# **EN2550 – Assignment 1**

## **Intensity Transformations and Neighborhood Filtering**

### Index No: 190696U

### Name: Wijegunawardana C.H.W.

#### Question 1 – Implementing the intensity transformation on an image

The transformation is generated as follows.  
Text, letter

Description automatically generated Chart, line chart

Description automatically generated  
The *LUT* function of the OpenCV library is used to apply the transformation to the given image.A screenshot of a computer

Description automatically generated with low confidence  
The original image and the transformed image are depicted in the results obtained.A person with a cigarette in her mouth

Description automatically generated with low confidence  
The pixels between 50-150 grayscale levels in the original image are enhanced using the transformation into higher gray-scale values. Therefore, those pixels have become whiter in the transformed image compared to the original image.  
  
Question 2 – Accentuating white and gray matter in the image  
To accentuate the white and gray matter in the grayscale image of the brain, the corresponding grayscale levels are transformed into 255 (absolute white).

|  |  |
| --- | --- |
| The transformation for the white matter  Text  Description automatically generated | Chart, line chart  Description automatically generated |
| The transformation for the gray matter  A screenshot of a computer  Description automatically generated with medium confidence |  |

The transformed images are depicted below.A picture containing text

Description automatically generated  
To accentuate the white matter, the transformation is designed such that the output will have absolute white in near-white pixels for better visualization. The near-white grayscale range is selected as 200-255.  
To accentuate the gray matter, the transformation is designed such that the output will have absolute white in gray pixels for better visualization. The gray grayscale range is selected as 50-200.

#### Question 3 – Gamma Correction to the L Plane

The gamma correction only needs to be applied to the L plane in the L\*a\*b\* color space. This was done using the following segment of code.Text

Description automatically generated   
First, the image is converted into the L\*a\*b\* color space using OpenCV *cvtColor* function. Then the gamma transform is constructed for the selected gamma value. Then the L field is extracted from the image using array slicing, the transformation is applied and appended with the remaining a\* and b\* fields.

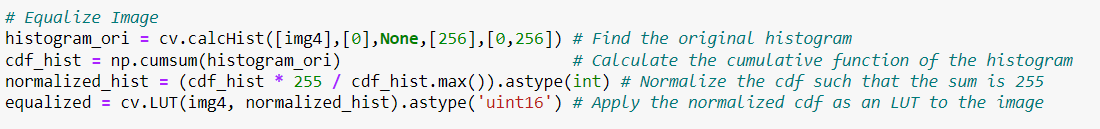
The histograms of the original image and the gamma-corrected image are generated using the OpenCV *calcHist* function.  
  
The results are depicted in the following figures.Graphical user interface

Description automatically generated with low confidence  
The gamma value **0.7** is selected to obtain a spread-out histogram for the transformed image.

#### Question 4 – Histogram Equalization

The histogram equalization is achieved by following the method below.

1. Calculating the cumulative distribution of the histogram of the original histogram.
2. Normalizing the CDF to values between 0-255.
3. Applying the CDF as a transformation to the original image.

This effectively spreads out the concentrated areas of the histogram within the range 0-255.  
The original image and the histogram equalized image are depicted in the following figure.A close-up of several rocks

Description automatically generated with low confidence

The corresponding histograms are as follows.Chart, histogram

Description automatically generated  
It can be observed that, after the histogram equalization, the histogram is distributed evenly across the 0-255 grayscale values. The transformed image has more details.

#### Question 5 – Zooming

The small images given in the assignment images folder were used for zooming. They were compared with the zoomed images provided.  
As instructed, the OpenCV function *resize* was used to do the zooming on the images using two different algorithms.

1. Nearest Neighbor
2. Bilinear Interpolation

To understand the behavior of the nearest neighbor algorithm, it was implemented using a written function as below.  
Text

Description automatically generated  
  
Bilinear interpolation was carried out using the *resize* function with *INTER\_LINEAR* as the interpolation argument.Text

Description automatically generated  
  
The small images are zoomed by a factor of 4 to be compared with the larger images.

|  |  |
| --- | --- |
| **Nearest Neighbor** | **Bilinear Interpolation** |
| A group of people holding swords  Description automatically generated with low confidence | A group of people holding swords  Description automatically generated with low confidence |
| SSD = 0.0004811121335890121 | SSD = 0.0004775562417232688 |

If observed closely, the image zoomed using the nearest neighbor method has a pixelated appearance because there are duplicate pixels. But the image zoomed using bilinear interpolation looks much smoother. The sum of squared differences is lower in the bilinear interpolation.

The SSD was calculated as follows.  
All the 3 images given were zoomed using the above two algorithms. The results are as follows.  
  
Nearest Neighbor  
A collage of people

Description automatically generated with low confidence  
Bilinear Interpolation

A collage of people

Description automatically generated with low confidence

#### Question 6 – Sobel Filtering

Using the *filter2D* function

Text

Description automatically generatedA picture containing text

Description automatically generatedUsing own function  
Zero padding was added to the image before applying the filter.Text

Description automatically generated  
Filter was applied convolutionally within a loop.Graphical user interface, text, application, Word

Description automatically generatedA picture containing text

Description automatically generatedSimilar results were obtained when the *filter2D* function and the own function were used.

Using Associative Property  
The Sobel vertical and Sobel horizontal kernels could be represented as following.

To generate the gradient image using this property, first the original images were convolved with the 3x1 array of the above compositions. Then, the intermediate image was again convolved with 1x3 array.

Text, letter

Description automatically generated

The same results could be obtained from this method as well.

A collage of a person

Description automatically generated with low confidence

#### Question 7 – Segmentation

The *grabCut* function was used to segment the flower out of the given image.  
First, the rectangle that completely contains the flower is defined. Then the *grabCut* function is applied and the mask is obtained where it estimates the foreground and the background.  
Then a mask is obtained for the foreground and the background based on the defined mask and the mask estimated by the *grabCut* function.

Foreground and background images are extracted by applying bitwise *AND* operation between the masks and the original images.

Text

Description automatically generated with low confidence

A picture containing text, plant, flower, daisy

Description automatically generated  
An enhanced image is obtained by applying *GaussianBlur* to the background image and adding it with the foreground image.  
  
A group of yellow flowers

Description automatically generated with medium confidence

When the Gaussian blur is applied to the background image, the edges of the black portion that was occupied by the flower gets blurred too. As a convolutional filter is applied, those pixels values might get increases above 0 in those edges. Then, when the foreground image is added to this blurred background, the pixel values around the edge will be increased than the yellow color in the foreground image. Therefore, they appear darker.

GitHub Link: <https://github.com/chathuni1999/EN2550.git>