DSA2 mini-project report

Teymur Rzali, Gismat Salimov

May 29, 2022

Introduction

The main idea of this project is to create a program that allows adjusting the brightness and contrast of the BMP image automatically. To obtain solid results the process should be done by obeying specific rules. To modify the brightness the minimum value of all RGB pixels will be subtracted from all pixels. Multiplying pixels by the maximizing factor which is gained by dividing the maximum brightness by the maximum RGB values allows obtaining maximum contrast in the image.

BMP image characteristics

The **BMP** image consists of two headers. The first BMP header (14 bytes) stores the general information about the image. The **DIB** header (the first 16 bytes) stores detailed information about the BMP image properties and pixel format.

Procedure

To interact and play with BMP image values the needed functions are created.

Functions for reading the BMP and DIB header

The **BMP getBmp** function opens the file setting the cursor to position zero and then starts to read the bytes obeying the values:

- 2 bytes for finding the format of the file
- 4 bytes for getting the size of the file in decimal

- the next 4 bytes are reserved for the application which unused
- 4 bytes for the end header which contains the byte where the image starts

The **DIB getDib** function starts to collect data from the image reading the correct amount of bytes:

- 4 bytes which contain the size of the DIB header
- 4 bytes for the width of the image
- 4 bytes for the height of the image
- 2 bytes of color planes
- 2 bytes which store the value for "bit per pixels"

The functions read all the header data and store them in an array to access them easily in the future when the resulting image is created

Functions for reading the pixel values

The reading of header data is done, so the next is to read pixels that contain RGB (also alpha values if any) values from the image. The main pixel reader function **Pixels fGetPixels** invokes several functions inside. The procedure starts with initializing the dynamic array with **Pixels fInitPixels** function for the pixels. The number of all RGB value groups (also the size of the array) could be found by multiplying the height by the width obtained from the header. The main read process starts with reading every RGB (also alpha if there is any) value with **PIX fGetPix** function at the same time comparing them to get maximum and minimum pixel values. The int ifMinPix and **int ifMaxPix** functions check if the current pixel is the minimum or maximum respectively. All the RGB values are stored in the array that is initialized at the beginning of the function.

Functions for adjusting the brightness and contrast

The **Pixels fAutoAdjust** function both finds the value to multiply the pixels to get maximum contrast and next modify the brightness and contrast of the image. Once more, initializing an empty array with the size of the pixels allows for adding modified pixel values to it. The **float* fGetCoefs** is responsible for finding maximizing coefficients for RGB values. As a next step, all the pixel values are read one-by-one from the array given as an

argument sent to **PIX fEditPixel** function for editing the values to adjust brightness and contrast. The function modifies the red, green, and blue values of the pixel returning a new RGB value. All the modified pixels are written to the newly created array.

Function to collect all arrays and convert to image

After all adjusting processes are done, **int fWritePixels** starts to write the header and result array to the file that opened in write mode. The void fWriteHeader writes the BMP and DIB headers to the file given as an argument. Next, **void fWritePix** appends the modified pixel values to the file. All the processes are done, and the modified image is ready as a result.

User manual

To see results visually, the user has to compile main.c file with –lm flag.

- gcc main.c -o autoadjust -lm

The –help command provides the message about how to run the code properly. Example run:

- ./autoadjust –help

The program gets the input path to the image as a command-line argument. All the necessary errors and pre-checkings such as file validation and correct input format are handled and provided in the code. To get the result, the user should first specify the path to the input image and then the path for storing the resulting image. Example run:

- ./autoadjust input.bmp > result.bmp

The second way is to run is to mention the flag "-o" to output the result into an output file. Example run:

- ./autoadjust image.bmp -o result.bmp

Results

Example 1

There is no difference between the initial and resulting image as its lowest value is 0,0,0 (black) and highest 255,255,255 (white) which means its RGB values are already at the highest state of brightness and contrast values.

Figure 1: Test on black white color

Example 2

Testing on the image which have different RGB values rather than 2 which seen in example 1 shows the difference between input and output.

Figure 2: Test on bridge image