BMP Image Processing Report

Name: Aryan Kumar

Roll Number: 21EC39008

Institution: Indian Institute of Technology, Kharagpur

Course: Digital Image and Video Processing Lab

Experiment Number: 2

Date: August 2024

1. Introduction

The objective of this experiment was to implement Python functions capable of reading, writing, and manipulating BMP image files. The experiment involved working with 24-bit RGB, 8-bit grayscale, and 8-bit color indexed BMP images. This report documents the process, implementation, and results of the BMP image processing tasks.

2. BMP File Format Overview

The BMP (Bitmap) file format is a raster image format primarily used for storing digital images. It comprises the following key components:

- **File Header**: Contains information about the file type, size, and the offset where the pixel data begins.
- **DIB Header (Bitmap Info Header)**: Stores information such as image dimensions, color format, compression type, and resolution.
- **Color Table**: Used in 8-bit and lower bit depth images to define the colors used in the bitmap.
- **Pixel Data**: The actual bitmap data, which represents the image pixel by pixel.

3. Methodology

3.1. Reading BMP Image

A function, `load_bmp_image(filepath)`, was developed to read and parse a BMP image file. The function extracts and prints key metadata including the image's width, height, bit depth, and file size. It also loads the pixel data into memory.

Key Steps:

- **File Header Parsing**: Verifies the BMP signature and reads the file size, reserved fields, and pixel data offset.
- **DIB Header Parsing**: Retrieves image width, height, color planes, bits per pixel, compression method, and other relevant details.
- **Color Table Loading**: If the image uses a color palette (e.g., 8-bit images), the color palette is loaded.
- **Pixel Data Loading**: Reads the pixel data into a structured array.

3.2. Writing BMP Image

The `save_bmp(image_info, output_path)` function was implemented to write BMP image data back to disk. This function uses the metadata and pixel data previously loaded to recreate the BMP file.

Key Steps:

- **File Header Construction**: Constructs and writes the BMP file header.
- **DIB Header Construction**: Creates and writes the DIB header based on the image information.
- **Color Table Writing**: If applicable, writes the color palette.
- **Pixel Data Writing**: Outputs the pixel data in the correct format.

3.3. Color Channel Manipulation

For this task, the function `modify_color_channel(image_info, channel_index, output_path)` was used to manipulate the color channels of the 'corn.bmp' image. Specifically, the red, green, and blue channels were individually set to zero, and the resulting images were saved.

4. Results

4.1. Image Metadata

The following table summarizes the metadata of the processed images:

Image	Width (px)	Height (px)	Bits per Pixel	File Size (bytes)
cameraman.bmp	512	512	24	786,486
corn.bmp	256	256	24	196,662
pepper.bmp	512	512	24	786,486

4.2. Color Channel Manipulation Results

The following images were generated by modifying the color channels of `corn.bmp`:

- **corn_red_channel_removed.bmp**: The red channel was set to zero.
- **corn_green_channel_removed.bmp**: The green channel was set to zero.
- **corn_blue_channel_removed.bmp**: The blue channel was set to zero.

These images demonstrate the effect of isolating each color channel in an RGB image.

5. Conclusion

This experiment successfully demonstrated the reading, writing, and manipulation of BMP image files using Python. The functions developed can handle 24-bit RGB, 8-bit grayscale, and 8-bit color indexed BMP images, fulfilling the objectives of the lab. The results show that basic image processing tasks, such as color channel manipulation, can be effectively performed using these modular functions.