CSE 107: Lab 2: Simple Image Manipulations in Python

Bryant Chon
Lab: Thursday 4:30-7:20pm
Yuxin Tian
October 3, 2022

Task 1: Computing the maximum value of an image. Rotating an image.

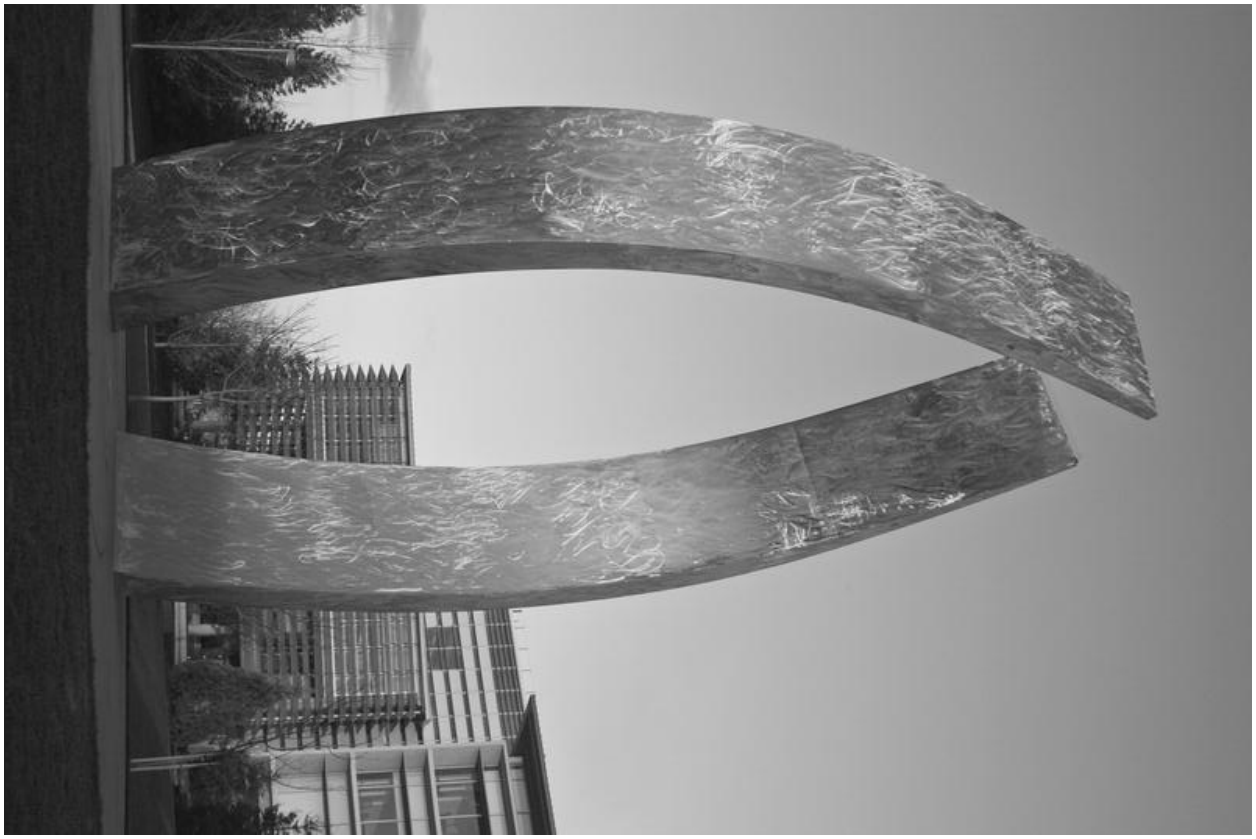

Figure 1: The Beginnings image rotated 90 degrees clockwise

Questions for Task1:

1. 246
2. 246
3. Yes, they should be the same because they are the same image just rotated.
4. Visualizing the pixel location of the rotated images and then coding the for loops correctly.

Task 2: Writing a function that computes the inverse of a grayscale image.

Figure 2: The inverse of the Watertower image.

Questions for Task 2:

1. 255
2. The difference between every pixel from the original and inverted photo is 255.
3. Realizing what inverted meant in terms of a grayscale image.

Task 3: Creating a gradient grayscale image. Computing the image average.

Figure 3: The gradient image.

Questions for task 3:
1. 127.5
2. I expected this because 127.5 is exactly half of 255.
3. Realizing I had to divide the total pixel by 256 in order to get the average.

Task1.py

```
#import pillow
from PIL import Image, ImageOps
#import numpy
import numpy as np
from numpy import asarray

# open a image from path
im = Image.open('Beginnings.jpg')

# show the image
im.show()

# convert image to grayscale, get path, save image
im_gray = ImageOps.grayscale(im)
im_gray_path = 'my_gray_beginnings.jpg'
im_gray.save(im_gray_path)

#show the image
im_gray.show()


# get pixel values, rows and cols
im_gray_pixels = asarray(Image.open(im_gray_path))
rows, cols = im_gray_pixels.shape
current_Maxpixel_value = 0

# get max pixel value
for row in range(rows):
    for col in range(cols):
        if current_Maxpixel_value < im_gray_pixels[row, col]:
            current_Maxpixel_value = im_gray_pixels[row, col]

# print max pixel value
print("current max pixel value is: ", current_Maxpixel_value)

# initialize counterclock empty array, rows and cols for counterclock
image, get rows and cols for original image
rowws = cols - 1
roww = rowws
```

```
coll = 0
rows, cols = im_gray_pixels.shape
counterclock = np.empty([cols,rows], dtype = int)

# rearrange pixels of original to counterclock image
for row in range(rows):
    for col in range(cols):
        counterclock[roww, coll] = im_gray_pixels[row,col]
        if roww >= 0:
            roww -= 1
        if roww == -1:
            roww = rowws
            coll += 1

# get pixel values, save/load image
data = Image.fromarray(counterclock)
data_path = 'im_gray_counterclock.png'
data.save('im_gray_counterclock.png')
im = Image.open('im_gray_counterclock.png')

# show the image
im.show()

# clockwise: initialize like counterclock
colls = rows - 1
rowws = cols - 1
roww = 0
coll = colls
rows, cols = im_gray_pixels.shape
clockwise = np.empty([cols,rows], dtype = int)

# rearrange pixel to create clockwise rotation
for row in range(rows):
    for col in range(cols):
        clockwise[roww, coll] = im_gray_pixels[row,col]
        if roww <= rowws:
            roww += 1
        if roww > rowws:
            roww = 0
            coll -= 1
```

```
# get pixel values, save/load image
data = Image.fromarray(clockwise)
data_path = 'im_gray_clockwise.png'
data.save('im_gray_clockwise.png')
im = Image.open('im_gray_clockwise.png')

# show the image
im.show()

# get pixel values
im = asarray(Image.open('im_gray_clockwise.png'))
rows, cols = im.shape
current_Maxpixel_value = 0
# get all the pixel values using the index
for row in range(rows):
    for col in range(cols):
        if current_Maxpixel_value < im[row, col]:
            current_Maxpixel_value = im[row, col]

#print max pixel value
print("current max pixel value for gray clockwise is: ",
current_Maxpixel_value)
```

Task2.py

```
#import pillow
from PIL import Image, ImageOps
#import numpy
import numpy as np
from numpy import asarray
import MyImageFunctions

# open a image from path
im = Image.open('Watertower.tif')
print("image mode is: ", im.mode)


# show the image
im.show()

#get pixel values
im_gray_pixels = asarray(im)

# invert original image, save/open image
inverse = MyImageFunctions.myImageInverse(im_gray_pixels)
data = Image.fromarray(inverse)
data_path = 'im_inverse.tif'
data.save('im_inverse.tif')
im = Image.open('im_inverse.tif')

# show the image
im.show()

# get pixels of inverted image
im_inverse_pixels = asarray(im)

# initialize rows cols max pixel variables
rows, cols = im_inverse_pixels.shape
current_Maxpixel_value = 0

# get all the pixel values using the index
for row in range(rows):
    for col in range(cols):
        if current_Maxpixel_value < im_inverse_pixels[row, col]:
```

```
            current_Maxpixel_value = im_inverse_pixels[row, col]

# print max pixel value
print("current max pixel value is: ", current_Maxpixel_value)
```

MyImageFunctions.py

```
#import pillow
from PIL import Image, ImageOps
#import numpy
import numpy as np
from numpy import asarray

# This function takes as input a numpy matrix representing a grayscale
image and
# outputs another numpy matrix which is the image inverse of the input.
# That is, for each pixel, output_value = 255 - input_value
#
# Syntax:
# out_numpy_matrix = myImageInverse( in_numpy_matrix )
#
# Input:
# in_numpy_matrix = the grayscale values of the input image
#
# Output:
# out_numpy_matrix = the grayscale of the inverse image
#
# History:
# Bryant Chon 10/3/22 Created

def myImageInverse(inImage):
    rows, cols = inImage.shape
    inverse = np.empty([rows,cols], dtype = int)
    for row in range(rows):
        for col in range(cols):
            inverse[row, col] = 255 - inImage[row,col]

    return inverse
```

Task3.py

```
#import pillow
from PIL import Image, ImageOps
#import numpy
import numpy as np
from numpy import asarray

# initialize gray gradient variable
GrayGradient = np.empty([100,256], dtype = int)
rows , cols = GrayGradient.shape
val = 0

# create gradient array
for row in range(rows):
    for col in range(cols):
        GrayGradient[row, col] = val
        val += 1
        if val > 255:
            val = 0

# save/open/show image
data = Image.fromarray(GrayGradient)
data_path = 'im_GrayGradient.tif'
data.save('im_GrayGradient.tif')
im = Image.open('im_GrayGradient.tif')
im.show()

# get average pixel value, print value
val = 0
for row in range(1):
    for col in range(cols):
        val += GrayGradient[row, col]

#print avg pixel
print("The average pixel value is: ", val/256)
```