Basic Image Editor using Tkinter GUI
EE 610: Image Processing: Assignment 1

Shreyas Nadkarni, 19D170029
Department of Electrical Engineering
IIT Bombay
Mumbai, India
19D170029@iitb.ac.in
Under the guidance of Prof. Amit Sethi, EE610 Instructor, Department of Electrical Engineering, IITB

***Abstract*—This work is a part of the course EE 610: Image Processing, running in Autumn 2021. An image enhancer application has been designed using the tkinter framework of Python, to carry out various image transformations on a custom image. This report summarises the details of the GUI design, functionality, and overall approach used for the task, along with some examples which highlight the importance of the respective operations.**

I. Introduction

The objective of this assignment is to design an interactive application to process and enhance an image. The user can browse and select an image from their device and display it in the GUI. They can then transform the image using operations namely histogram equalization, gamma correction, logarithmic transformation, blurring, sharpening and binary thresholding. Several buttons have been designed to choose the operation and these can be performed sequentially. To reverse the operation(s), "Undo" and "Undo All" buttons have been designed. The code comprises several functions which have been written for each specific operation and for other tasks like browsing, undoing and saving the image. The main block of code defines a tkinter window which launches the main window of the GUI containing the display area and the buttons. The design of the GUI and calls to each function have been made in this main code. The details of the design, operations, experiments and results have been presented in this report.

II. GUI Design

The GUI window contains two parts, one is the panel for displaying the image (henceforth called 'panel' and the other one is a frame for placing the buttons (henceforth called 'buttonframe'). The buttonframe contains the buttons namely: Browse, Save As, Undo, Undo All, Equalize Histogram, Gamma Correction, Log Transform, Blur, Sharpen, Threshold, and then a label displaying the last action performed on the image, followed by some text giving instructions to the user. The actions for browsing, saving, undoing, undoing all, log transformation and histogram equalization use a single function which is called after the button is pressed. The other operations need the user to input some parameter (like the cutoff for thresholding) so they have been designed using two functions each, one for creating a new window and prompting the user to input the value while the other for using the value and carrying out the operation.

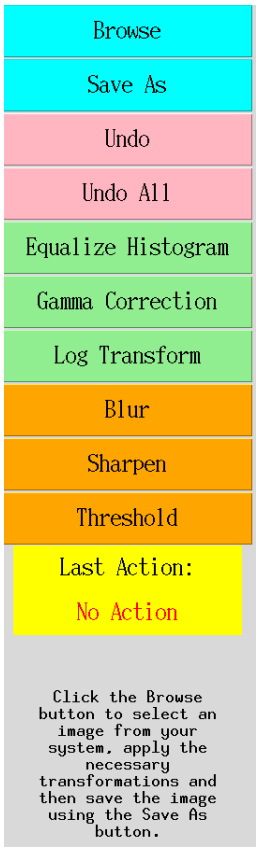


Fig. 1: The frame designed for buttons

III. Image Processing Operations

A. Histogram Equalization

This operation increases the contrast in an image by equalizing the histogram of the pixel values in the image so that each part of the spectrum has sufficient pixels. The transformation used is the discrete analog of the histogram equalization

performed in continuous domain.

$$s_k = T(r_k) = (L-1) \times \sum_{j=0}^{k} p_r(r_j) \tag{1}$$

where r_j is the old pdf value of the j^{th} pixel value and s_k is the corresponding new pixel value of the k^{th} pixel. L is the number of possible intensity levels in the image (here L = 256)

B. Gamma Correction

This is a transformation in which the pixel values are are raised to a power γ where the value of γ is user defined. The pixels are scaled to values between 0 and 1, and can be later re-scaled to [0, 255] after the transformation.

$$s = cr^{\gamma} \tag{2}$$

where r is the old pixel value, s is the new pixel value, and c is an optional scaling factor. In this assignment, c has been kept equal to 1, and scaling has been applied but not inside the operation itself.

C. Log Transformation

This applies a logarithmic function on the pixel values so that the intensity values (scaled to be between 0 and 1) rise. They can then be scaled back to [0, 255]

$$s = c \times log(1 + r) \tag{3}$$

Here, c is chosen to be $\frac{1}{log(2)}$ so that the output values are between zero and 1. After this transformation they are further scaled using a factor of 255, for displaying the image

D. Blurring

Blurring of the image has been implemented using convolution with a box filter whose size is input by the user. The kernel size is stored in a variable named ksize and a zero padding of width $\frac{ksize-1}{2}$ is applied to the original image, before carrying out the convolution. The larger the kernel size, the stronger is the blur, because a larger kernel takes more surrounding pixels into account while finding the output value of one particular pixels, resulting in stronger blurring on the whole.

E. Sharpening

Sharpening of the image has been implemented by unsharp masking. The image is first blurred using a 9x9 box filter, then the blurred image is subtracted from the original image to create a "mask" and then this mask is added with a weight 'c' to the original image. The weight 'c' can be input by the user.

$$m(x,y) = f(x,y) - f_{blur}(x,y) \tag{4}$$

$$g(x,y) = f(x,y) + c \times m(x,y) \tag{5}$$

where $f(x,y)$ is the original image, f_{blur} is the blurred image, $m(x,y)$ is the mask, and $g(x,y)$ is the output (sharpened) image.

F. Binary Threshold (extra operation)

This operation sets each pixel value equal to the minimum (0: black) or to the maximum (255: white) depending on whether it is less than the cutoff or greater than he cutoff. The cutoff value here is input by user. Preferably the image should be gray-scaled, otherwise the colours for the minimum and maximum will depend on the hue and saturation values in the HSV image.

IV. Experiments and Results

Some custom images were chosen and the enhancer was tested on these. Different images have been chosen to show the utility of each operation. The GUI Window looks as follows:

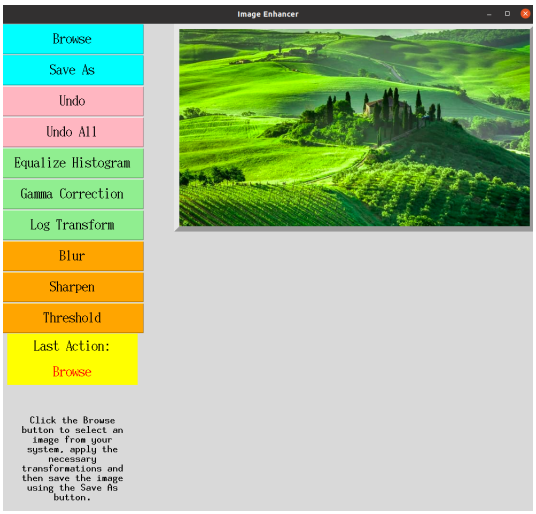


Fig. 2: The Image Enhancer Window

A. Histogram Equalization and Sharpening

Histogram equalization increases the contrast in a low contrast image while sharpening (unsharp masking with c = 1) helps us to see the finer details in more detail, however it introduces some artefacts towards the edges. A low contrast image has been chosen to show the use of these two operations.

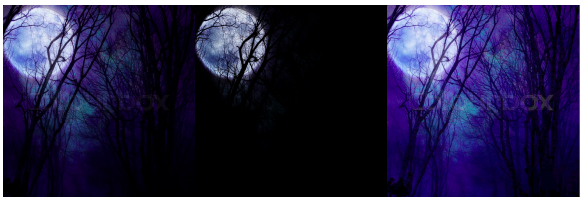
Fig. 3: Original

Fig. 4: Histogram Equalized

Fig. 5: Sharpened with c = 1

B. Gamma Correction

Gamma correction with a gamma¿1 darkens the overall image and increases the contrast in the dark regions, while that with gamma¡1 increases the contrast in the bright parts and brightens the overall image.



(a) Original (b) Gamma = 4 (c) Gamma = 0.5

C. Log Transformation

This looks similar to gamma correction with gamma ¡ 1. It lightens the image and increases contrast in the light parts.

D. Blurring

This can be used for tasks like de-wrinkling where we want to hide some intricate details. In the following image, after blurring with a 3x3 kernel, some wrinkles seem to have reduced leading to the person appearing younger.

E. Thresholding

Thresholding an image converts it to a black and white image which can be used for various applications like designing, advertisements, cartoons, etc. The following image has been gamma corrected by 4 and the thresholded with a cutoff of 127, to get a 'silhouette' of a city.

(a) Original (b) After series of Log Transforms

(a) Original (b) Blurred

Fig. 9: Original

F. Series of Transforms for Enhancing

The following image has a lot of glow due to the sun and the parts of the background like the buildings and the trees are not very clearly visible. To enhance this image, we apply a gamma correction of 4 (which reduces the 'glow'), then

Fig. 10: Thresholding with cutoff = 127 after Gamma Correction with gamma = 4

a log transform twice (which brightens it overall again) and then equalize the histogram to get the final image in which all aspects are clearer: people, tress, leaves on the road, buildings, etc.

Fig. 11: Original

Fig. 12: Enhanced

V. Conclusion and Discussion

Thus, an image editor application has been implemented using tkinter and other Python libraries. The image processing operations have been written without using built-in functions from Open-CV, and have been implemented using only Numpy, which gives a feel of how they are actually implemented mathematically. The challenges faced during the assignment were that of GUI design and interfacing it with the operations using Pillow and tkinter functions, since this was the first time I did GUI Programming. Given more time, I would have liked to implement more functionality such as increasing and decreasing the blurring extent in real time (which would need increasing the efficiency of the operation) and other advanced techniques such as frequency domain filtering for denoising, etc.

References

[1] Digital Image Processing: 4th Edition, Rafael C. Gonzalez, Richard E. Woods
[2] https://www.geeksforgeeks.org/python-gui-tkinter/
[3] https://www.tutorialspoint.com/python/python_gui_programming.html
[4] https://www.pyimagesearch.com/2016/05/23/opencv-with-tkinter/
[5] https://stackoverflow.com/questions/57033158/how-to-save-images-with-the-save-button-on-the-tkinter-in-python
[6] https://www.geeksforgeeks.org/file-explorer-in-python-using-tkinter/
[7] https://stackoverflow.com/questions/10133856/how-to-add-an-image-in-tkinter