# BASIC IMAGE EDITOR USING PYTHON

M.V.MANI KUMAR, 203070012
Assignment Report
Department of Electrical Engineering
Indian Institute of Technology, Bombay, India
203070012@iitb.ac.in

***Abstract*—In this report we are designing a GUI that consist of different image processing techniques using python.In this can handle both gray scale and rgb images.The image processing techniques we are using here are histogram equilization,gamma transform,log transform,blurring and sharpening.Gui layout and different features of gui layout are discussed in this report**

## I. Introduction

### A. Objectives:

The objective is to built a Gui which includes loading an image it should handle both rgb and grayscale and image processing operations like

*1) Histogram Equilization:* Execute the cumulative sum:
cs = cumsum(hist)
re-normalize the cdf
nj = (cs - cs.min()) * 255
N = cs.max() - cs.min()
cs = nj / N

*2) Gamma Correction:* Gamma correction python code:
gamma$_corrected = np.array(255 * (img/255) * *gamma, dtype =' uint8')$

*3) Log Transform:* Log Transform python code.
c = 255/(np.log(1 + np.max(img)))
log$_transformed = c * np.log(1 + img)$

*4) Blurring Operation:* Gaussian(Low pass filter) = (1 / 16.0) * np.array([[1., 2., 1.], [2., 4., 2.], [1., 2., 1.]])
image = convolve2d(image, kernel, 'same', boundary = 'fill', fillvalue = 0)

*5) Sharpening Operation:* Laplacian= np.array([[0., -1., 0.], [-1., 5., -1.], [0., -1., 0.]])
image = convolve2d(image, kernel, 'same', boundary = 'fill', fillvalue = 0)

*6) Sobel Operation:* Laplacian= np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])
image = convolve2d(image, kernel, 'same', boundary = 'fill', fillvalue = 0)

*7) Important Note On RGB Image Operations:* While performing image processing operations on color images they should be converted to HSI/HSV or Lab, and only the I/V/L channel should be manipulated.
HSVimg = cv2.cvtColor(img,cv2.COLORBGR2HSV).

## II. GUI Design

### A. Approach of GUI Programming :

Here we have used Tkinter Python Gui for the creation of the GUI and Canvas function is used for displaying the images.Pillow Library is used for opening and loading of images.In overall Gui place function in Tkinter is used to place buttons ,canvas, control bars on the screen.Black background and gold foreground is use for buttons.All the image processing operations are given as command to buttons in the Gui.Size of Gui is given as 800*800

### B. Gui Features :

*1) load an image and display:* This task is executed with a button named(Select Image).A function named(def selected()) for loading image from local directory is defined and it is given as a command to the button.Canvas is created for the display of the loaded image.Here the same canvas is used for all other features in the Gui.
def selected() function:
imgpath = filedialog.askopenfilename(title="select a file",filetypes=(("png files", "*.jpg"),("all file", "*.*")))
img = Image.open(imgpath)
img1 = ImageTk.PhotoImage(img)
myStack.append('v')
canvas2.create$_image(300, 210, image = img1)$
$canvas2.image = img1.$
$Canvas\ and\ Button\ Creation:$
$canvas2 = Canvas(root, width = "600", height = "420", relief = RIDGE, bd = 2)$
$canvas2.place(x = 15, y = 150)$
$btn1 = Button(root, text = "SelectImage", bg =' black', fg =' gold', font = ('ariel15bold'), relief = GROOVE, command = selected)$
$btn1.place(x = 20, y = 595)$

*2) Histogram Equilize:* This task is executed with a button named(Histogram Equilize).A function named(def histogram equilization()) is defined for histogram equilization operation.It is given as a command to the button.The image path and image variables which are used in the loading of images

are defined as global variables and utilized here.
Button Creation and Placement:

btn2 = Button(root, text="Histogram Equilize", bg='black', fg='gold', font=('ariel 15 bold'), relief=GROOVE, command=selected)
btn2.place(x=160, y=595)

*3) Log Transform:* This task is executed with a button named(Log Transform).A function named(def Log Transform()) is defined for Log Transform operation.It is given as a command to the button.The image path and image variables which are used in the loading of images are defined as global variables and utilized here.
Button Creation and Placement:

btn3 = Button(root, text="Log Transform", bg='black', fg='gold', font=('ariel 15 bold'), relief=GROOVE, command=selected)
btn3.place(x=335, y=595)

*4) Gamma Correction:* This task is executed with a button named(Gamma correct).A function named(def gamma Transform()) is defined for Gamma Correction operation.It is given as a command to the button.The image path and image variables which are used in the loading of images are defined as global variables and utilized here.Here for the input of the gamma from user another function called (def click()) is defined and Entry.get() is used in the function to get the user input gamma value.In the Gui it is designed in such a way that upon clicking gamma correction Entry widget will appear on the screen for the input value .
def click() function:
Get the input
val = myentry.get()
try:
Try to make it a float
val = float(val)
return val
except ValueError:
return val
Button Creation and Placement: btn4 = Button(root, text="Gamma Correction", bg='black', fg='gold', font=('ariel 15 bold'), relief=GROOVE, command=selected)
btn4.place(x=505, y=595)

*5) Blurring operation bar:* This task is executed with a control scale named(Blur).A function named(def blurring(event)) is defined for Blur operation.It is given as a command to the scale.The image path and image variables which are used in the loading of images are defined as global variables and utilized here.The scale value is taken into blur operation function as a iteration for convolve function.Here variable V1 is used as a scale value
Scale Creation and Placement: scale1= ttk.Scale(root, from$_=0, to = 10, variable = v1, orient = HORIZONTAL, command = blurring)scale1.place(x = 150, y = 10)$

*6) sharpening operation bar:* This task is executed with a control scale named(sharp).A function named(def sharp(event)) is defined for sharp operation.It is given as a command to the scale.The image path and image variables which are used in the loading of images are defined as global variables and utilized here.The scale value is taken into sharp operation function as a iteration value for convolve function.Here variable V2 is used as a scale value
Scale Creation and Placement: scale2= ttk.Scale(root, from$_=0, to = 10, variable = v1, orient = HORIZONTAL, command = blurring)scale2.place(x = 150, y = 55)$

*7) SAVE:* This task is executed with a button named(SAVE).A function named(def Save()) is defined for saving the image in local directory.
def save():
global img$_path, imgg, img1$

ext = img$_path.split(".")[-1]$
$file = asksaveasfilename(defaultextension = f".ext", filetypes = [("AllFiles", "*.*"), ("PNGfile", "*.png"), ("jpgfile", "*.jpg")])$
$iffile : ifcanvas2.image == img1 : imgg.save(file)$

*8) UNDO:* This task is executed with a button named(UNDO).A function named(def Undo()) is defined for Undo last change in the image.For this function i have stored output image of every operation in local directory and assigned a stack variable to it upon executing any particular image processing operations mentioned above the stack will append that variable.On executing undo function top value in the stack will be popped out and the image output assigned to the stack value now which is at the top will be displayed on the canvas
def undo() function:
global img$_path, img, imgr$
$myStack.pop()$
$if(myStack[len(myStack) - 1] == "a") :$
$img = Image.open("hist.jpg")$
$elifmyStack[len(myStack) - 1] == "b" :$
$img = Image.open("log.jpg")$

elif myStack[len(myStack)-1]=="c":
img = Image.open("gamma.jpg")

elif myStack[len(myStack)-1]=="d":
img = Image.open("blur.jpg")

elif myStack[len(myStack)-1]=="e":
img = Image.open("sharp.jpg")
elif myStack[len(myStack)-1]=="v":
img = Image.open(img$_path)$

*9) RESET:* This task is executed with a button named(UNDO).A function named(def reset()) is defined for revert to original image.This function works similar to

undo but it will display original image def reset() function global img$_path, img, imgr$
$myStack.pop()$
$if(myStack[0] == "v"):$
$img = Image.open(img_path)$

## III. IMAGE PROCESSING OPERATIONS

*1) Histogram Equilization:* Histogram equalization is a technique for adjusting image intensities to enhance contrast.// Let f be a given image represented as a $m_r$ by $m_c$ matrix of integer pixel intensities ranging from 0 to $L1$. $L$ is the number of possible intensity values, often 256. Let p denote the normalized histogram of $f$ with a bin for each possible intensity.
The cdf is a cumulative sum of all the probabilities lying in its domain and defined by

$$CDF(x) = \sum_{k=-\infty}^{k=\infty} P(k) \tag{0}$$

*2) Log Transform:* Log transform means replacing each pixel with the logarithim value.
The log transformation can be defined by the formula

$$s = c\log(1+\gamma)$$

where $s$ and $r$ are the pixel values of output and the input image and c is a constant .Here 1 is added in case pixel value is zero then $log(0)$ will be infinity so to avoid it and to have a minimum value of 1

*3) Gamma correction:* Gamma correction or power-law curves with fractional values of $/gamma$ map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input levels Gamma correction (or) Power-law transformations have the formula

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

*4) Low Pass Gaussian Blur:* Smoothing (also called averaging) spatial filters are used to reduce sharp transitions in intensity. Because random noise typically consists of sharp transitions in intensity, an obvious application of smoothing is noise reduction. Linear spatial filtering consists of convolving an image with a filter kernel. Convolving a smoothing kernel with an image blurs the image, with the degree of blurring being determined by the size of the kernel and the values of its coefficients

$$h(x,y) = e^{\frac{-x^2+y^2}{2\sigma^2}}$$

*5) Laplacian Sharpening:* The Laplacian operator is an example of a second order or second derivative method of enhancement. It is particularly good at finding the fine detail in an image. Any feature with a sharp discontinuity (like noise, unfortunately) will be enhanced by a Laplacian operator. Thus, one application of a Laplacian operator is to restore fine detail to an image which has been smoothed to remove noise. (The median operator is often used to remove noise in an image.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

$$g(x,y) = f(x,y) + c[\nabla^2 f]$$

where $f(x,y)$and $g(x,y)$ are the input and sharpened images, respectively.Here in our programming we used $c = -1$

## IV. EXPERIMENTS AND RESULTS

### A. GUI SCREEN

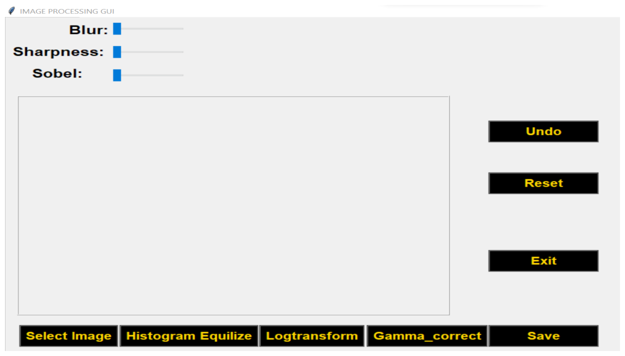

Fig. 1: GUI SCREEN

### B. Histogram Equilization

This Grayscale Image is choosen to enhance the image quality and image contrast is also improved.

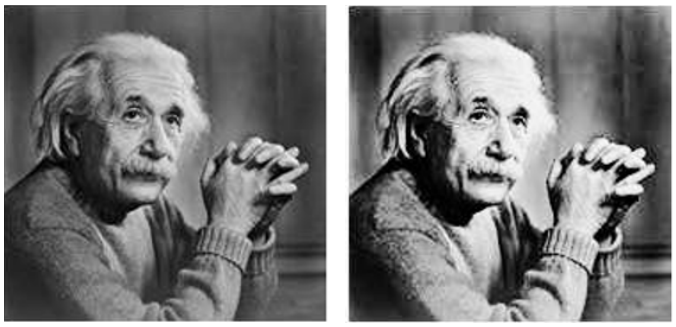

Fig. 2: Original Image [left] and Histogram Equilize Image[right]

### C. Gamma Correction

$\gamma = 1.5$

In this image we can see that dark areas contrast is increased and light areas contrast is decreased overall image looks pleasant to the eyes.

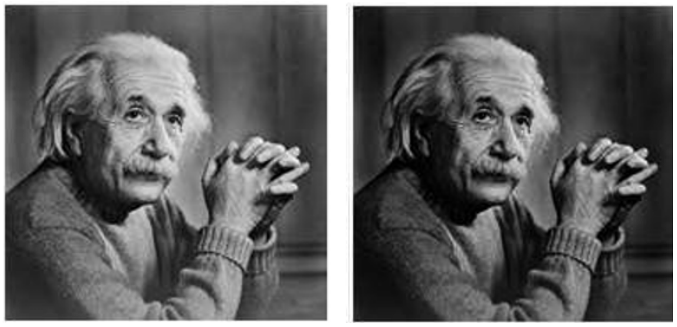

Fig. 3: Original Image [left] and Gamma correction Image [right]

### D. Log Transform

In this image by executing the log transform we came to see some unnoticed details in the transformed image which are not visible in the original image

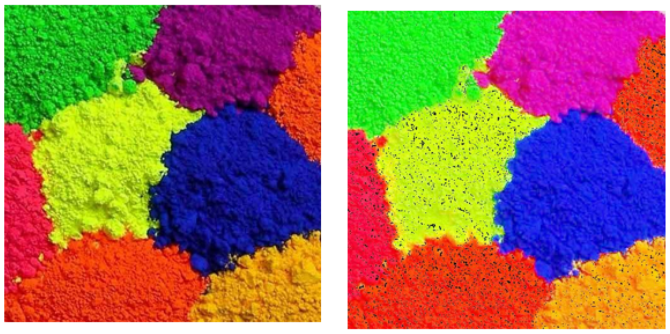

Fig. 4: Original Image [left] and Log Transform Image[right]

### E. Blur

Here i have taken this image to see the blur effect clearly

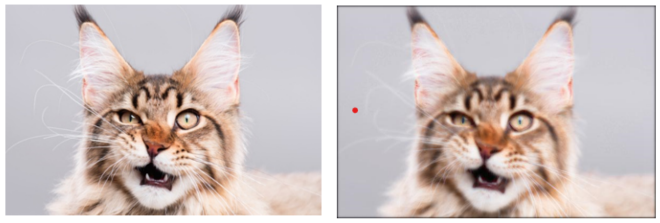

Fig. 5: Original Image [left] and Blurred Image [right]

### F. Sharpen

We can see some unnoticed details clearly after sharpening the image especially facial features of cat

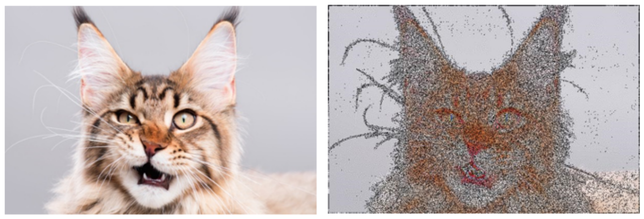

Fig. 6: Original Image [left] and Sharpen Image [right]

### G. Sobel

We can see some unnoticed details clearly after sharpening the image especially along the edges and hair of cat

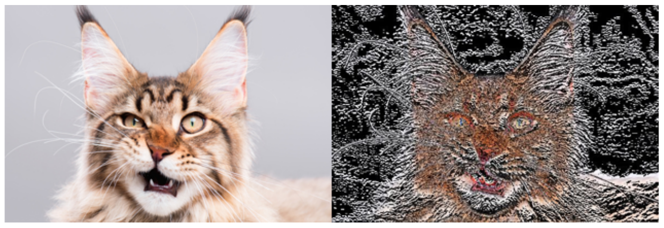

Fig. 7: Original Image [left] and Sobel Image [right]

## V. Conclusion and Discussion

The main challenge is creation of gui using tkinter .Especially in taking the user input for gamma value in gamma correction changing the int to float and getting the entry widget.Tried to do the convolve function on own but failed if the time is more this would be possible.Setting up the dimensions for canvas and placement of buttons taken lot more time.Initially working with pillow library and open cv is difficult while creating Gui.Given more time i would have made gui more fluid and blur operation smooth and visual asthetics of Gui can be improved.Blurring and Sharpening can be done in various filters if more time is allocated

## References

[1] https://github.com/SVLaursen/Python-RGB-to-HSI/blob/master/converter.py

[2] https://stackoverflow.com/questions/50578876/histogram-equalization-using-python-and-opencv-without-using-inbuilt-functions

[3] https://github.com/Jpoliachik/PythonImageProcessing/blob/master/ImageProcessingGUI.py

[4] https://github.com/ashwin-pajankar/Python-OpenCV3/tree/master/01%23%20Basics

[5] https://towardsdatascience.com/image-processing-with-python-blurring-and-sharpening-for-beginners-3bcebec0583a

[6] https://www.pyimagesearch.com/2016/05/23/opencv-with-tkinter/#download-the-code

[7] Rafael C. Gonzalez and Richard E. Woods, *Digital Image Processing Fourth Edition-Pearson Publicatons-2018*.