.imshow("Original", img)
cv2.imshow("Reconstructed", decoded)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Expected Outcome:

RLE effectively compresses areas with uniform pixel values (e.g., backgrounds), but not detailed or noisy images.

Task 3: Evaluating Compression Quality

Objective:

Compute PSNR and MSE to quantify compression quality.

Code Example:

```
import numpy as np
```

```
import cv2

def mse(img1, img2):
    return np.mean((img1.astype("float") - img2.astype("float")) ** 2)

def psnr(img1, img2):
    mse_val = mse(img1, img2)
    if mse_val == 0:
        return 100
    max_pixel = 255.0
    return 20 * np.log10(max_pixel / np.sqrt(mse_val))

# Load original and JPEG images
original = cv2.imread('image_png.png')
compressed = cv2.imread('image_jpeg.jpg')

print("MSE:", mse(original, compressed))
print("PSNR:", psnr(original, compressed))
```

Expected Outcome:

Lower MSE and higher PSNR indicate better compression quality.

Assignments for Students

Assignment Task 1 — Lossless Compression using PNG and RLE

Compress the same image using PNG and RLE methods. Compare file sizes and reconstructed quality.

Assignment Task 2 — Lossy Compression using JPEG

Compress the image with different JPEG quality levels (100, 70, 50, 20). Measure PSNR for each.

Assignment Task 3 — Hybrid Approach

First apply a smoothing filter (Gaussian or median) before JPEG compression. Evaluate how preprocessing affects file size and quality.

Assignment Task 4 — Compression Efficiency Analysis

Create a table comparing formats (BMP, PNG, JPEG, WebP) in terms of file size, compression ratio, and PSNR.

Control Questions

1. What is the main purpose of image compression?
2. Explain the difference between lossless and lossy compression.
3. What type of images benefit most from lossless compression?
4. How does JPEG compression reduce data size?
5. What is the function of the quantization step in JPEG?
6. Define and interpret the PSNR metric.
7. What does a high MSE value indicate about image quality?
8. Why does RLE perform poorly on highly detailed images?
9. How does DCT help in transforming image data for compression?
10. What is the main advantage of using frequency-domain compression?
11. Compare PNG and JPEG in terms of visual quality and file size.
12. How can pre-filtering improve compression results?
13. Describe how Huffman coding contributes to compression.
14. Mention two real-life applications where image compression is crucial.