UFAZ DSA2 2021-22 mini-project

Manipulating Bitmap (BMP) images

You can download an example BMP image there: `https://tinyurl.com/yaj99chn`. It is a white image with a black strip in its middle.

BMP images store pixels, but because they can use different colours or formats, the file starts with a "header" that describes how pixels are stored. You can find a description of the BMP format and its headers there: `https://en.wikipedia.org/wiki/BMP_file_format`

The first header of a BMP image is as follows (important information in **bold face**):

- 2 first Bytes: 42 4D which are the ASCII values for B M (cf. BMP format)
- **4 next Bytes: the size of the file** 48 C0 12 00 in the `example.bmp` file. NB: this is "little endian"[1]: as you saw in Computer Architecture you need to inverse the bytes to convert them to an unsigned integer, So the value is 00 12 C0 48, which corresponds to 1 228 872 in decimal.
- 4 next Bytes : reserved for application. Not used.
- 4 Bytes : the offset which tells at which byte the image starts. In the `example.bmp` file : 00 00 00 46 = 70 (this is the size of both BMP and DIB headers).

Now comes the second header, the DIB header:

- 4 Bytes in little endian : size of the header starting counting from this point. Here 00 00 00 38h = 56d. Looking in the BMP format, we see this means BITMAPV3INFOHEADER (and because before the DIB, we have 14 Bytes, 14+56 = 70 size of the header before pixels data so all is well.
- **4 Bytes in little endian for image width**: here 80 02 00 00 in little endian, meaning 00 00 02 80 = 640 (the width of the image - you can verify yourself). **You will need to modify this**.
- 4 Bytes in little endian for image height: here E0 01 00 00 in little endian, means 00 00 01 E0 = 480 (you can verify for yourself).
- 2 Bytes : number of color planes (here it should be 1) (this is the case in our example bmp image).
- 2 Bytes : number of bits per pixels : here 20 00 which means 00 20 => 32
- 24 remaining Bytes : not important.

Now should come the colour table but... because we are using 32 bits per pixels, the colour table is not needed :-)

So we now start with pixels at offset 70. We find FF FF FF 00 (this means R V B are FF = white colour, then 0 for alpha channel) 270 times, then 00 00 00 00 (black) 100 times, then FF FF FF 00 270 times = 640 pixels for one line.

Hint1 : If the number of pixels is not multiple of 4, there is padding with the values 0.
Hint2 : Pixels are arranged from left to right BUT from bottom to top!

[1] https://en.wikipedia.org/wiki/Endianness

1 Automatic brightness and contrast of BMP image

Photos taken by cameras can have poor contrast or be overexposed or underexposed depending on the lighting conditions.

The objective of this project is for you to create an automatic brightness and contrast program that will take as an input a BMP image, and will automatically adjust brightness and constrast along the following principles.

1.1 Definition of Brightness

The brightness of a pixel is defined as its value. In a black/white image with pixels taking values between 0 and 255, the brightness it he value of the pixel. If you want to add brightness to a pixel, simply **add** 10 (this is an example) to the pixel value.

If you want to add brightness to a whole image, add value 10 to all the pixels of the image but of course, if the value of a pixel is already 250, adding 10 to it will change its value to 4, meaning that it will turn into a black spot, so the value of the pixel must be maximised by 255.

When this happens, information will be lost and will result in overexposed areas.

Please note that all senses from animals are logarithmic, meaning that your "feeling" that an image is 1 step brighter means that the pixel value is multiplied by 2 (or by the base of the exponential that you choose to apply).

This means that removing value 10 from a pixel value 100 may not have the same visual effect than removing 10 from a pixel value of 20: when you remove 10 from 100, you reduce brightness by 10%. When remove 10 from 20, you reduce brightness by 50%!!!

So you may want to use a more refined function than simply adding or removing an identical value to all pixels for brightness adjustment, which brings us to the notion of contrast. Please indicate so in your report if you do not use simple addition or subtraction to adjust brightness.

1.2 Definition of contrast

Contrast can be defined as the range of values that are taken by the pixels of an image. If, in a black/white image with possible pixel values between 0 and 255, all the pixels of an image have values between 100 and 150, then, contrast will be poor.

If you want to adjust contrast, you need to **multiply** pixel values by a factor, to extend (or reduce) the range of values taken by all the pixels of the image.

2 Automatically adjusting brightness and contrast

A simple way to adjust brightness and contrast consists of looking at the current contrast of the image (finding the range of the used pixels in the image, i.e. between 100 and 150).

If contrast is to be maximised, then, we will postulate that an image with maximal contrast should contain at least a pixel of brightness 0 and a pixel of brightness 255.

The simple way to obtain a maximal contrast image is (starting with an image whose pixel values vary between 100 and 150) to remove value 100 to all pixels of the image.

By doing so, all values of the image will range between 0 and 50 (brightness has been lowered by 100 for all pixels).

Then, one needs to maximize contrast by extending the range from $[0, 50]$ to $[0, 255]$. This can simply be done by multiplying all values by $255/50 = 5.1$.

After this is done, the pixel values of the image will range from 0 to 255.

3 Objective of this project

For simplicity, the example above has been taken with black and white images with pixel values ranging from 0 to 255. However, for colour images, each pixel is represented with 3 values (Red, Green, Blue) + an Alpha value (for transparency).

For a **simple** automatic brightness and contrast adjustment tool, you will find the lowest value of R, V, B components of all pixels and reduce brightness by **removing** this value from all the R, V, B components of all the pixels.

Then, you will find the highest value of R, V, B components of all pixels and maximise contrast by finding the factor by which to multiply it to get maximum brightness for this pixel, and **multiply** all the R, V, B components of all pixels of the image by this value to obtain maximal contrast.

You can find an example of a similar tool online here:
https://www.peko-step.com/en/tool/brightness_contrast.html
You can better understand about image brightness and contrast by using histograms:
https://www.creative-photographer.com/exposure-lesson-5-read-camera-histogram/

3.1 What must be done

Create a program called `autoadjust` that will take a `.bmp` file as an argument and will output (standard output) the modified values of the file.

Usage:
```
$ autoadjust image.bmp > image2.bmp
```

Please offer option "`-o`" to output the result into an output file.

Usage:
```
$ autoadjust image.bmp -o image2.bmp
```

Do not forget to display a help message if the file is called without an image file name or if the file name passed as parameter does not end with `.bmp`.

Example:

```
$ autoadjust
ERROR: you must provide a .bmp file name
$ autoadjust image.jpg
ERROR: you must provide a .bmp file name
```

4 Skills you will acquire with this project

1. Understand what is a file format.
2. Learn to deal with bytes and not integers.
3. Learn about how to use big little-endian values.
4. Learn how to open / write files.

Use a hexadecimal editor to view (and possibly modify) the BMP file to make sure everything is correct.

5 What must be handed back

1. Source codes of your project.
2. A 3 to 5 pages description of your project **written in LaTeX**, which you can have freely access to using the free UFAZ overleaf instance.

   In this description, you must provide a "user manual" describe how to launch the project, describe the command line options, error messages, ... and show the result on some images.

The project can be done by groups of 2 students.

Groups must not copy on each other.
All groups must work on their own!!!