Assignment 2

by

Jassimran Kaur
jassimran@gatech.edu
GTID: jkaur9

Georgia Institute of Technology

CS 6475- Computational Photography
Assignment 2
May 19, 2015

1. “numberOfPixels”

Properties of an Image can be accessed by `img.shape`. It returns a tuple of number of rows, columns and channels (if image is color). Since given image is grayscale, tuple returned contains only number of rows and columns. Since we know that number of pixels in an image is product of number of rows and number of columns, I used:

```
image.shape[0]*image.shape[1]
```

2. “averagePixel”

Sum() is among the basic array operations provided by NumPy. It is use to add the items in an array. It was advised to use the “numberOfPixels” method written above. Hence I divided `img.sum()` with output of “numberOfPixels” method to get the average color value of the given grayscale image.

```
a = image.sum()/numberOfPixels(image)
```

Since an “int” value was expected to be returned, I used conversion `int(a)` to get the desired output.

3. “convertToBlackAndWhite”

My first approach was to loop through each pixel and check if pixel is greater than 128 or not and update the values accordingly. But it took more than 10s to process the image. Hence I used NumPy’s “where” method which was much more efficient.

```
numpy.where(condition[, x, y])
```

It returns elements, either from *x* or *y*, depending on *condition*. The given condition is if values are greater than 128, then set pixel to 255, else to 0. Hence condition is equal to image > 128 ; x = 255 and y = 0.

```
np.where(image > 128, 255, 0)
```

Using this approach, here are results:

Input Image

Output Image

4. “averageTwoImages”

My first approach was to loop through each pixel and average the values of two images and store in a resultant image. But, it was very inefficient and took about 30s, so I use NumPy to add two images and divide them by 2.

```
res = (img1+img2)/2
```

But input images were of “uint8” type and due to addition values would often go above 255. Hence I converted input images to “float” type before applying array addition, as shown below:

```
img1 = image1.astype('float')
img2 = image2.astype('float')
```

Once the average image was formed, I converted it back to “uint8”, as shown below:

```
result = res.astype('uint8')
```

Using this approach, here are results:

InputImage 1

InputImage2

Average Output Image

5. “flipHorizontal”

There is flipping function “fliplr” available in NumPy, it flips an array in the left/right direction, `numpy.fliplr(m)`. I used it on the input image to get the desired horizontally flipped output image as shown below:

```
res = np.fliplr(image)
```

Using this approach, here are results:

Input Image

Output Image