Vector Quantization Image Compression

Detailed Code Explanation

Fares Wael
ID: 202201260

Contents

1 Introduction

This report explains a Java program that implements **Vector Quantization (VQ)** for image compression. The idea is to reduce image size by representing blocks of pixels with a smaller codebook of representative blocks.

2 Libraries Used

- `javax.imageio.ImageIO`: Reading and writing images.
- `javax.swing.*`: (Unused in this code, possibly for future GUI).
- `java.awt.*` and `java.awt.image.BufferedImage`: Image handling.
- `java.io.*`: File reading and writing.
- `java.util.*`: Handling lists, arrays, and input.

3 Main Method

The program starts execution from the `main` method:

- Takes three inputs from the user:
  - Path to an RGB image.
  - Block size (e.g., 4x4 pixels per block).
  - Codebook size (number of representative vectors, K).
- Calls `processImage` to perform the compression.

4 Image Processing Steps

4.1 Convert Image to Grayscale

The method `convertToGrayscale` is used to simplify the data, converting the colored image to grayscale, reducing complexity.

4.2 Adjust Image Size

Since we work with fixed-size blocks, the image dimensions must be divisible by block size. If not, the image is resized without interpolation.

4.3 Block Division

`createBlocks` method divides the adjusted grayscale image into non-overlapping square blocks of the given size.

4.4 Codebook Generation (LBG Algorithm)

- The `generateCodebook` function starts with the average of all blocks.
- It splits the existing vectors by a small perturbation (multiplying by 1.01 and 0.99).
- Then, it iteratively applies K-means clustering until convergence is reached (distance change $< \varepsilon$).

4.5 Compression

Each block is replaced with the nearest codebook vector (measured by Euclidean distance).

4.6 Image Reconstruction

The compressed blocks are used to reconstruct the compressed version of the image using `reconstructImage`.

4.7 Performance Metrics

- **Mean Square Error (MSE)**: Measures the error between the original and reconstructed images.
- **Compression Ratio**: Indicates how much the data size has been reduced.

4.8 Report Generation

A text report (`report.txt`) is generated that summarizes:

- Inputs used.
- MSE value.
- Original and compressed sizes.
- Compression ratio.
- Notes about quality vs compression.

5 Important Functions Explained

5.1 convertToGrayscale

- Creates a new grayscale image from the RGB input.

5.2 createBlocks

- Divides the image into small blocks (vectors) of size blockSize $\times$ blockSize.

5.3 generateCodebook

- Implements the Linde-Buzo-Gray (LBG) algorithm.
- Splits codewords and refines them with K-means clustering.

5.4 findNearestVector

- Calculates the Euclidean distance between a block and each codebook vector, and returns the index of the closest.

5.5 euclideanDistance

- Computes the standard Euclidean distance between two vectors.

5.6 reconstructImage

- Uses the labeled image to reconstruct the final compressed image using codebook vectors.

5.7 calculateMSE

- Computes the Mean Square Error between the grayscale original and reconstructed image.

5.8 generateReport

- Creates a human-readable report summarizing all important results.

6 Outputs Produced

- **grayscale.png**: Grayscale version of the original image.
- **adjusted_grayscale.png**: (Optional) Adjusted image if resizing was needed.
- **reconstructed.png**: The compressed and reconstructed image.
- **report.txt**: Summary report including MSE and compression ratio.

7 Test Cases

This section presents examples of inputs and outputs:

7.1 Input 1

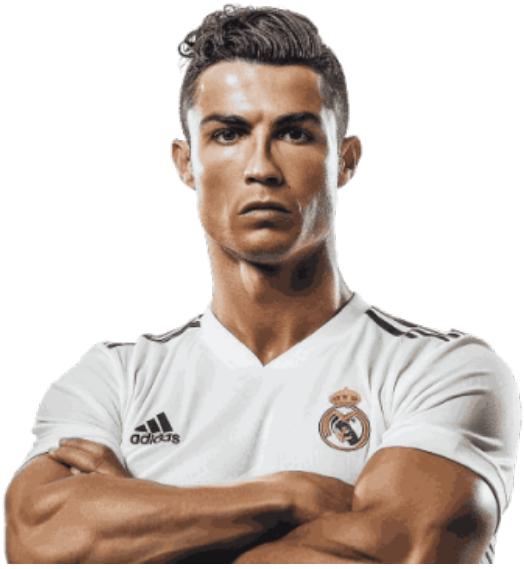

Figure 1: Original Input Image 1

7.2 Input 2

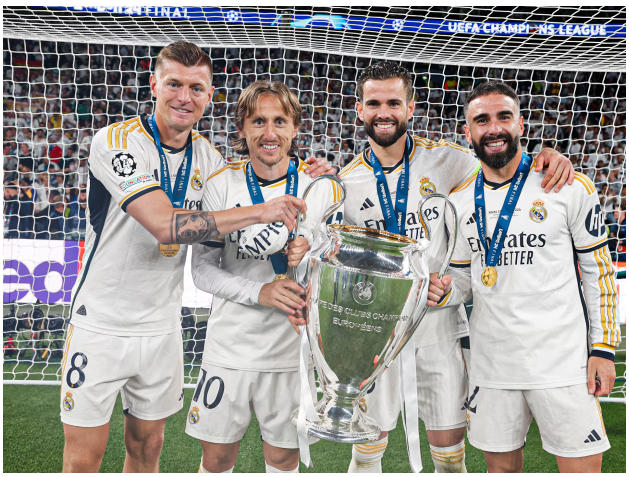

Figure 2: Original Input Image 2

7.3 Calculation Results and Report

input > ≡ report.txt

```
Vector Quantization Image Compression Report
==========================================

Input Parameters:
- Original Image: C:\Users\FARES WAEL\Downloads\assignment_4\input.png
- Block Size: 4x4
- Codebook Size (K): 4

Compression Results:
- Mean Square Error (MSE): 269.008875
- Original Data Size (bits): 1280000.0
- Compressed Data Size (bits): 20512.0
- Compression Ratio: 62.402496099843994:1

Notes:
- Lower MSE indicates better image quality
- Higher compression ratio indicates better compression
- Blocking effects may be visible with larger block sizes

Files Generated:
- grayscale.png: Original image converted to grayscale
- reconstructed.png: Image after compression and decompression

```

Figure 3: Calculation Results and Report Summary

7.4 Output 1

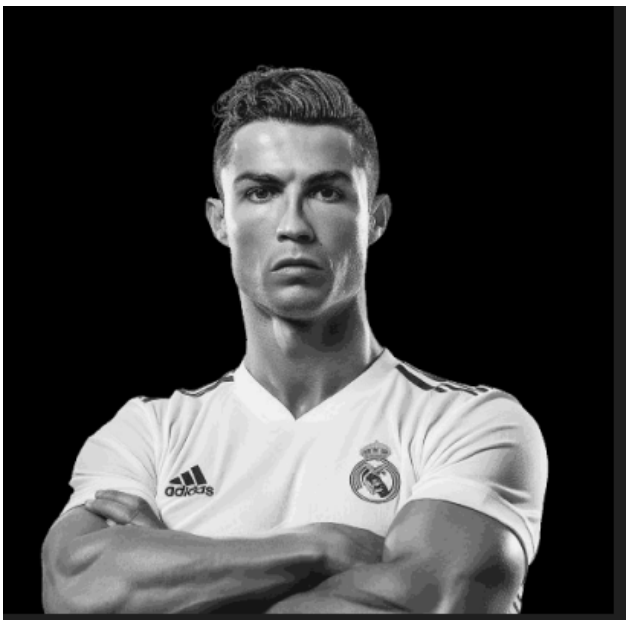

Figure 4: Reconstructed Image for Input 1

7.5 Output 2

Figure 5: Reconstructed Image for Input 2

8 Conclusion

This Java program successfully implements vector quantization-based image compression with report generation. It efficiently applies the LBG algorithm and K-means clustering to minimize image size while preserving quality as much as possible.