Basic Image Editor

Kushal Kejriwal
190110040
Department of Electrical Engineering

***Abstract*—A lot of the times, the images obtained in real time have problems with them and some information is not visible. We can perform different type of enhancements necessary to extract certain useful information. We will also develop a Graphical User Interface (GUI) which provides an interface to the users to be able to load, manipulate and save the enhanced images**

***Index Terms*—Image enhancement, GUI, python**

I. Introduction

This report starts of with the design of the Graphical User Interface (GUI) describing the library, overall approach and the final interface which was developed. Then we describe the various image enhancement operations which are Log Transform, Gamma Transform, Sharpening, Blurring, Histogram Equalisation and Negative Transform. We then provide some examples of each operation which consists of original image and the enhanced version. We also take some images through multiple enhancements. We will also describe the main challenges faced here.

II. Graphical User Interface (GUI) Design

A. Overall Approach

The GUI was developed using a python library **tkinter**.It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Using buttons, the images were enhanced (the particular transformation code ran in the backend) and displayed in the image area.

B. GUI features

Initially, the user has to load an image and hence will need a **Load Image**. Upon clicking this button, the user will have to navigate and select a particular image (**jpeg** and **png** formats allowed) which will be displayed in the image display area. Now the image enhancement, undo once, undo all and save buttons will appear on the GUI

- **Image Enhancement**: 6 buttons each doing a particular enhancement. Some of them need an input to control the extent. There is a box below each of these buttons where the user can enter the input and then click on the button
- **Undo Once/All**: Undo once will restore the previous image whereas undo all will revert back to the original image
- **Save**: The user can save the enhanced image using this button

Identify applicable funding agency here. If none, delete this.

C. Code Blocks and Description

Upon clicking the load file button, a function of called which did two things. One, it opens up a dialog box to select the file and two, it introduces the plethora of buttons mentioned above. This was done using *tk.button* and each button had a particular name and was linked to a particular function. Code for an example button:

```
    log_button = tk.Button(
     window,
     text = "Log Transform",
     command=lambda: convert_log(im_store)
     )
     log_button.place(relx=0.3, rely=0.7,
     anchor=CENTER)
```

Some functions needed an input to control the extent of the enhancement. For this I kept a text box below the button. The user will enter input in this button and upon clicking the text box is passed as an argument to the function. The function will extract the input from the label and perform the enhancement

```
input_sharpen = tk.Text(window,
height = 2,width = 10)
input_sharpen.place(
relx=0.7, rely=0.75,
anchor=CENTER)
```

III. Image Processing Operations

A. Image Processing Operations Implemented

- Equalize histogram
- Gamma correct (ask for input gamma upon pressing the button)
- Log transform
- Blur with a mechanism to control the extent of blurring
- Sharpening with a mechanism to control the extent of sharpening
- Negative Transformation

B. Purposes of the implemented operations

- **Equalize histogram**: Helps in stretching the clumped intensity values to the full range thus improving contrast
- **Gamma correct**: Controls the overall brightness. Makes a very dark image bright and vice versa
- **Log transform**: Convex function which improves the overall brightness of an image

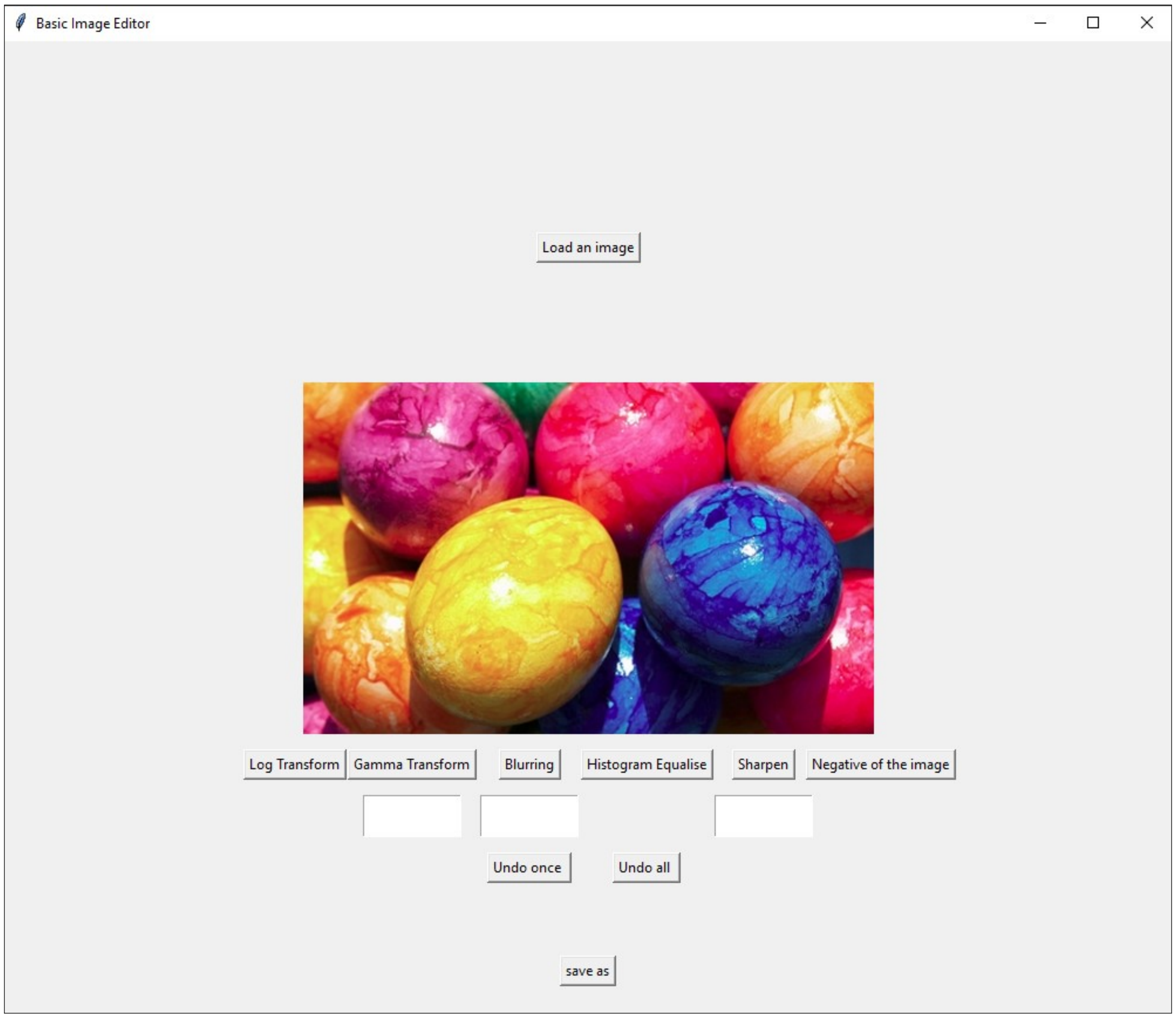

Fig. 1. The GUI Interface

- **Blur**: Helps in reducing the noise and smoothens the image
- **Sharpening**: Helps in sharpening the image by enhancing the edges. Increases noise
- **Negative Transform**: Inverts the image

C. Mathematical Formulae

r is the input and s is the output. r_k denotes the k^{th} pixel intensity. c is a constant for scaling. M, N are the dimensions of the image

- **Equalize histogram**:

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

The implemented version uses the formula

$$s_k = round\,(255\,(\frac{cdf(r_k) - cdf_{min}}{MN - cdf_{min}}))$$

- **Gamma correct**:

$$s = T(r) = cr^{\gamma}$$

- **Log transform**:

$$s = T(r) = c\,log(r+1)$$

- **Blur**:

$$s = T(r) = f * r$$

f is a kernel of size m (depends on the extent) with each value being $\frac{1}{m^2}$

- **Sharpening**:

$$s = T(r) = r + cf * r$$

f is a laplacian $3 * 3$ kernel

- **Negative Transform**

$$s = T(r) = 255 - r$$

IV. Experiments and Results

V. Conclusions and discussions

Main Challenges

The main challenges I faced were as follows:

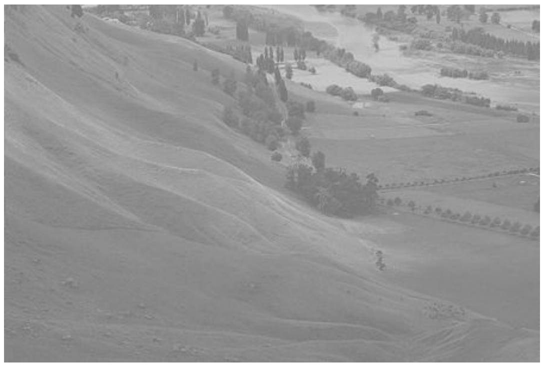

Fig. 2. Original Image

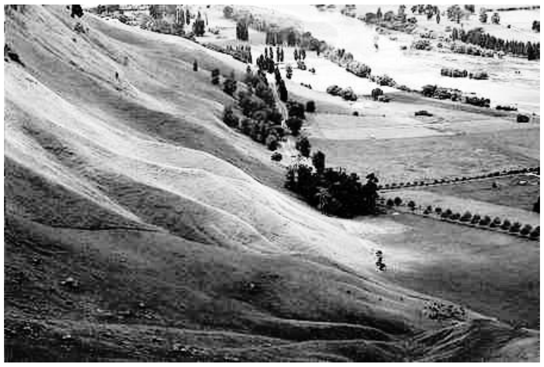

Fig. 3. Histogram equalised

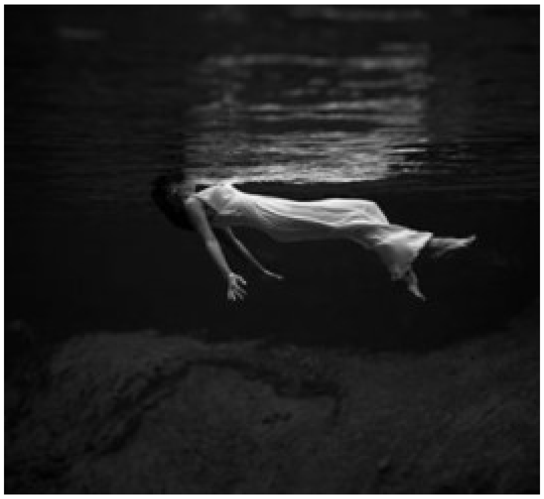

Fig. 4. Original Image

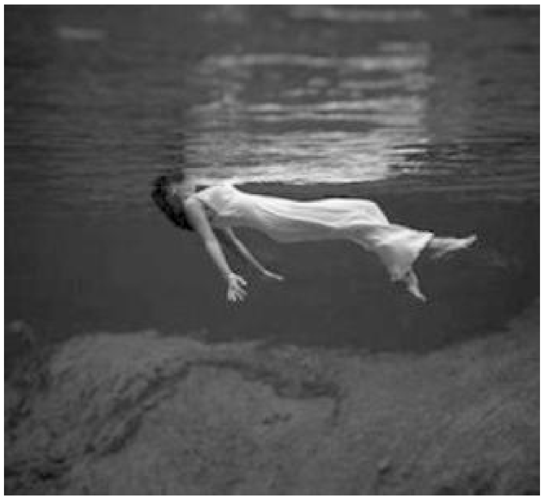

Fig. 5. $\gamma = \frac{1}{2}$

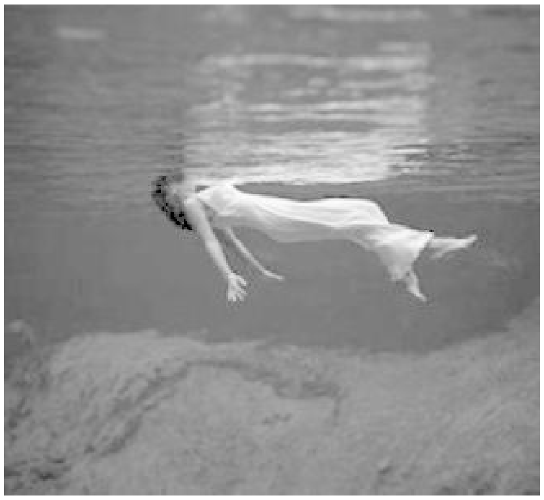

Fig. 6. $\gamma = \frac{1}{4}$

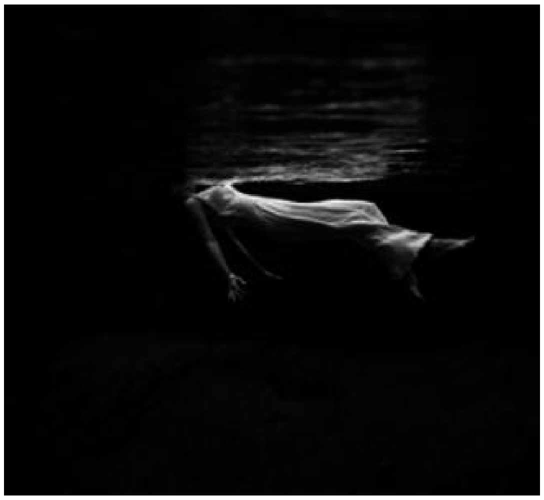

Fig. 7. $\gamma = 2$

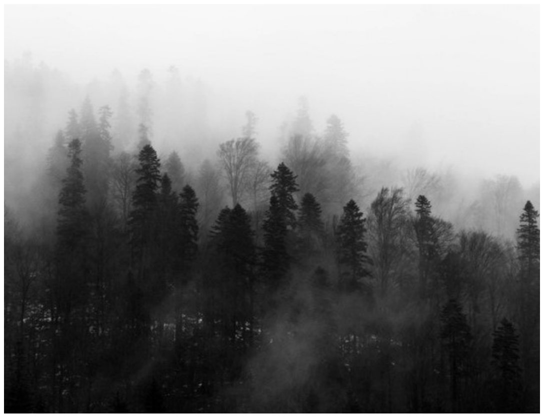

Fig. 8. Original image

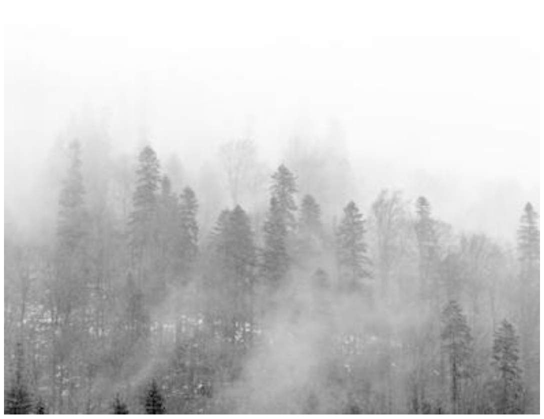

Fig. 9. Log Transformed enhanced image

- I had no prior experience of any GUI package with Python. This was a big challenge since I needed to create an extensive GUI for the image enhancement operations. I overcame this by using the most basic and well supported library **tkinter** . The extensive documentation helped me get a brief overview of the various features and I could then implement them in my project
- Image inputs can be of two types - grayscale and colour. Hence I needed to write code which could handle both these types of images. I converted the RGB images to HSV and manipulated only the V channel for colour images whereas I directly manipulated the grayscale images since they are 2D images
- There were many issues with the image enhancement operations since these were implemented from scratch. Hence I had to take care of various cases such as overflowing due to 8bit representation, RGB ordering, scaling, encountering zero in log transforms

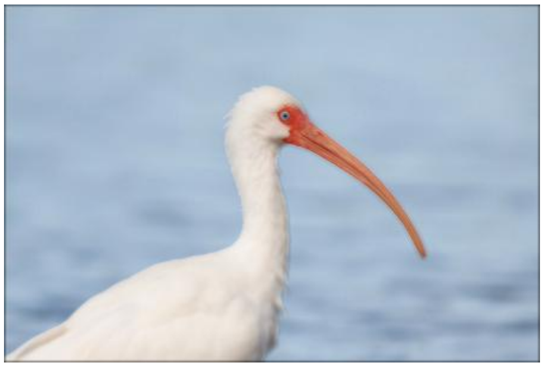

Fig. 11. Blurred Image

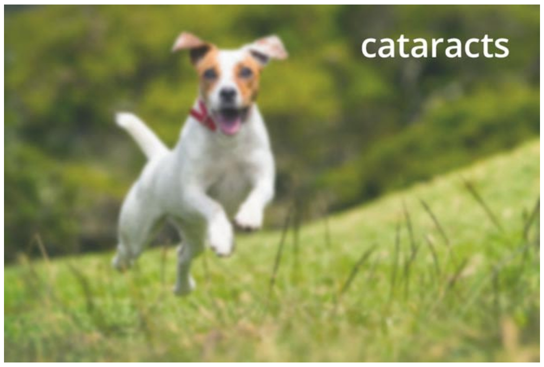

Fig. 12. Original Image

Further improvements

Given more time, I would have liked to

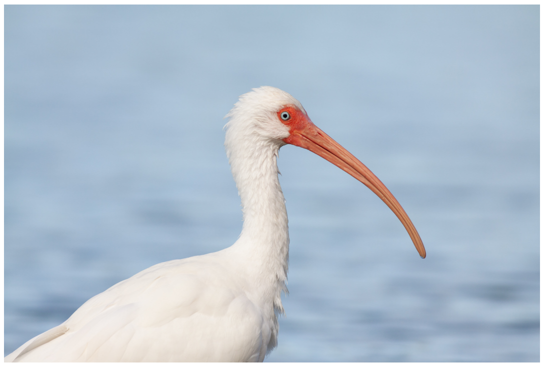

Fig. 10. Original Image

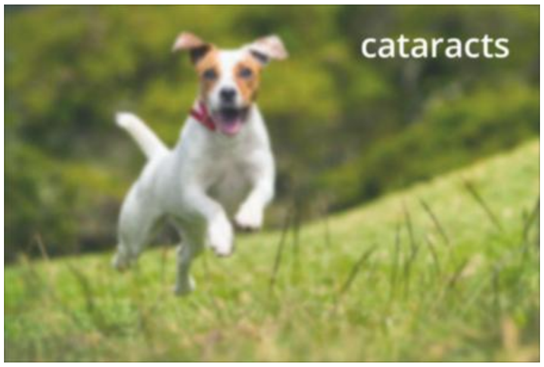

Fig. 13. Sharpened Image

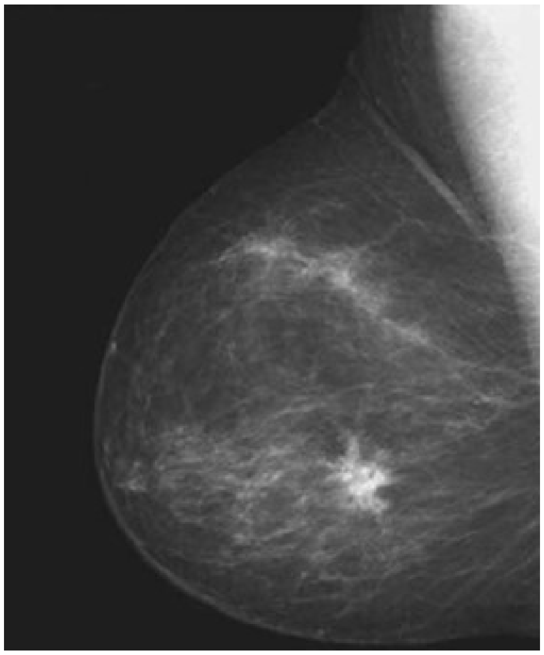
Fig. 14. Original Image

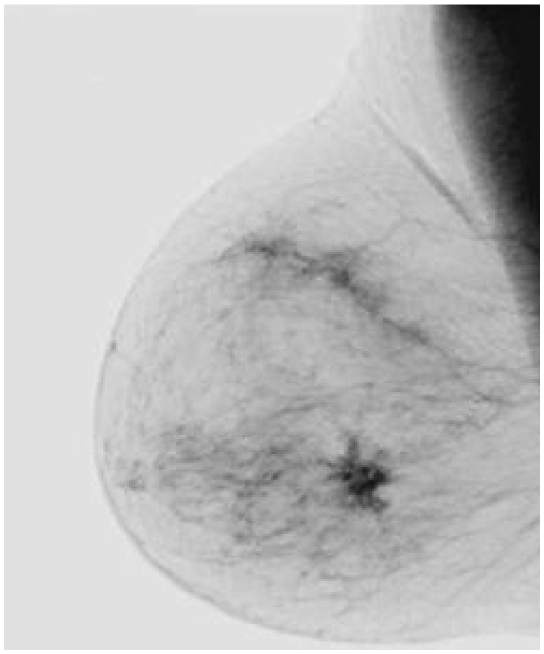
Fig. 15. Negative Image

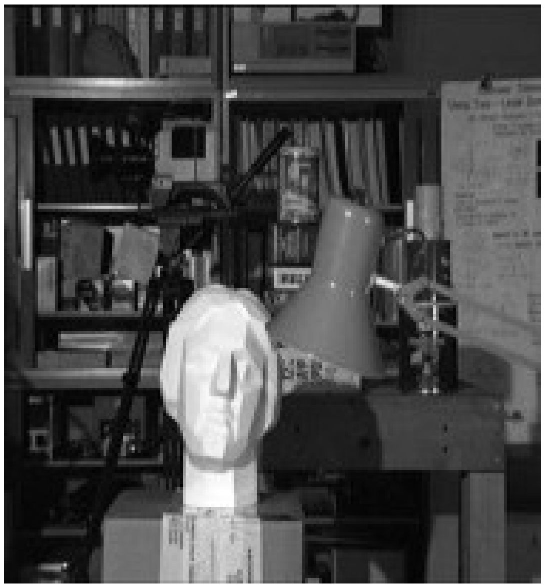
Fig. 16. Multiple Enhancements - Original Image

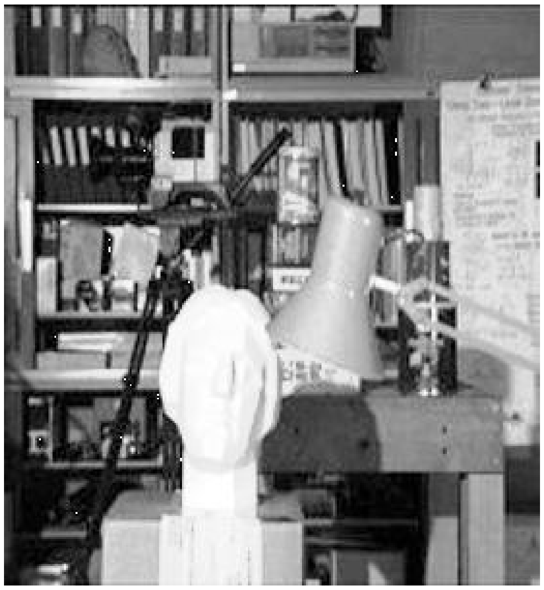
Fig. 17. Multiple Enhancements - Histogram Equalisation

- Improve my sharpening filter. The current laplacian implementation does not work that well on the images. I would like to try out more techniques such as unsharp masking and frequency filtering to obtain better results
- Give warnings on GUI. Currently if there is an error such as undoing too many times or no image selected, I only get a warning on my python terminal. I would like to show the error to the user and suggest solution

References

[1] Digital Image Processing, Gonzalez and Woods, Prentice Hall
[2] Stack Overflow https://stackoverflow.com/
[3] Tkinter Documentation https://docs.python.org/3/library/tkinter.html
[4] Histogram Equalisation https://en.wikipedia.org/wiki/Histogram_equalization

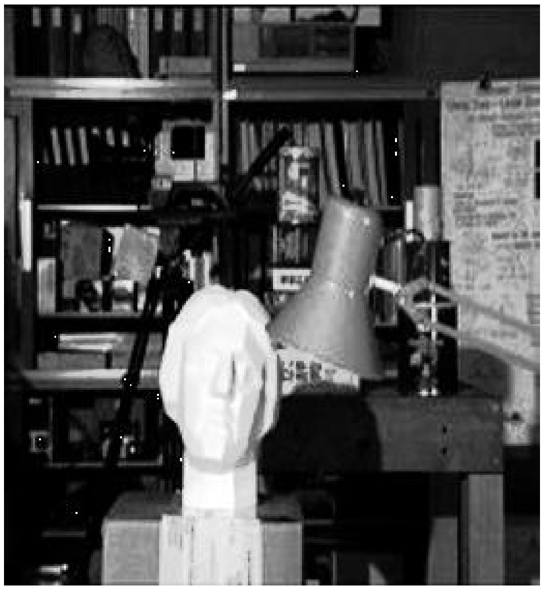
Fig. 18. Multiple Enhancements - Gamma Transform

[5] Tkinter Tutorials https://www.tutorialspoint.com/python/tk_button.htm