CSE 107: Lab 02: Simple Image Manipulations in Python

Angelo Fatali
LAB: R 7:30-10:20pm
Liang, Haolin
September 20, 2022

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

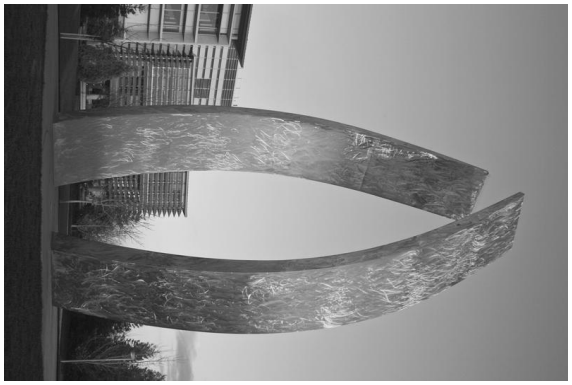

Figure 1: The Beginnings.jpg image rotated 90 degrees clockwise.

Questions for task 1:

1. What is the maximum pixel value of your grayscale Beginnings image?
   - 240
2. What is the maximum pixel value of your clockwise rotated grayscale image?
   - 240
3. Should these be the same? Why or why not?
   - They should be the same since none of the pixel values were changed, only the position changed.
4. What was the most difficult part of this task?
   - Figuring out how to rotate the image with a nested loop rather than using the numpy.rot90() function.

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

Figure 2: The inverse of the Watertower.tif image

Questions for task 2:

1. What is the maximum pixel value of your inverse image?
   - 255
2. How is this maximum value related to the values of the original image?
   - The max value to the original image is also 255. Even though the pixel values are changing, we're still relatively keeping the same values just the inverse, so anything pure black is now the highest value and anything pure white is now the lowest value.
3. What was the most difficult part of this task?
   - Writing a comment to the method header I find to be more tedious than difficult; but otherwise respectively easy.

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

Figure 3: The gradient.tif image.

Questions for task 3:

1. What is the average pixel value in your gradient image?
   - 127.5
2. Why did you expect to get this value from the gradient image?
   - This value was expected because there are 0-255 values and since we're doing an incremental value in each column, we can expect our average to be 255/2.
3. What was the most difficult part of this task?
   - Remembering to do -1 since pixels go up to 255 not 256.

Task1.py

```
# Import pillow
from PIL import Image, ImageOps

# Import numpy
import numpy as np
from numpy import asarray

# Read the image “Beginnings.jpg”.
img = Image.open('Beginnings.jpg')

# Display the image on the screen.
img.show()

# Convert the image from color to grayscale.
img_gray = ImageOps.grayscale(img)

# Display the image on the screen again.
img_gray.show()

# Create a numpy matrix that has the pixel values from the image.
img_gray_pixels = asarray(img_gray)
rows, cols = img_gray_pixels.shape
# print(img_gray_pixels)

# Use nested for loops to compute the maximum pixel value in the numpy matrix
and print this value out to the terminal. Note, you cannot use any built-in
functions to compute the maximum—you must loop through the pixel values.
max = img_gray_pixels[0,0]
for row in range(rows):
    for col in range(cols):
        if(img_gray_pixels[row,col] > max):
            max = img_gray_pixels[row,col]
print("max pixel value: " + str(max))

# Create a new numpy matrix which is the original matrix rotated by 90
degrees
# counterclockwise. The Beginnings grayscale image should have
dimensions 800 (rows) x
# 533 (columns). Your new matrix should have dimensions 533 x 800
(but don’t hardcode
# these values—use the number of rows and columns of the Beginnings
grayscale image).
# Steps to do this:
# - Create a new blank numpy matrix.
# - Use nested for loops to copy the pixel values from the original matrix
image to
# the counterclockwise rotated one.
rows, cols = img_gray_pixels.shape
img_gray_pixels_rotated_CCW = np.zeros(shape=(cols, rows))
for col in range(0, cols):
    for row in range(0, rows):
        img_gray_pixels_rotated_CCW[col][row] = img_gray_pixels[row][col]

# Create an image from the rotated matrix.
img_gray_rotated_CCW = Image.fromarray(np.uint8(img_gray_pixels_rotated_CCW))
```

```
# Display the counterclockwise rotated image on the screen.
img_gray_rotated_CCW.show()

# Write the counterclockwise rotated image to a file.
img_gray_rotated_CCW.save("Beginnings_grayscale_rotated_CCW.jpg")

# Create a new numpy matrix which is the original matrix rotated by 90
degrees clockwise.
rows, cols = img_gray_pixels.shape
img_gray_pixels_rotated_CW = np.zeros(shape=(cols, rows))
for col in range(0, cols):
    for row in range(0, rows):
        img_gray_pixels_rotated_CW[col][row] = img_gray_pixels[rows-row-1]
[cols-col-1]

# Create an image from the rotated matrix.
img_gray_rotated_CW = Image.fromarray(np.uint8(img_gray_pixels_rotated_CW))

# Display the counterclockwise rotated image on the screen.
img_gray_rotated_CW.show()

# Write the counterclockwise rotated image to a file.
img_gray_rotated_CW.save("Beginnings_grayscale_rotated_CW.jpg")

# Compute and print the maximum pixel value of the clockwise rotated image.
Again, you must compute the maximum value yourself using nested for loops.
max = img_gray_pixels_rotated_CW[0,0]
rows, cols = img_gray_pixels_rotated_CW.shape
for row in range(rows):
    for col in range(cols):
        if(img_gray_pixels_rotated_CW[row,col] > max):
            max = img_gray_pixels_rotated_CW[row,col]
print("max pixel value of clockwise rotated image: " + str(int(max)))
```

Task2.py

```
# Import pillow
from PIL import Image, ImageOps

# Import numpy
import numpy as np
from numpy import asarray

# Import Inverse Function
import MyImageFunctions

# Read the image “Watertower.tif”
img = Image.open('Watertower.tif')

# Display the image on the screen.
img.show()

# Create a numpy matrix that has the pixel values from the image.
img_pixels = asarray(img)
rows, cols = img_pixels.shape
```

```
# Call  the  function  myImageInverse()
inverse_img_pixels = MyImageFunctions.myImageInverse(img_pixels)

# What is the maximum pixel value of your inverse image?
max = inverse_img_pixels[0,0]
for row in range(rows):
    for col in range(cols):
        if(inverse_img_pixels[row,col] > max):
            max = inverse_img_pixels[row,col]
print("max pixel value of inverse image: " + str(max))

# How is this maximum value related to the values of the original image?
max = img_pixels[0,0]
for row in range(rows):
    for col in range(cols):
        if(img_pixels[row,col] > max):
            max = img_pixels[row,col]
print("max pixel value of original image: " + str(max))


# Create an image from the returned matrix.
inverse_img = Image.fromarray(np.uint8(inverse_img_pixels))

# Display the image on the screen.
inverse_img.show()

# Write the image to a .tif file.
inverse_img.save("Watertower_inverse.tif")
```

MyImageFunctions.py

```
# Import pillow
from PIL import Image, ImageOps

# Import numpy
import numpy as np
from numpy import asarray

def myImageInverse(inImage):
    # This function takes in a numpy matrix of grayscale image pixels and
    # outputs another numpy matrix of grayscale image pixels that is inverse
    #
    # Syntax:
    # out_matrix = myImageInverse(in_matrix)
    #
    # Input:
    # in_matrix = the grayscale pixel values of the input image
    #
    # History:
    # A. Fatali 9/26/22 Created
    rows, cols = inImage.shape
    outImage = np.zeros(shape=(rows, cols))
    for row in range(0, rows):
        for col in range(0, cols):
            outImage[row][col] = 255-inImage[row][col]

    return outImage
```

Task3.py

```
# Import pillow
from PIL import Image, ImageOps

# Import numpy
import numpy as np
from numpy import asarray

# Create a grayscale image of size 100 rows x 256 columns in which the value
of each row varies from 0 in the left column to 255 in the right
column. Thus, the image contains a grayscale gradient from black on the
left to white on the right.
img_gray_pixels = np.zeros(shape=(100, 256))
rows, cols = img_gray_pixels.shape
for row in range(0, rows):
    for col in range(0, cols):
        img_gray_pixels[row,col] = col

# Create an image from the new matrix.
img_gray = Image.fromarray(np.uint8(img_gray_pixels))

# Display the image on the screen.
img_gray.show()

# Save the image as a .tif file.
img_gray.save("Gradient.tif")

# Compute the average pixel value of the gradient image. You must use nested
for loops to do this. You are not allowed to use any built-in functions to
compute the average.
sum = 0
for row in range(rows):
    for col in range(cols):
        sum = sum + int(img_gray_pixels[row][col])
avg = sum / (cols*rows)
print("avg pixel value of gradient image: " + str(avg))
```