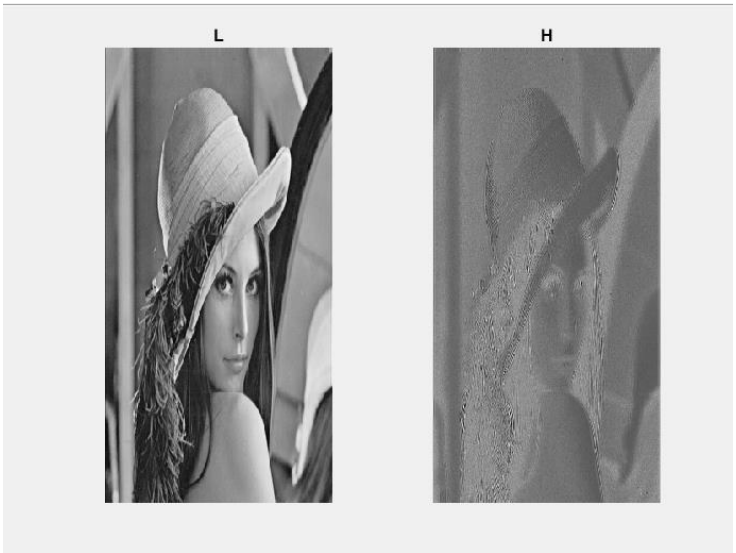
Image Sub-Band Coding

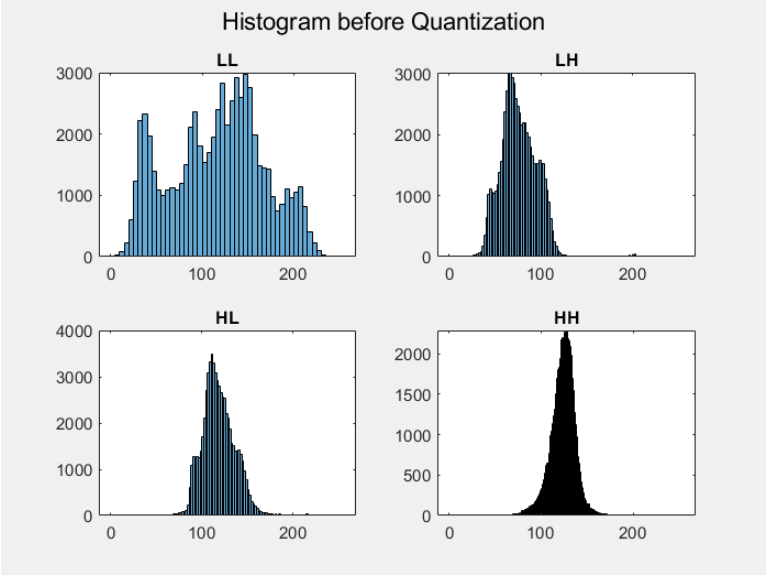
The image compression is divided into several steps which are as follows:

1. Read the image.

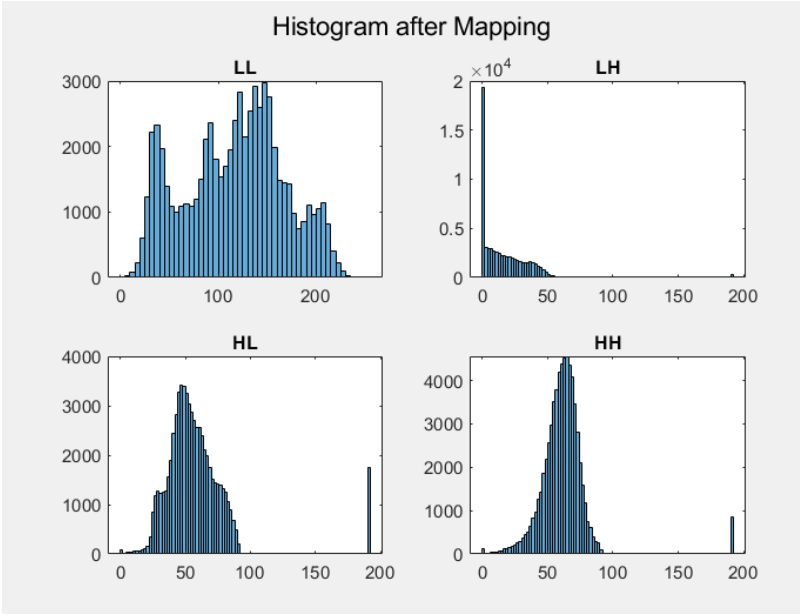
2. **First level of filtering and decimation (2 Subbands):** Then filter the image row by row using low pass and high pass filter, and downsample the the filtered image. This is an intermediate image. This is called L and H respectively.



3. **Second level of filtering and decimation (4 Subbands):** Next, do the same thing with each column of the above image. This way we obtain HH, HL, LH, LL.
4. To check what's the distribution of the pixel values of the images, we plot the histogram of the image.

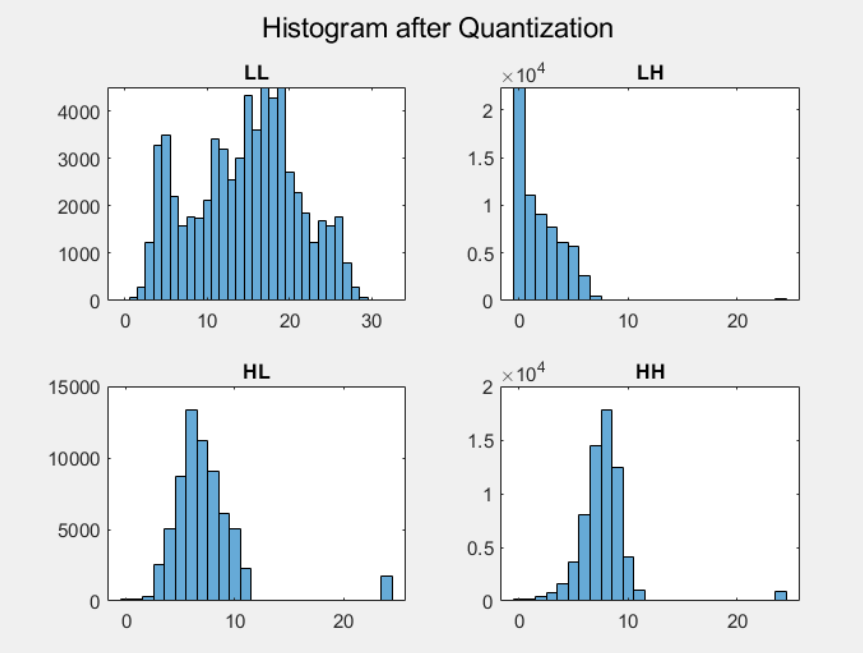


5. **Uniform and Non-Uniform Quantization:** Following are the observation from the above histogram:
    a. We see that except LL all the other subbands are concentrated at some region.
    b. We can use uniform quantizer for LL.
    c. Most of the pixel values in LH, HL, HH are within 80-180.
    d. For them we need to use non-uniform quantizer.
        i. First we map the subband region using a mapping function so that spread of the image pixel values increase. For this we have chosen a function **$f(x) = ax^2+b$**, subject to f(80) = 0 and f(180) = 255. On solving this we get a = 9.8e-3 and b = -67.72
    e. After applying this mapping function to the 3 subbands following are the new subbands:



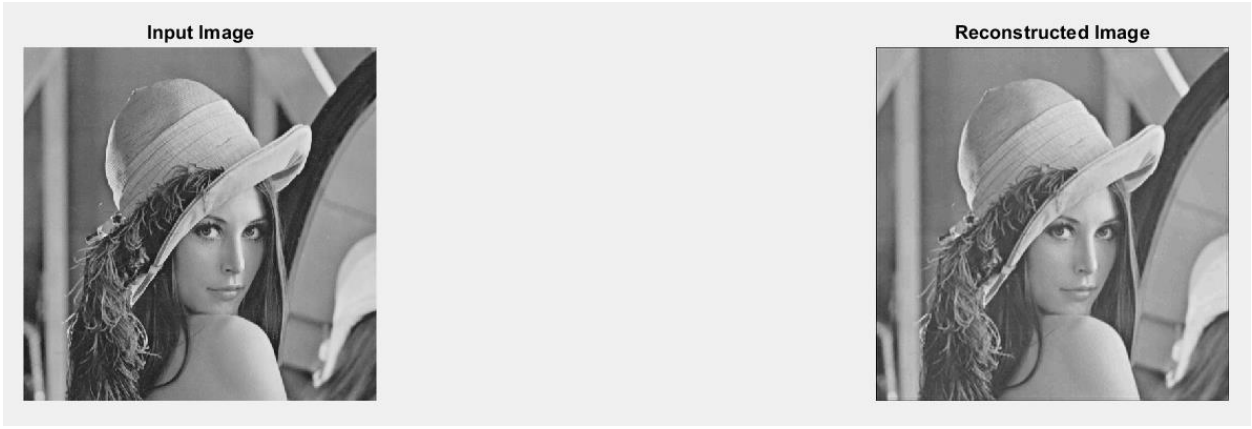
Although the pixel values in LH, HL and HH have spread a little, there's still a big scope of improvement.

f. Now finally we quantize our subbands into 32 levels. After quantizing, we get:

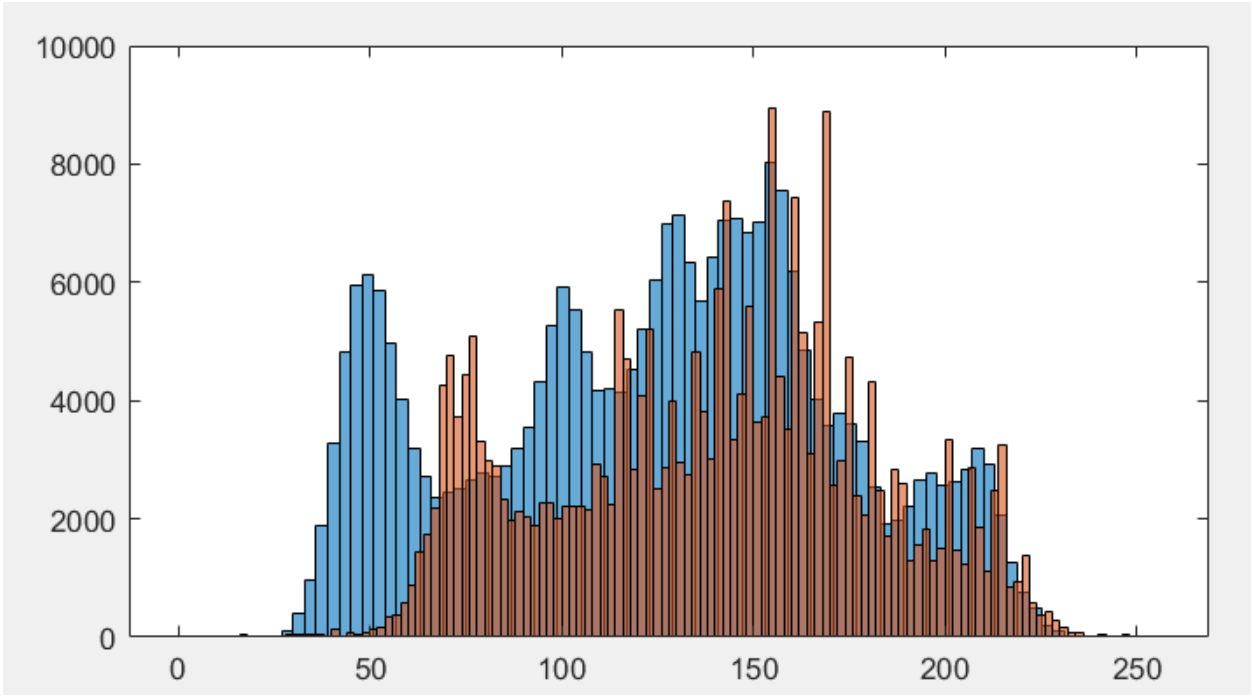


Till here we have compressed the image. We have **achieved the compression of 37.5%**. That is because earlier we were using 8 bits to represent our image whereas we are using 5 bits to represent the same now. Although there would be some loss associated with the compression which we can calculate after seeing further.

6. **Reconstruction:** Now, we start reconstruction of the image. For that we need to first use the filter f_0 and f_1. This filtering is done column-wise and then the image is upsampled. It is followed by filtering (row-wise) it again using the same filters and upsampling again. Eventually we get the original image size 4 images. We add them all together and we have reconstructed our image.



This histogram of the pixel values of both the images is shown below. Red ones are for the reconstructed images and the blues are for the input image.



Performance Index: Finally, if we try to find the root mean square error we get **0.0356.** If we reduce the number of quantization levels, this error goes up. Therefore, there's a tradeoff between compression and loss.

Discussion:

1. The loss can go down more if we optimize our filter more or go to higher order filter. Pixel values are seeing some gain due to our non-ideal filter. But this would negatively impact our code run time.
2. If we keep on increasing our filter bank tree, we would go on decreasing our quantization loss but again our code would take more time to run.

3. Instead of dividing our filter bank into regions like low pass and high pass, we could have taken more regions and started with it. This also would have led to better compression.

   All the points above need more computation but eventually going to lead to better compression.

Additional Observations:

The above compression has been done where image size has been kept the same. Whereas in real, convolution would be linear, and the size of the image would keep on changing after further processing.
Say, we have filter of length 13 and image of 512x512. If we try to use 4 subbands, after reconstruction, we are going to obtain the image of size 536x536 because every time linear convolution happens number of that dimension would increase by filter length -1. Below is the reconstructed image.

Here, edges have the artifacts in the form of borders which was bound to happen due to linear convolution.