CS 4475 Project 2 Report

Sanghyun Chun

Shc1991@gatech.edu

902759995 (schun41)

Introduction

This project involves taking multiple images to create “tiles” that will be used to modify a given image and turn it into a “mosaic” type aesthetics.

Workflow

Here’s an image I’d like to make into a mosaic:

Let’s go step by step in the process.

1. Choose an image you want to transform, and save it in the same folder as myproject3.py as “original.jpg.” The format needs to be JPG. The size/dimension of the image does not matter.
2. You have to analyze what colors constitute your image. In this example, the flag has white, black, red, and blue.
3. After you have your basic idea of what colors you need, you need to find enough images to replace those pixels with. Your images may come from several sources; your personal photos, or from the Internet. Save them all in the “sources” folder. Once again, size does not matter, but the format needs to be a JPG.
4. Run myproject3.py. Let’s break down each function:
    a. resizeImage: This takes in a list of all the images you got in your “sources” folder, resizes

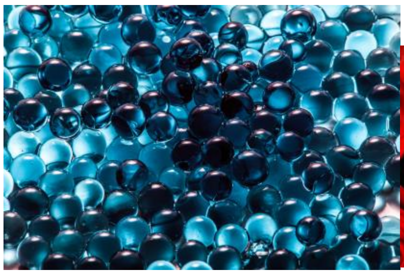
each image to a 15 x 12 image, and saves them all in the “mosaics” folder. I used os.listdir(\path) to automatically turn all the images into a list.

b. averageRGB: This function calculates the average RGB value of a given image.
c. averageTileRGB: This function uses averageRGB to calculate the average RBG value of a given image within its dimension.
   pixelSum = sum of the image’s RBG values.
   pixelCount = total number of pixels in that image.
   The average is simply pixelSum / pixelCount.
d. createMosaic: This function compares each of the original image’s pixels, and replaces it with the tile image that has the closest RGB value. The pixels in the original image is calculated in a block of 15 x 12, the same size as the tile images.
e. createNewImage: This is where the new image is created. I first make mosaic_img, which is just an empty array of 0s which will be filled up. The pixels must be replaced in a 15 x 12 basis. It is important to note that OpenCV stores image coordinates **backwards (height / width)**. So when running a double for loop, we must start with the height value:

   ```
   for h in image.height
       for w in image.width
   ```

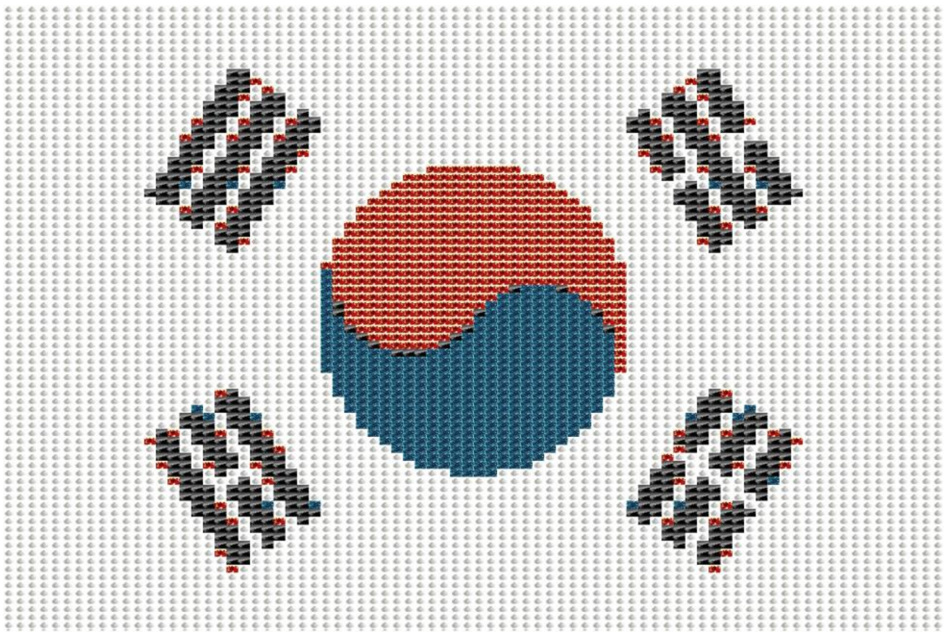
   I thresholded the RGB difference using np.sqrt(3 * np.power(128, 2)). 128 is my arbitrary choice: it is the middle intensity value of a grayscale (0 – 255). The RGB difference is calculated with the following formula: origImage[i][j] – averageRGB(img).
   If the difference is smaller than the threshold, replace threshold value with the difference value. For each j runs in the loop, call createMosaic to replace pixels with appropriate tile image.
f. Save createNewImage as a variable, then cv2.imwrite it, calling it “output.jpg.”

Continuing our example, here are the tiles images I will use:



After resizing, they will look like this:

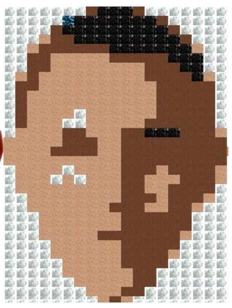
The final image will look like this:



Discussion

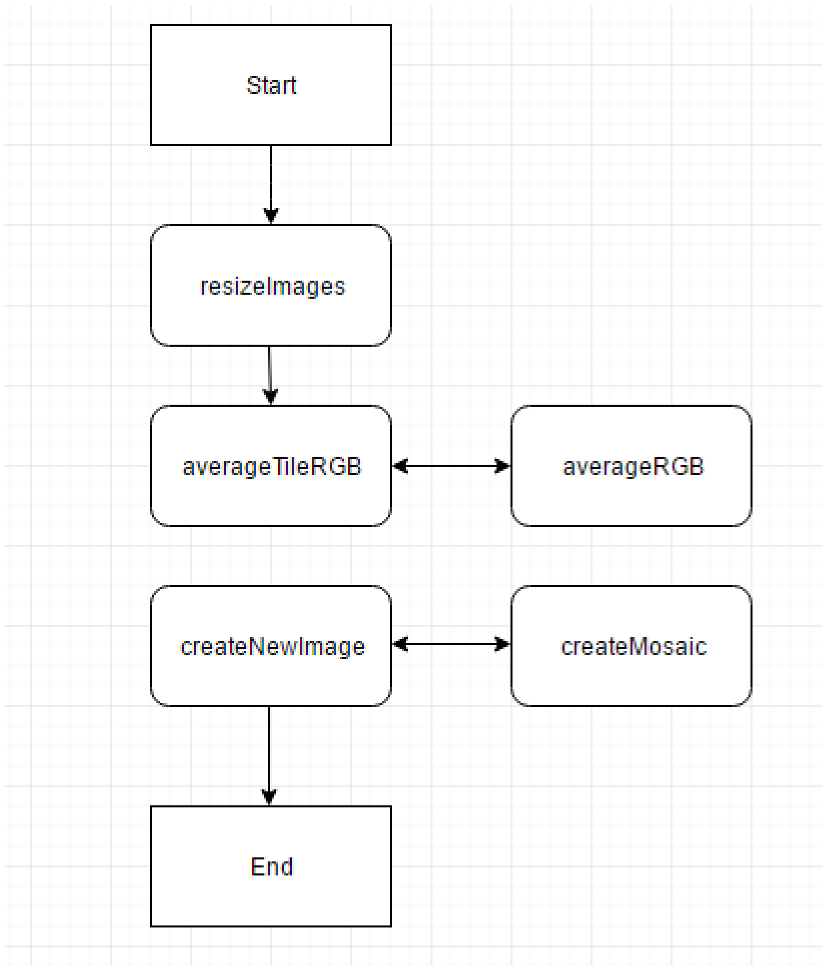
There are a few things to note about this program:

1. The more source images the user has, the more accurate the output will be. I used flags because they are relatively easy to turn into mosaics. Images with complex RGB schemes and contrasts will be harder to render. For instance, this is my Obama face:

   As you can see, I only have a limited number of sources images, therefore the end result is much less detailed.
2. Generally, the higher the original image's resolution is, the better the output will be, since each tile will be able to be placed more compactly.

Flowchart



Artistic / Technical Emphasis

This is the longest coding I've done out of the 3 projects. I put some thoughts in how to calculate the RGB values and replacing them with source images. This project incorporates a lot from what we learned in class, especially in terms of manipulating dimensions and colors. Also, formulas such as calculating thresholds (np.sqrt(3 * np.power(128, 2))) involves mathematical and technical knowledge.

On the other hand, the program yields aesthetical results, if done correctly. Admittedly, the more sources images one has, the more detailed and accurate the final image will be, especially when the original image has lots of colors. If one puts in the effort, however, the output is pleasing to the eye and original.

Technicality: 25%

Artistic Merit: 25%

Credit

Images were found at:
https://upload.wikimedia.org/wikipedia/commons/thumb/0/09/Flag_of_South_Korea.svg/2000px-Flag_of_South_Korea.svg.png
http://specials-images.forbesimg.com/imageserve/573c6b334bbe6f63618536ea/0x0.jpg?fit=scale&background=000000
https://wallpaperscraft.com/image/roses_petals_flower_red_90476_3840x2160.jpg
https://static.pexels.com/photos/1346/blue-abstract-balls-spheres.jpg
http://eskipaper.com/images/black-backgrounds-11.jpg
http://www.parloursupply.com/media/catalog/product/cache/14/image/9df78eab33525d08d6e5fb8d27136e95/p/r/productshot_960x500_pulver.png
http://pngimg.com/upload/face_PNG5660.png
http://orig13.deviantart.net/037d/f/2012/042/f/4/seamless_human_skin_by_hhh316-d4pfpwv.jpg
http://kingofwallpapers.com/brown/brown-024.jpg

RESOURCES

https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV/LABS_2013_2014/Lab_ImageMosaic.html
http://answers.opencv.org/question/88376/convert-raw-image-into-mosaic-image-representation/
http://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html
http://docs.scipy.org/doc/numpy/reference/generated/numpy.zeros.html