CSE 107: Lab 02: Simple Image Manipulations in Python.

Martin Urueta
LAB: Thursday 7:30pm-10:20pm
Liang, Haolin
September 20, 2022

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

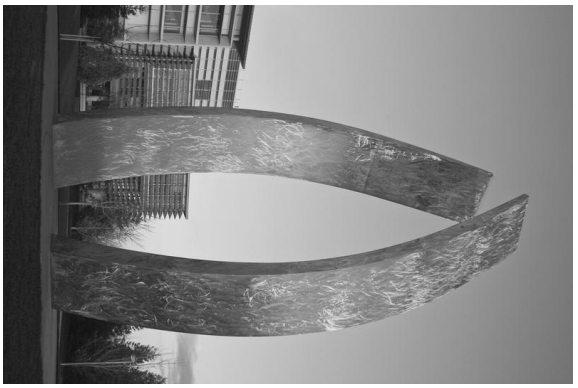

Figure 1: The beginnings Image rotated 90 degrees clockwise.

Questions:

1. The maximum pixel value of my grayscale Beginnings image is 240
2. The maximum pixel value of my clockwise rotated grayscale image is 240
3. These should be the same since we have not affected the pixel values. The only thing that was affected in the matrixes is that the pixel values have been relocated.
4. The most difficult part of this task is the rotation of the image clockwise

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

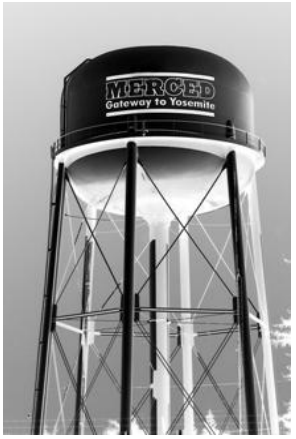

Figure 2: The inverse of the Watertower image.

Questions:

1. The maximum pixel value of my inverse image is 255
2. The maximum value of the original image is 255 as well. This is because inverting a dark pixel that has a value of 0 turns to the brightest pixel of 255 being white. Since the original image has the minimum value to be 0 and max value to be 255. And to inverse of that value it will still has a minimum value of 0 and max value of 255.
3. The most difficult part of this task was doing the function

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

Figure 3: The gradient image.

Questions:

1. The average pixel value in my gradient image is 127.5
2. I expect to get this value because the lowest pixel value is 0 (black) and the highest pixel value is 255 (white) and you would get 127.5 since you would add up all the values possible to get and divide the amount of possible values you will get 127.5. Another way to figure it out is by adding the (max + min) / 2 to get your average value.
3. The most difficult part is creating the matrix to create a image

Task1.py

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

# Import numpy
import numpy as np
from numpy import asarray

# Read the image
im = Image.open('Beginnings.jpg')

# Show the image.
im.show()

# Convert image to gray scale.
im_gray = ImageOps.grayscale(im)

#Show gray image
im_gray.show()

#Save gray image
im_gray.save("Beginnings_grayscale.jpg")

# Get access to the pixel values through the matrix im_gray_pixels.
im_gray_pixels = asarray(im_gray)
rows, cols = im_gray_pixels.shape

# Determine the dimensions of the image.
print("Image size is: ", rows, "rows x", cols, "columns")

# Default value to find max value pixel
maxpixel = 0

# Find max pixel value in for loop
for row in range(0, rows):
    for col in range(0, cols):
        if maxpixel < im_gray_pixels[row, col]:
            maxpixel = im_gray_pixels[row, col]

# Print max pixel value
print("maximum pixel value is :", maxpixel)

# Change the rows and cols of the new matrix
ninetydegrees_rows = cols
ninetydegrees_cols = rows
ninetydegrees_ccw_pixels = np.zeros(shape=(ninetydegrees_rows,
ninetydegrees_cols))

# Copy values of from old image but change the rows and cols to ccw rotate image
for row in range(0, ninetydegrees_rows):
```

```
    for col in range(0, ninetydegrees_cols):
        ninetydegrees_ccw_pixels[row, col] = im_gray_pixels[col, row]

# Create image
ninetydegrees_ccw_im_gray = Image.fromarray(np.uint8(ninetydegrees_ccw_pixels))

# Show image
ninetydegrees_ccw_im_gray.show()

# Save Image
ninetydegrees_ccw_im_gray.save("Beginnings_grayscale_counterclockwise.jpg")

# create new matrix
ninetydegrees_cw_pixels = np.zeros(shape=(ninetydegrees_rows,
ninetydegrees_cols))

# Copy values of from old image but change the rows and cols to cw rotate image
for row in range(0, ninetydegrees_rows):
    for col in range(0, ninetydegrees_cols):
        ninetydegrees_cw_pixels[row, col] =
im_gray_pixels[(ninetydegrees_cols-col-1), (ninetydegrees_rows-row-1)]

# Create image
ninetydegrees_cw_im_gray = Image.fromarray(np.uint8(ninetydegrees_cw_pixels))

# Show image
ninetydegrees_cw_im_gray.show()

# Save Image
ninetydegrees_cw_im_gray.save("Beginnings_grayscale_clockwise.jpg")

# Find max pixel in for loop
for row in range(0, ninetydegrees_rows):
    for col in range(0, ninetydegrees_cols):
        if maxpixel < ninetydegrees_cw_pixels[row, col]:
            maxpixel = ninetydegrees_cw_pixels[row, col]

# Print max pixel value
print("maximum pixel value is :", maxpixel)
```

Task2.py

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

# Import numpy
import numpy as np
from numpy import asarray

#import MyImageFunctions.py
from MyImageFunctions import myImageInverse


# Read the image
im = Image.open('Watertower.tif')

# Show the image.
im.show()

# Get access to the pixel values through the matrix.
im_pixels = asarray(im)
rows, cols = im_pixels.shape

# Default value to find max value pixel
max_pixel = 0

# Find max pixel value in for loop
for row in range(0, rows):
    for col in range(0, cols):
        if max_pixel < im_pixels[row, col]:
            max_pixel = im_pixels[row, col]

# Print max pixel value
print("maximum pixel value of org image is :", max_pixel)

#inversed the Image/matrix using myImageInverse function and output matrix
inversed_pixels = myImageInverse(im_pixels)

# Default value to find max value pixel
max_inversed_pixel = 0

# Find max pixel value in for loop
for row in range(0, rows):
    for col in range(0, cols):
        if max_inversed_pixel < inversed_pixels[row, col]:
            max_inversed_pixel = inversed_pixels[row, col]

# Print max pixel value
print("maximum pixel value of inversed image is :", max_inversed_pixel)

# Create image
```

```
inversed_im = Image.fromarray(np.uint8(inversed_pixels))

# Show image
inversed_im.show()

# Save Image
inversed_im.save("inversed_Watertower.tif")
```

MyImageFunctions.py

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

# Import numpy
import numpy as np
from numpy import asarray

def myImageInverse(inImage_pixels):
# This function takes as input of numpy matrix representing a image and outputs
another numpy matrix which is the inverse of the input
# For each pixel, output_value = 255 - input_value
#
# Syntax:
#   inversed_pixels = myImageInverse(im_pixels)
#
# Input:
#   im_pixels = the matrix values of the orginal input image
#
# Output:
#   inversed_pixels = the matrix values of the inverse image
#
# History:
#   M. Urueta   9/19/22     Created

    rows, cols = inImage_pixels.shape
    Copy_inImage_pixels = np.zeros(shape=(rows, cols))

    for row in range(0, rows):
        for col in range(0, cols):
            Copy_inImage_pixels[row, col] = 255 - inImage_pixels[row, col]

    return Copy_inImage_pixels
```

Task3.py

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

# Import numpy
import numpy as np
from numpy import asarray

# Create cols and rows of the matix
cols = 256
rows = 100

# Create a 100x256 matrix that has 0 in each value
grayscale_pixels = np.zeros(shape=(rows, cols))

# Change values of each pixel = to that column number they are on
for row in range(0, rows):
    for col in range(0, cols):
        grayscale_pixels[row, col] = col
# Create image from the matrix
grayscale_im = Image.fromarray(np.uint8(grayscale_pixels))

# Show image
grayscale_im.show()

# Save image
grayscale_im.save("grayscale.tif")

# Create a default value for average pixel
Averagepixel = 0

# Add up all the values in the matrix
for row in range(0, rows):
    for col in range(0, cols):
        Averagepixel += grayscale_pixels[row, col]

# find the average value (add up all the values) / (total # of values)
Averagepixel = Averagepixel / (cols * rows)

# Print the average pixel value
print ("Average Pixel:", Averagepixel)
```