CS0008: Introduction to Computer Programming with Python

Department of Computer Science, University of Pittsburgh

Assignment 2: Image Processing

DUE 11/3/2019 11:59PM

Goal

The goal of this assignment is to learn a little bit about image processing by programming a few simple image handling functions. Some test images are available for download on this assignment's page, with them, you will also find auxiliary functions that allow you to read and write images to your computer. Make sure to download the file bmp.py, save it to the same directory as your program's and to import it in your code. You do not need to make any modifications to bmp.py.

About images, pixels and RGB

Before diving into the code, please watch this video to learn more about RGB, pixels and bitmaps:

https://www.youtube.com/watch?v=15aqFQQVBWU

The Auxiliary Functions

In the file bmp.py you will find 2 main functions:

- **ReadBMP(filename)**: reads a .bmp file and returns a list of pixels.
- **WriteBMP(pixels, filename)**: gets a list of pixels and writes them to a file with the name specified in the 2nd argument.

When you call **ReadBMP(),** the return will be a 3-dimension list. The first dimension corresponds to the lines of the image, the second corresponds to the image columns, and the third is a list with values for the colors red, green and blue, of the pixel in that line and column. In the images for this assignment the values of each color range from [**0 – 255]**. If a pixel has RGB values equal to **[255,0,0] the red color will be the brightest**, if the pixel has the **RGB value equal to [255,255,255] it will be white**, if it's **[0,0,0] it will be black**.

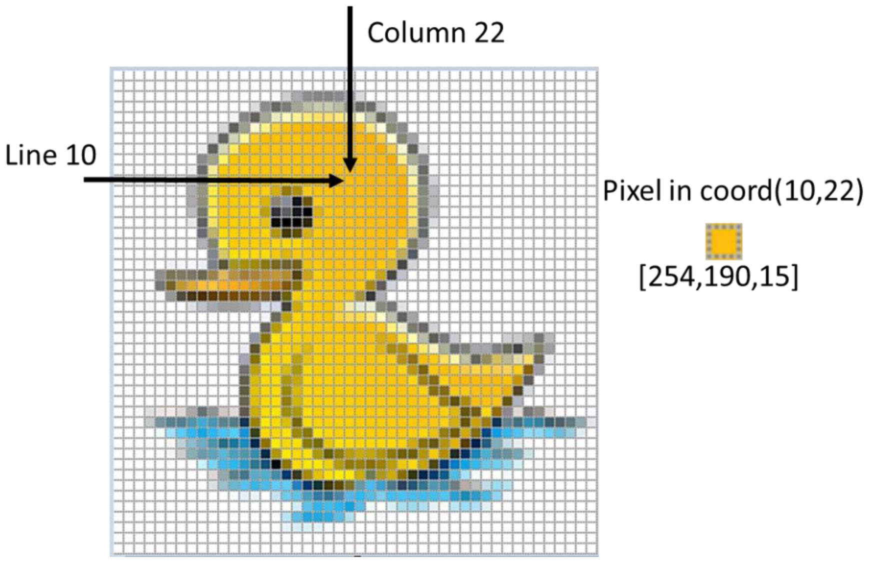
Example

pixels = ReadBmp('example.bmp')

If you load an image that's 46 pixels wide and 46 pixels high, each X and Y pair will contain a list with length 3 containing the values for the 3 colors/channels. If you want to access the **values for Red** for the pixel in **line 10 column 22** you would do:

pixels[9][21][0]

To access green and blue values you would do respectively: pixels[10][22][1] and pixels[10][22][2].



The Second Auxiliary function, **WriteBMP(pixels, filename)** is used to save the results after the processing operations were applied to the image.

Processing Functions:

For this assignment you will write **7 functions** as described below.

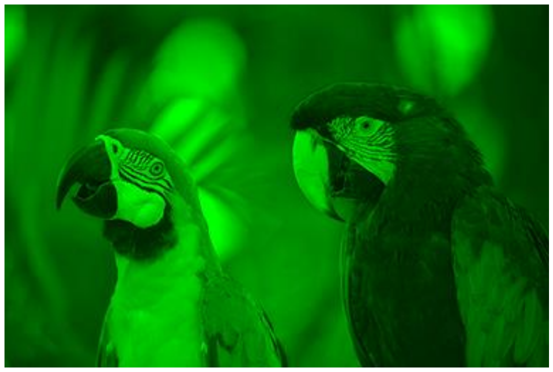
- **FUNCTION 1: Invert colors**
  Write a function **Invert()** that will read the file 'img.bmp' and process the pixel's list by assigning it to each individual pixel with the value (to each channel):

  255 - current_value

  The function must save the result with the modified colors for each pixel to a file named '**inverted.bmp'**. Remember that you have to modify the values for each channel (Red, Green and Blue). For the image provided, the final result should look something like this:

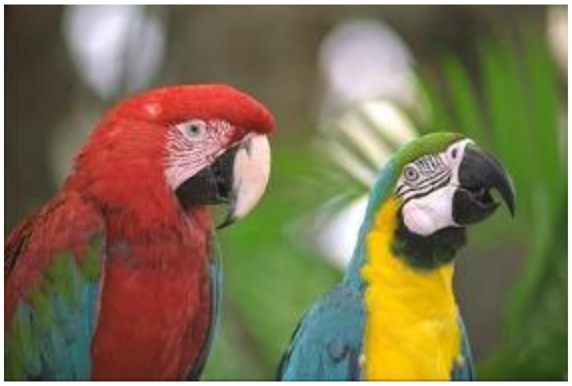
- **FUNCTION 2: Display Channel**

  Write a function **DisplayChannel()** that will read the file 'img.bmp' and ask the user which channel they want to display **(0) for Red, (1) for Green and (2) for Blue**. Depending on the answer, the function will **change the values for the other 2 channels to 0** i.e. for a pixel with RGB values of **[210,56,12]** and the user said "**green**", that pixel must be changed to **[0,56,0]**. After that, the function must save the image to a file named '**channel.bmp'.** For the image provided the result for "green" should look like this:



- **FUNCTION 3: Flip Image**

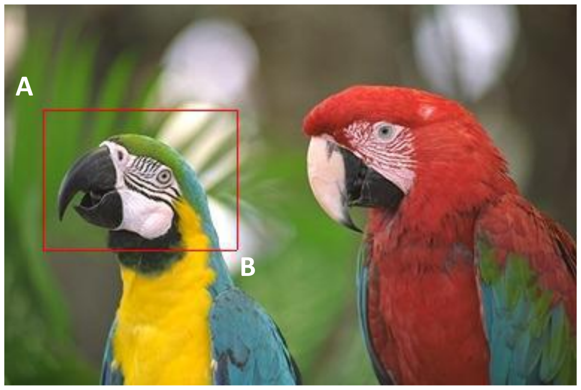
  Write a function **Flip()** that will read the file 'img.bmp' and ask the user if the image should be flipped **vertically or horizontally, flips the pixels accordingly,** and then saves the result to a file called '**flipped.bmp**'. i.e. in a 48x48 image where the input is 'horizontally' the pixel in the coordinates (30,10) should be reallocated to (18,10). *Note that the coordinates representation may change in Python
  For '**horizontally'** and the image provided, the image should look like this after the transformation:



- **FUNCTION 4: Draw Rectangle**

  Write a function **DrawRect()** that will ask the user for coordinates: A **(x1, y1)** and B **(x2, y2)**, where **x1** and **y1** are the coordinates for **the top left corner of a rectangle** and **x2** and **y2** are the **coordinates for the bottom right corner of a rectangle**. Then, for each pixel that belongs to the sides of that rectangle change the RGB value to **[255,0,0]**. The resulting image should be saved to a file named 'rectangle.bmp'

For the file img1.bmp and coordinates (26,165) for x1 and y1 and (156,71) for x2 and y2 the final image should look like this:



- **FUNCTION 5: Grayscale**

Write a function **Grayscale()** that will read the file 'img.bmp' and convert it into grayscale. To convert to grayscale you must apply the following formula pixel by pixel:

(Red + Green + Blue) //3 = gray

And assign the value "**gray**" to all 3 channels. For example: if you have a pixel **(30,128,255)**
(30+128+255)/3 = 138 then the resulting pixel will be **(138,138,138)**, after that, your program must save the result to a file called '**grayscale.bmp**'. For the sample image the result would be the following:

- **FUNCTION 6: Brightness**

Write a function **Brightness()** that will read the file 'img.bmp' and asks the user if they want to increase (**1**) or decrease (**2**) brightness in a **loop** until the user hits (**0**) to quit.
To increase the brightness in your picture you must add **25 points** to each channel for every time the user asks to increase image brightness. For a pixel [250, 20,100] the new values would be [255, 45, 125], **make sure that the values will not go over 255.**
To decrease the brightness in your picture you must subtract **25 points** from each channel for every time the user asks to reduce the image brightness. For a pixel [250, 20,100] the new values would be [225, 0, 75], **make sure that the values will not go under 0.**

After that, the function must save the image to a file named ‘**brightness.bmp’.**

For the sample image the result would be the following for **3 times increase brightness**:

For the sample image the result would be the following for **3 times decrease brightness**:

- **FUNCTION 7: Contrast**

Write a function **Contrast()** that will read the file ‘img.bmp’ and asks the user if they want to increase (**1**) or decrease (**2**) image contrast in a **loop** until the user hits (**0**) to quit.
To increase the contrast in your picture you must set **C** to **45 points,**
To decrease the contrast you must set **C** to **-45 points**,
After that you must calculate the contrast change factor using the formula:

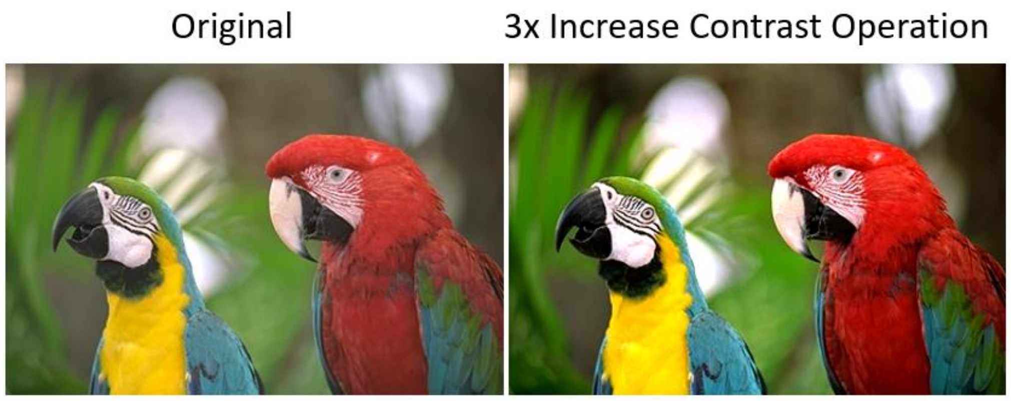
$$factor = \frac{259 \times (C + 255)}{255 \times (259 - C)}$$

And then apply the following formula for each channel for every pixel:

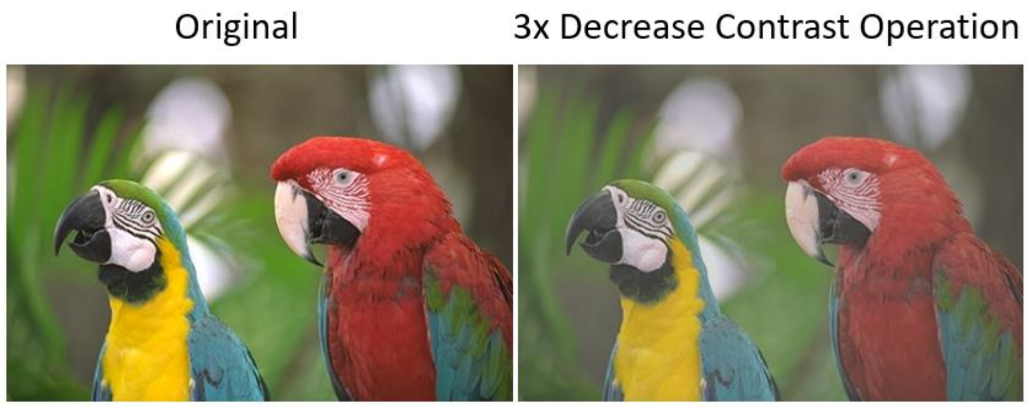
$$new_{value} = factor \times (cur\_value\_channel - 128) + 128$$

For the increase contrast operation and a pixel [250, 20,100], the factor would be 1.4238 and the new values would be [255, 0, 88], **make sure that the values will not go under 0 or over 255.**
For the decrease contrast operation and a pixel [250, 20, 100], the factor would be 0.7016 and the new values would be [213, 52, 108]
After that, the function must save the image to a file named ‘**contrast.bmp’.**

For the sample image the result would be the following for **3 times increase contrast** operation:



For the sample image the result would be the following for **3 times decrease contrast operation**:



GOOD LUCK!!!