Homework Assignement 3 Extra Credit

CSE 5940

Alan L. Perez
School of Computer Science and Engineering
California State University, San Bernardino
San Bernardino, Ca. United States of America

***Abstract*—The goal of this extra credit is to get familiar with other aspects of OpenCV when it comes to Power-law (Gamma) Transformation. Applying the Thresholding, and Histogram Equalization.**

Keywords— Power-law (Gamma) Transformation, Thresholding, and Histogram Equalization.

I. Motivation

In this assignment our goal was to take a picture (marilyn.bmp) that the Professor provided us from our assignment 1 and apply Power-law (Gamma) Transformation, the Thresholding, and Histogram Equalization. All this using the module OpenCV for python. This is because students in CSE 5940 want to get familiar with the image processing in terms of Intensity Transformation from Image Enhancement. Through this students will be able to receive a deeper understanding of the concepts as previously stated.

II. Main Idea

In this assignment the students were allowed to use libraries to solve the main objective. The library used for all three part of the extra credit assignment was the python library "OpenCV" or "cv2" [1]. Then numpy was used which is a "python library for working with arrays" [2]. Numpy was only used when implementing the Power-law (Gamma) transformation. These were the only libraries needed to implement every part of the assignment.

III. Implementaion

The first part to be implemented for this assignment was the Power-law (Gamma) Transformation. Here we start by importing "cv2" [1]. Then we also "import numpy" [2]. In the first line we use "imread" from the "cv2" module to read the path of the image so we can than use that information later in our code [1]. Next we creat a list of gamma values and store them in a list named "gammas" [1]. Then we create a for loop so we can go through each gamma value one by one [1]. Next in the for loop we apply the gammas and apply the gammas by creating an array to store our image using numpy.array [1]. Inside we apply the equation of the Power Law (Gamma) Transformation ($s = c*r^{\gamma}$). Then for "dtype" we set it as "uint8", which means it is an 8-bit unsigned integer [3]. Lastly we save our image using "cv2.imwrite" into a .jpg file for each of our gamma values [1]. This is done within the for loop.

Next we implemented thresholding to our image. We use a similar process as the first one by importing "cv2" [4]. Afterward, we apply the "cv2.threshold" function to our image [4]. This function give two results. We save it in the variable "thresholdValue" because the result given is our threshold value [4]. Next value we get is our is our threshold image which we save in our variable "threshold" [4]. The first parameter is our image that we read earlier. The next parameter is 127 which is the "threshold value we set [4]. After 255 is the maximum value that a pixel is set to if it is greater than 127 [4]. Then we set the threshold type as "THRESHOLD_BINARY" [5].

The last part we need to implement is Histogram Equalization. Like we did in the previous two parts we "import cv2" [6]. Again we use "cv2.imread" in order to read the image, however, this time we set the of the flag to zero to let the function know that the image is in greyscale [7]. We must do this or else we can't use the next the next function "equalizeHist" because it cannot work if the image is in Blue Green Red or BGR. When we apply the "cv2.equalizeHist" we save the image into the variable "equalization" [6]. Lastly like the other parts we use "cv2.imwrite" to save the equalization histogram image into a .jpg file[6].

IV. Expermentail Results

In this section, you need to discuss your exponential results for this assignment. (1-3 paragraphs).

The original image we started out with was this "marilyn.bmp":

For the first implementation we needed to use the Power-law (Gamma) Transformation. Below is an example of this being implemented with a gamma value of 0.1:

Below is an example of a gamma value of 2.5:

Next to be implemented was the thresholding:

Last to be implemented was Histogram Equalization:

V. Issues and potential solutions

The only problem that occurred was when trying to use the function "cv2.equalizeHist". The attempt to read it as BGR in the "cv2.imread" was tried, but an error occurred. It was only later through further research that the "cv2.equalizeHist" function does not work on a BGR image, only on a greyscale image. The flag had to be set to zero [6].

References

[1] Patel, Karan. "Power Transformation in Digital Image Processing | Image Enhancement | Medical Imaging Technique." *YouTube*, YouTube, 24 June 2020, https://www.youtube.com/watch?v=aGHQNbXkeg0.

[2] "Numpy Introduction." *Introduction to NumPy*, https://www.w3schools.com/python/numpy/numpy_intro.asp.

[3] "[MS-DTYP]: UINT8." *[MS-DTYP]: UINT8 | Microsoft Docs*, https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-dtyp/a88ed362-a905-4ed2-85f5-cfc8692c9842.

[4] Patel, Karan. "Binary Image || What Is a Binary Image? How You Convert an Image into Binary Using Python? Lec: 1." *YouTube*, YouTube, 11 June 2020, https://www.youtube.com/watch?v=opii_HRx1bg&t=290s.

[5] "Miscellaneous Image Transformations." *OpenCV*, https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html#gga a9e58d2860d4afa658ef70a9b1115576a147222a96556ebc1d948b372b cd7ac59.

[6] "Histograms - 2: Histogram Equalization." *OpenCV*, https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalizatio n.html.

[7] "Python Opencv: Cv2.Imread() Method." *GeeksforGeeks*, 2 Aug. 2019, https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/.