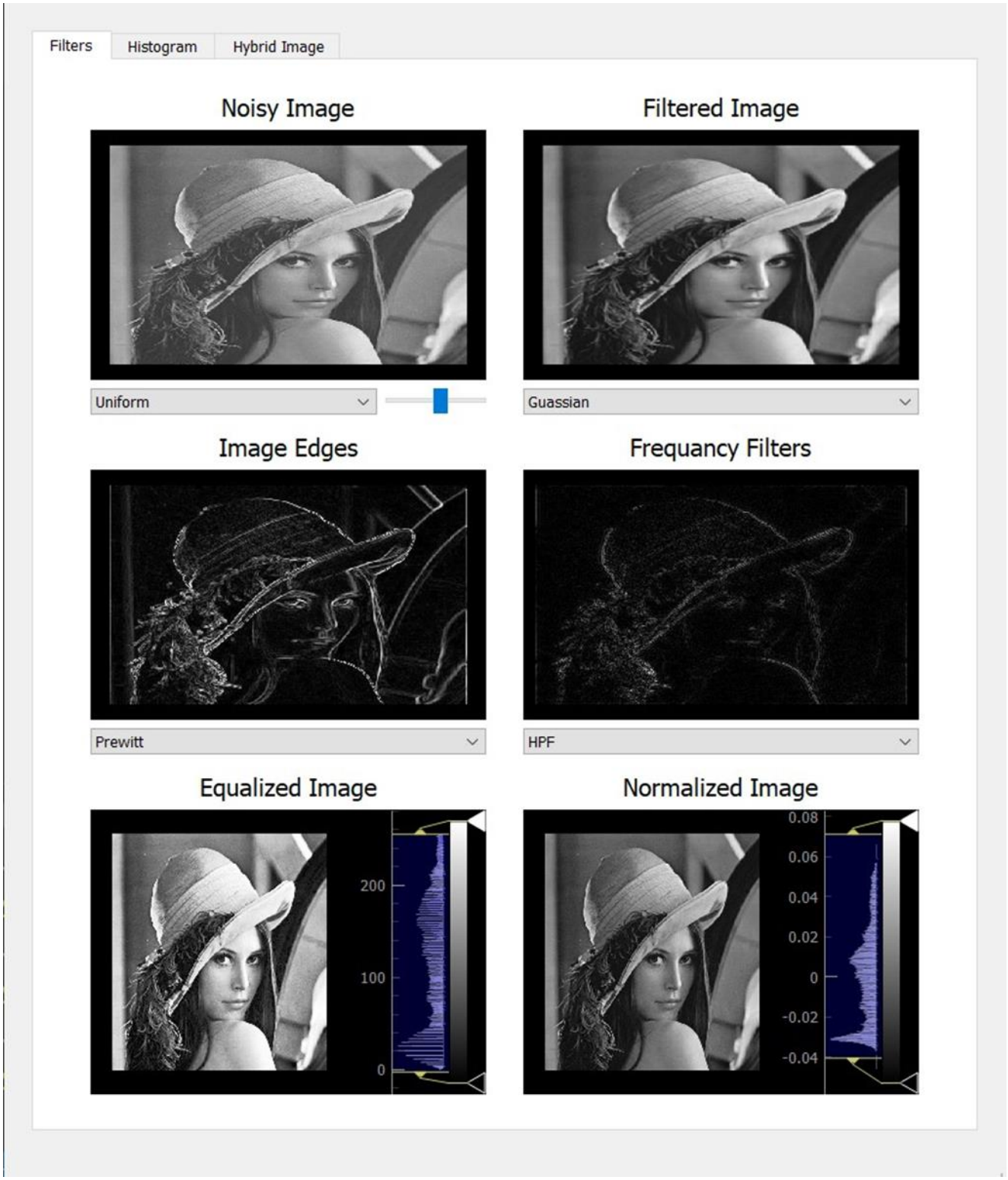
2023

Task1

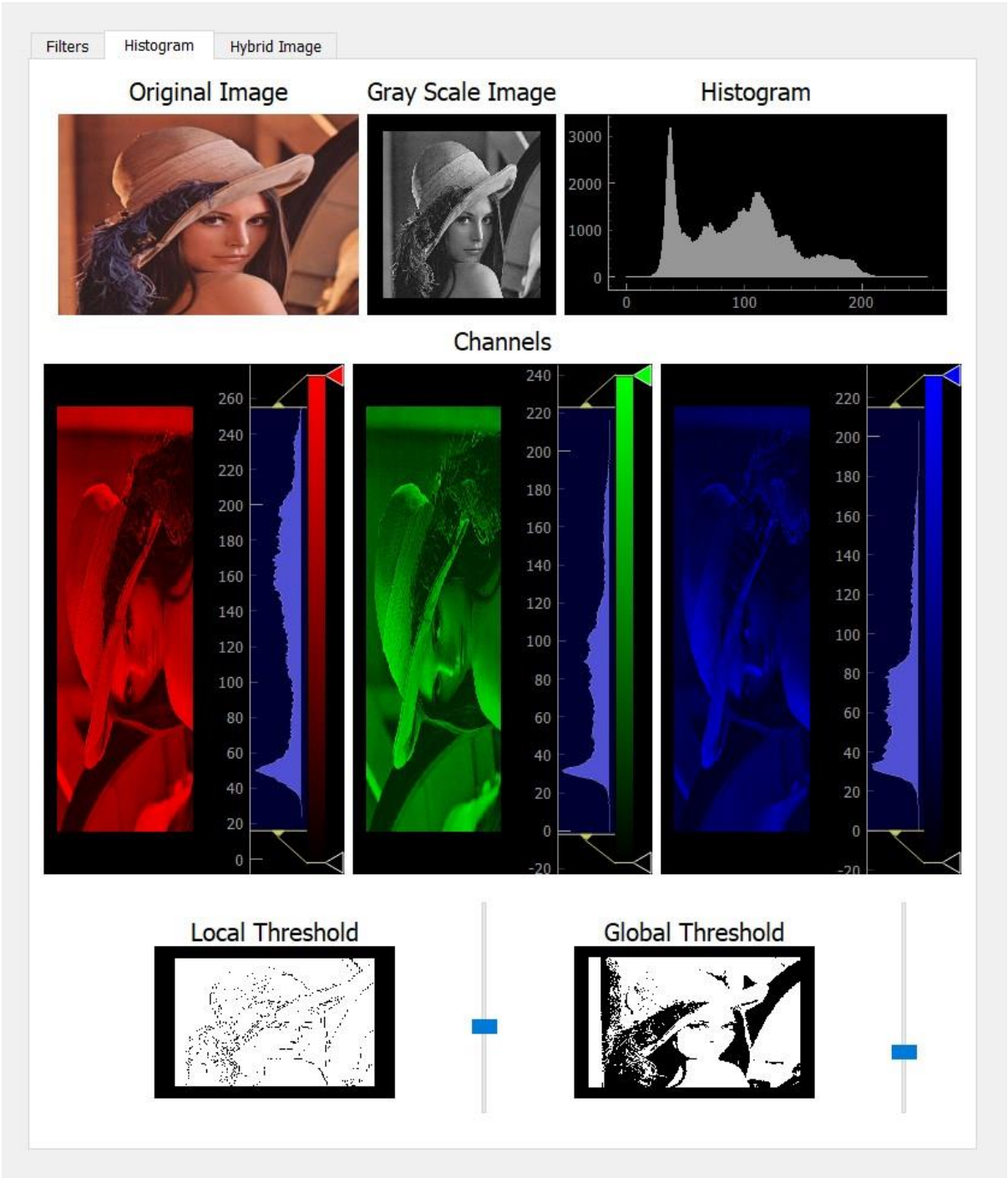
ENG:peter salah , ENG:Laila abbas

Name	SEC	BN
Bassma alaa eid	1	20
Abdelrhman ail farouq	1	55
Amr kamal fathi	2	6
Mahammad sayed zaki	2	17

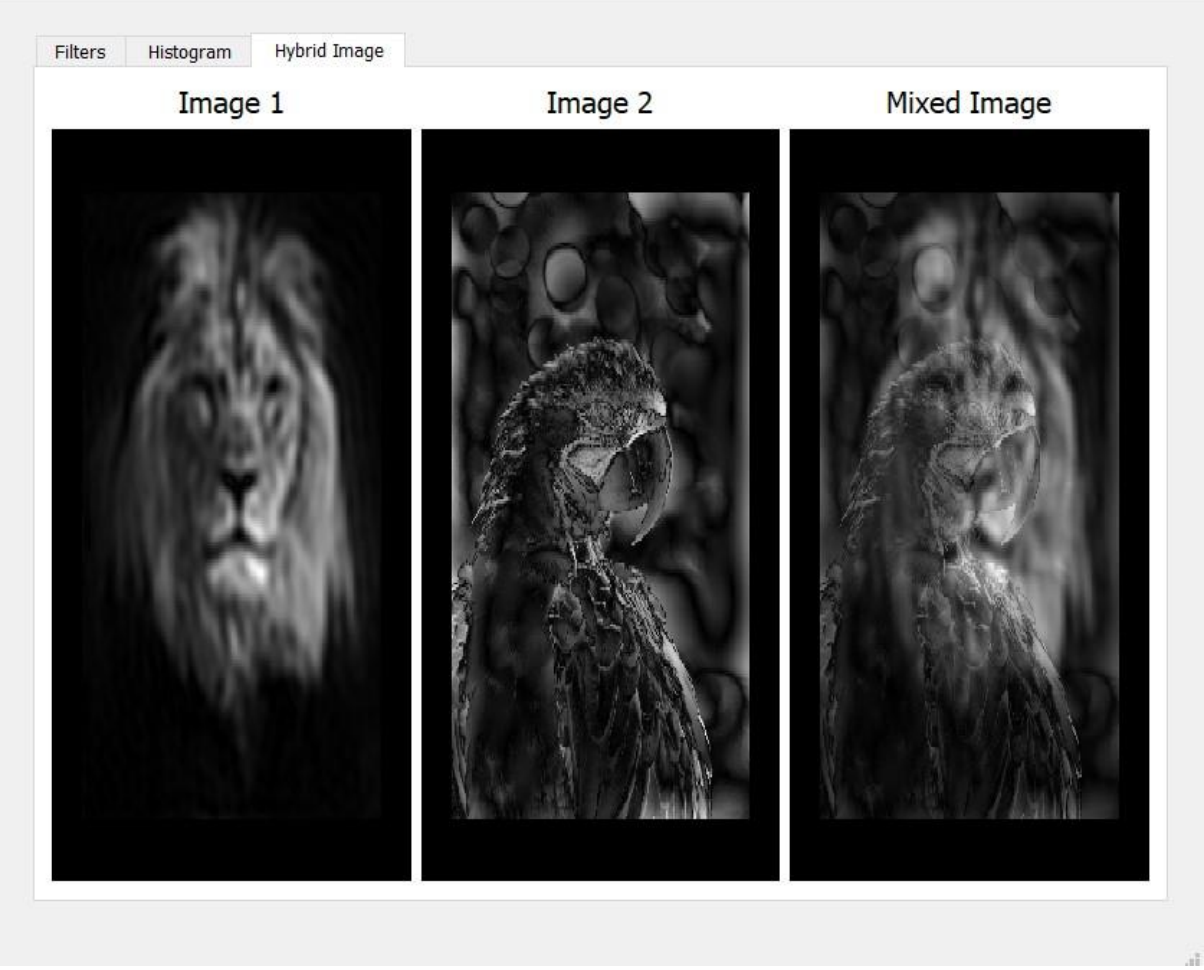
Tab 1:


Filters
Histogram
Hybrid Image
Noisy Image
Filtered Image
Uniform
Guassian
Image Edges
Frequancy Filters
Prewitt
HPF
Equalized Image
Normalized Image
200
100
0
0.08
0.06
0.04
0.02
0
-0.02
-0.04

tab 2:-


Filters
Histogram
Hybrid Image
Original Image
Gray Scale Image
Histogram
3000
2000
1000
0
0
100
200
Channels
Local Threshold
Global Threshold

Tab 3:-



Part 1:-

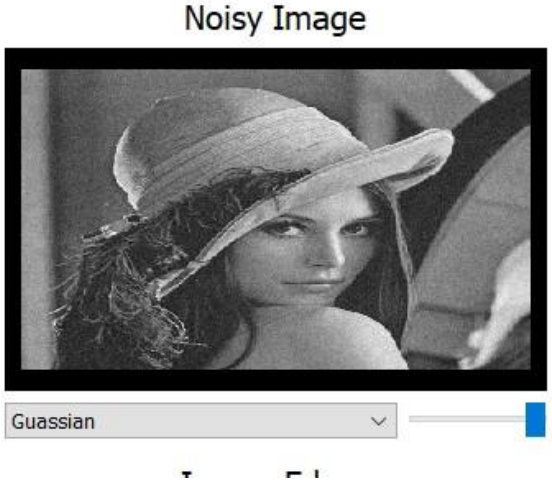
We add three different type of noise :

- Uniform
- Guassian
- Salt & Pepper

You can change between them using combo box and change the amount of noise using slider.

❖ Uniform

❖ Guassian



❖ Salt & Pepper

Part 2:-

Filter the noisy image using the following low pass filters:-

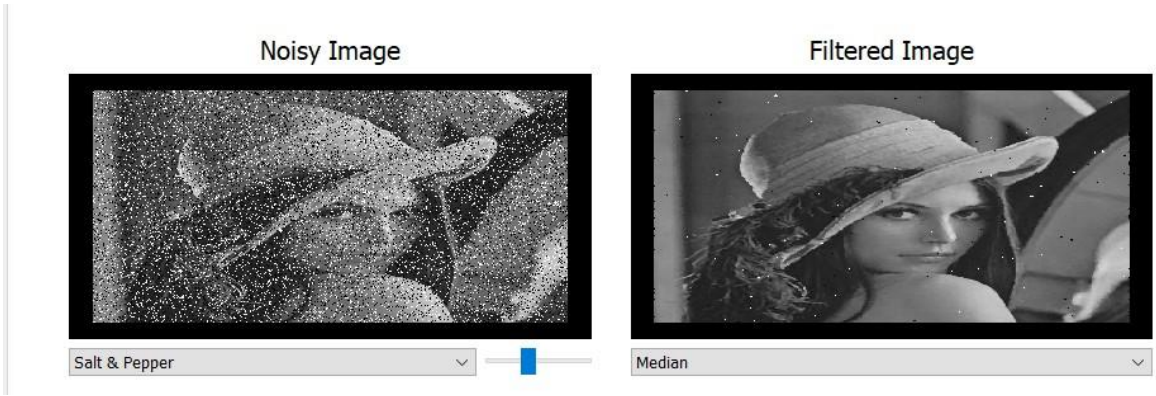
we use function "avgFilter" to manipulate the noisy image with a specific filter we choose and the result is:

- original image

- mean filter

- Guassian filter

- Median filter



And we found that median filter can handle the image salt and pepper noise better,

And guassian filter is better in the image uniform and guassian noise

Part 3:-

Sobel , Roberts , Prewitt and Canny edge detectors:-

we use function "edgFilters" to manipulate the image with a specific filter we choose and the result is:

- original image

- Sobel filter

- Roberts filter

- Prewitt filters

- canny edge detection

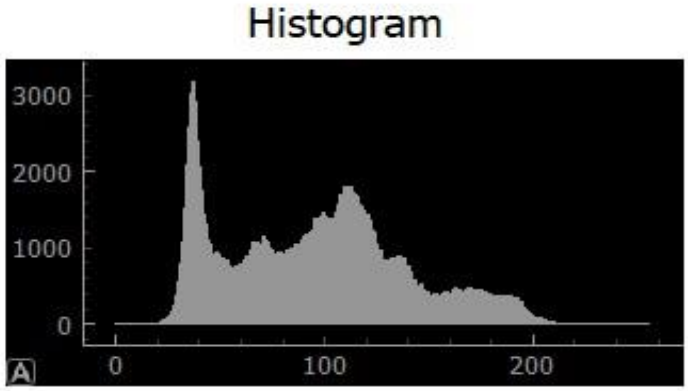
we found that canny edge detection algorithm produces smoother, thinner and cleaner images than sobel , prewitt and Roberts filters.

Part 4:-

Draw histogram:

In this part we've implemented a function called "df" that takes an image data array and return the histogram values for each intensity value.

Using that values to draw a "BarGraphItem" on pyqtgraph we got the following output:



Part 5:-

Equalize the image:

Using the previous histogram to generate a histogram equalization function by looping over the whole image array and equalize the output of the process we got the following image:

We can see the difference in histograms, now it's values distributed over larger range of data.

And we found that histogram equalization is a good technique for adjusting image intensities to enhance contrast but it is not important for all images.

Part 6:-

Normalize the image:

Normalization process doesn't depend on histogram.

By calculating the mean and standard deviation for image data array and using the following equation:

```
New Value = (Original Value - mean) / std^2
```

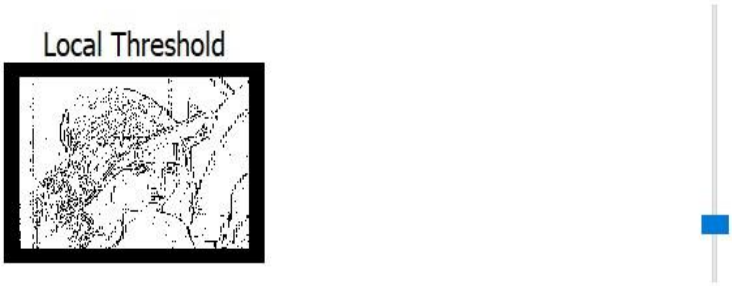
We got the following image:

Part 7:-

Local and global thresholding:

Global Thresholding: We implemented the global function where it iterates over all the image's pixels and assign it to a new value where it becomes 255 or 0 according to whether or not it's greater than a given threshold provided by the user.

Local Thresholding: We implemented the local function to work as it divides the image to many smaller windows where their size is selected by the user, the mean is calculated to the selected window and then is used as the threshold value to this window by applying the same technique used in the global function, the user have the option to choose if a constant is needed to be subtracted from the mean before applying it to the window.

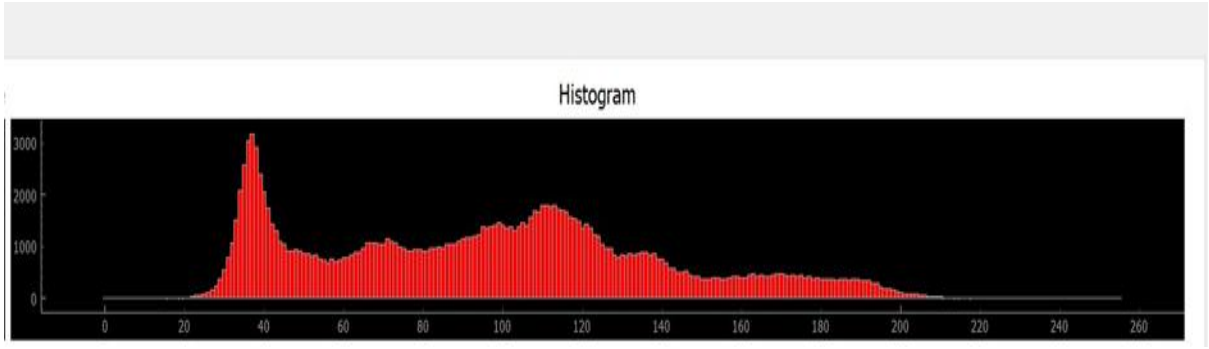


we found that local thresholding is suitable for some images and global thresholding is suitable for another ones it depends on the image.

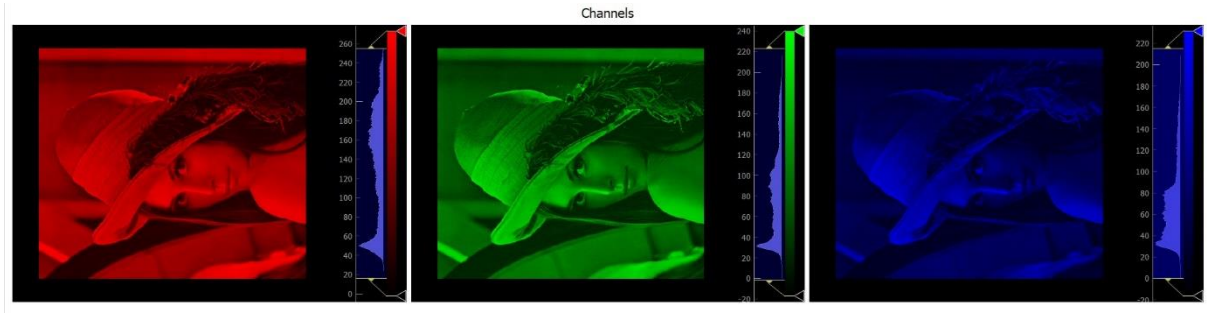
Part 8:-

Transformation from color image to gray scale image and plot of R, G, and B histograms with its distribution function (cumulative curve that you use it for mapping and histogram equalization).

The transformation of a colored image to a gray scale image was done by selecting each corresponding pixel in the 3 channels(ie. RGB) and multiplying these value by certain constant values respectively and add them together. the result is s then equal to the pixel value needed to acheive the suitabale grey color for that pixel. Where its histogram is displayed as follows:



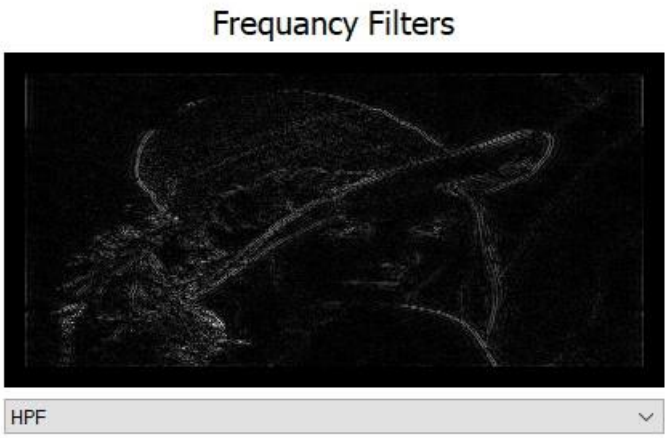
RGB channels and its histograms :



Part 9:-

Frequency domain filters (high pass and low pass):-

we use function "freqFilters" to manipulate the image with a LPS or HPS in the frequency domain and get the image back to time domain and the result is :



We noticed that LPF is a blurring filter and HPF is an edge detection filter.

Part 10:-

Hybrid images:

After implementing the high pass and the low pass filters required in part 9, hybrid images could be acheiving by simply applying the high filter on an image and the low pass on another and adding the results of each together to reach the merging between the two images needed.