Lab Assignment #2

Due Monday, September 26 by 11:59pm

(upload a single PDF to CatCourses)

Title: Simple Image Manipulations in Python.

The goal of this lab is to make sure you are able to read and write images, convert an image from color to grayscale, perform simple operations like rotating an image 90 degrees both clockwise and counterclockwise, create a function that computes the inverse of an image, and prepare a single PDF to submit as your lab report.

The lab consists of three tasks.

See `Task_example.py` for an example Python file that creates a new image which is the "Beginning.jpg" image cropped by 20 pixels along each edge. This example file demonstrates a number of things that can be useful for the tasks below such as reading an image from a file, creating a grayscale image from a color one, creating a `numpy` matrix with the pixel values from an image, creating a blank `numpy` matrix and changing its values, and creating an image from a `numpy` matrix which contain its pixel values.

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

Create the Python file `Task1.py` to do the following:

- Read the image "Beginnings.jpg".
- Display the image on the screen.
- Convert the image from color to grayscale.
- Display the image on the screen again.
- Create a `numpy` matrix that has the pixel values from the image.
- 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.*
- 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.
- Create an image from the rotated matrix.
- Display the counterclockwise rotated image on the screen.
- Write the counterclockwise rotated image to a file.
- Create a new `numpy` matrix which is the original matrix rotated by 90 degrees *clockwise*. (Follow similar substeps as above.)
- Create an image from the rotated matrix.

- Display the clockwise rotated image on the screen.
- Write the clockwise rotated image to a file.
- Compute and print the maximum pixel value of the clockwise rotated image. Again, you must compute the maximum value yourself using nested `for` loops.

Questions for task 1:

1. What is the maximum pixel value of your grayscale Beginnings image?
2. What is the maximum pixel value of your clockwise rotated grayscale image?
3. Should these be the same? Why or why not?
4. What was the most difficult part of this task?

What to turn in for task 1:

- Your clockwise rotated image.
- Your answers to the four questions above.
- Your code as file `Task1.py`.

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

Create the Python file `Task2.py` to do the following:

- Read the image "Watertower.tif". Note that this image should already be in grayscale. You can check this with `print("image mode is:", im.mode)` which will print "L" if it is grayscale. See the documentation for `pillow` about other image modes. (The reason I'm using the .tif format is so it is not jpg compressed. This will be important for future labs. Note that .tif images can be color.)
- Display the image on the screen.
- Create a `numpy` matrix that has the pixel values from the image.
- Call the function `myImageInverse()` from the package `MyImageFunctions.py`. (You will write this package and function—see below.) The input to this function will be the `numpy` matrix with the pixel values. The output will be a `numpy` matrix with the inverse of the pixel values.
- Create an image from the returned matrix.
- Display the image on the screen.
- Write the image to a .tif file.

Create the file `MyImageFunctions.py` which defines the function `myImageInverse(inImage)` that takes as input a `numpy` matrix and outputs a `numpy` matrix which is the image inverse of the input. The function will thus need to:

- Determine the size of the input matrix.
- Create a new `numpy` matrix of the same size.
- Use nested for loops to copy the values from the input matrix to the output matrix using the equation `out_value = 255-in_value`. This is the image inverse of a grayscale image.
- Returns the output matrix.

Questions for task 2:

1. What is the maximum pixel value of your inverse image?
2. How is this maximum value related to the values of the original image?
3. What was the most difficult part of this task?

What to turn in for task 2:

- Your inverse image.
- Your answers to the three questions above.
- Your code as the files `Task2.py` and `MyImageFunctions.py`.

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

Create the Python file `Task3.py` to do the following:

- 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.
- Display the image on the screen.
- Save the image as a .tif file.
- 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.

Questions for task 3:

1. What is the average pixel value in your gradient image?
2. Why did you expect to get this value from the gradient image?
3. What was the most difficult part of this task?

What to turn in for task 3:

- Your gradient image.
- Your answers to the three questions above.
- Your code as the file `Task3.py`.

Submit your lab report as a single PDF to CatCourses.

Create a single PDF which contains:

- Lab number and title, your name, your lab section, your TA's name, and the date.
- For each of the three tasks:
  - The images you were instructed to submit. The images should have brief captions.
  - Your answers to the questions.
- Your code can appear at the end of the PDF.

See Lab_02_sample_report.pdf as an example. (This sample report also has some code that you can use/copy.)

Comment your code

- Provide comments for key lines of code.

- Provide headers for your functions. Headers are comments which appear right after the function declaration that summarize what the function does its calling syntax, the input and output parameters, and the function history. For example, I wrote the following header for my function myImageInverse:

```
def myImageInverse( inImage_pixels ):
# 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:
#   S. Newsam   9/10/22     Created
#   S. Newsam   9/11/22     Modified to use numpy instead of PIL image
```

What your report will be graded on

- Whether your report includes the images you were asked to include and they have appropriate captions.
- Whether you answered the task questions correctly.
- Whether your report includes the code you were asked to include.
- Whether your code is correct.
- Whether your code is commented and your functions have headers.