CS663 Course Project

Ayan Tanwar, Harsh Kumar , Sarthak Chaurasia
Roll Number: 22b0931 , 22b0973 , 22b1014

November 25, 2024

Navigation

1 Introduction

Image compression is a critical component in modern multimedia applications. In this project, we implemented a compression algorithm using Discrete Cosine Transform (DCT) and Huffman Encoding. The implementation is designed for both grayscale and color images and evaluates the performance using metrics like Root Mean Squared Error (RMSE) and Bits Per Pixel (BPP).

2 Algorithm Description

The implemented image compression algorithm is inspired by the JPEG compression standard and consists of the following key steps:

2.1 1. Discrete Cosine Transform (DCT)

- **Purpose:** The DCT is applied to reduce spatial redundancy in image data by transforming it into the frequency domain. This allows high-energy (low-frequency) components to be concentrated while reducing the impact of less visually significant high-frequency components.
- **Implementation Details:**
  - The image is divided into non-overlapping 8×8 blocks.
  - Each block undergoes a 2D DCT transformation, which computes the frequency coefficients for the block.
  - The result is a matrix where the top-left corner represents the DC coefficient (low-frequency component), and the rest are AC coefficients (higher-frequency components).

2.2 2. Quantization

- **Purpose:** Quantization reduces the precision of the DCT coefficients, prioritizing low-frequency components while discarding less significant high-frequency details. This step achieves lossy compression by exploiting the human visual system's reduced sensitivity to high-frequency variations.
- **Implementation Details:**

  - A predefined quantization matrix M, scaled by a **quality factor (Q)**, is used to divide the DCT coefficients block by block.
  - The operation is mathematically expressed as:

$$\text{Quantized Coefficients} = \text{Round}\left(\frac{\text{DCT Coefficients}}{Q \cdot M}\right)$$

  - A higher quality factor Q preserves more detail, while a lower Q leads to higher compression but more information loss.

2.3 3. Differential Encoding

- **Purpose:** Differential encoding exploits redundancy across blocks to further compress the data. It stores the difference between successive blocks rather than their absolute values.
- **Implementation Details:**
  - For each block, the algorithm calculates the difference between the current block's quantized coefficients and the previous block's quantized coefficients.
  - The first block's coefficients are stored directly since no previous block exists.
  - This step ensures that the stored values have smaller ranges, improving subsequent compression.

2.4 4. Huffman Encoding

- **Purpose:** Huffman encoding is a lossless compression technique that efficiently represents data based on symbol frequency.
- **Implementation Details:**
  - A frequency table of the quantized coefficients is generated.
  - A Huffman tree is constructed from the frequency table, assigning shorter binary codes to more frequent values and longer codes to less frequent values.
  - The quantized coefficients are encoded into a binary string using the Huffman tree.
  - This step minimizes the size of the compressed file without introducing further loss.

2.5 5. File Input/Output (I/O)

- **Purpose:** To enable the storage and reconstruction of the compressed image.
- **Implementation Details:**
  - The compressed data, including metadata such as the image dimensions, quantization matrix, Huffman table, and the encoded binary string, is stored in a binary file format.
  - During decompression:
    * The Huffman table is used to decode the binary string back into the quantized coefficients.
    * The differential encoding is reversed to retrieve the original quantized blocks.
    * The quantized coefficients are dequantized using the same quantization matrix M and quality factor Q.
    * The inverse DCT (IDCT) is applied to the dequantized coefficients to reconstruct the image in the spatial domain.

2.6 Mathematical Formulations

- **DCT Transformation:**

$$F(u,v) = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

  where $\alpha(u)$ and $\alpha(v)$ are scaling factors to normalize the transformation.

- **Inverse DCT (IDCT):**

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} \alpha(u)\alpha(v)F(u,v)\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

- **Quantization:**

$$Q_{i,j} = \text{Round}\left(\frac{D_{i,j}}{M_{i,j}\cdot\frac{50}{Q}}\right)$$

  where $D_{i,j}$ are the DCT coefficients and $M_{i,j}$ is the quantization matrix entry.

This multi-step algorithm balances compression efficiency with visual fidelity, producing a compressed image that approximates the original with minimal perceptual loss.

This implementation is inspired by the JPEG algorithm and supports varying quality factors to control compression.

3 Dataset Description

We tested the algorithm on the following datasets:

- **COIL-20 Dataset:** Contains grayscale images of 20 objects with backgrounds removed, focusing on the smallest square region containing the object.
- **Color Images:** A collection of standard RGB images representing diverse scenes to evaluate the algorithm's effectiveness on color data.

4 Results

We evaluated the performance of our algorithm using the following experiments:

1. **Grayscale Images:**
    - RMSE vs BPP plot for 20 images from the COIL-20 dataset.
    - Quality factor varied from 10 to 100 in steps of 10.
    - A QF range of 50–70 offered a balanced trade-off, achieving significant compression without noticeable degradation in visual quality.
    - Images with higher spatial frequency content (e.g., detailed textures) exhibited slightly higher RMSE values due to the aggressive quantization of high-frequency DCT coefficients.
2. **Color Images:**
    - Similar RMSE vs BPP plots were generated for color images.

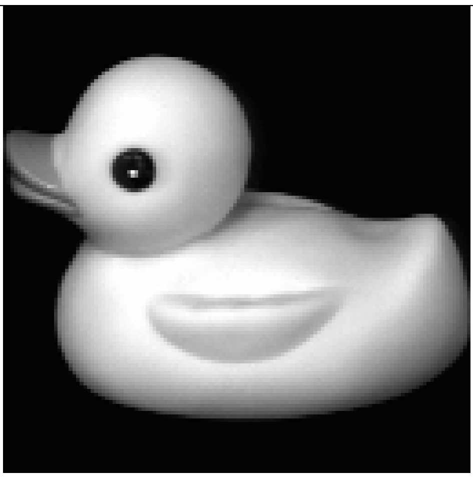
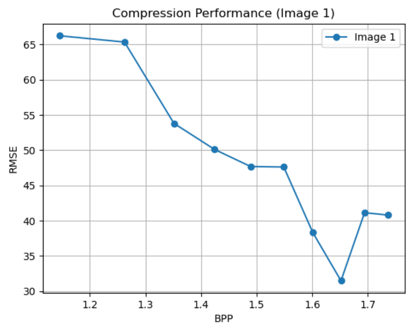
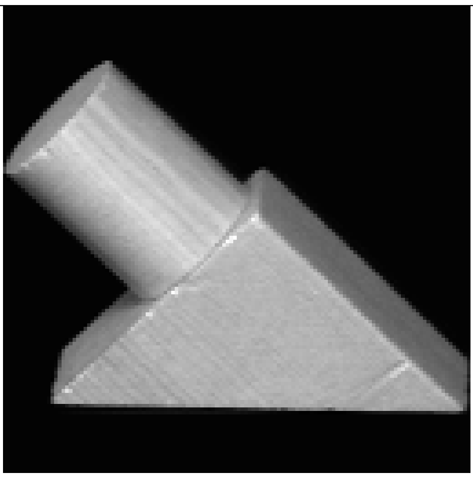
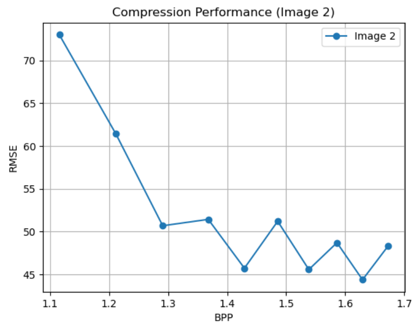
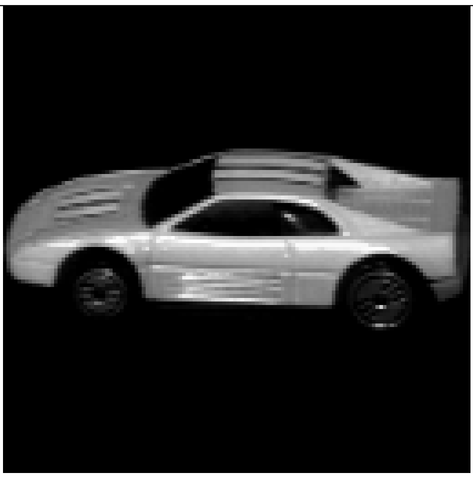
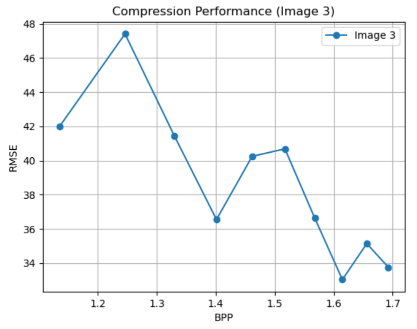
 **obj1__0.png**	 **image_1_plot.png**
 **obj2__0.png**	 **image_2_plot.png**
 **obj3__0.png**	 **image_3_plot.png**

Figure 1: RMSE vs BPP plots for images 1 to 3.

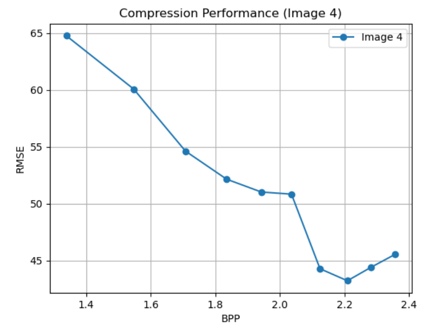
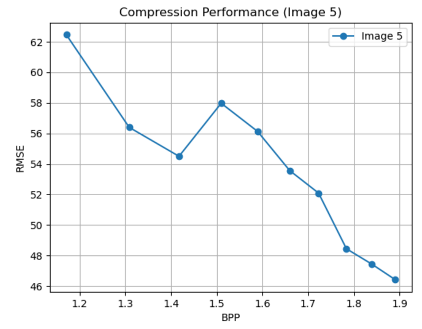
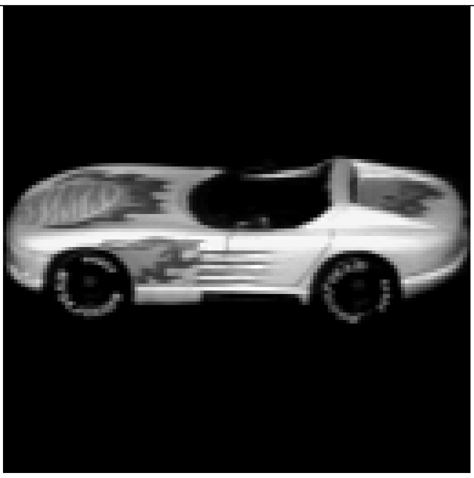
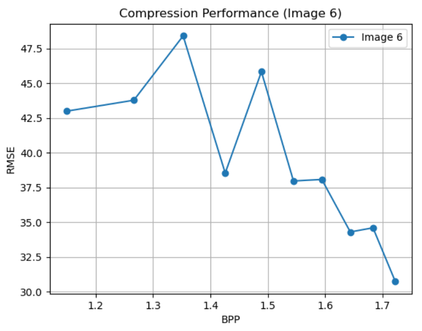
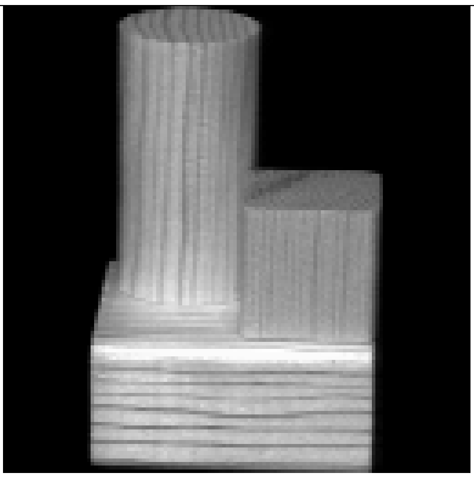
obj4__0.png	 **image_4_plot.png**
obj5__0.png	 **image_5_plot.png**
 **obj6__0.png**	 **image_6_plot.png**

Figure 2: RMSE vs BPP plots for images 4 to 6.



obj7__0.png

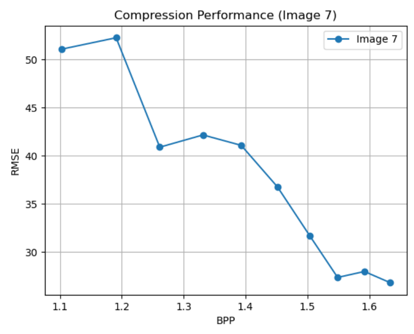


image_7_plot.png

obj8__0.png

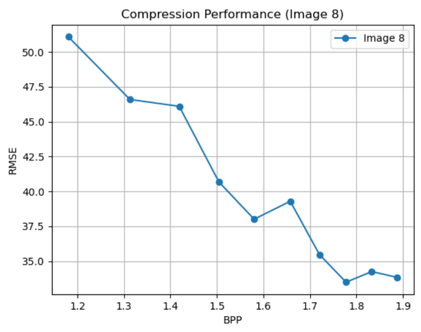
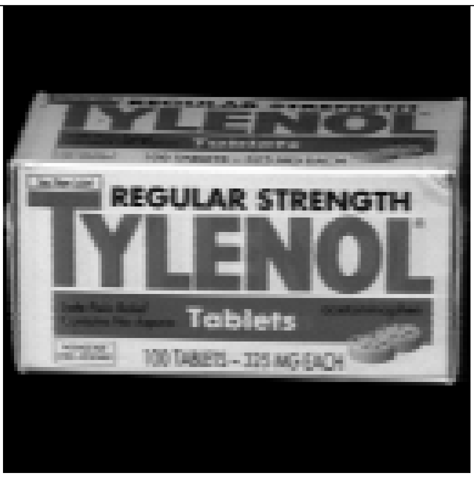


image_8_plot.png



obj9__0.png

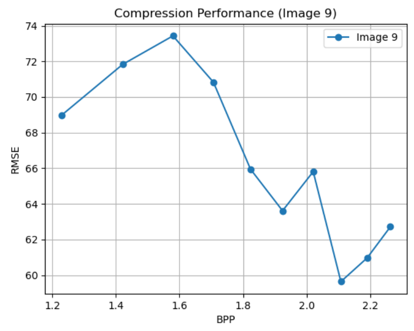


image_9_plot.png

Figure 3: RMSE vs BPP plots for images 7 to 9.

obj10__0.png

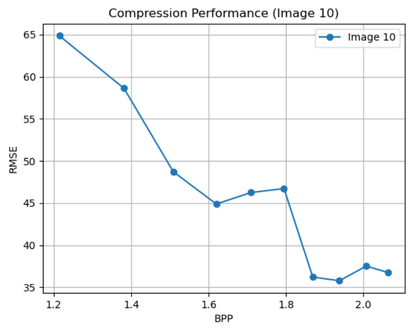
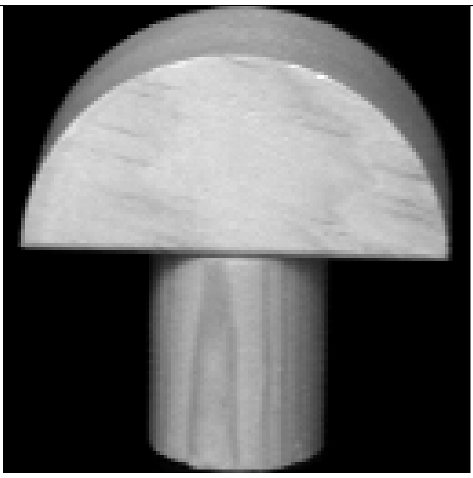


image_10_plot.png



obj11__0.png

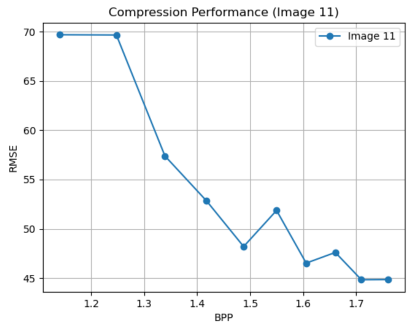
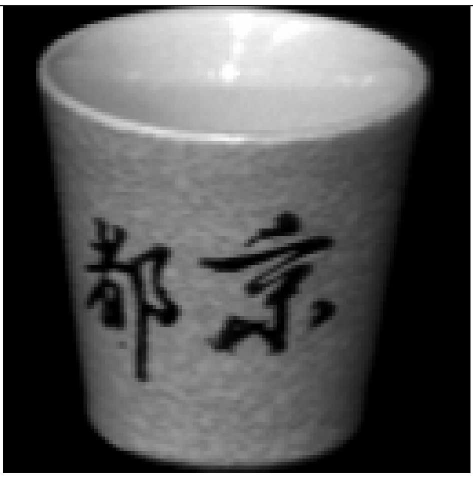


image_11_plot.png



obj12__0.png

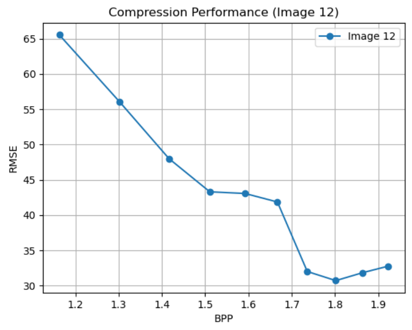


image_12_plot.png

Figure 4: RMSE vs BPP plots for images 10 to 12.

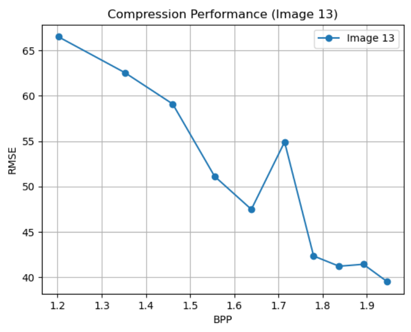
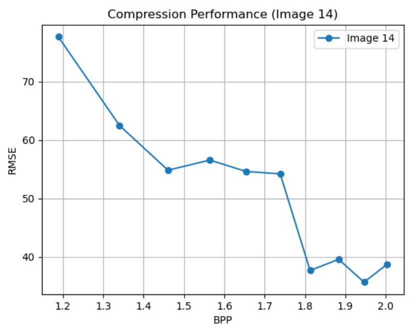
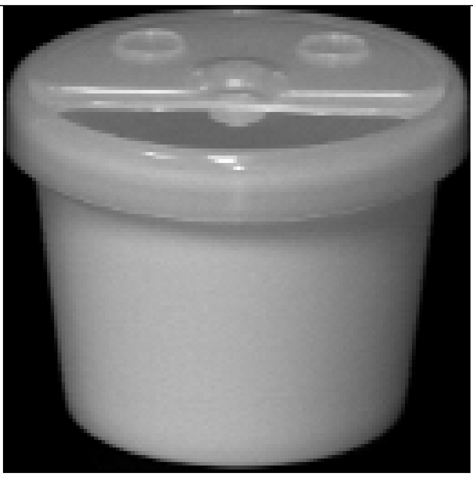
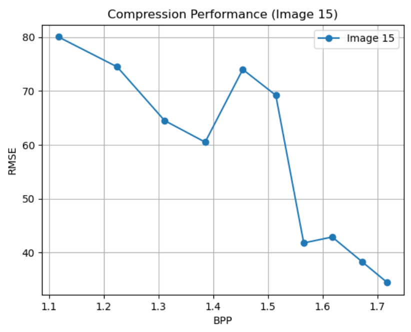
obj13__0.png	 **image_13_plot.png**
obj14__0.png	 **image_14_plot.png**
 **obj15__0.png**	 **image_15_plot.png**

Figure 5: RMSE vs BPP plots for images 13 to 15.

obj16__0.png

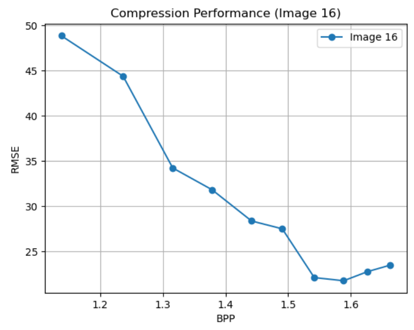


image_16_plot.png

obj17__0.png

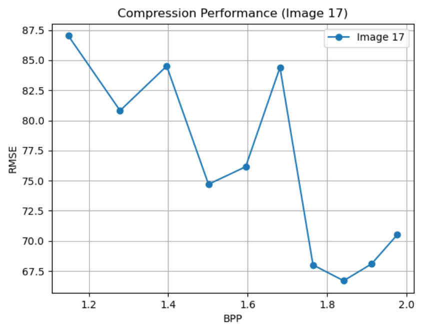
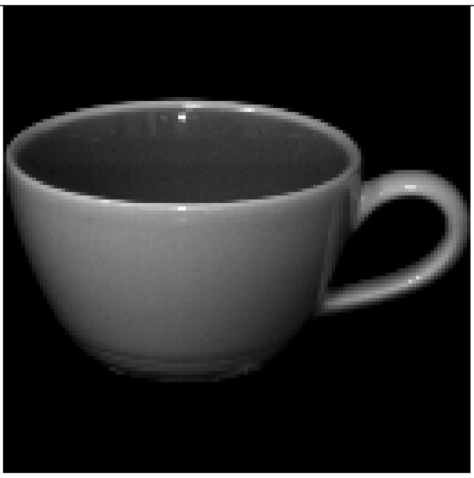


image_17_plot.png



obj18__0.png

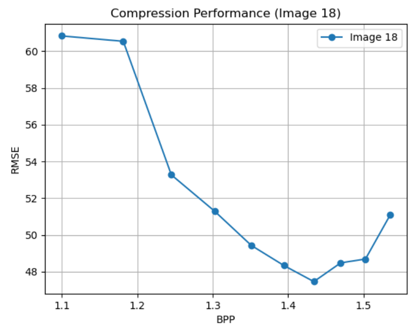


image_18_plot.png

Figure 6: RMSE vs BPP plots for images 16 to 18.

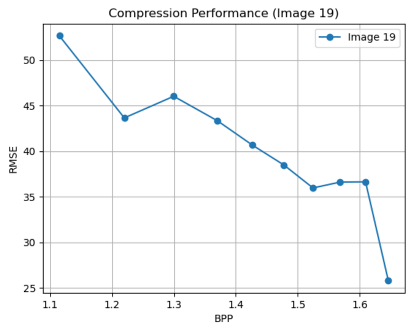
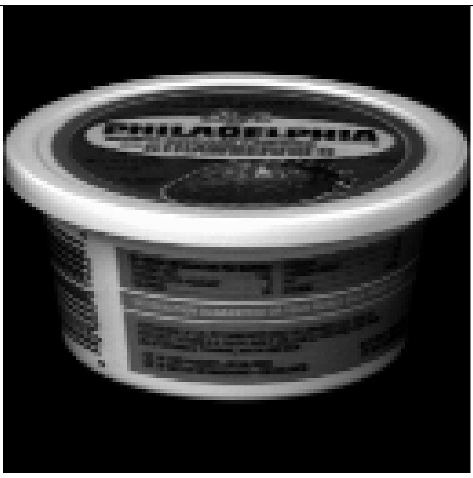
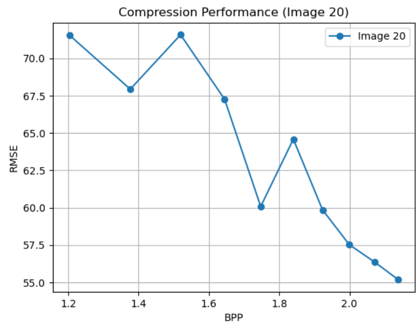
obj19__0.png	 **image_19_plot.png**
 **obj20__0.png**	 **image_20_plot.png**

Figure 7: RMSE vs BPP plots for images 19 to 20.

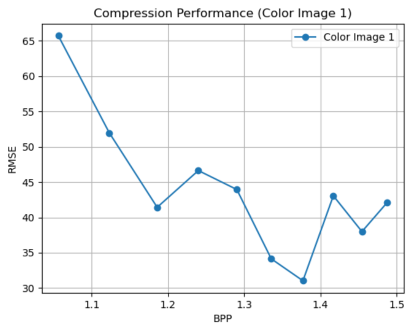
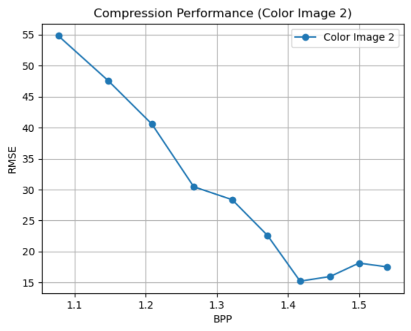
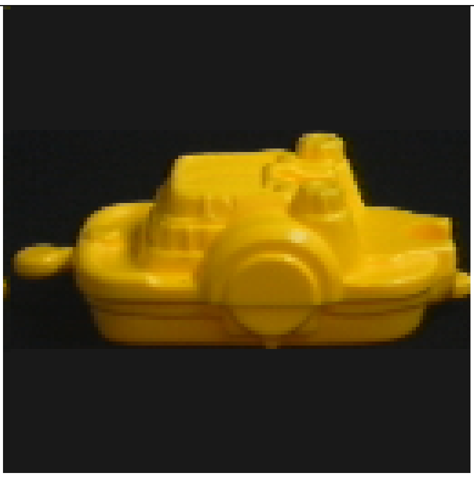
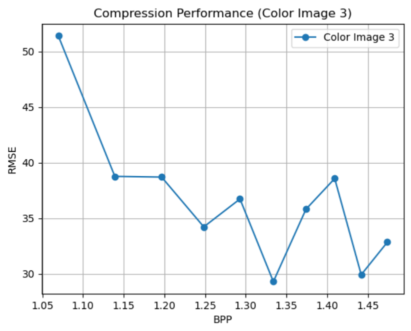
obj1__0.png	 **color_image_1_plot.png**
obj2__0.png	 **color_image_2_plot.png**
 **obj3__0.png**	 **color_image_3_plot.png**

Figure 8: RMSE vs BPP plots for images 1 to 3.

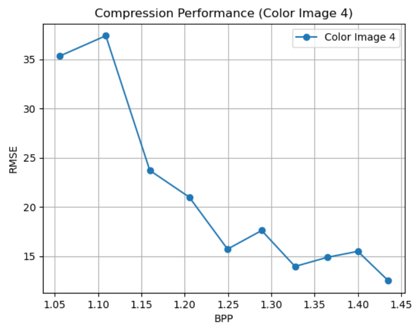
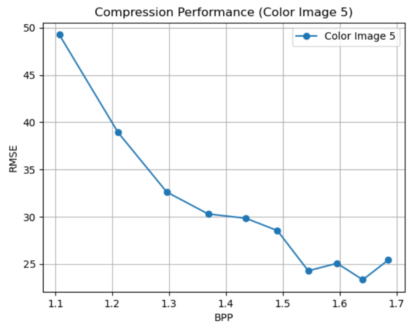
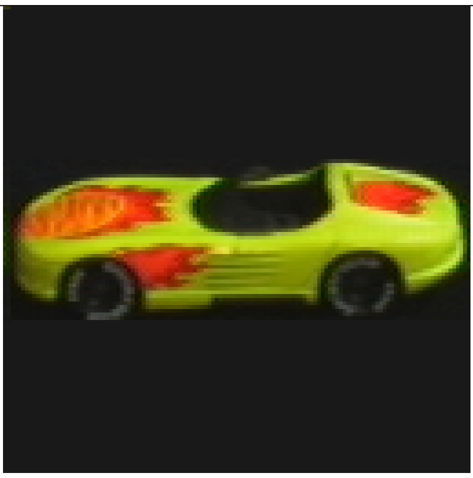
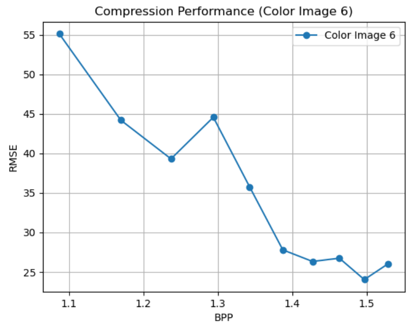
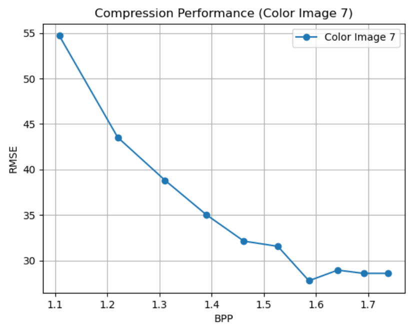
obj4__0.png	 **color_image_4_plot.png**
obj5__0.png	 **color_image_5_plot.png**
 **obj6__0.png**	 **color_image_6_plot.png**

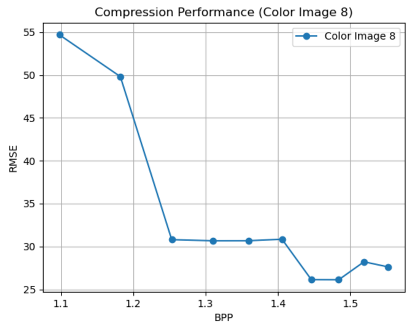
Figure 9: RMSE vs BPP plots for images 4 to 6.

obj7__0.png

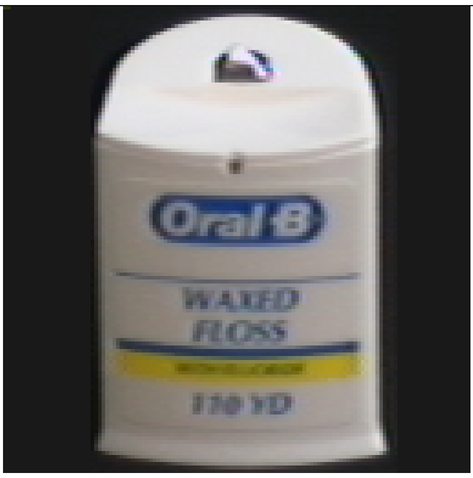


color_image_7_plot.png

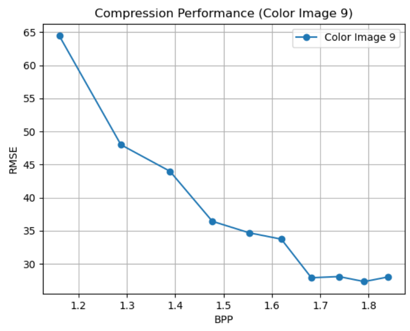
obj8__0.png



color_image_8_plot.png



obj9__0.png



color_image_9_plot.png

Figure 10: RMSE vs BPP plots for images 7 to 9.

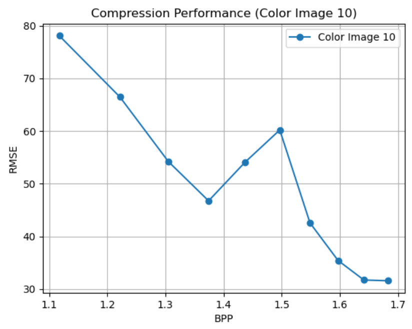
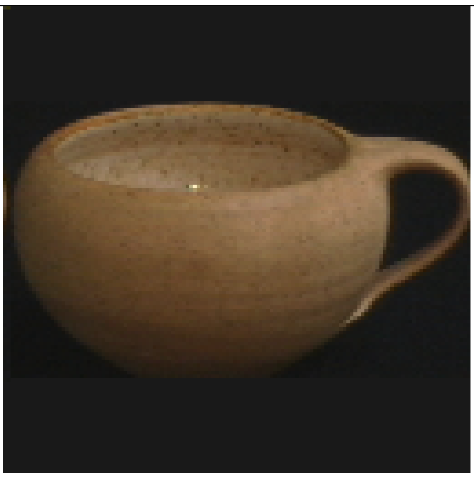
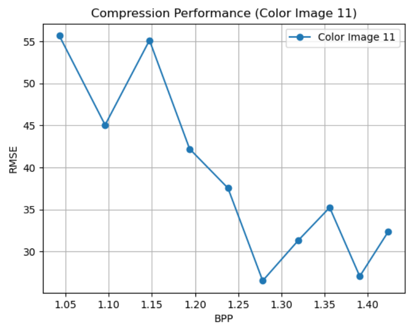
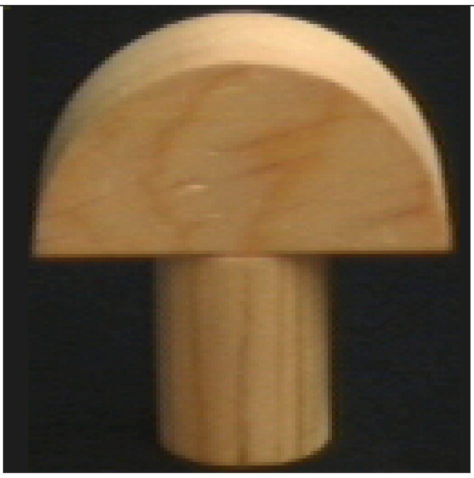
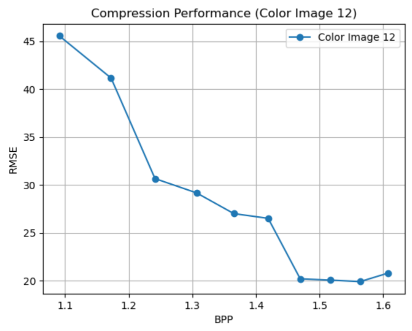
obj10__0.png	 **color_image_10_plot.png**
 **obj11__0.png**	 **color_image_11_plot.png**
 **obj12__0.png**	 **color_image_12_plot.png**

Figure 11: RMSE vs BPP plots for images 10 to 12.

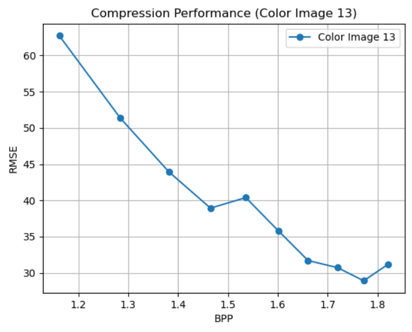
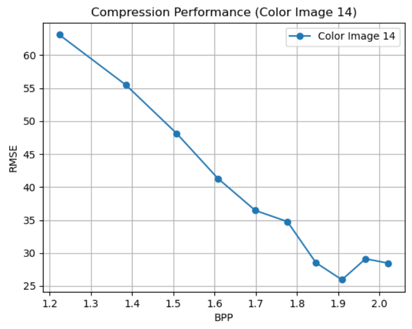
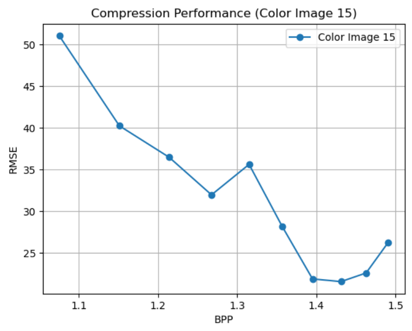
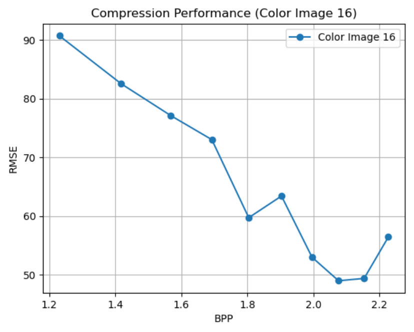
obj13__0.png	 **color_image_13_plot.png**
obj14__0.png	 **color_image_14_plot.png**
obj15__0.png	 **color_image_15_plot.png**

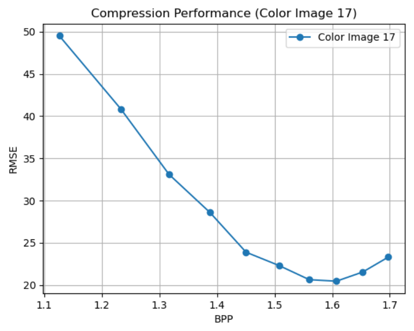
Figure 12: RMSE vs BPP plots for images 13 to 15.

obj16__0.png

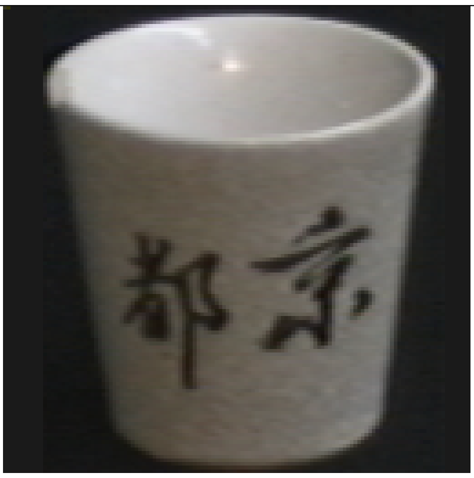


color_image_16_plot.png

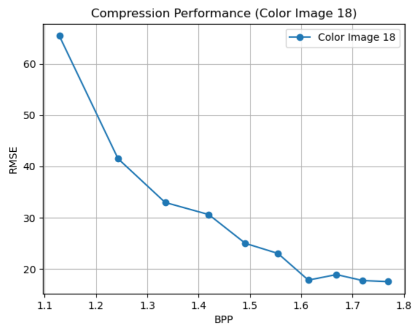
obj17__0.png



color_image_17_plot.png



obj18__0.png



color_image_18_plot.png

Figure 13: RMSE vs BPP plots for images 16 to 18.

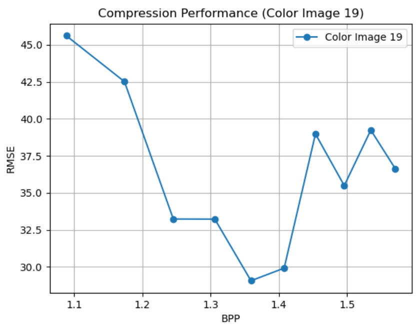
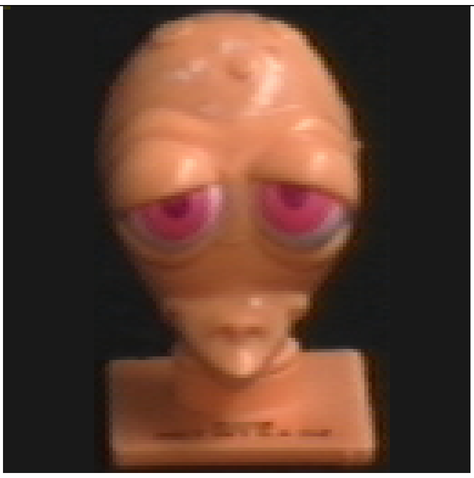
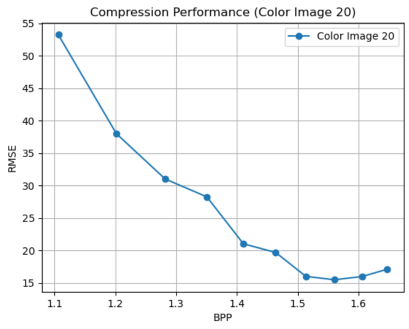
obj19__0.png	 **color_image_19_plot.png**
 **obj20__0.png**	 **color_image_20_plot.png**

Figure 14: RMSE vs BPP plots for images 19 to 20.

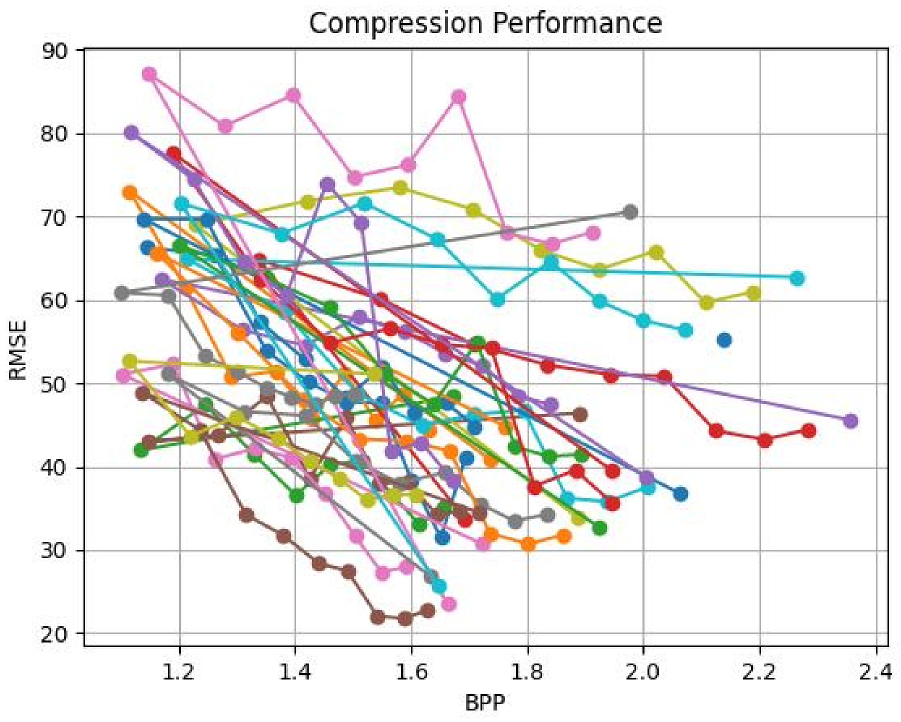


Figure 15: RMSE vs BPP plot for all images.

5 Conclusion and Analysis

The implementation of the image compression algorithm demonstrated significant potential for achieving high compression efficiency with manageable trade-offs in visual quality. The observations are detailed below:

- **Advantages:**
  - **Efficient Compression:** The algorithm achieved substantial reduction in image size, with higher quality factors maintaining almost indistinguishable visual fidelity compared to the original image. This is particularly advantageous for storage and bandwidth-constrained applications.
  - **Frequency-Domain Representation:** The Discrete Cosine Transform (DCT) effectively concentrates image energy into a few significant coefficients, facilitating efficient quantization and subsequent compression.

  - **Lossless Entropy Encoding:** The application of Huffman encoding preserved essential details in the compressed data by minimizing redundancies in quantized coefficients without introducing any additional artifacts.
  - **Scalability with Quality Factors:** The adjustable quality factor offers flexibility, allowing users to prioritize either compression ratio or image quality based on application requirements, such as archival storage or web-based media transmission.

- **Challenges:**
  - **Computational Overhead:** Processing large images with an 8x8 block-based DCT approach introduces significant computational complexity. This can impact real-time performance in resource-constrained environments or for high-resolution images.
  - **Quality Sensitivity:** Lower quality factors result in amplified quantization artifacts, such as blocking effects and a loss of fine details. These artifacts become especially noticeable in areas with gradual intensity variations or fine textures.
  - **Edge Artifacts:** The block-based processing occasionally introduces discontinuities along block boundaries, particularly in regions with high-frequency content, which could degrade perceptual quality.
  - **Fixed Block Size Limitation:** The use of fixed 8x8 blocks may not adapt optimally to complex image content, limiting the potential for compression efficiency in highly detailed regions.
  - **Quantization Matrix Dependency:** The results are sensitive to the design of the quantization matrix, which may require fine-tuning for specific types of images to balance compression performance and quality retention.