Image Processing Assignment: Basic GUI Editor

Vinit Awale
Roll No. 18D070067
Department of Electrical Engineering
Indian Institute of Technology, Bombay
Mumbai, India
email: 18D070067@iitb.ac.in

***Abstract*—This document specifies the implementation details of a Basic GUI Editor built as a part of an Assignment for the course EE610: Image Processing. Various image processing operations such as Histogram Equalisation, Gamma Correction, Log transformation, Blurring, Sharpening and Contrast Stretching have been implemented in this Basic GUI Editor. This paper includes the mathematical and various implementation details of the image manipulation operations implemented. Through this assignment we understand the use and applications of the operations mentioned earlier.**

***Index Terms*—Histogram Equalisation, Gamma Correction, Log transformation, Blurring, Sharpening, Contrast Stretching, OpenCV, Image Processing**

I. Introduction

Humans are visual beings. Major potion of the perception of environment by human beings is done through their visual system. Image Processing operations attempt to enhance the images captured through various methods (eg. Camera, MRI, CT SCan) to find out the patterns in the images that could not be observed earlier.
In this assignment we have implemented various image processing operations such as:

- Histogram Equalisation
- Gamma Correction
- Log transformation
- Blurring
- Sharpening
- Contrast Stretching

For implementing these image manipulation operations, we have the Python Programming Language. However, all of the operations are implemented from scratch with using any inbuilt functions in several libraries. The only additional libraries used for implementing the above mentioned operations are:

- Open CV (CV2)
- NumPy

Also, we have implemented a basic Graphical User Interface (GUI) to enable easier transformation of the images. The implementation of the Graphical User interface has been done using the following tools:

- PyQt5
- Designer (of PyQt5)

Following is an example of an image enhancement operation implemented in the assignment

Fig. 1: Original Image

Fig. 2: Image after Histogram Equalisation

II. GUI Design

A. Approach

The approach of building the GUI was as follows:

- Using PyQt5 Designer, designing the User Interface i.e. generating the GUI.ui file
- Using the library PyQt5, adding the image manipulation functions to the UI designed i.e. making the GUI.py file

The GUI.ui file included in the code folder was generated using PyQt5 Designer. Following is the GUI interface design in Designer.

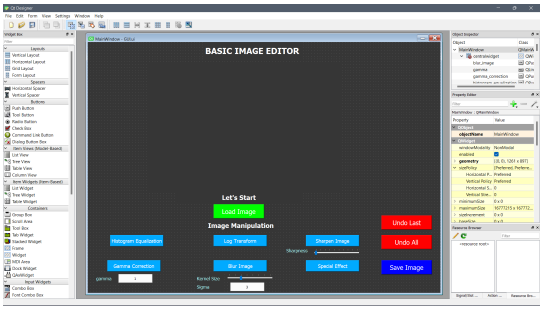


Fig. 3: GUI design using PyQt5 Designer

Then using the following command:

```
pyuic5 -x GUI.ui -o GUI.py
```

the **GUI.py** file was generated corresponding to the UI designed. Further the various buttons in the UI were connected to their corresponding functions by writing following lines of code in **GUI.py**

```
# Connection specific function to the buttons clicked
self.load_image.clicked.connect(self.Load_Image)                    # Connect button to Load Image function
self.histogram_equalization.clicked.connect(self.HISTOGRAM_EQUALIZATION)    # Connect button to Histogram Equalization function
self.gamma_correction.clicked.connect(self.GAMMA_CORRECTION)        # Connect button to Gamma Correction function
self.log_transform.clicked.connect(self.LOG_TRANSFORMATION)         # Connect button to Log Transformation function
self.blur_image.clicked.connect(self.BLUR_IMAGE)                    # Connect button to Blur Image function
self.sharpen_image.clicked.connect(self.SHARPEN_IMAGE)              # Connect button to Sharpen Image function
self.undo_last.clicked.connect(self.UNDO_LAST)                      # Connect button to Undo Last function
self.undo_all.clicked.connect(self.UNDO_ALL)                        # Connect button to Undo All function
self.save_image.clicked.connect(self.SAVE_IMAGE)                    # Connect button to Save Image function
self.special_effect.clicked.connect(self.SPECIAL_EFFECT)            # Connect button to a Special Effect function
```

The linked functions are further defined in the **GUI.py** file included along with the code folder. For defining these functions we have used the implementations of the image processing manipulations mentioned above were used.
The implementations of the following:

- Histogram Equalisation
- Gamma Correction
- Log transformation
- Blurring
- Sharpening
- Contrast Stretching

are included in the **Image_Manipulations.py** file of the code folder. This file has been imported in the **GUI.py** file to access the implementations of the Image Manipulation operations

B. Components of the GUI

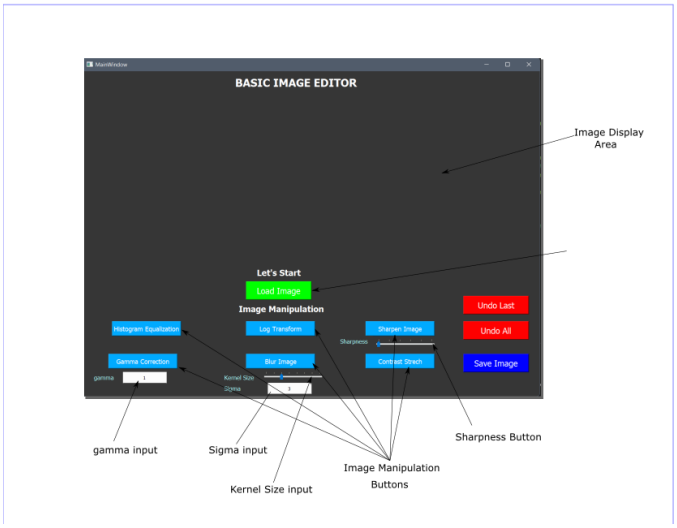
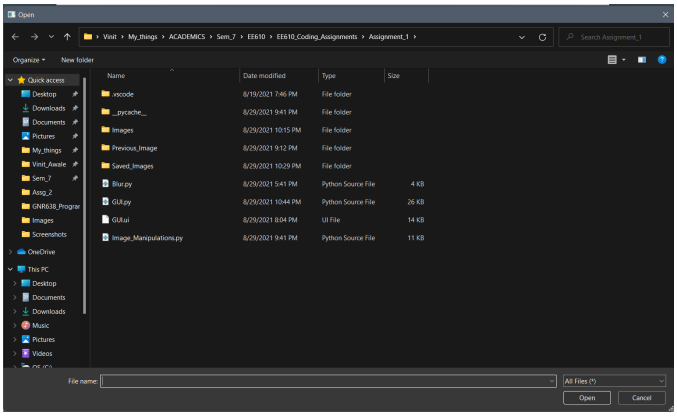


Fig. 4: Components of GUI

We can see the various components of the GUI in the above image.

1) Image Display Area: The result of various manipulations on the image are shown in this area.

2) Load Image Button: On clicking this button a pop-up window appears which can be used to load the image on which manipulations are performed.



3) Image Manipulation Buttons: These buttons perform the image manipulation operation mentioned on them

4) Undo Last Button: This button undoes the last image manipulation operation performed

5) Undo All Button: This button undoes all image manipulation operation performed

6) Save Image: This button saves the image shown int the image display area in the **Saved_Images** folder included in the code folder

III. Image Processing Operations

A. Histogram Equalization

This operation is performed to make the Probabilty Distribution Function (PDF) of the image to be as close as possible to the uniform distribution

Mathematical Equation:

$$s_k = T(r_k) = (L-1)\sum_{j=o}^{k} p_R(r_j)$$

Where s_k are the pixel intensity values of the transformed image and r_k are the pixel intensity values of the original image R, whose PDF is $p_R(r_j)$
The implementation of this operation is done in the function **Histogram_Equalization()** included in **Image_Transformations.py** file

Purpose: This operation is used when we the PDF of the image is skewed and we want to make the PDF close to uniform distribution.

Earliest Citation: This method was first cited in paper **"Multiobjectives bihistogram equalization for image contrast enhancement"** by Hum, Yan Chai; Lai, Khin Wee; Mohamad Salim, Maheza Irna in the year **2014**

B. Gamma Correction

Gamma correction or gamma is a nonlinear operation used to encode and decode luminance or tristimulus values in video or still image systems

Mathematical Equation:

$$s_k = cr_k^{\gamma}$$

Where s_k are the pixel intensity values of the transformed image and r_k are the pixel intensity values of the original image R The implementation of this operation is done in the function **Gamma_Correction()** included in **Image_Transformations.py** file

Purpose: Gamma correction is used when we want to map the intensity range of the image with a non linear equation. For $\gamma > 1$, the low intensity range in the input image is mapped to a larger range in the transformed image. For $\gamma < 1$, the high intensity range in the input image is mapped to a larger range in the transformed image.

Earliest Citation: This method was first cited in paper **"Digital Video and HDTV: Algorithms and Interfaces"** by Charles A. Poynto in the year **2003**

C. Log Transformation

Log transformation is a non linear transformation which maps the image intesity of the original image using the logarithm function

Mathematical Equation:

$$s_k = clog(1 + r_k)$$

Where s_k are the pixel intensity values of the transformed image and r_k are the pixel intensity values of the original image R The implementation of this operation is done in the function **Log_Transformation()** included in **Image_Transformations.py** file

Purpose: By using the log transform, components that were multiplicatively combined become combined additively. This makes it easier to separate them by linear filtering.

Earliest Citation: This method is quite an old one, however the earliest citiation could not be found online.

D. Image Blurring

This has been implemented using Gaussian Blurring for this assignment.

Mathematical Equation:

$$I_s = I_r * K$$

Where I_s is the output image and I_r is the input image and K is the Gaussian kernel (discussed in lectures) and * represents the convolution operation. The implementation of this operation is done in the function **GaussianBlur()** included in **Image_Transformations.py** file

Purpose: Image Blurring is Primarily done to reduce the noise in the image

Earliest Citation: This method was first cited in **"Computer Vision", page 137** by Shapiro, L. G. Stockman in the year **2001**

E. Image Sharpening

This has been implemented using Unsharp Masking for this assignment.

Mathematical Equation:

$$I_s = I_r + (I_r I_r * K) \times c$$

Where I_s is the output image and I_r is the input image and K is the Gaussian kernel (discussed in lectures) and * represents the convolution operation. Here, c is the sharpness that we want in our image. The implementation of this operation is done in the function **Sharpen()** included in **Image_Transformations.py** file

Purpose: Image Sharpening is primarily done to make the image stand out and look sharper. This less clear patterns in an image become clear when sharpening is done.

Earliest Citation: This method was first cited in **"Guide to Image Sharpening", Cambridge in Color**. However, the exact year could not be found out

IV. Experiments and Results

A. Histogram Equalization

Fig. 5: Original Image

We see that the image is underexposed. It means that the lower intensities (darker pixels) have a higher probabilty. Hence, histogram equalization will improve the appearance of the image.
One more example has been included in the starting of the assignment for histogram equalization.

Fig. 6: Image after Histogram Equalisation

The original image was chosen since it did not have any details in it. However, after histogram equalization the different patterns in the image become clear

B. Gamma Correction

Fig. 7: Original Image

We can easily see that this image is washed out. Hence, we can enhance it by mapping the lower intensities to a higher range in the output image. Hence we can use gamma correction with $\gamma > 1$

Fig. 8: $\gamma = 2$

Fig. 9: $\gamma = 3$

Fig. 10: $\gamma = 4$

Hence, we can easily see that the washed out image is improved after gamma correction with $\gamma > 1$

C. Log transformation

Fig. 11: Original Image

We can easily see that this image is underexposed. Hence, we can enhance it by mapping the higher intensities to a higher range in the output image. Hence we can use gamma correction with $\gamma > 1$

Fig. 12: Log transformation

We can easily see how the details in the underexposed image become clear after log transformation.

D. Image Blur

Fig. 13: Noisy Lenna Image

The Lenna image shown above has a lot of noise. Hence, blurring can improve the image.

Fig. 14: Gaussian Blur on Noisy Lenna Image

We can easily see how the blurring has improved the image

E. Image Sharpening

Fig. 15: Blurred Barbara Image

The Barbara image shown above is slightly blurred. Hence, sharpening can improve the image

Fig. 16: Sharpened Image

We can easily see how the blurring has improved the image.

F. Contrast Streching

We have the following low contrast image

Fig. 17: Low Contrast Image

After Contrast Streching, we get

Fig. 18: After Contrast Streching

G. Multiple Image transformations

As can be seen in Fig. 12, after **Log transformation**, the image has become a bit noisy. Hence, we can apply Gaussian Blur to it with a large value of σ. However, the generated image gets too blurred. Hence we sharpened it, however, this time with a smaller sigma.

Fig. 19: Original Image

Fig. 20: Enhanced Image

V. Conclusions

All the image manipulation methods have been successfully implemented. Also from the last example in the experiments we can easily that applying multiple transformation on an image can enhance it in a better way.

VI. GitHub Repository Link of the Implementation

All the above mentioned image transformation have been implemented along with the GUI in the following GitHub repository https://github.com/vinitawale/EE610_Coding_Assignments/tree/master/Assignment_1
Note: The link will be made public after the completion of the course conducted in Autumn 2021

VII. References

- https://en.wikipedia.org/wiki/Unsharp_masking
- https://en.wikipedia.org/wiki/Gaussian_blur
- https://en.wikipedia.org/wiki/Gamma_correction