

Hieu Nguyen
GTID: 903185448
Email: hieu@gatech.edu
8/31/15

CS6475 Computational Photography Assignment #2

In the PDF we want you to write how you approached each of the five functions above (a brief description), and we want you to include input / output images (of your choosing) for functions 3, 4, and 5. Please include your name and GTID in this PDF.

1. `numberOfPixels()`

This function returns the total number of pixels in an image (i.e. `image_width * image_height`). The input grayscale image is represented as an array, with each element corresponding to a pixel. Thus, the size of the array would also give the amount of pixels. I verified that this matched the calculated value of `width * height`.

2. `averagePixel()`

This function returns the average color value of a grayscale image, which is the sum of all pixels divided by the total number of pixels. I used the `numpy.sum()` function to add all elements in the array, then divided by the total pixels calculated from `numberOfPixels()`. I also rounded the average result (a float) and casted to `int` data type to match the expected output in the test program. I was initially worried about possible overflow issues, but my result was reasonable and fell within the 0 to 255 range.

3. `convertToBlackAndWhite()`

This function generates a 1-bit image based on if the color value is higher or lower than 128. I iterated over the image array with a `for` loop, and checked each pixel value to assign a new “binary” value in an output array. This function took a few seconds to run, as it had to iterate through the entire array. I wonder if there is a faster method using vector operations...

Input:



Output:



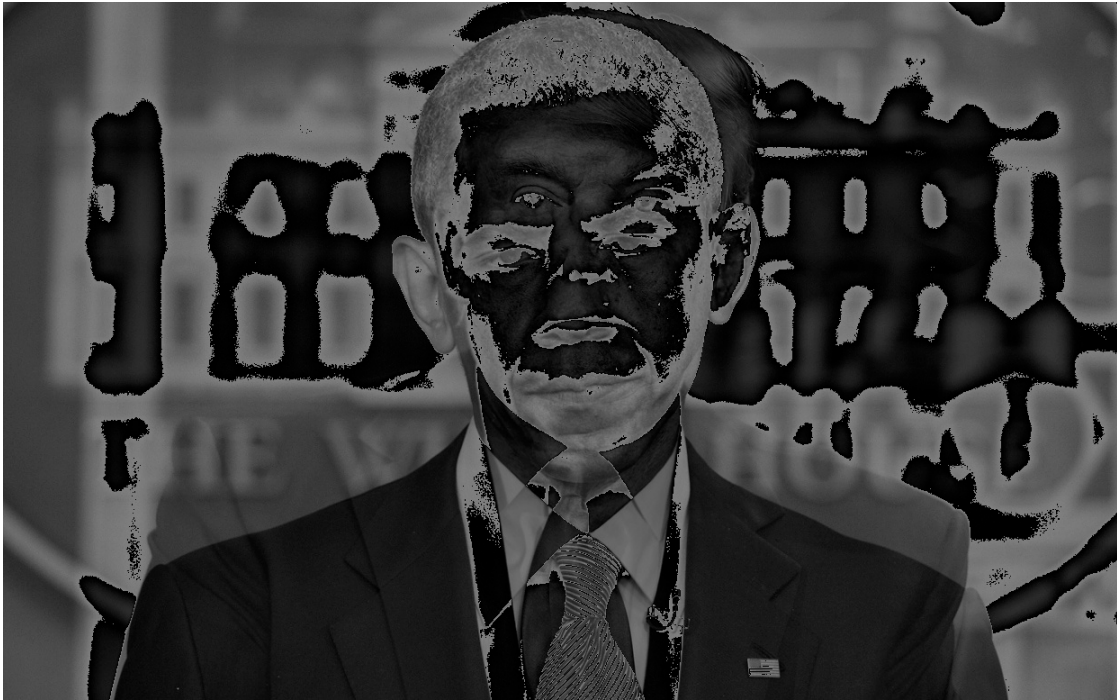
4. `averageTwoImages()`

This function averages the pixels of two input images. For this, I just simply added the corresponding elements in each array, then divided by two. The output seemed to make sense as a merging of both images (like a flattened overlay). I didn't use any special functions, but I'm wondering if one exists for this particular use case.

Input:



Output:



5. `flipHorizontal()`

This function essentially mirrors the image across the horizontal axis. I used a special OpenCV function, `flip()`, to perform this image transformation. The `flip()` function takes in an input image and a parameter to signify the type of flip. I used a '1' to indicate a flip on the horizontal axis.

Input:



Output:

