End Term Project Report: Vector Quantization for Image Compression

[Mohamed Darwish]
[202201273]

Contents

Abstract

This report presents the implementation and results of the End Term Project for the Information Theory course, focusing on vector quantization for image compression. The project involves compressing color images from three categoriesnature, faces, and animalsusing vector quantization in both RGB and YUV color spaces. The implementation uses a block size of 2x2 pixels and a codebook size of 256, with 30 training images (10 per category) to generate codebooks and 15 test images (5 per category) to evaluate performance. The main requirements include generating separate codebooks for RGB components, calculating compression ratios, and comparing the quality of original and reconstructed images using Peak Signal-to-Noise Ratio (PSNR). Additionally, the bonus requirements are fulfilled by implementing YUV compression with 50% subsampling of U and V channels, followed by up-sampling and conversion back to RGB. The results demonstrate effective compression with varying PSNR values across categories, with RGB generally outperforming YUV in quality but YUV achieving better compression ratios due to subsampling.

1 Introduction

The objective of this project is to implement vector quantization for image compression as part of the End Term Project for the Information Theory course. Vector quantization is a lossy compression technique that maps blocks of pixels to a smaller set of representative vectors (codewords) stored in a codebook, achieving high compression ratios at the cost of some image quality. This project processes color images in three categoriesnature, faces, and animalsusing both RGB and YUV color spaces. The main requirements involve collecting 45 images (15 per category), using 30 images for training (10 per category) to generate codebooks, and 15 images for testing (5 per category). The block size is set to 2x2 pixels, and the codebook size is 256 vectors. The project evaluates compression performance by calculating the compression ratio (without codebook overhead) and the quality of reconstructed images using PSNR. The bonus section extends the implementation to YUV color space with 50% subsampling of U and V channels, followed by up-sampling and comparison with RGB results.

This report details the project structure, methodology, test cases, results, and conclusions, providing a comprehensive overview of the implementation and its performance.

2 Project and File Structure

The project is organized to ensure modularity and ease of execution. The root directory, named End od semster project, contains the following structure:

- **bin/**: Directory containing compiled Java class files.
- **src/**: Directory containing the source code files:
  - VectorQuantizationMain.java: The main class that orchestrates the vector quantization process, including user input, image processing, codebook generation, and testing.
  - VectorQuantizer.java: A utility class with methods for image processing, vector quantization, codebook generation using K-means clustering, and image reconstruction.
- **dataset/**: Directory containing the image dataset:
  - train/: Training images, divided into subdirectories:
    - nature/: Contains nature_1.jpg to nature_10.jpg.
    - faces/: Contains face_1.jpg to face_10.jpg.
    - animals/: Contains animal_1.jpg to animal_10.jpg.
  - test/: Test images, divided into subdirectories:
    - nature/: Contains nature_11.jpg to nature_15.jpg.
    - faces/: Contains face_11.jpg to face_15.jpg.
    - animals/: Contains animal_11.jpg to animal_15.jpg.
- **output_rgb/**: Directory where RGB-compressed images are saved (e.g., nature_compressed_img1
- **output_yuv/**: Directory where YUV-compressed images are saved (e.g., nature_compressed_img1

The project is implemented in Java, using the JDK 11.0.16.1 environment. The source code is compiled into the bin/ directory and executed using a PowerShell command with an 8GB heap size to handle memory-intensive image processing.

3 Methodology

The vector quantization process is implemented as follows:

1. **Dataset Preparation**: 45 color images are collected from the internet, with 15 images per category (nature, faces, animals). These are split into 30 training images (10 per category) and 15 test images (5 per category).
2. **User Input**: The program prompts the user for the block size (set to 2) and codebook size (set to 256).
3. **Training Phase**:
   - Images are resized to a maximum dimension of 1024 pixels to manage memory usage.
   - Images are cropped to ensure dimensions are divisible by the block size (2x2).
   - For RGB mode, each image is split into R, G, and B channels, and 2x2 blocks are extracted to form training vectors.
   - For YUV mode, images are converted to YUV color space, with U and V channels subsampled to 50% of their original width and height. Blocks are then extracted.
   - K-means clustering (with a maximum of 20 iterations) is applied to generate one codebook per channel (R, G, B for RGB; Y, U, V for YUV), each containing 256 codewords.
4. **Testing Phase**:
   - Test images undergo the same resizing and cropping preprocessing.
   - In RGB mode, images are quantized using the RGB codebooks, reconstructed, and saved to output_rgb/.
   - In YUV mode, images are quantized using the YUV codebooks, U and V channels are up-sampled, converted back to RGB, and saved to output_yuv/.
   - Performance metrics are calculated: PSNR for quality, compression ratio (without codebook overhead), and processing time.
5. **Evaluation**: Results are summarized, comparing RGB and YUV modes based on PSNR, compression ratio, and processing time.

4 Test Cases

The project includes test cases for both training and testing phases across all categories. Below is a detailed account of the test cases executed:

4.1 Training Phase

The training phase processes 10 images per category to generate codebooks. The images are resized and cropped as needed to align with the 2x2 block size.

- **Category: Nature**
  - nature_1.jpg: Resized from 4000x6000 to 682x1024.
  - nature_2.jpg: Resized from 6000x4000 to 1024x682.
  - nature_3.jpg: Resized from 4032x3021 to 1024x767, cropped to 1024x766.
  - nature_4.jpg: Resized from 2736x3648 to 768x1024.
  - nature_5.jpg: Resized from 5184x3456 to 1024x682.
  - nature_6.jpg: Resized from 3293x4935 to 683x1023, cropped to 682x1022.
  - nature_7.jpg: Resized from 1881x2690 to 716x1024.
  - nature_8.jpg: Resized from 3872x2592 to 1024x685, cropped to 1024x684.
  - nature_9.jpg: Resized from 4320x3240 to 1024x768.
  - nature_10.jpg: Resized from 2200x1414 to 1024x658.
- **Category: Faces**
  - face_1.jpg: Resized from 3368x5052 to 682x1024.
  - face_2.jpg: Resized from 4000x6000 to 682x1024.
  - face_3.jpg: Resized from 1836x3264 to 576x1024.
  - face_4.jpg: Resized from 6000x4000 to 1024x682.
  - face_5.jpg: Resized from 7076x4533 to 1023x655, cropped to 1022x654.
  - face_6.jpg: Resized from 3456x5184 to 682x1024.
  - face_7.jpg: Resized from 4664x3840 to 1024x843, cropped to 1024x842.
  - face_8.jpg: Resized from 5184x3456 to 1024x682.
  - face_9.jpg: Resized from 2400x3000 to 819x1024, cropped to 818x1024.
  - face_10.jpg: Resized from 6000x4000 to 1024x682.
- **Category: Animals**
  - animal_1.jpg: Resized from 4000x3000 to 1024x768.
  - animal_2.jpg: Resized from 2472x3709 to 682x1024.
  - animal_3.jpg: Resized from 5184x3888 to 1024x768.
  - animal_4.jpg: Resized from 1920x1080 to 1024x576.
  - animal_5.jpg: Resized from 3047x2362 to 1024x793, cropped to 1024x792.
  - animal_6.jpg: Resized from 2592x3872 to 685x1024, cropped to 684x1024.
  - animal_7.jpg: Resized from 2359x1887 to 1024x819, cropped to 1024x818.
  - animal_8.jpg: Resized from 2392x2500 to 979x1024, cropped to 978x1024.
  - animal_9.jpg: Resized from 2939x2583 to 1024x899, cropped to 1024x898.
  - animal_10.jpg: Resized from 2848x4288 to 680x1024.

All images were successfully processed, and codebooks for both RGB and YUV modes were generated.

4.2 Testing Phase

The testing phase evaluates 5 images per category, compressing them in both RGB and YUV modes, and calculating PSNR, compression ratio, and processing time.

- **Category: Nature**
  - nature_11.jpg: Resized from 2200x1376 to 1024x640.
  - nature_12.jpg: Resized from 7301x4873 to 1024x683, cropped to 1024x682.
  - nature_13.jpg: Resized from 4000x6000 to 682x1024.
  - nature_14.jpg: Resized from 4470x3024 to 1024x692.
  - nature_15.jpg: Resized from 3007x4511 to 682x1024.
- **Category: Faces**
  - face_11.jpg: Resized from 2592x3888 to 682x1024.
  - face_12.jpg: Resized from 3461x2307 to 1024x682.
  - face_13.jpg: Resized from 5760x5075 to 1024x902.
  - face_14.jpg: Resized from 3456x5184 to 682x1024.
  - face_15.jpg: Resized from 6000x4000 to 1024x682.
- **Category: Animals**
  - animal_11.jpg: Resized from 2848x4272 to 682x1024.
  - animal_12.jpg: Resized from 3888x2592 to 1024x682.
  - animal_13.jpg: Resized from 4275x2539 to 1024x608.
  - animal_14.jpg: Resized from 3456x5184 to 682x1024.
  - animal_15.jpg: Resized from 2250x2625 to 877x1024, cropped to 876x1024.

All test images were successfully processed, with results detailed in the next section.

5 Results and Discussion

The test results provide a comprehensive evaluation of the vector quantization implementation in both RGB and YUV modes. The key metrics are PSNR (dB) for image quality, compression ratio (CR), and processing time (ms). The results are summarized in the following tables.

5.1 Nature Category Results

5.2 Faces Category Results

5.3 Animals Category Results

5.4 Analysis

The results show that RGB mode generally achieves higher PSNR values compared to YUV mode, indicating better image quality after reconstruction. For example, animal_12.jpg

Table 1: Test Results for Nature Category

Image	RGB			YUV		
	PSNR (dB)	CR	Time (ms)	PSNR (dB)	CR	Time (ms)
nature_11.jpg	27.42	768.00	1663	26.06	512.00	858
nature_12.jpg	32.00	779.43	1915	29.68	519.62	918
nature_13.jpg	34.13	779.43	1773	31.30	519.62	928
nature_14.jpg	36.08	772.47	1969	33.85	519.00	990
nature_15.jpg	26.03	779.43	1760	24.58	519.62	928

Table 2: Test Results for Faces Category

Image	RGB			YUV		
	PSNR (dB)	CR	Time (ms)	PSNR (dB)	CR	Time (ms)
face_11.jpg	35.10	779.43	1862	31.05	519.62	995
face_12.jpg	34.51	779.43	1880	33.74	519.62	965
face_13.jpg	37.18	773.14	2300	33.98	515.43	1196
face_14.jpg	31.76	779.43	1762	30.07	519.62	902
face_15.jpg	36.03	779.43	1733	34.53	519.62	952

Table 3: Test Results for Animals Category

Image	RGB			YUV		
	PSNR (dB)	CR	Time (ms)	PSNR (dB)	CR	Time (ms)
animal_11.jpg	34.13	779.43	1756	30.32	519.62	917
animal_12.jpg	40.01	779.43	1749	35.30	519.62	1016
animal_13.jpg	36.09	768.00	1509	34.56	512.00	816
animal_14.jpg	36.89	779.43	1790	33.33	519.62	967
animal_15.jpg	31.14	778.67	2281	30.44	519.11	1249

achieves an RGB PSNR of 40.01 dB compared to a YUV PSNR of 35.30 dB, a difference of 4.71 dB. However, YUV mode benefits from subsampling, resulting in a lower compression ratio (around 512519:1) compared to RGB (around 768779:1), which translates to more efficient storage due to the reduced data in U and V channels.

Processing times for YUV mode are consistently lower than for RGB mode (e.g., 816ms vs. 1509ms for animal_13.jpg), likely due to the subsampling reducing the amount of data processed. The PSNR values vary across categories, with animals showing the highest peak (40.01 dB for animal_12.jpg in RGB) and nature the lowest (24.58 dB for nature_15.jpg in YUV), reflecting the diversity in image content and complexity.

6 Conclusion

This project successfully implements vector quantization for image compression, meeting both the main and bonus requirements of the End Term Project for Information Theory. The implementation processes 45 color images across three categories, using 30 for train-

ing and 15 for testing, with a block size of 2x2 and a codebook size of 256. The RGB mode achieves higher image quality (PSNR) but lower compression ratios compared to YUV mode, which benefits from 50% subsampling of U and V channels. The results provide a clear comparison between the two techniques, with RGB offering better quality and YUV offering better compression efficiency. Future improvements could include optimizing the K-means clustering algorithm for faster convergence, exploring larger block sizes, or implementing adaptive subsampling based on image content.