Image Processing Assignment 1

Basic Image Editor

Sunaina Saxena
Roll Number-213070001
EE1-Communication Engineering
Indian Institute of Technology, Bombay
213070001@iitb.ac.in

***Abstract*—This report contains detailed description of basic image editing tools along with their python code and output images for the same. For making these image enhancement techniques more intuitive, basic GUI features such as loading image from files present in the system and creating buttons for indicating different operations are included. The approach of GUI design is clearly explained in the report. The challenges that were faced while working on this assignment are also stated.**

***Index Terms*—**

I. Introduction

A. Objectives

This assignment mainly focus on explaining some basic operations that can be performed on an image to enhance its features and improving its quality. It also allows one to apply the theoretical knowledge of image enhancement techniques to reality through applying some coding skills.

B. Main parts of code

- Create image display area.
- Generate image load button that opens file selector.
- Create several image manipulation buttons to perform various operations such as Histogram equalization,Image negative, Log transform, Gamma correction, Image blurring, Image sharpening.
- Create Buttons to undo last change, undo all changes, save the current image, etc.

II. Ease of Use

The code is well organised, variables are given names according to their operations and comments are written after each line of code that requires explanation.

III. Graphical User Interface(GUI) design

Graphical user interfaces are the standard of user-centered design in software application programming, providing users the capability to intuitively operate computers and other electronic devices through the direct manipulation of graphical icons such as buttons, scroll bars, windows, tabs, menus, cursors, and the mouse pointing device. Many modern graphical user interfaces feature touch-screen and voice-command interaction capabilities.
There are many GUI frameworks available in python such as Kivy, Libavg, PyQT, PySimpleGUI, Pyforms, Tkinter, etc.
The one that is used for designing GUI in this assignment is **Tkinter**.

A. Overall Approach of GUI design

The tkinter package is a thin object-oriented layer on top of Tcl/Tk. Tkinter is a set of wrappers that implement the Tk widgets as Python classes.

B. GUI Features

- **from tkinter import ***
  Support for Tkinter is spread across several modules. Most applications will need the main tkinter module,as well as the tkinter.ttk module, which provides the modern themed widget set and API.In this assignment only tkinter module is used.
- **from tkinter import filedialog**
  It is a dialog to allow the user to specify a file to open or save. Here it is used to open the system files for selecting image to load.
- **from tkinter import simpledialog**
  The SimpleDialog module is used to create dialog boxes to take input from the user in a variety of ways. SimpleDialog allows us to take input of varying datatypes from the user, such as float, string and integer. Here it is used for controlling the extend of blurring,sharpening,etc by taking input from the user.
- **root=Tk()**
  To initialize tkinter, we have to create a Tk root widget, which is a window with a title bar and other decoration provided by the window manager.

- **The Canvas widget**
  The canvas widget provides the basic graphics facilities for Tkinter, and more advanced functions.Each item on a canvas is enclosed by a bounding box, which is defined using 2 points: the top-left corner and the bottom-right corner of the box.Window is used to place other widgets on the canvas (makes the canvas widget act like a

geometry manager).

– **Tkinter Buttons**
The Button widget is a standard Tkinter widget, which is used for various kinds of buttons. A button is a widget which is designed for the user to interact with, i.e. if the button is pressed by mouse click some action might be started.

```
root=Tk()
root.geometry('800x500')

#=========================define background image=================================
img=ImageTk.PhotoImage(file='C:/Users/sunfo/IdeaProjects/GUI/at images/back.jpg')
#=========================create canvas=================================
my_canvas=Canvas(root,width=800,height=500)
my_canvas.pack(fill='both',expand=True)
#=============================set image in canvas============================
my_canvas.create_image(0,0,image=img,anchor='nw')
#=================================add a label=======================================
my_canvas.create_text(400,50,text='IMAGE PROCESSING:)',font=50,fill='white')
#=================================add some buttons=================================
root.title('image processing assignment 1')
button1=Button(root,text='load_image',command=image_load)
button2=Button(root,text='gamma_correct',command=gamma_correction)
button3=Button(root,text='log_transform',command=log_transform)

button1_window=my_canvas.create_window(500,100,anchor='nw',window=button1)
button2_window=my_canvas.create_window(600,100,anchor='nw',window=button2)
button3_window=my_canvas.create_window(700,100,anchor='nw',window=button3)
```

Fig. 1. GUI Code

C. Image Processing Operations

The image processing operations implemented are as follows:

– **Conversion of image from RGB to HSV and GRAYscale and vice versa**
If we see an image, we see multiple combinations of color(s). Every value carries information about the image and we can alter the values according to our requirement. If we talk about the color sensors present in a human eye, cones are the sensors responsible for color vision. Approximately 65 percent of all cones are sensitive to Red, 33 percent are sensitive to green and left 2 percent are sensitive to blue light. However, blue cones are most sensitive.There are a handful of colour spaces to represent an image such as RGB, BGR, HSV, CMYK etc. But there is something they all have in common. They are the channels, which these colour spaces use, to collectively form an image.
Grayscale = R / 3 + G / 3 + B / 3
The weighted method, also called luminosity method, weighs red, green and blue according to their wavelengths. The improved formula is as follows:
Grayscale = 0.299R + 0.587G + 0.114B
HSV (hue, saturation, value) and HSL (hue, saturation, lightness or luminance) are transformations of a Cartesian RGB color space.

– **Histogram Equalization**
Histogram equalization is one of the Pixel brightness transformations techniques. It is a well-known contrast enhancement technique due to its performance on almost all types of image.

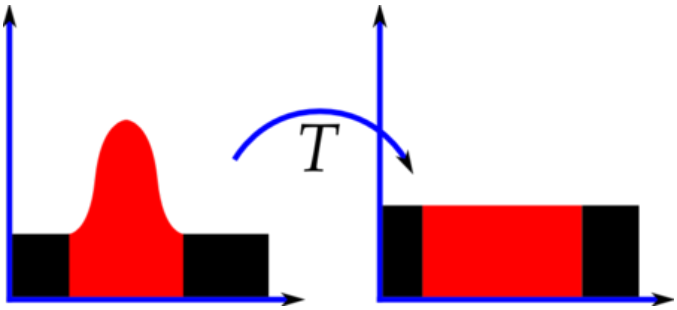

Fig. 2. Histogram before and after equalization

Steps involved
1-Get the input image.
2-Generate the histogram for the image.
3-Find the local minima of the image.
4-Divide the histogram based on the local minima.
5-Have the specific gray levels for each partition of the histogram.
6-Apply the histogram equalization on each partition.

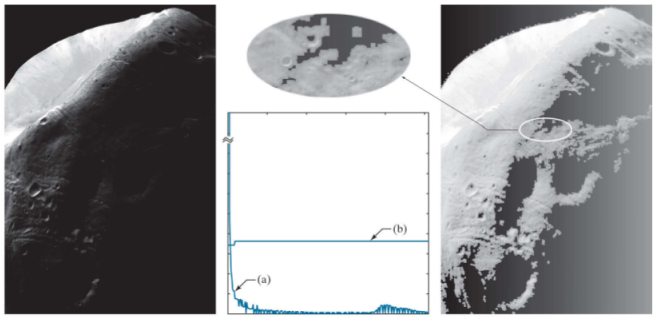

Fig. 3. NASA'S Mars Global Surveyor

– **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. Images which are not properly corrected can look either bleached out, or too dark. Trying to reproduce colors accurately also requires some knowledge of gamma. Varying the amount of gamma correction changes not only the brightness, but also the ratios of red to green to blue.

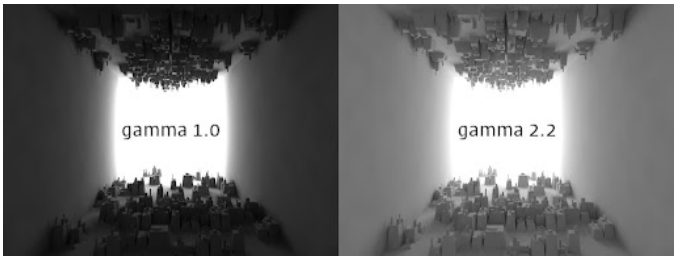

Fig. 4. Image with different Gamma values

$$S = Cr^{\gamma}$$

The reason gamma correction is used is because the input signal, or voltage, sent to a monitor is not high enough to create a bright image. Therefore, if the gamma is not altered, the images on your screen would be dark and difficult to see. By applying gamma correction, the brightness and contrast of the display are enhanced, making the images appear brighter and more natural looking.

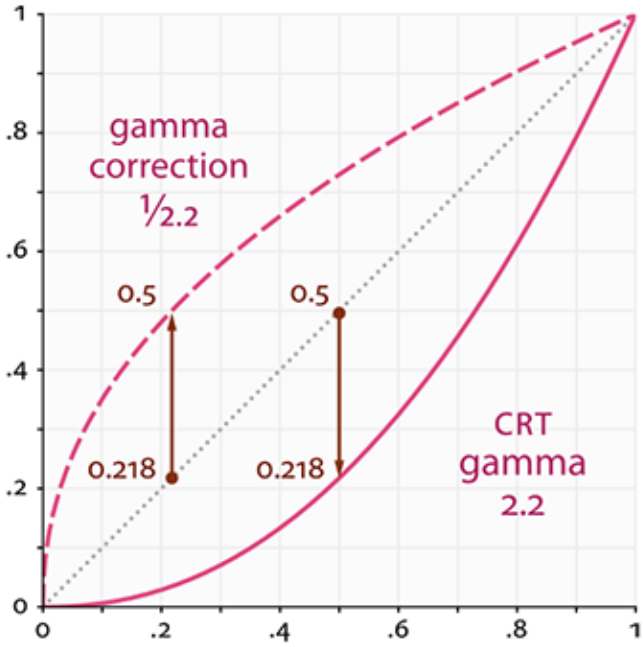

Fig. 5. Pixel intensity transformation for gamma and inverse gamma

– **Log Transform**
Log transformation means replacing each pixel value with its logarithm. The general form of log transformation function is:
$s = T(r) = c * log(1 + r)$
Where, 's' and 'r' are the output and input pixel values and c is the scaling constant represented by the following expression (for 8-bit):
$c = 255/(log(1 + maxinputpixelvalue))$

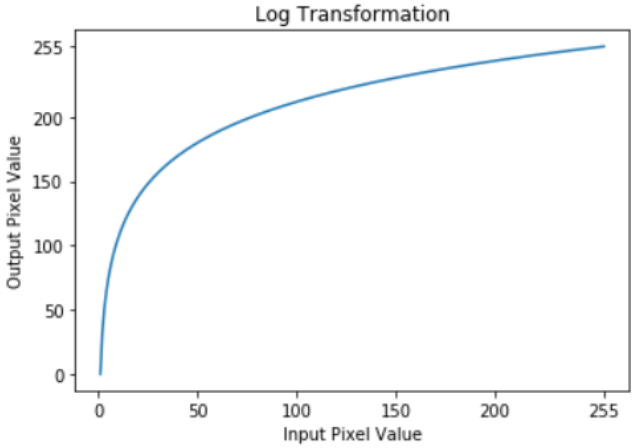

Fig. 6. Graph showing low intensities mapped to higher pixel values

Fig. 7. Image before and after log transform

– **Filtering operations**
Filtering is a technique for modifying or enhancing an image. For example, you can filter an image to emphasize certain features or remove other features. Image processing operations implemented with filtering include smoothing, sharpening, and edge enhancement.
– **1-Smoothing of image**
Average filters take the mean value of the pixels in a neighborhood, which is defined by the size of a mask (m-columns and n-rows). It is important to divide by the sum of the values in the neighborhood to normalize output values.

Another linear filter is performed with a **weighted average filter** to multiply pixels by different coefficients, thus giving more importance (weight) to some other pixels.The mask size increases the blurring factor, because at each convolution it calculates the average output with more pixels values. For example, a mask with m=15 will blends small objects in the background.

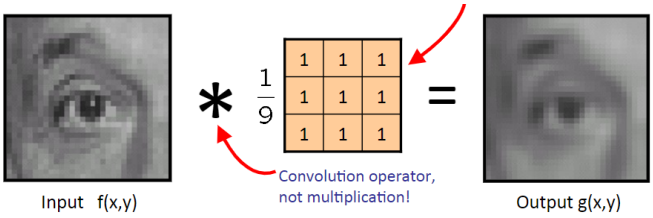

Fig. 8. Average mask

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Fig. 9. Weighted average filter mask

There are also non linear filters such as **Gaussian filter,Median filter** which provides smoothing effect.

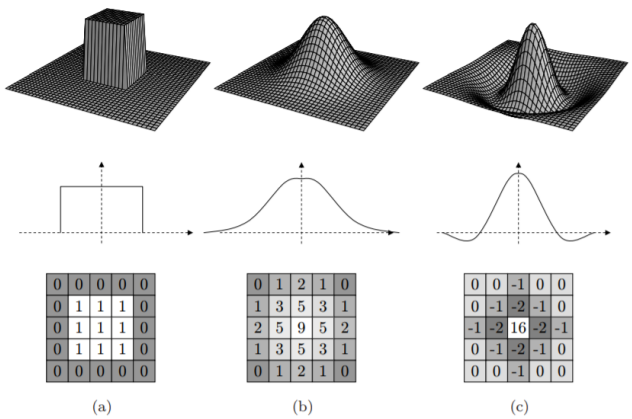

Fig. 10. Gaussian mask

– **2- Edge Detect and Sharpening**

Most of the shape information of an image is enclosed in edges. So first we detect these edges in an image and by using filters and then by enhancing those areas of image which contains edges, sharpness of the image will increase and image will become clearer.

Sharpening is opposite to the blurring. In blurring, we reduce the edge content and in Sharpening, we increase the edge content. So in order to increase the edge content in an image, we have to find edges first.

Common operators for edge detection followed by sharpening are **Laplacian operator and Sobel operator**

$$\nabla^2 = \nabla \cdot \nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

Fig. 11. Laplacian- Second Derivative

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}$$

Fig. 12. Laplacian masks

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Vertical **Horizontal**

Fig. 13. Sobel operators

D. Experiments and Result

To show proper effects of different operations, different images are chosen accordingly.

– **Image load function**
This function will load image from system files as shown below.

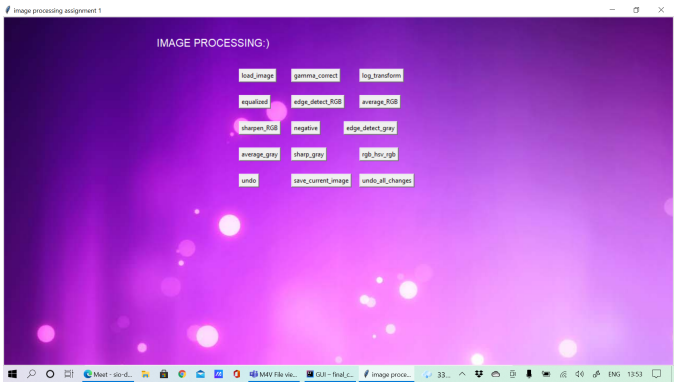

Fig. 14. Image load area

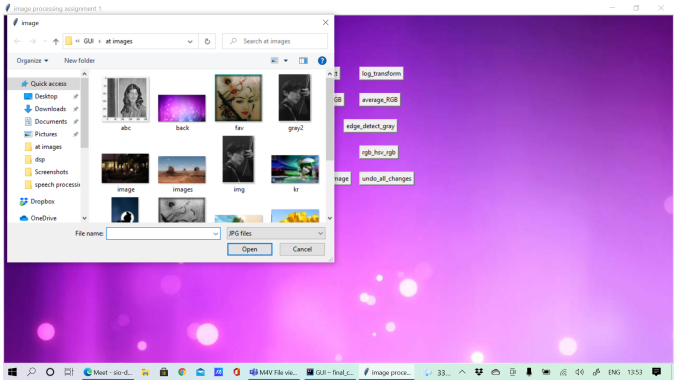

Fig. 15. Window after clicking on load image button

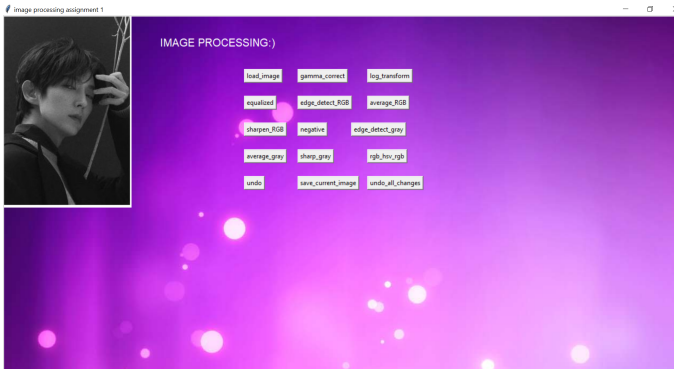

Fig. 16. Selected image appearing in canvas

– **Gamma correction function**
This function generates image for gamma=2.2 and gamma=(1/2.2).

Fig. 17. 1:Gamma=2.2, 2:original loaded image,3:Gamma=(1/2.2)

– **Function for log transform of an image**

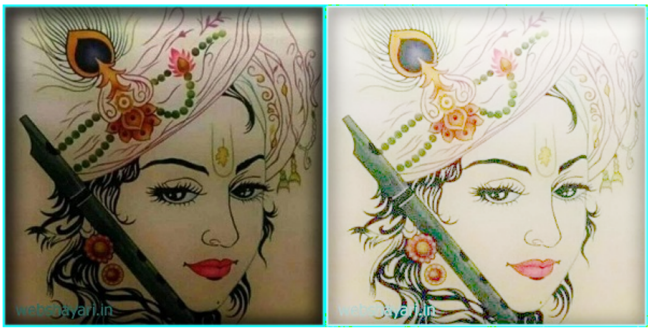

Fig. 18. Original and log transformed image

– **Function for equalized image**

Fig. 19. Original and equalized image

– **Function for smoothing RGB images**
The smoothing operation is performed on different channels of RGB image seperately and the all channels are combined to get smooth RGB back.
Range of smoothing can also be given as input by user as shown below in figure 20.
Edge detection and sharpening is also performed on RGB image but unable to remove excessive noise

content.So sperately performed for single channel images again and that is giving proper results.

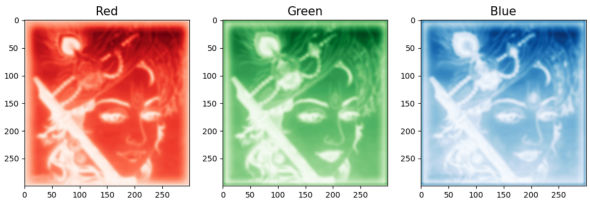

Fig. 20. All the channels after smoothing operation

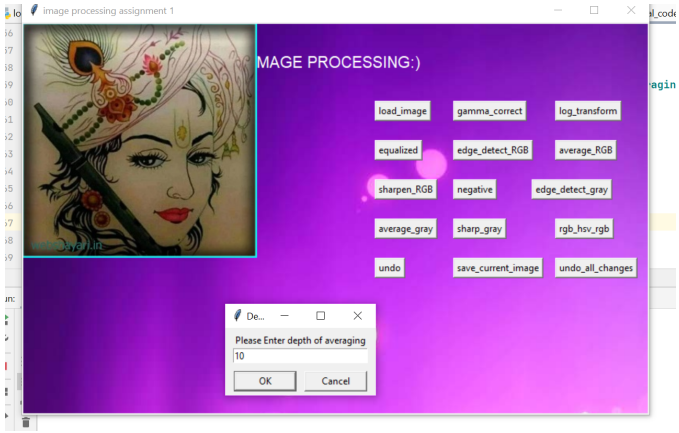

Fig. 21. Window for user to input range of blurring

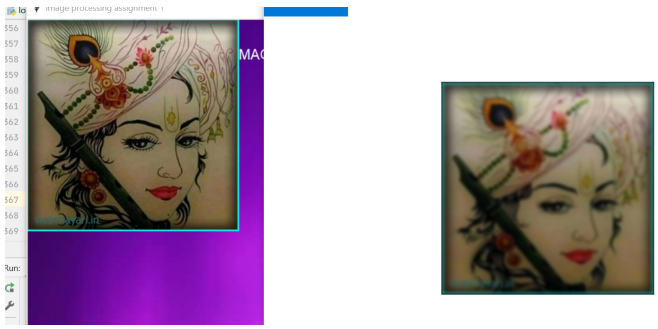

Fig. 22. Original and blurred image

– Function for negative of image

Fig. 23. Original and negative image

– Function for smoothing single channel images

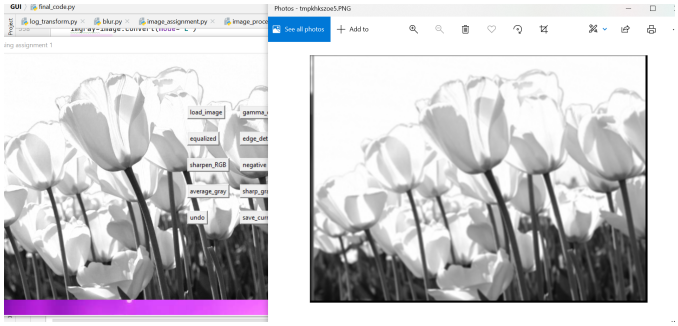

Fig. 24. Original and smooth image

– Function for sharpening single channel images

Fig. 25. Original and sharpened image

– Function for edge detection of single channel images

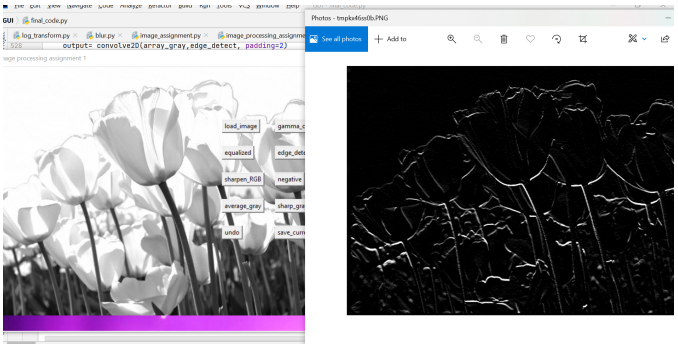

Fig. 26. Original image and edge detected image

- **Other functions**
  Undo: This function reverts the current effect on an image and displays the previous one.
  Save current image: As the name suggests it saves the current displayed image.
  Undo all changes: This function reverts all changes and gives back the original image back.

E. Conclusion and Discussion

- It was a great experience working on this assignment, as it involved deeper learning of theoretical concepts and then applying them to images made the concepts very clear.

- GUI integration was a really great experience. It was exhilarating to work on it and it also enhanced the skills which we can apply in future projects also. If given more time, I would have tried to explore and implement more features.

- For me the challenging part was image filtering operations. It took me a lot of time to get the correct result.First I used gray scale images and the result was fine for that, but for colour image sharpening operation, I was not able to remove the high frequency noise till the end.

- Another challenge was using V channel for performing operations on it and then incorporating it to finally get the transformed image, I was able to do it with individual codes but incorporating it with GUI image loading feature was difficult for me.If more time was given then I would have tried that, although I will still try to come up with the solution of this problem.

REFERENCES

- Digital image processing by Gonzalez, Rafael C and Woods, Richard E and others,year=2002,publisher=Prentice hall Upper Saddle River, NJ.

- GeeksforGeeks,2019, August 2,Python — Intensity Transformation Operations on Images. https://www.geeksforgeeks.org/python-intensity-transformation-operations-on-images/

- Edeza, T. (2021, January 2). Image Processing with Python — Blurring and Sharpening for Beginners. Medium. https://towardsdatascience.com/image-processing-with-python-blurring-and-sharpening-for-beginners-3bcebec0583a

- Blurring images – Image Processing with Python. (n.d.). Data Carpentry - Image Processing with Python. https://datacarpentry.org/image-processing/06-blurring/