ECE1512 Digital Image Processing and Application - Winter 2021
Assignment 1

Zhe Su (1002113116)

Handed: January 31, 2021
Due: February 1, 2021

Technical Discussion

Part A: Image Enhancement Using Intensity Transformations

The first part of this assignment is to perform the intensity transformation on the targeted image for manipulating its contrast ratio. The operations we perform are in spatial domain, which is to manipulate the image pixels directly. We are going to explore two different techniques: log transformation and power-law transformation.

1. Log Transformation

Log transformation is to map a narrow range of input grey level value to a wider range of output value for increasing intensity or brightness of the image. This is done by the mathematical nature of the log function. The general formula is described as the following:

$$s = c * log(1 + r) \tag{1}$$

where r refers to the original intensity level of a pixel, s represents the final output of intensity level of a pixel, and c is a scaling factor. In this assignment, we set value of c to (L-1) / log (1+maximum pixel value) to avoid the final output value from exceeding the valid range, where L is the maximum value of the bit channel, in this case 8 bits channel having L equals to 256. Notice that the value of r could be zero, which might generate a divide-by-zero warning while implementing in python. Therefore, we might want to add a relatively small dummy value inside log term to avoid this warning. Log transformation is not suitable to apply on any images. For instance, an image with many high range of pixel values might get lost of its original information in the image after applying log transformation. Thus, we generally apply this technique on images where many lower pixel values are presented.

2. Power-law Transformation

Similar to log transformation, power-law transformation is also to map a dark input value to a wider range of output value or the other way around. By observing the mathematical natural of power law, the output values is indeed controlled by the power factor. The formula is described as the following:

$$s = c * r^{\gamma} \tag{2}$$

where varying γ gives different shape of curves. When γ is smaller than zero, the output value is brighter, otherwise the output value becomes darker. The rest of the terms are identical to the log transformation, and we also set c to 1. We would examine different value of γ to see how it varies the output pixel value, the results are shown in the later section.

Part B: Histogram Equalization

The histogram of image shows the grey scale distribution in the image. A high contrast image generally has a equally spread distributed histogram. Therefore, the technique of equalizing the image histogram could be applied to improve the contrast ratio and intensity level of dark or washed out images. The histogram

equalization function is described as following:

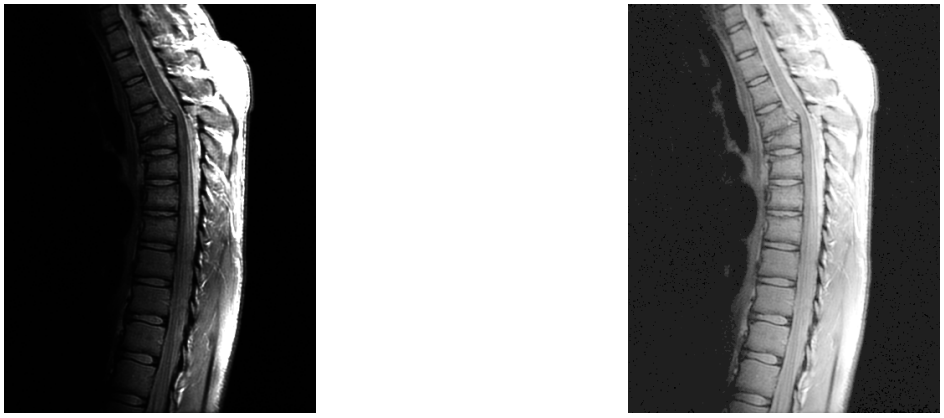
$$s_k = (L-1)\sum_{j=0}^{K} p_r(r_j) \tag{3}$$

where L represents the image bit channel, which is a 8 bits channel in our case. Pr is the probability density function (PDF) of input value. In discrete case, the summation of the PDF becomes the cumulative density function (CDF) of input value, which is defined as the transformation function. This is also the accumulated normalized histogram value of the original input. Applying the transformation function to the input, we are able to get the output value with being histogram equalized. We implemented histogram equalization functionality in Python by the help of utilizing OpenCV. OpenCV provides handful tools in Python to read image and to parse image histogram values. The result would be discussed in the later section.

Results Discussion

1. Log Transformation

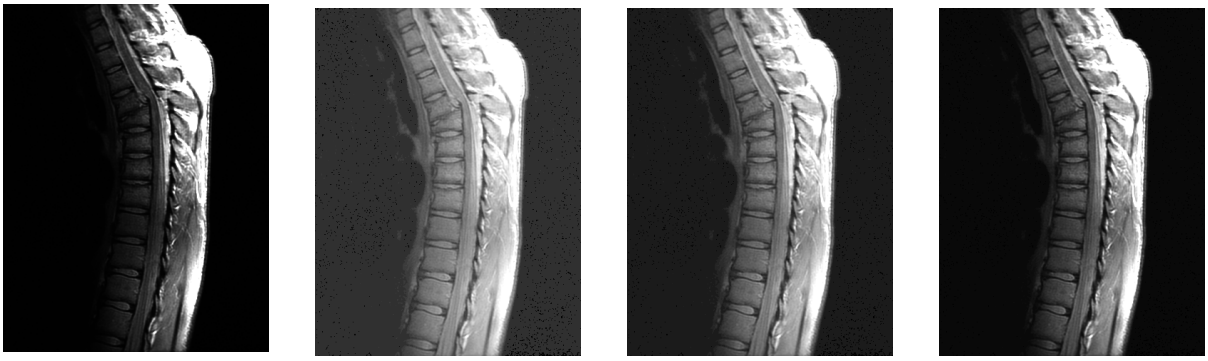
Figure 1 shows the comparison between the original image and the image after being log transformed. We could clearly see that the intensity value increased and the image is more brighter than before, and the output image meets our assumption and expectation.



(a) Original (b) Log transformed

Figure 1: Applied log transformed

2. Power-law Transformation

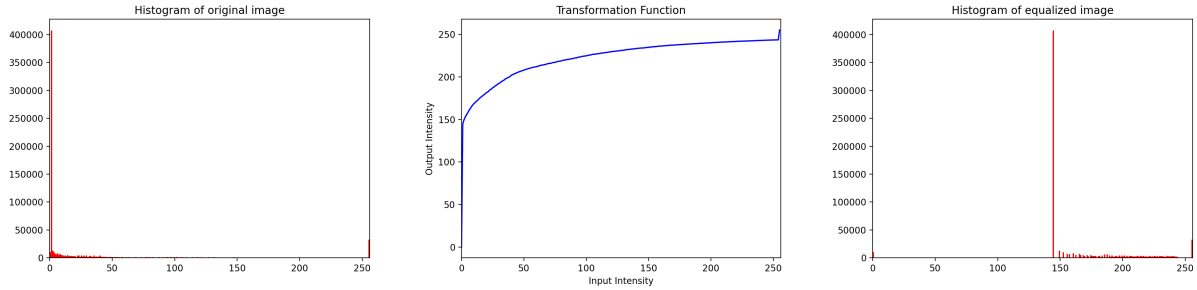


(a) Original (b) $\gamma = 0.3$ (c) $\gamma = 0.4$ (d) $\gamma = 0.6$

Figure 2: Applied power-law transformed with different γ

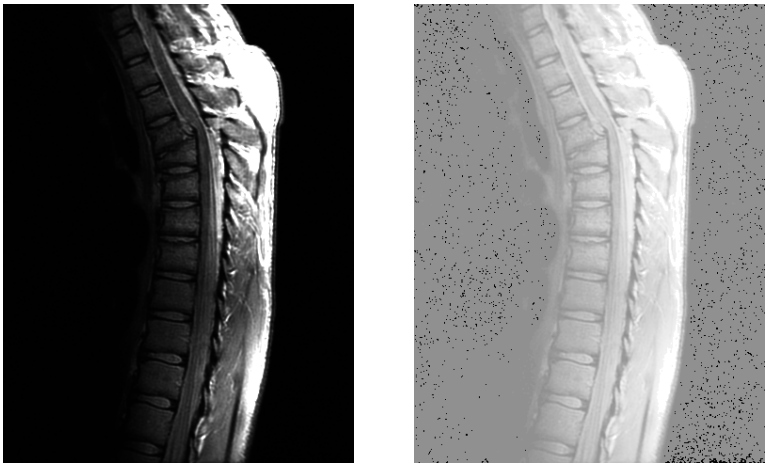
Figure 2 shows the results of power-law transformed with γ values of 0.3, 0.4, and 0.6. We found that the smaller the γ value it applied to the transformation, the higher intensity value we could get from the output. This makes sense because the higher the γ value is applied, the flatter the power-law shape is given by the power-law formula, resulting less low pixel values being boosted up. The output images meet our original assumption and expectation.

Histogram Equalization



(a) Original histogram (b) Transformation function (c) Equalized histogram

Figure 3: Image histogram and transformation function



(a) Original (b) Histogram equalized

Figure 4: Applied histogram equalization

Figure 3a shows the original image histogram, where the value are mostly presented in the lower value section and it also has many values near zero. This make sense since the original image is dark. After applying the transformation function (accumulated normalized value of input intensity) to the original histogram, we acquire the equalized histogram as shown in figure 3c and the output image is illustrated in figure 4b. We found that the equalized histogram values are mostly grouped in the higher value section, resulting that the output image is brighter than usual. There are various ways to resolve this issue, including applying a different transformation function to the image, or specifying a output histogram and transforming the image by the technique of histogram specification.

Appendix

Implementation in Python and OpenCV

```
def logTransformed(imgPath, imgName):
   img = cv2.imread(imgPath+"/"+imgName)
   dummyValue = 0.00000001
   c = 255/(np.log(1 + dummyValue + np.max(img)))
   imgTransformed = c * np.log(1 + dummyValue + img)
   imgTransformed = np.array(imgTransformed, dtype = np.uint8)
   imgPathOut = imgPath + "/logTransformed.tif"
   cv2.imwrite(imgPathOut, imgTransformed)
```

Listing 1: Log Transformation Implementation

```
def powerTransformed(imgPath, imgName):
   img = cv2.imread(imgPath+"/"+imgName)
   c = 1
   # values from the slide
   for gamma in [0.3, 0.4, 0.6]:
      gammaCorrected = np.array(255*(img/255) ** gamma, dtype = np.uint8)
      imgPathOut = imgPath + "/gamma" + str(gamma)[-1] + ".tif"
      cv2.imwrite(imgPathOut, gammaCorrected)
```

Listing 2: Power-law Transformation Implementation

```
def historgramEqualized(imgPath, imgName):
   img = cv2.imread(imgPath+"/"+imgName, 0)
   hist,bins = np.histogram(img.flatten(),256,[0,256])

   # plot of original histogram
   plt.figure(0)
   plt.hist(img.flatten(),256,[0,256], color = 'r')
   plt.xlim([0,256])
   plt.title("Histogram of original image")

   # CDF of input intensity value
   cdf = hist.cumsum()
   cdfTrans = np.ma.masked_equal(cdf,0)
   cdfTrans = (cdfTrans - cdfTrans.min())*255/(cdfTrans.max()-cdfTrans.min())

   # plot of transformation function
   plt.figure(1)
   plt.plot(cdfTrans, color = 'b')
   plt.xlim([0,256])
   plt.title("Transformation Function")
   plt.xlabel("Input Intensity")
   plt.ylabel("Output Intensity")

   # apply transformation function
```

```
cdf = np.ma.filled(cdfTrans,0).astype(np.uint8)
imgEnhanced = cdf[img]
hist,bins = np.histogram(imgEnhanced.flatten(),256,[0,256])

# plot of enhanced image histogram
plt.figure(2)
plt.hist(imgEnhanced.flatten(),256,[0,256], color = 'r')
plt.title("Histogram of equalized image")
plt.xlim([0,256])


imgPathOut = imgPath + "/historgramEqualized.tif"
cv2.imwrite(imgPathOut, img2)
```

Listing 3: Histogram Equalization Implementation

Resources

Acknowledge the following resources in assist to complete the assignment.

1. ECE1512_Chapter_03a_Intensity_Transformations_GW.pdf
https://q.utoronto.ca/courses/199947/files/11845572?module_item_id=2218274
2. ECE1512_Chapter_03b_Intensity_Transformations_GW.pdf
https://q.utoronto.ca/courses/199947/files/11845572?module_item_id=2218274
3. opencv-python-tutroals.readthedocs.io
https://opencv-python-tutroals.readthedocs.io/en/latest/index.html
4. Log transformation of an image using Python and OpenCV
https://www.geeksforgeeks.org/log-transformation-of-an-image-using-python-and-opencv/?ref=lbp