Image and Video Compression and Network Communication - Final Project Implementation

——12432735 陈秋明

Overview

This project implements a simple image compression and communication system using wavelet transforms, quantization, entropy coding, and network transmission. Below is the detailed implementation.

Encoder Implementation

1. Image Reading

- Use OpenCV to read a 512×512 grayscale image
- Verify image dimensions and convert to grayscale if needed
- Turn data type to np.float64 to insure accuracy when doing DWT

2. 5-level DWT Decomposition

- Implement the (5,3) wavelet transform `dwt_53_forward()`
- Perform 5-level decomposition
- Structure coefficients in a pyramid hierarchy `[cA_n, (cH_n, cV_n, cD_n), ... (cH_1, cV_1, cD_1)]`

3. Quantization

- Apply uniform quantization with step size q

4. Prediction for LL Subband

- Use simple prediction (e.g., previous pixel in raster order)
- Encode prediction residuals instead of direct values

5. Scanning and Symbol Generation

- **Low-frequency subband**: Raster scan (left-right, top-bottom)
- **High-frequency subbands**: Zero-tree scan with EZT symbols

- Implement zero-tree structure (ZTR can only be found on the deepest level)
- Generate symbols: IZ (isolated zero), ZTR (zero-tree root)
- For non-zeros: (size, amplitude) representation

6. Huffman Coding

- Build frequency tables for some of the symbols: `['ZTR', 'IZ', 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`
- Generate optimal Huffman codes for each symbol type
- Encode the 1-D symbol sequence
- Write Huffman codes and encoded data into file

7. Bitrate

- Get total bits used for compressed representation
- Compute bitrate R(q) in bits per pixel (bpp)

Network Communication Module

8. Socket Implementation

- Use Python's socket library for TCP communication
- Implement error handling for network transmission

Decoder Implementation

9. Huffman Decoding

- Read and parse Huffman code tables from bitstream
- Decode symbol sequence using the tables
- Reconstruct the 1-D symbol sequence

10. Inverse Scanning

- Reconstruct subbands from the 1-D sequence:

- LL subband from raster scan
- High-frequency subbands from zero-tree scan

- Rebuild wavelet coefficient pyramid structure

11. Inverse Quantization

- Multiply quantization indices by step size q

12. Inverse DWT

- Apply inverse (5,3) wavelet transform
- Reconstruct image from the 5-level subband decomposition
- Scale values to original bit depth (np.unit8)

13. PSNR Calculation

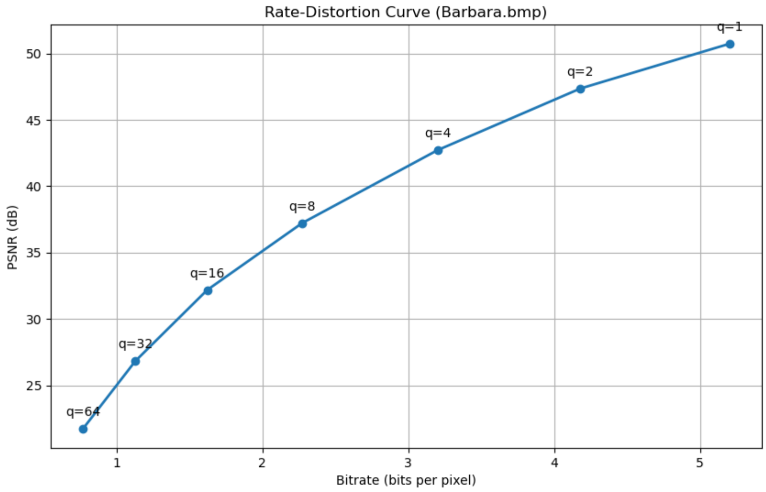
- Compute MSE between original and reconstructed image
- Calculate PSNR = $10 \cdot \log10(MAX^2/MSE)$, where MAX is maximum pixel value

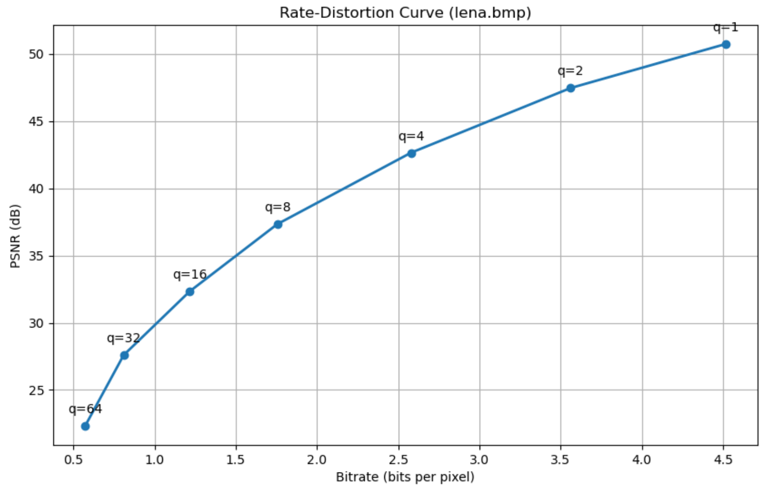
14. Rate-Distortion Analysis (in plot_rd_curve.py)

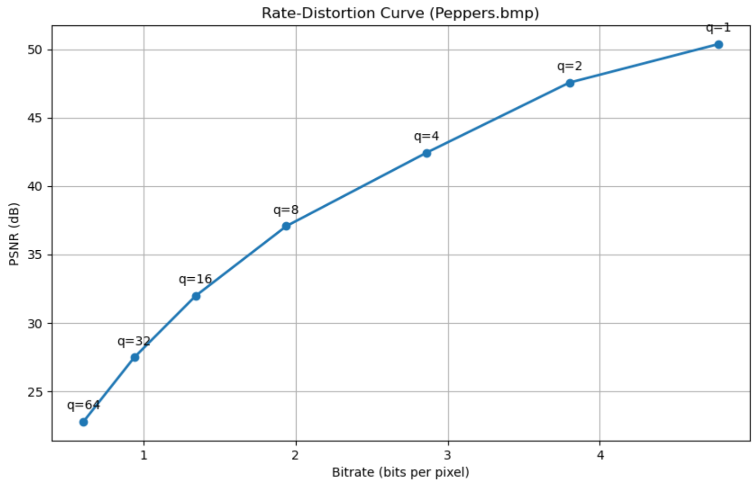
- Test with multiple quantization step sizes (e.g., q = 1, 2, 4, 8, 16, 32, 64)
- Plot R-D curve with bitrate R(q) vs PSNR D(q)
- Evaluate on 3 standard test images (Lena, Barbara, Peppers)

Figures：

1. R-D curves of the three test images:







It can be seen that smaller quantization steps require more bitrate, but gets higher PSNR (better image quality).

2. Comparison of original and reconstructed images:

Quantization step size = 8.0

Quantization step size = 32.0