Image Editor - Image Processor

Submitted in partial fulfilment of the requirements of the course of EE 610 Image Processing, Electrical Engineering

Vipin Singh (19D070069)
Electrical Engineer
IIT Bombay

***Abstract*—The coal of this assignment was to build a Graphical user Interface (GUI) for an image editor capable of performing multiple image transformation on an image. Image Editor is a GUI tool built with help of Tkinter and can perform basic operation like Histogram Equalization, Gamma Correction, Log transform, blur image, sharpen the image and invert colors. All the functions for the operation have been written from scratch efficiently, using vectorization operations. The tools allows the user to load a custom image and save the displayed image as and when he wants to.**

***Index Terms*—Image Processing, Python, Tkinter, Intensity Transform, Convolution**

I. Introduction

The GUI tool Image Editor is developed for intensity transform on color as well as gray scale images. The tool provides easy access to the features using GUI made in Tkinter.

A. Objective

The objective of the assignment was to design a GUI tool capable of performing image transformation in Python.

B. Design

The TKinter library in python was used to make the GUI tools and for each intensity transform corresponding functions were implemented. The help of numpy was taken for vectorization which helped in efficient execution of the code. The RGB image were converted to HSV image and the transformation were applied on the V channel.

C. Features

Following features were implemented in the tools

- Load an Image
- Save current image
- Undo the last transformation
- Revert back to original image
- Histogram Equalization
- Gamma Correction
- Log Intensity Transform
- Blur the image (Mean filter was used)
- Sharpen the Image
- Invert the color of the image

II. GUI Design

The GUI for the tool was made using TKinter. Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit.

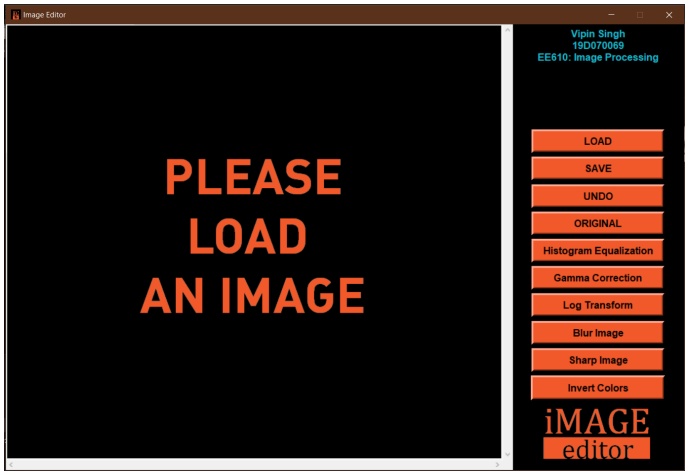


Fig. 1: Image Editor on start

The design of the tool was made using a 2 color scheme (Black and Orange). Standard tkinter button, frames and canvas were used to implement the GUI of the tool. The size of the tool is fixed and cannot be changed. The canvas is build inside a frame of size (800 X 700). The canvas has scroll bar to scroll over images of bigger size. The button are stacked on the right and give access to image manipulation feature of the toolbox. At the bottom right corner is a custom designed logo on Adobe Illustrator. Some feature of the GUI not related to Image transformation have been explained below.

A. Load

The function allow the user to load an image on the canvas. With the help of filedialog, the user can load an image onto the tool. The loaded image is converted to its matrix using numpy array. The array is saved in an global variable called currimagematrix and also pushed to the back of the list (imglist). The list imglist is a stack, it allows the user to undo changes(Explained in detail later). If the image loaded is a grayscale image, it is first converted to RGB and the correspoding image and matrix are stored. The global variable currimage stores the images, that is to be displayed on canvas(i.e. the image form the array).

B. Save

The function allow the user to save the current file. The devault extension for saving the file is .jpg format. The filedialog is used to get the file path and file name for the image for saving purpose. The final image to be saved is obtained by converting the global variable currimagematrix to its image form using PIL.

C. Undo

The undo allows the user to revert back the last transformation applied by the user. The undo in toolbox is performed using the stack data structures, where the image is stored in the matrix form after each transformation. On applying the UNDO, the last entry to the stack is deleted or pop out and the last entry currently on the stack is our required image. The stack is implemented using the list imglist and the global variable currimagematrix is updated to the last entry in the updated stack(imglist).
Figure

D. Original

This button allows the user to go back to the original image which he loaded using the load feature of the toolbox. On using this feature the first entry in the stack is preserved and all other entry are discarded and deleted. The currimagematrix variable is updated to the first entry of the imglist stack. And, now the stack has only one entry,

E. Directory Structure

The directory structure of the toolbox is as followed:

- imageeditor.py : Contains the code for the toolbox with all image transformation and basic GUI implemetation code.
- SampleImages/: This folder contains some sample images used for verification of toolbox.
- README.md: github readme file
- Requirements.txt
- Report.pdf: Report of the assignment
- Probelms.pdf: Problem Statement of the assignment

III. Image Processing Operations

A. Histogram Equalization

Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. This method usually increases the global contrast of many images, especially when the image is represented by a narrow range of intensity values. Through this adjustment, the intensities can be better distributed on the histogram utilizing the full range of intensities evenly.

The RGB image format is first converted to HSV image format. Then we apply the histogram equalization on the V channgel of the HSV image format and finally convert it back to RGB format.

First the histogram is made for all the intensity level and is then converted to density by dividing with total number of pixels. After this we calculate the cumulative density and then multiple the cumulative density with the highest intensity level possible for all the intensity level. This provide us a unique transformation from one intensity level to another.
Let the histogram distribution be represented by H(k), than for an intensity i the the corresponding intensity after transformation ($i^{'}$) is given by the formula given below.

$$i^{'} = \frac{255 \times \sum_{k=0}^{i} H(k)}{MN} \tag{1}$$

where the image size is M X N. If $i^{'}$ is not an integer, we

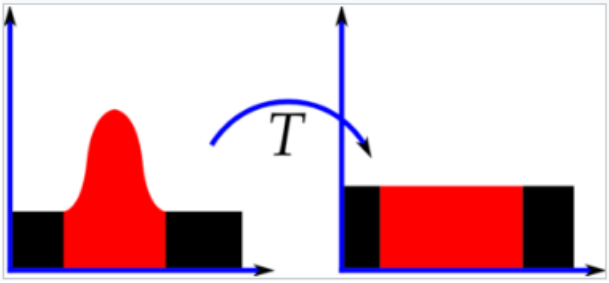


Fig. 2: Histogram Equalization, Pic Credits: TheAIlearner

round it off to the nearest integer.
Uses:

- Improves contrast in images
- Can be used to display noises in images at some areas
- Better contrast for images with histogram distribution closely packed

Citation:

- https://www.nxp.com/docs/en/application-note/AN4318.pdf
- https://www.sciencedirect.com/science/article/abs/pii/S0734189X8780186X

B. Gamma Correction

Gamma correction matters if you have any interest in displaying an image accurately on a computer screen. Gamma correction controls the overall brightness of an image.
The toolbox ask the user to enter the value of gamma(γ. And on receiving the value from the user it transform the image using the relation given below. Here i is the input intensity and $i^{'}$ is the transformed intensity.

$$i^{'} = \frac{i^{\gamma}}{255(\gamma - 1)} \tag{2}$$

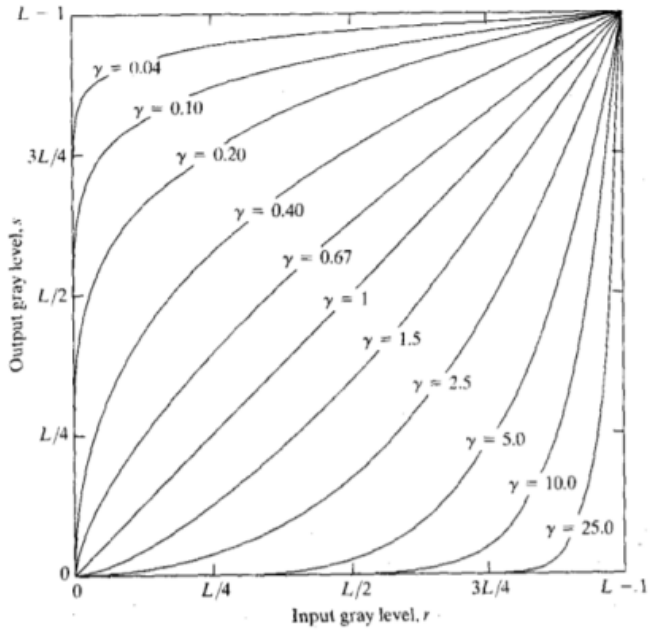


Fig. 3: Gamma Correction, PIC Credits : TheAIlearner

If $i^{'}$ is not an integer, we round it off to the nearest integer.

Uses:

- Controls the overall brightness of an image
- Was used in mac earlier

Citation:

- https://poynton.ca/notes/TIDV/index.html
- http://www.libpng.org/pub/png/spec/1.2/PNG-GammaAppendix.html

C. Log Transform

Log transformation of an image means replacing all pixel values, present in the image, with its logarithmic values. Log transformation is used for image enhancement as it expands dark pixels of the image as compared to higher pixel values. The tool transform the image using the relation given below. Here i is the input intensity and $i^{'}$ is the transformed intensity.

$$i^{'} = \frac{255 \times log(1+i)}{log(256)} \tag{3}$$

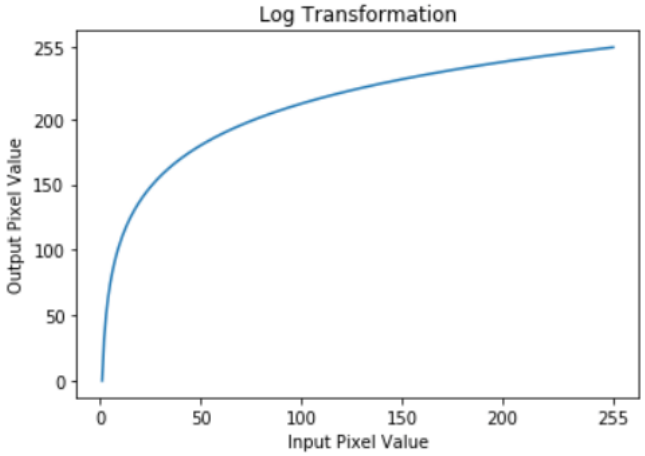


Fig. 4: Log Transformation, PIC Credits : TheAIlearner

If $i^{'}$ is not an integer, we round it off to the nearest integer.

Uses:

- To transform skewed data to approximately conform to normality
- Expand dark intensity pixels in image, and contract light density pixels

Citation:

- https://www.researchgate.net/publication/282265733_An_Integrated_Approach_of_Logarithmic_Transformation_and_Histogram_Equalization_for_Image_Enhancement

D. Blur Image

On blurring an image we losses its finer details but helps in removing noise.
We use a box filter of variable size (depending on the extent of blur) as our kernel and than on padding with 0 we use convolution.
On using this feature the tool ask the user for the extent of burring (say n). The kernel used for box filer is given below

$$B = [b_{ij}]_{2n-1 \times 2n-1} \;\; where \;\; b_{ij} = \frac{1}{(2n-1)^2} \tag{4}$$

The convolution function takes the HSV matrix as the input and the kernel and convolve the kernel with V channel of the image.

$$g(x,y) = w \star f(x,y) = \sum_{s=-n}^{n} \sum_{t=-n}^{n} w(s,t) f(x+s, y+t) \tag{5}$$

Here g(x,y) is the blurred image matrix, f(x,y) is the original image matrix and w(x,y) is the kernel matrix and in this case the box filter.

$\frac{1}{25}$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Fig. 5: Box Filter Kernel of size 5 (n =3)

Uses:

- To remove noise
- To sharpen Images

Citation:

- https://www.researchgate.net/publication/221573689_Blurred_Image_Region_Detection_and_Classification

E. Sharpen Image

This feature helps to sharpen the edges of image and makes the look more sharper.
We use the Gaussian filter to smoothen the image and than subtract the smoothened image to get the edges of the image. We add the image with edges to original image to get a sharpened image.
The tool request the user to input the extent of sharpening, and depending on the input it scales the intensity of the image with edges and than adds it to the original image.
We implement the Gaussian filter by using a 5 x 5 Gaussian blur kernel and than use the convolution function to get the blurred image. Let us suppose the result of Gaussian blur is g(x,y) than the image containing the edges will be

$$m(x, y) = f(x, y) - g(x, y) \tag{6}$$

where f(x,y) is the original image.
Now the sharpened image will be as followed where k is the input by the user for the extent of sharpening.

$$s(x, y) = f(x, y) + k \times m(x, y) \tag{7}$$

where s(x,y) is the sharpened image.

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Fig. 6: Gaussian Filter Kernel of Size 5x5

Uses:

- To highlight edges

Citation:

- https://www.researchgate.net/publication/305985620_A_Review_on_the_Image_Sharpening_Algorithms_Using_Unsharp_Masking

F. Invert Colors

This is a fun features which is provided by the tool to give cool feature to the image.
The intensity is transformed as followed, where i is the intensity transformed to new intensity $i^{`}$.

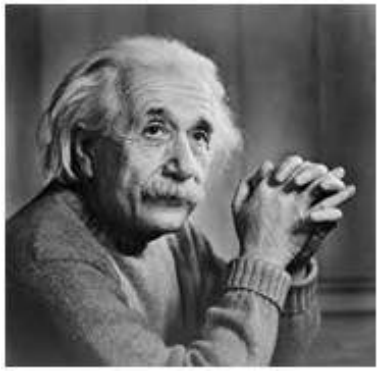
$$i^{`} = 255 - i \tag{8}$$

The intensity transforms makes the light part darker and dark part lighter. Here also the intensity transform is applied to the V channel in HSV image format

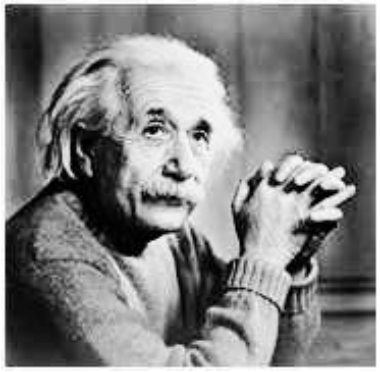
IV. Experiments Result

The images before and after each transformation have been given below:

A. Histogram Equalization



(a) Before



(b) After

Fig. 7: Histogram Equalization, Pic Credit: Tutorial Point

The image used to show histogram equalization has intensity histogram concentrated to darker region and hence the image becomes an ideal candidate for the histogram equalization for abetter detail.
We observer that at beginning we had lot of darker intensity, after histogram equalization the intensity histogram is more flat.

B. Gamma Correction

The image used lot of darker intensity and hence a gamma correction of γ ¡ 1 is an ideal transformation for extracting more information from the image. I gave γ = 0.4 as my input

(a) Before

(b) After

Fig. 8: Gamma Correction, Pic Credit: [8]

C. Logarithmic Transform

We used the log transform as we have lot of dark pixels and can bear the loss of some light pixel to extract a clear image from the image.

(a) Before

(b) After

Fig. 9: Log Transform, Pic Credit: [10]

D. Blur

For the above feature, I gave an input 2 for the blur extent. So a 3X3 box filter was used and following images were obtained.

(a) Before

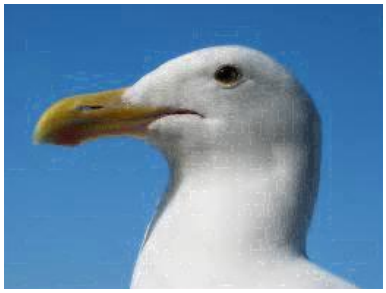
(b) After

Fig. 10: Blur Image, n=2 Pic Credit: [9]

E. Sharpen Image

For the feature, I gave an input 0.7 for the sharpen extent. The following images were obtained.

(a) Before



(b) After

Fig. 11: Sharp Image, k =0.7, Pic Credit: [9]

1) Invert Colors: For this feature, i choose the picture because it had large variety of intensity and color inversion can easily been seen.

(a) Before

(b) After

Fig. 12: Color Inversion, Pic Credit: [11]

V. Conclusion and Discussion

A. Challenges Overcome

- Completed my first Programming assignment
- Completed my first GUI project
- Worked first time with Images
- Implemented vectorization for first time

B. Further Work

- Implement the algorithm to variety of image type
- Live feedback of dynamic parameter on image transformation
- Resizing of image
- Better Sharpening algorithm to avoid artifacts occurrence in image

References

[1] Rafael C. Gonzales, Richard E. Woods, “Digital Image Processing”.
[2] Log Transform https://www.researchgate.net/publication/282265733_An_Integrated_Approach_of_Logarithmic_Transformation_and_Histogram_Equalization_for_Image_Enhancement
[3] Tkinter Tutorial https://www.javatpoint.com/python-tkinter
[4] Tkinter Video Tutorial https://www.youtube.com/watch?v=YXPyB4XeYLA
[5] Canvas in Frame https://stackoverflow.com/questions/7727804/tkinter-using-scrollbars-on-a-canvas
[6] Gaussian Filters https://bohr.wlu.ca/hfan/cp467/12/notes/cp467_12_lecture6_sharpening.pdf
[7] Tkinter Documentation https://docs.python.org/3/library/tk.html
[8] Gamma Image Sample https://i0.wp.com/www.dfstudios.co.uk/wp-content/uploads/2014/02/mandrill_gamlo.jpg?resize=150%2C150&ssl=1
[9] Panda Picture https://wallpapershome.com/animals/wild/
[10] Logarithmic Transformation Picture https://wallpaperaccess.com/full/2110.jpg
[11] Elephant Picture https://www.art.com/gallery/id--b1976-c23951/elephants-black-and-white-photography-prints.htm