EE 610 Image Processing Assignment 1: Basic Image Editor

1

Harsh Diwakar
Department of EE
IIT Bombay
Mumbai, India
213070018@iitb.ac.in

***Abstract*—In this assignment, a Basic Image editor is implemented to perform several operations on images. In today's world we have a lot of images because of the embedded cameras in mobile phones, however there are also other sources of images. After the image is captured sometimes it requires some enhancements for human eyes to see the details or just to beautify the image for human view. This basic image editor provides features such as: Histogram Equalization, Gamma Transform, Log Transform, Local Histogram Equalization, Edge Detection, Sharpening and Blurring. All of these operations are implemented in GUI using the Tkinter package in Python. This report will explain everything related to the assignment including a few results of before and after a feature is implemented to image.**

I. Introduction

"A picture is worth a thousand words", which means that complex ideas can be conveyed by a single image, which might not be as convenient to convey the same idea using some written text or verbal description that makes images a very important source of capturing and transferring information. and ideas. Sometimes the Image might contain the information but it is not visible to the human eyes or not pleasant for the human eyes. This is where Image editors come into picture that makes these tasks very easy.

In this assignment an Image editor is designed for applying complex transformations on images by providing the user a very simple GUI implementation. There are multiple transform operations provided for the image that are: Histogram Equalization[1], Gamma Correction[1], Local Histogram Equalization[1], Edge Detection[1], Log Transform[1], Sharpening of image[1] and Blurring[1] the image. All of the mathematical, conceptual and implementation details of the image editor is explained in detail in this report.

II. GUI Design

The GUI (graphical user interface) is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicators such as primary notation, instead of text-based user interfaces, typed command labels or text navigation[2]. GUI provides users to implement complex operations without writing the codes for each operation. In this assignment GUI is implemented using the Tkinter[3] package of Python. Tkinter is one of the most commonly used package for implementation of GUI in python as it is a standard Python interface. Tkinter provides a lot of elements for implementation of user interface. Some that are used in this Assignments are listed below:

- Buttons: It performs operations when pressed.
- Labels: These are used to put information on the GUI screen
- Canvas: This is specifically used in this assignment to display the images on the GUI screen.
- Entry: This element of Tkinter is helpful in getting input from the user.

A very simple approach was chosen for designing the GUI. There are buttons for general operations: Load, Save, Undo and Reset. Then there are Buttons for Image Editing Operations: Equalize Histogram, Gamma Correction, Local Histogram, Edge Detection, Log Transform, Sharpen and Blur. Two Canvas are used for displaying the images on screen.

Features of GUI:

- Background colour is chosen to be Purple so that it is soothing to the eyes.
- Load button-. This button allows the user to choose an image of any image format by opening a file selector. After the image is selected it is then loaded into the GUI and displayed on the screen.
- Save- On clicking this button a filedialog will open and allows the user to choose the location in files for saving the modified image. And after the location is chosen the image is saved at that location.
- Undo button- undos the last operation performed on the image. For implementing this function we have used a stack for storing all the images and popping up the last image from the stack on clicking this button.
- Reset button- resets the image to the original image that was loaded from the file. This is again implemented using the same stack that is cleared and the original image is pushed into the stack.
- Exit Button- When clicked it will close the GUI program.
- Equalize Histogram button- This button converts the unequalized histogram image to an equalized histogram image.
- Gamma Correction button- On clicking this button, few more features will pop up on the screen:
    - A label asking the user for the value of gamma.
    - An Entry element where the user needs to enter the value of gamma.
    - OK Button that needs to be pressed after entering the value of gamma to perform the gamma transformation.Then the transformed image will be displayed.
    - After the transformation is applied to the image and it is displayed on the Canvas these features will disappear.

- Local Histogram button- Sometimes Histogram Equalization is unable to give all the required details from the image that is why this Local Histogram Equalization button is provided that equalizes the Histogram of image locally and displays the original image with some percentage of locally histogram equalized image added to it.
- Edge Detection button- Detects the edges from the image and displays in grayscale for better visualization, Log Transform- Applies log transformation on the image and displays the transformed image on screen.
- Sharpen button- This button is for sharpening the image using different operators. Like Gamma Correction button, few more features will pop up:
    - A label asking the user for a method for sharpening.
    - Laplacian 1st order button that should be clicked for sharpening using laplacian 1st order operator.
    - Laplacian 2nd order button that should be clicked for sharpening using laplacian 2nd order operator.
    - Subnet Mask button that should be clicked for sharpening using subnet mask method.
    - Sobel Operator button should be clicked for sharpening using Sobel Operator.
    - After a method of sharpening is chosen, all the above features will disappear and new set of features will appear:
    - A label asking the user for Extent of sharpening.
    - 10 buttons numbered from 1 to 10 for choosing the extent of sharpening.
    - After one button is clicked and extent of sharpening is chosen again the features will disappear and sharpened image will be displayed
- Blur button- This button is designed for blurring the image. Here also some more features will pop up on the screen:
    - A label asking user for type of filter to be applied for blurring operation
    - Button Box Filter when pressed uses Box Filter to blur the image.
    - Gaussian Filter button when pressed uses a Gaussian filter for blurring the image.
    - After a filter is chosen by pressing the above two buttons again these buttons and labels disappear and a new label appears asking the user for the extent of blurring.
    - 10 buttons numbered from 1 to 10 appear on the screen for choosing the extent of blurring.
    - After the extent is chosen again the buttons and label will disappear.
- Two Canvas features are used in the GUI for displaying the image.
    - One main canvas where the modified image will be displayed.
    - One small canvas that can be found on the left side of the GUI that will always show the original image.

Main features of the overall GUI implementation have been explained above. Tkinter provides a very rich resource of libraries for doing the GUI operations. For loading the image into the GUI and saving the image into a desired location filedialog module of Tkinter is used. For exiting from the GUI destroy() method of Tkinter is used, similarly for making some features disappear (as in Gamma Correction Button, Sharpen Button and Blur Button) from the screen after() and destroy() method of Tkinter is used.

III. Image Processing Operations

Various image editing operations that has been implemented in this GUI are:

A. Log Transform
B. Gamma Correction
C. Histogram Equalization
D. Local Histogram Equalization
E. Blur
F. Edge Detection
G. Sharpen

A. Log Transformation

Log Transform comes under the umbrella of Basic intensity transformation. In intensity transform the values of pixels are processed and transformed into new values by some method. Values of pixels before applying the transform are denoted as r and after the transformation is applied the value of pixels becomes s. Then the transform T is defined as:

$$s = T(r) \quad (1)$$

Coming back to Log transform, T is changed to a log function then the eq. (1) becomes:

$$s = c\,log(r + 1) \quad (2)$$

Where c is a constant, it is assumed that $r \geq 0$ and 1 is added inside the log so that we can avoid 0 inside the log function that is not defined. When we apply log transform to an image then the dark pixels of the image are spreaded as compared to light pixels. This gives an effect of making the details hidden in dark regions to be more visible that can be seen from plot below:

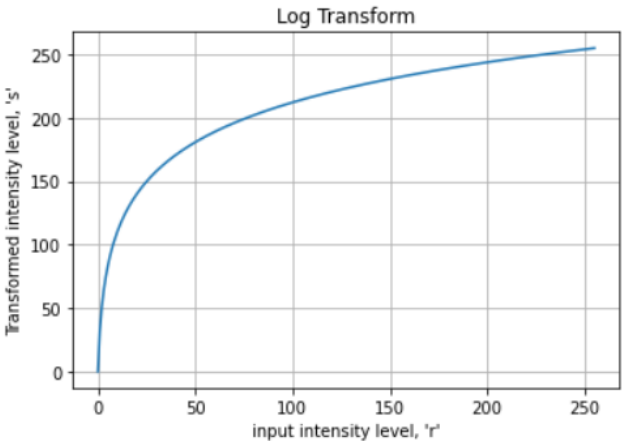


Figure 1. A plot showing the Log transformation

As it is clearly visible from the above plot that the intensities of low values are getting stretched to a higher

range in transform 's'. Log Transform is used to show the details hidden in the dark region of the image (Illustration for Log Transform is shown in section V.)

B. Gamma Correction

Gamma Correction is also an intensity transform that works by changing the intensities of the pixels of image by the transformation law:

$$s = cr^{\gamma} \tag{3}$$

Where c and γare positive constants. Transform depends on the value of γ, when the value of γ> 1 then the low valued pixels will be stretched to a higher range in transform 's' like the Log transform. Exactly Opposite effect can be observed when γ< 1 that is the high valued pixels are now stretched to a higher range.

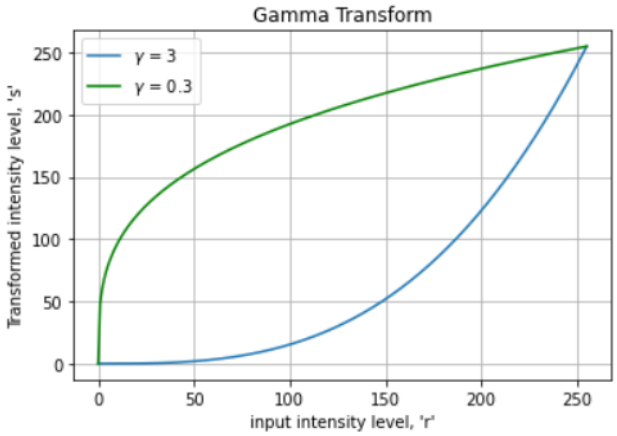


Figure 2. A Plot showing the Gamma Transform with $\gamma = 3$and $\gamma = 0.3$

As it is visible from above plot, that $(\gamma = 3) > 1$gives increased range of low valued pixels and $(\gamma = 0.3) < 1$ gives an increased range of high valued pixels. Gamma Transform is used to show the details hidden in the dark or light region of image depending upon γ.

C. Histogram Equalization

Before this we have seen Log ang Gamma transform that increased the range of a particular region (dark or bright) of the image. Sometimes it happens that the values of pixels are spread in a very narrow range of intensity values. These images can be enhanced using Histogram Equalization. Let us take the Histogram of an image (Figure 15(a)).

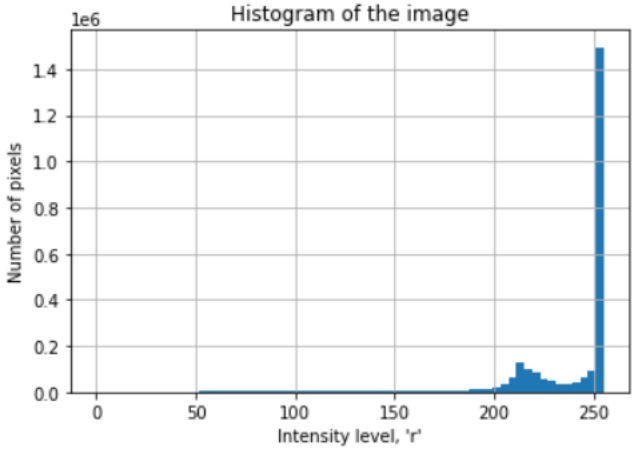


Figure 3. Original Histogram of an image (Image 15(a))

It is visible from the above histogram that intensities levels from 0 to 200 are not properly utilized and all the pixel's intensity value is converged at the higher values. Histogram equalization uses cumulative probability distribution (CDF) function to equalize the histogram of an image. CDF for a discrete PDF (probability distribution function) is defined as a cumulative sum of all the probabilities:

$$CDF(r) = \sum_{j=-\infty}^{\infty} P(j) \tag{4}$$

For finding the P(j) of an image we will normalize the histogram distribution by dividing the number of pixels at each intensity level by the size of the image i.e. if the size of image is M✕N pixels then each pixel count will be divided by M✕N. After a probability function is found then the transformed is applying by using the equation:

$$s = (L - 1)\, CDF(r) \tag{5}$$

After applying the transform, HIstogram of the image looks like:

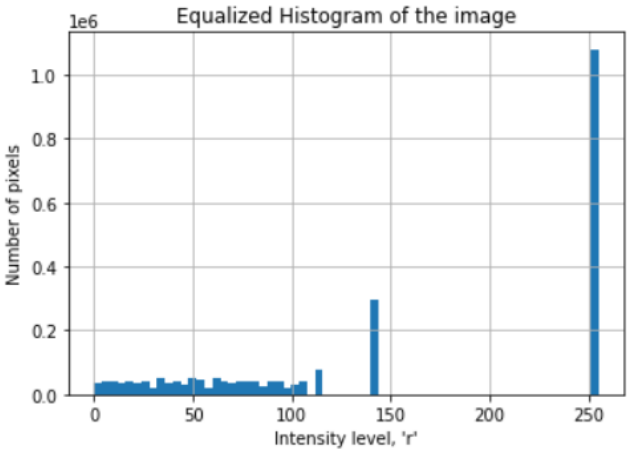


Figure 4. Histogram of the Image after applying Histogram Equalization

As it is observed from the above histogram that intensity levels are not well utilized and the output image after applying the transform is shown in Figure 15(b).

D. Local Histogram Equalization

Sometimes Histogram Equalization alone is not able to show all the hidden details in the image. For these kinds of images Local Histogram Equalization may be used. The main idea behind local histogram equalization is that we apply histogram equalization on small portions of image and combine it with some percentage of original image to show the details. We add original image also for displaying as the local Histogram may increase the noise also.

E. Blur

For Blurring an image we use the concept of Spatial Filtering. In spatial Filtering we take a filter of size J✕K and convolve it with the Image. This must be noted that J and K must be odd and let us assume they are of the form 2m+1 and 2n+1 respectively. Convolution operation for finding one pixel of output image (g(x,y)) is given as:

$$g(x,y) = \sum_{s=-m}^{m} \sum_{t=-n}^{n} w(s,t) f(x+s, y+t) \quad (6)$$

Where, w(s,t) is the value of $(s,t)^{th}$ pixel of filter, f(x,y) is the value at $(x,y)^{th}$ pixel of input image. w visits every pixel in f by varying the value of x and y. This can also be interpreted as taking neighbors of the input image for calculating the output and when m and n are equal to zero, this converges to intensity transform.

For Blurring or smoothing an image we convolve the image with the gaussian filter or a box filter. A box filter is a filter with all equal elements and the elements are equal to 1/JK where J✕K is the size of filter. For a gaussian filter the value of element at position j and k is defined by the distance of j and k from the middle element of the filter. Let the x- distance be x and y- distance be y then the value for element at that location filter(x,y) will be:

$$filter(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (7)$$

Where σ^2 is the variance of gaussian that will increase with increase of filter size. A 3✕3 box filter and approximation of Gaussian filter is shown below:

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

(a)

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

(b)

Figure 5. (a) Box Filter. (b) Gaussian Filter

Convolving the image using the above filters gives a smoothened image. It can also be interpreted as a low pass filter and used to remove high frequency components from images. We can also control the extent of blurring or smoothing by changing the size of filters.

F. Edge Detection

Edge detection as the name suggests detects the edges present in an image. This operation can be performed using the Sobel Operators. We convolve the image with sobel operator in the horizontal axis and with the sobel operator in the vertical direction. Convolving the image with horizontal or vertical filter gives edges in horizontal and vertical direction respectively. Sobel Operators for horizontal and vertical direction is shown below:

-1	-2	-1
0	0	0
1	2	1

(a)

-1	0	1
-2	0	2
-1	0	1

(b)

Figure 6. (a) Sobel Operator in horizontal and (b) vertical direction.

After we have detected the edges in both the direction we now use the following equation for finding the overall edges:

$$g(x,y) = \sqrt{g(x)^2 + g(y)^2} \quad (8)$$

Where g(x) and g(y) are the result of convolution operation with sobel horizontal and vertical operators.

G. Sharpen

For Sharpening the image 4 methods are used here. However, the basic idea is same in all the methods i.e. we first find the edges or high frequency components and then add a small percentage of the result to the original image for enhancing the image. It is used to highlight the edges in images. We find edges using Sobel Operator (explained in the above section.), Laplacian first order operator, Laplacian Second order Filter or Subnet mask. Discrete approximations of 3✕3 Laplacian operators are shown below:

0	1	0
1	-4	1
0	1	0

(a)

1	1	1
1	-8	1
1	1	1

(b)

Figure 7. Discrete approximation of (a) Laplacian First order operator, (b) Laplacian Second Order Operator for a 3✕3filter.

Steps for sharpening using subnet mask:

1. Blur the image using box or Gaussian filter
2. Subtract blurred image from original image.
3. Add the result of the above operation to the original image.

For controlling the extent of sharpening, percentage of masks to be added is varied.

IV. Experiments and Results

In this section results are displayed. Screenshots of GUI are displayed for showing the overall GUI implementation. Different image editing operations are performed on several different images to show the effects of operations.

A. GUI Screenshots

Overall GUI screenshot:

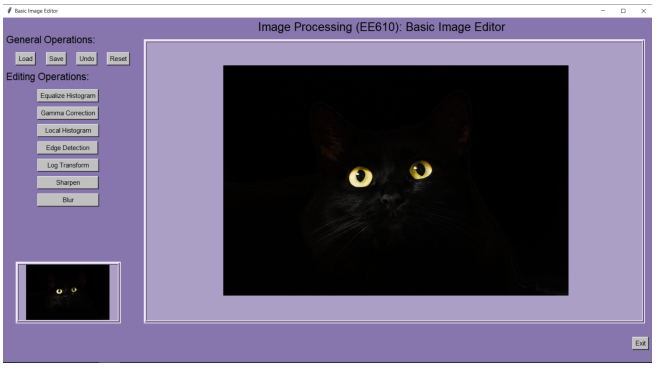
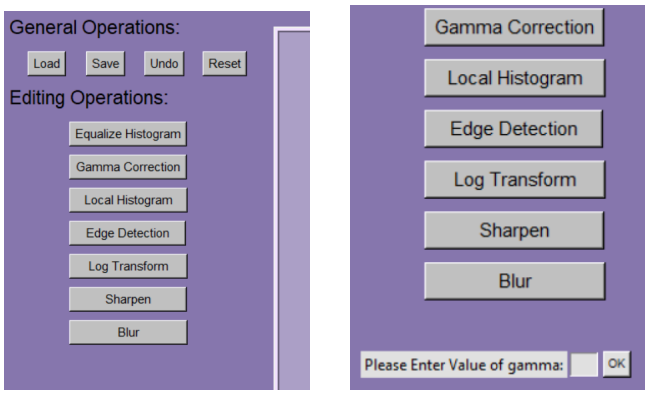


Figure 8. Screenshot of the GUI with an image.

As it can be seen from the above screenshot, there is a main canvas for displaying the image after operation and a small canvas on the left side for always displaying the original loaded image. Image used in the above screenshot of GUI is downloaded from PixelBay[4]. and is free for commercial

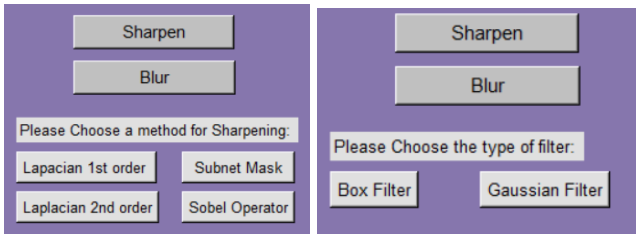
use. Also the operational buttons are not easily visible in this screenshot for that more snippets are shown.



(a) (b)

Figure 9. Snippets of GUI. (a) close view of operational buttons, (b) View after the Gamma Correction button is clicked.

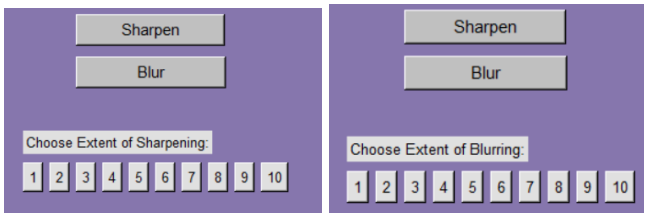
Operational buttons are clearly visible in Figure 9(a) and the features that pop up after pressing the Gamma Correction are shown in Figure 9(b).



(a) (b)

Figure 10. Snippets of GUI after the (a) Sharpen and (b) Blur Buttons are clicked.

In the above Figure, features that pop up after clicking the Sharpen button (Figure 10(a)) and the Blur Button (Figure 10(b)) are clearly visible. The label asks the user to choose a method for sharpening or the type of filter for smoothing.



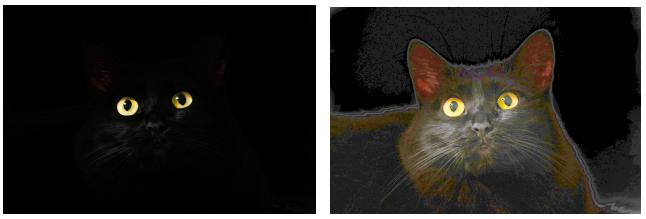
(a) (b)

Figure 11. Snippets of GUI after the (a) method for sharpening and (b) type of filter for smoothing is clicked.

In the above images features after choosing the method of sharpening or filter for smoothing is shown. After the filter or method is chosen user is asked for the extent of blurring or sharpening and allowed to choose from the range of 1 to 10 that is available by the use of buttons.

B. Log Transform

For showing the effect of log transform an image of a cat is chosen. In Figure 12(a) the cat is not properly visible because of the dark background. We can only see the eyes of the cat but after applying the Log transformation Figure 12(b) we can see more details of the cat. That is Log transformed is showing the details hidden in the dark regions

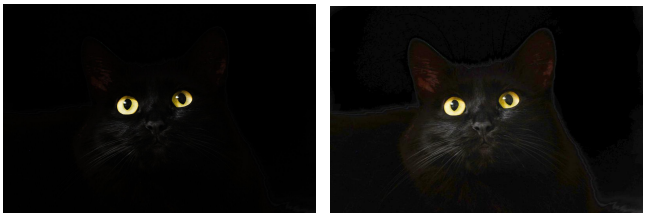


(a) (b)

Figure 12. (a) image of cat before and (b) after applying Log Transform.

C. Gamma Correction

Different values of gamma will give different results for an image. Same cat image is chosen to show the effect for gamma < 1.



(a) (b)

Figure 13. (a) image of cat before and (b) after applying Gamma Correction with $\gamma = 0.6$.

Similar effects as Log Transform is observed here (Figure 13). However, in gamma correction we have control over how much of details in the dark region we want to show be varying γ. For displaying the effect of $\gamma > 1$an image with light background is chosen.

(a)

(b)

Figure 14 (a) Image before applying gamma correction, (b) after applying gamma correction with $\gamma = 6$.

As it is clearly visible from the Figure 14 that gamma correction with $\gamma > 1$makes the details hidden in bright regions more visible. Trees are more clearly visible in the Figure 14(b) after applying gamma correction with $\gamma = 6$.

D. Histogram Equalization

An image of a woman is chosen from PixelBay[4] (Figure 15(a)). In the image most of the pixels are towards higher values of intensities that is shown in the Figure 3.

(a)

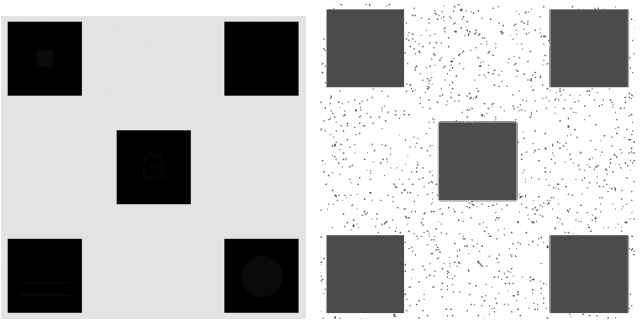
(b)

Figure 15. (a) Image before applying Histogram Equalization and (b) after applying Histogram Equalization.

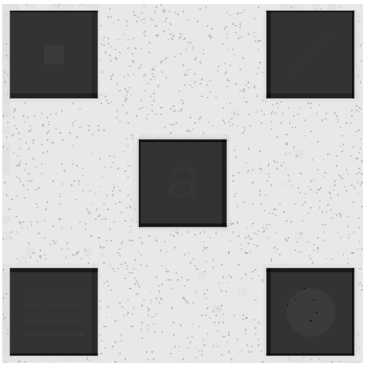
After Histogram equalization is applied to the image, details of the woman's face are visible more clearly and Histogram of transformed image can be seen in Figure (4).

E. Local Histogram Equalization

For displaying the effect of Local Histogram Equalization an image from ImageProcessingPlace[5] is downloaded. Images from this website can be used for personal education or research purposes. It will be shown that how Local Histogram Equalization can perform better than Histogram Equalization. In the Figure 16(a) original image with hidden features is shown, and similarly in Figure 16(b) and 16(c) Histogram Equalized Image and Local Histogram Image is shown respectively.



(a) (b)



(c)

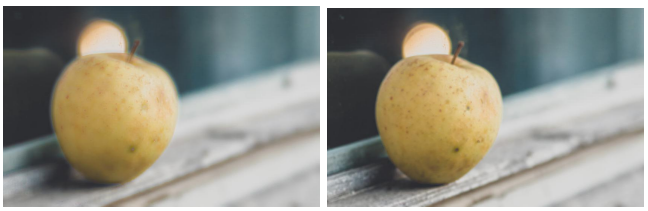
Figure 16. (a) Original Image. (b) Histogram Equalized Image (c) Local Histogram Equalized Image.

As observed from Figure 16, Histogram Equalization is just adding noise to the original image and is not able to get the details hidden inside the black boxes. However, in the case of Local Histogram Equalization we can see some details inside the black boxes like a small letter 'a', a circle, a box etc.

F. Blur

To display the blur operation we will choose an image having sharp details and make it blur.

(a) (b)



(c) (d)

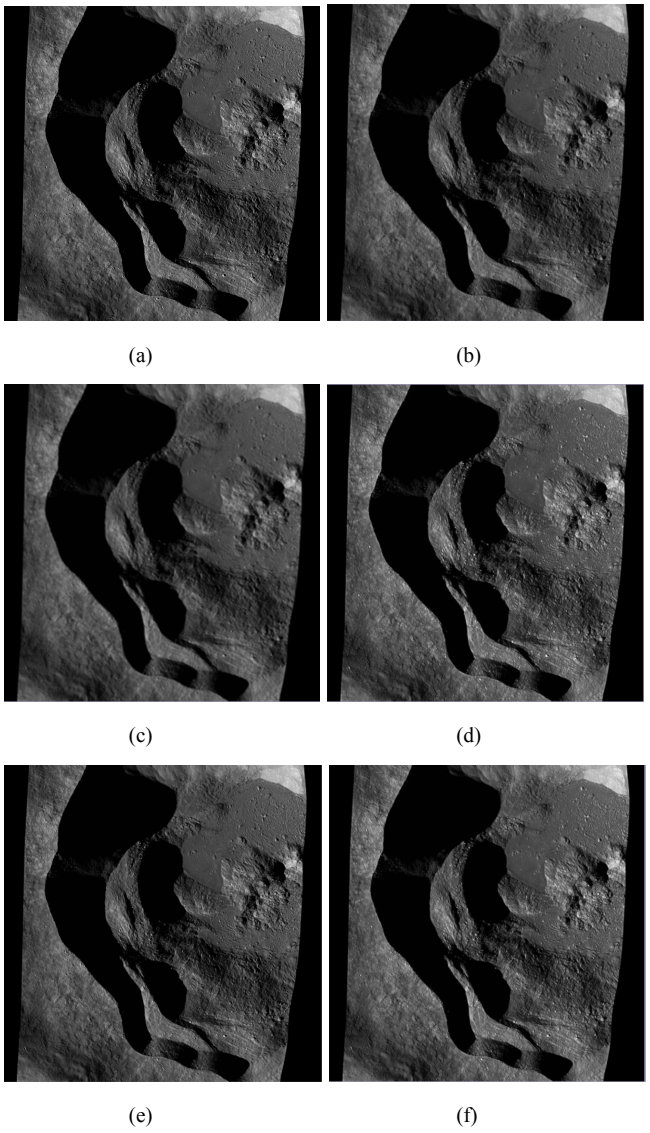
Figure 17. (a) Original Image (b) Blurred using Gaussian Filter with an extent of 9. (c) Blurred using Box Filter with an extent of 9. (d) Blurred using Box filter with an extent of 3.

Figure 17 shows some results obtained by using different filters and extent of filtering. It can be observed from the above results that the image is becoming more blurred as the

extent of blurring is increased. Also a slight difference can be seen in between smoothing using Box filter and Gaussian Filter.

G. Sharpen

For displaying we will use an image downloaded from NASA[6]. Edges detected from the Edge detection method is now added in the original image to sharpen it.



(a) (b)

(c) (d)

(e) (f)

Figure 19. (a) Original Image, Sharpening with (b Laplacian First order Operator and to an extent of 9 (c) Laplacian Second order Operator and to an extent of 9 (d) Sobel Operator and to an extent of 9 (e) Subnet Mask and to an extent of 9 (f) Sobel Operator to an extent of 2.

We can observe from Figure 19, a small enhancement at the edges when we apply sharpening using different techniques. However, in this image we can not see a lot of difference between the different methods used for sharpening but for several images some methods are better than another.

H. Edge Detection

Same image used in the blur operation is used to display the Edge detection operation.

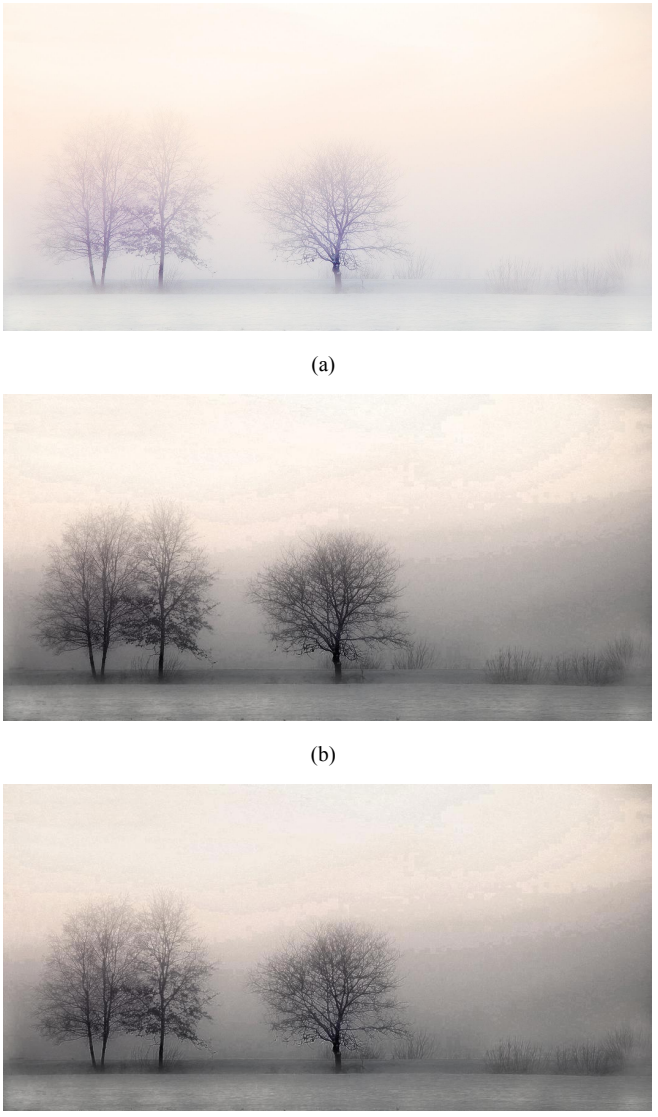
(a) (b)

Figure 18. (a) Original Image (b) Detected edge of Original Image

From Figure 18. it can easily be observed that edges are very nicely detected from the original image. Only the sobel operator is used to display the edge detection. However, other methods such as Laplacian operators may also be used for edge detection.

I. Multiple operations

We will use the image of the tree for showing the enhancement of an image using multiple operations.



(a)

(b)

(c)

Figure 20. (a) Original Image (b) Image after Gamma Correction with $\gamma = 5$.(c) Gamma corrected image after sharpening using Sobel Operator.

Original Image is first undergone through Gamma correction for making the trees more clearly visible. After this image was sharpened using the sobel operator to an

extent of 9. We can see the finally enhanced image with clearly visible trees and boundaries.

V. Conclusion and Discussion

Basic Image Editor with the basic operations such as Histogram Equalization, Gamma Correction, Log Transform, Local Histogram Equalization, Smoothing, Edge Detection and Sharpening was implemented in Python using Tkinter Package. All the mathematical, conceptual and implementation aspects were presented in the report. Main challenges faced in implementation of the image editor were in vectorised implementation of Convolution, Controlling the Extent of Blur or sharpening operations and challenge in padding.

A. Vectorised Implementation of Convolution

One of the major challenges in implementing this image editor was implementing the convolution using vectorisation. This was important because convolution implemented using loops is very slow in Python in comparison with vectorised implementation. This is because vectorised implementation efficiently uses computational resources to decrease the time taken for execution. This challenge was overcome by the use of few numpy[7] features. List of numpy features required to implement the vectorised implementation:

1. Numpy.repeat()- Repeats elements of array.

```
# import numpy
import numpy as np
# define an array arr
arr = np.arange(0,3)
# Print result of np.repeat
print(np.repeat(arr,3))
```

```
[0 0 0 1 1 1 2 2 2]
```

Figure 21. Code snippet to show np.repeat()

2. Numpy.tile()- Construct an array by repeating the array number of times given by repetitions.

```
# import numpy
import numpy as np
# define an array arr
arr = np.arange(0,3)
# Print result of np.tile
print(np.tile(arr,3))
```

```
[0 1 2 0 1 2 0 1 2]
```

Figure 22. Code snippet to show np.tile()

3. array[i]- Returns an numpy array with the values of array at locations i, where i is also a numpy array.

```
# import numpy
import numpy as np
# define i
i = np.array([0,0,0,1,1])
# define an array arr
arr = np.array([10,11,12,13])
# Print result of np.tile
print('arr = ',arr,' arr[i]= ',arr[i])
```

```
arr =  [10 11 12 13]  arr[i]=  [10 10 10 11 11]
```

Figure 23. Figure to illustrate numpy feature

4. Broadcasting Feature- if we add a scalar to an array it broadcasts the scalar and adds it to all the values. Same properties is used in low row or column vector.

```
# import numpy
import numpy as np
# define i
i = np.array([1,2])
# define an array arr
print(i.reshape(-1,1)+i)
```

```
[[2 3]
 [3 4]]
```

Figure 24. Broadcasting feature of Numpy.

Now, Let's look at the conceptual detail for vectorised implementation. Let us assume we have an image with following indexes and a filter of size 3✕3:

00	01	02	03	04	05
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

Figure 25. Image with first window of convolution.

Now, for finding the next element of output window will slide towards right as:

00	01	02	03	04	05
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

Figure 26. Image with second window of convolution.

For using the vectorised implementation we will use np.dot feature of numpy that returns the dot product of two arrays.

However the way our images and filters are at present we can not perform dot products directly. So for that we will flatten the filter i.e. filter of size 3✕3 will be reshaped to 1✕9. Still our problem with image is not solved. But if we can get all the sliding windows in a column vector of 9✕1 then we can easily perform dot product and use vectorised implementation. Let us see how will image look after we represent it in columns of 9✕1.

00	01	02	03	10	11	12	13	20	21	22	23	30	31	32	33
01	02	03	04	11	12	13	14	21	22	23	24	31	32	33	34
02	03	04	05	12	13	14	15	22	23	24	25	32	33	34	35
10	11	12	13	20	21	22	23	30	31	32	33	40	41	42	43
11	12	13	14	21	22	23	24	31	32	33	34	41	42	43	44
12	13	14	15	22	23	24	25	32	33	34	35	42	43	44	45
20	21	22	23	30	31	32	33	40	41	42	43	50	51	52	53
21	22	23	24	31	32	33	34	41	42	43	44	51	52	53	54
22	23	24	25	32	33	34	35	42	43	44	45	52	53	54	55

Figure 26. Image after representing each window in columns.

If we observe from the above image properly there is a pattern in i and j that can easily be obtained. All we need to do is use the four numpy features explained before and we can get the above receptive window[8]. After the above receptive window is obtained we will now take the dot product of the flattened filter of 9✕1 with the calculated receptive window.

```
# 9. Function for Vectorised Convolution:
def convo(img, fil):
    '''
    Takes two numpy arrays as input: img and fil and returns the
    result after convolution
    '''
    # Filter Hight and Width are same and equal to the size of filter
    fil_hw = fil.shape[0]
    # Calculating and storing the shape of Convolution Result
    out_shape = np.array(img.shape) - np.array(fil.shape) + 1
    # Creating the i and j values to broadcast the original
    #image so that we can apply vectorised
    #Implementation of Convolution
    i0 = np.repeat(np.arange(fil_hw), fil_hw)
    j0 = np.tile(np.arange(fil_hw), fil_hw)
    i1 = np.repeat(np.arange(out_shape[0]), out_shape[1])
    j1 = np.tile(np.arange(out_shape[1]), out_shape[0])
    # Broadcasting i0, i1, j0, j1 to get i and j
    i = i0.reshape(-1, 1) + i1.reshape(1, -1)
    j = j0.reshape(-1, 1) + j1.reshape(1, -1)

    # Applying i and j to get broadcasted image
    image_rec = img[i, j]
    # Flatten the Filter
    fil_flat = fil.reshape(fil_hw * fil_hw, -1)
    # Apply dot product to get convoluted image
    convol_res = np.dot(fil_flat.T, image_rec)
    # Reshape the Output
    convol_res = convol_res.reshape(out_shape[0], out_shape[1])
    return convol_res
```

Figure 27. Code snippet for vectorised implementation of Convolution.

After obtaining the dot product we have calculated our output using vectorization. All that is left now is to reshape the output and get the final result of convolution.

B. Controlling extent of Blur or Sharpen

Tkinter provides a very nice slider for varying a variable. The name of the Tkinter widget is scale. However the problem I faced by using this widget was that it was not directly going from one value to the other value, let's say 1 to 5. It was taking all the intermediate steps in between like 1,2,3.. then it was reaching to 5. So, for overcoming this challenge, Automatically disappearing buttons were used that will disappear after their work is done. This also helped in saving the space in the GUI screen for other operations and the screen looked cleaner.

C. Challenge in padding operation

When I used only minimal padding required for getting the output shape equal to the input image shape then there was a problem. The problem was that the V component of the image in which I was applying the transform and putting it back into the original image without changing the H and S component. Then the V component was shifting one pixel below and to the left.

(a)

(b)

Figure 28. (a) Original Image (b) Distorted image after applying transforms

As it can be observed from the above example that the V component was shifting as convolution with minimal padding was applied. To overcome this problem I used extra padding and clipped the output so that it matches the input image shape.

Overall, it was so interesting to work on this image editor. I got to learn a lot of new things like GUI, image editing operations from scratch, vectorised implementation of Convolution and many more. Given more time I would like to use other transforms to enhance an image like Frequency transforms, Wavelet transforms. Also I would like to use Machine Learning models to enhance the image.

REFERENCES

[1] R. C. Gonzalez and R. E. Woods, 'Intensity Transformations and Spatial Filtering' in Digital Image Processing. Upper Saddle River, NJ, USA: Prentice-Hall, 2008.

[2] Technopedia.com, 'Graphical User Interface (GUI)', 2021. [Online] Available: https://www.techopedia.com/definition/5435/graphical-user-interface-gui. [Accessed: 28- Aug- 2021].

[3] Python Software Foundation, ‘tkinter - Python interface to Tcl/Tk’, Available: https://docs.python.org/3/library/tkinter.html. [Accessed: 28 - Aug - 2021].

[4] Pixelbay, ‘Stunning free images & royalty free stock’, Available: https://pixabay.com. [Accessed: 28 - Aug - 2021].

[5] ImageProcessingPlace, ‘DIP3/e—Book Images Downloads’, Available: http://www.imageprocessingplace.com/DIP-3E/dip3e_book_images_downloads.htm. [Accessed: 28 - Aug - 2021].

[6] NASA, ‘Earth’s Moon’ Available: https://moon.nasa.gov/. [Accessed: 28 - Aug - 2021].

[7] NumPy, ‘Array manipulation routines’, Available: https://numpy.org/doc/stable/reference/routines.array-manipulation.html. [Accessed: 28 - Aug - 2021].

[8] Medium, ‘Implementing Convolution without for loops in Numpy!!!’, Available: https://medium.com/analytics-vidhya/implementing-convolution-without-for-loops-in-numpy-ce111322a7cd. [Accessed: 28 - Aug - 2021].

[9] freeCodeCamp, ‘Learn How to Use Tkinter to Create GUIs in Python’ Available: https://www.freecodecamp.org/news/learn-how-to-use-tkinter-to-create-guis-in-python/. [Accessed: 28 - Aug - 2021].

[10] Pillow, ‘Concepts’, Available: https://pillow.readthedocs.io/en/stable/handbook/concepts.html. [Accessed: 28 - Aug - 2021].

[11] matplotlib, ‘Sample plots in Matplotlib’, Available: https://matplotlib.org/stable/tutorials/introductory/sample_plots.html#sphx-glr-tutorials-introductory-sample-plots-py. [Accessed: 28 - Aug - 2021].

[12] Python Software Foundation, ‘math — Mathematical functions’, Available: https://docs.python.org/3/library/math.html. [Accessed: 28 - Aug - 2021].