MINI PROJECT - JPEG COMPRESSION

Submitted by: Sneha Dash (21EC39023), Unnati Singh (21EC39027)

Overview

This project explores JPEG compression, a commonly used technique for reducing image file sizes while maintaining visual quality. The pipeline involves converting an RGB image to the YCbCr colour space, subsampling, discrete cosine transformation (DCT), quantization, and entropy encoding.

Files Included

1. `Mini_Project.ipynb`: Jupyter notebook with python code implementing the JPEG compression pipeline.
2. `mini_project.py`: Python file, implementing the same code.
3. `Compress_Image.ipynb`: Secondary code file implementing the compression pipeline till quantization and then decompressing to view the compressed image.
4. `Input/`: Directory containing the input images.
5. `Output/`: Directory containing the output binary files.
6. `Mini_Project_Report.pdf`: The experiment report detailing the objective, methodology, results, and conclusion.
7. `Mini_Project_PPT.pptx`: The presentation for this experiment.
8. `Mini_Project_PPT.pdf`: The presentation in PDF format.
9. `ReadMe.pdf`: This ReadMe file.

Functions Overview

1. `read_image(imagepath):`
   - **Purpose**: Loads an image and converts it to RGB format.
   - **Working**: Reads the image in BGR, then converts it to RGB.
   - **Input**: Path to the image file.
   - **Output**: RGB-format image as a NumPy array.

2. `rgb_to_ycbcr(image):`
   - **Purpose**: Converts an RGB image to YCbCr colour space.
   - **Working**: Multiplies the RGB image with a conversion matrix and adds an offset to obtain YCbCr values.
   - **Input**: An RGB image (NumPy array).
   - **Output**: The converted YCbCr image (NumPy array with values clamped between 0 and 255).

3. `chroma_subsampling(ycbcr_image, ratio1h = 2, ratio1v = 2, ratio2h = 2, ratio2v = 2):`
   - **Purpose**: Performs chroma subsampling on a YCbCr image.
   - **Working**: Subsamples the Cb and Cr channels by specified horizontal and vertical ratios, while keeping the Y channel intact.
   - **Input**: A YCbCr image (NumPy array) and optional subsampling ratios for Cb and Cr channels.
   - **Output**: The Y channel and subsampled Cb and Cr channels (NumPy arrays).

4. `block_formation(channel, block_size=8):`
   - **Purpose**: Divides a channel of an image into smaller blocks of a specified size.
   - **Working**: Reshapes the channel into blocks of the given size by adjusting the dimensions and swapping axes.
   - **Input**: A 2D image channel (NumPy array) and an optional block size.
   - **Output**: A 2D array of image blocks (NumPy array).

5. `dct_block(block):`
   - **Purpose**: Computes the Discrete Cosine Transform (DCT) of a block of data.
   - **Working**: Applies the DCT formula using cosine functions and scaling factors to the input block.
   - **Input**: A 2D block (NumPy array).
   - **Output**: The DCT-transformed block (NumPy array).

6. `quantize(block, quant_matrix):`
   - **Purpose**: Quantizes a block using a provided quantization matrix.
   - **Working**: Divides the block by the quantization matrix and rounds the result to the nearest integer.
   - **Input**: A 2D block (NumPy array) and a quantization matrix (NumPy array).
   - **Output**: The quantized block (NumPy array of integers).

7. `run_length_encode(block):`
   - **Purpose**: Performs run-length encoding (RLE) on a block after zigzag scanning.
   - **Working**: Scans the block in a zigzag pattern, then applies run-length encoding to compress consecutive zeros.
   - **Input**: A 2D block (NumPy array).
   - **Output**: The run-length encoded data (list of integers and tuples).

8. `huffman(rle_data):`
   - **Purpose**: Performs Huffman encoding on the run-length encoded data.
   - **Working**: Builds a Huffman tree from symbol frequencies, generates binary codes for each symbol, and encodes the sequence.
   - **Input**: Run-length encoded data (list of integers and tuples).
   - **Output**: The Huffman codes (dictionary) and the encoded binary sequence (string).

9. `compress_save(image_path):`
   - **Purpose**: Compresses an image and saves the compressed data and Huffman-encoded sequences to files.
   - **Working**: Converts the image to YCbCr, applies chroma subsampling, performs DCT, quantization, run-length encoding, and Huffman coding, then saves the compressed data.
   - **Input**: The path to the image file.
   - **Output**: The complete compressed data (dictionary) saved as a pickle file and printed file size information.