

EEE 404/591 – Real Time DSP –

Project 1 Real Time Image Processing

-Hemanth Balaji Dandi

Objective:

The objective of this lab is to apply simple real time image enhancement techniques to live video stream captured from a webcam attached to the PC via the Universal Serial Bus (USB).

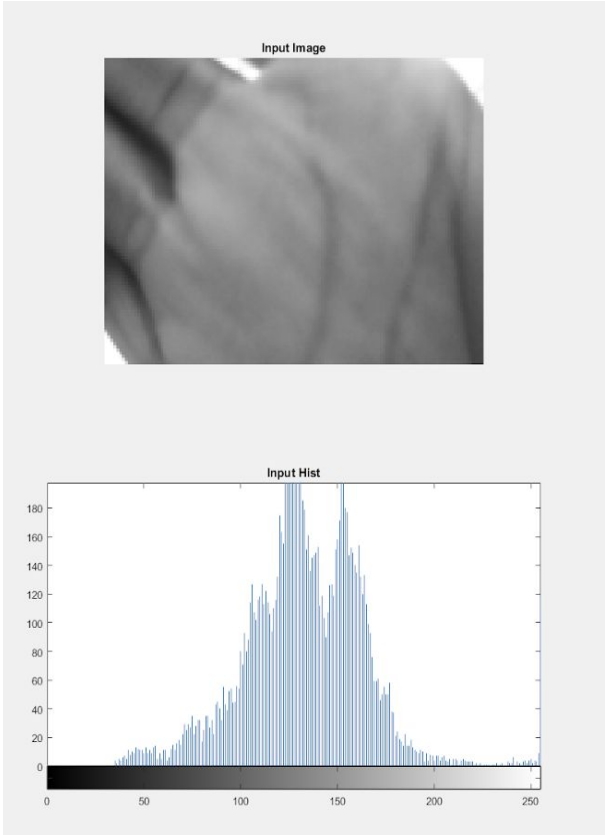
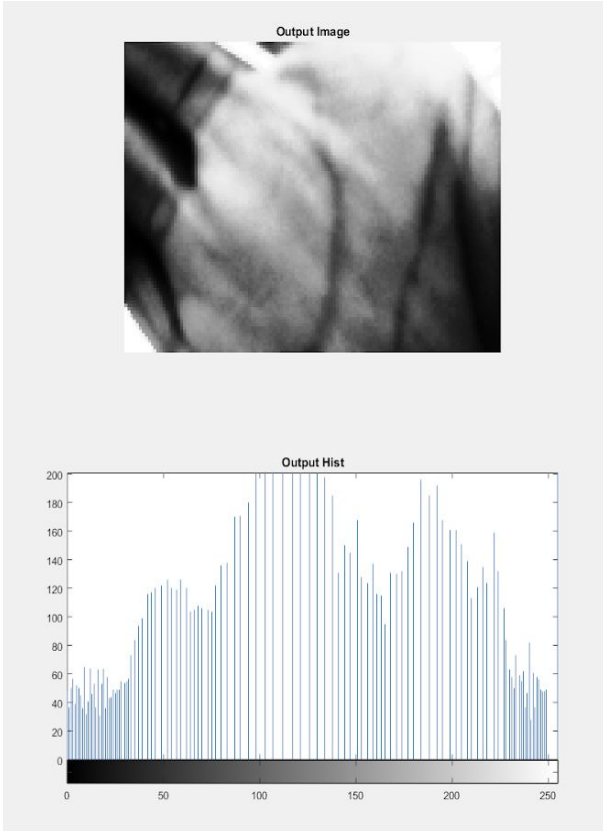
The following transforms are implemented-

- ❖ Histogram Equalization
- ❖ Segmentation
- ❖ Grayscale Quantization
- ❖ Grayscale Transformations to enhance brightness and contrast

Before starting to modify the “rtimage.mcp” template, try to scan the code and answer the following questions:

- *What are the beans used in Processor Expert?*
 - *What are the settings for the “AsynchroSerial” bean?*
 - *What is the size of the input and output buffers, respectively?*
 - *What is the baud rate used?*
 - *What is the frame rate?*
 - *What methods of the “AsynchroSerial” bean are used in “rtimage.c”?*
-
- ☐ Only the **Serial-AsynchroSerial bean** is used
 - ☐ Input Buffer size-**128** ; Output Buffer Size-**1024**
 - ☐ Baud Rate- **57600 baud**
 - ☐ Frame Rate- $\frac{57600 \text{ bits/sec}}{98304 \text{ bits/frame}} = \mathbf{0.586 \text{ frames/sec}}$
 - ☐ Methods of the “AsynchroSerial” bean are **Recvchar, Sendchar, RecvBlock, SendBlock, ClearRxBuf, ClearTxBuf, GetCharsInRxBuf, GetCharsInTxBuf.**

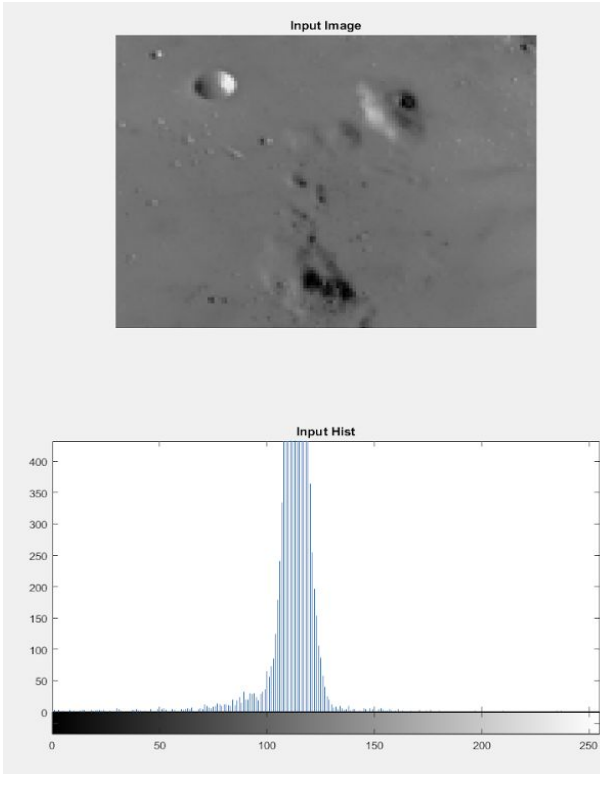
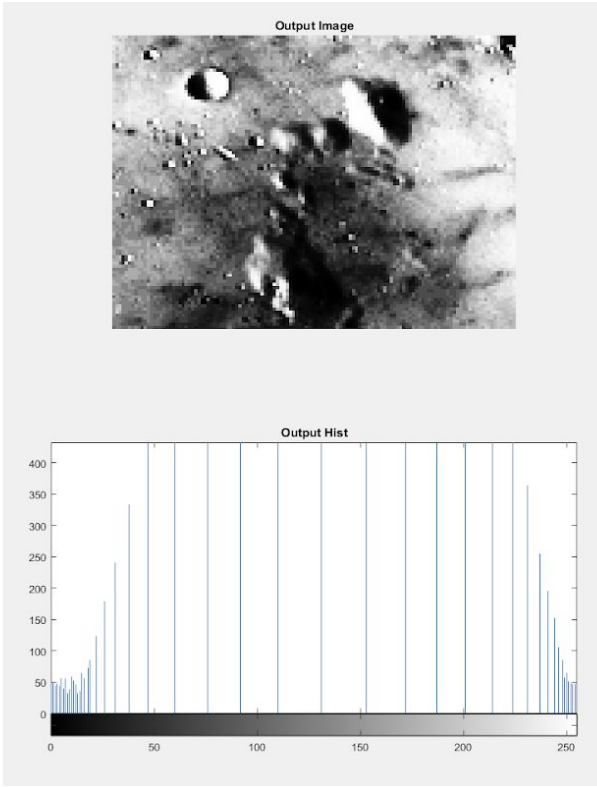
A. Histogram Equalization (For both C and Hybrid)

Input Image	Output Image
 <p>The input image is a grayscale photograph of a palm leaf, appearing somewhat dark and low-contrast. Below it is the 'Input Hist' (histogram), which shows a narrow, bell-shaped distribution of pixel intensities, primarily concentrated between 100 and 180 on the x-axis (representing pixel values from 0 to 255). The y-axis represents frequency, ranging from 0 to 180.</p>	 <p>The output image is the same palm leaf, but with significantly enhanced contrast, making the veins and edges much clearer. Below it is the 'Output Hist' (histogram), which shows a much wider and more uniform distribution of pixel intensities across the entire range from 0 to 255. The y-axis represents frequency, ranging from 0 to 200.</p>
Transform : Histogram Equalization	
<p>Parameters: image_org -> input image pixels hist_org -> histogram mapped_levels -> new mapped levels of histogram image_eq -> image after histogram equalization N1 -> Total number of pixels in a frame-96x128 size -> Total number of levels in the histogram-256</p>	

Observation:

We can see that, by histogram equalization, the histogram of the output image is spread out across the whole dynamic range of the image.

Also, because of this, we can see the lines/ edges of the palm much clearer.

Input Image	Output Image
	
Transform : Histogram Equalization	
Parameters: image_org -> input image pixels hist_org -> histogram mapped_levels -> new mapped levels of histogram image_eq -> image after histogram equalization N1 -> Total number of pixels in a frame- 96x128 size -> Total number of levels in the histogram- 256	

Observations:

We can observe, both in the C and the Hybrid case, the initial histogram and the histogram obtained after histogram equalization are different for the old and transformed images respectively. The reason behind this, is simply because, if we look at the histogram of the initial images (can be seen in the 'moon.png' result much clearer), it is populated between only a few intensities, while the rest of the intensities are very low. In-order to spread the intensities across

the whole dynamic range of the image, ie, across 0->256, we perform histogram equalization, and hence we see a much more spread-out histogram in the output.

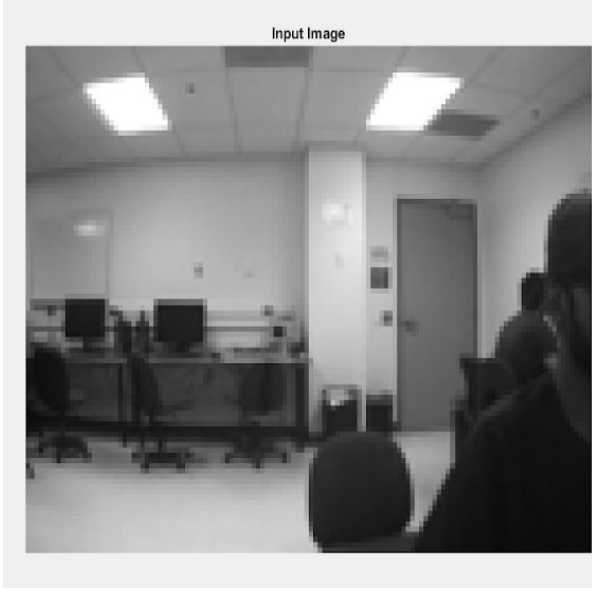
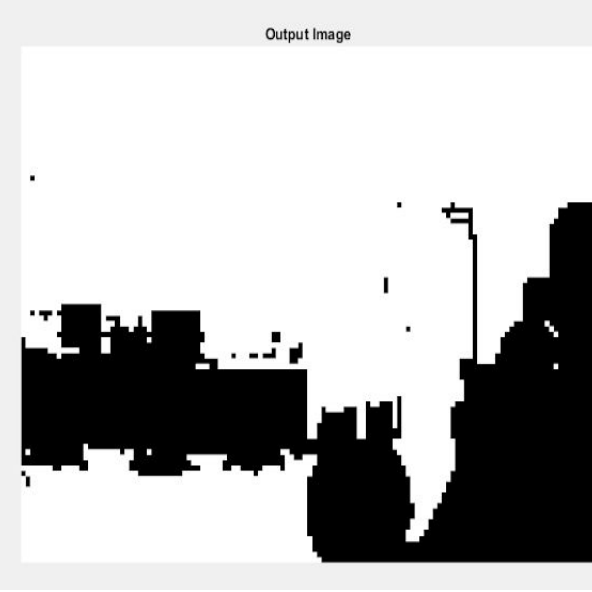
The purpose of this is to view regions of an image which could not be viewed from the original image, like for example, in the 'moon.png', the original image had little information about the crevices on the surface, but through histogram equalization, those crevices can be seen.

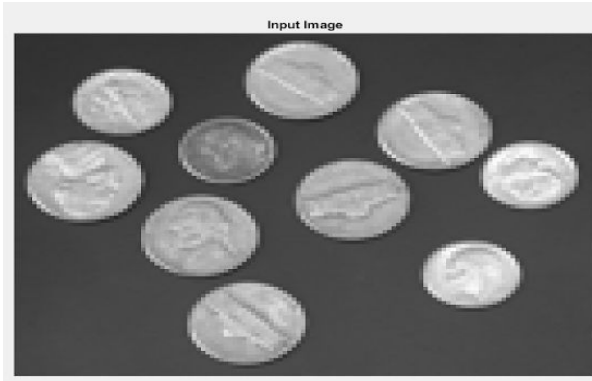
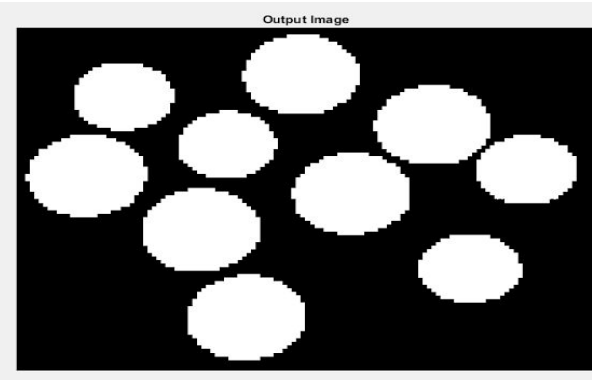
Techniques to optimize the C Code for Histogram Equalization

- In the Map_levels() function, instead of dividing by N in every pdf calculation in one iteration, we can calculate the cdf of that iteration in 'sum' and divide that value at the end of the iteration by 'N' instead of dividing multiple times for each pdf in each iteration. This saves computational time and makes the code more compact
- In the same Map_levels() function, instead of creating another variable to multiply with 255 and take that variable and divide by 'N' and store it in another variable, the above two operations can be performed in one line, which saves program memory by not using redundant local variables.
- For calculate_histogram(), instead of doing a tedious and longer process of going through a search algorithm to increment the positions where the input image pixels have occurred on the histogram, thereby increasing the computation time, memory, unnecessary loops and if conditions (which require a lot of jumps), we use a pointer to shift to the said pixel position and increment, which saves memory, speed and more efficient.
- Again in Map_levels(), instead of doing $sum = sum + ((255 * sum) / N)$, use $sum += (255 * sum) / N$.

B. Segmentation

B.1 Global Thresholding

Input Image	Output Image
	
Transform : Global Thresholding	
Parameters: temp -> input image pixel T -> Threshold-80	



Input Image	Output Image
	
Transform : Global Thresholding	

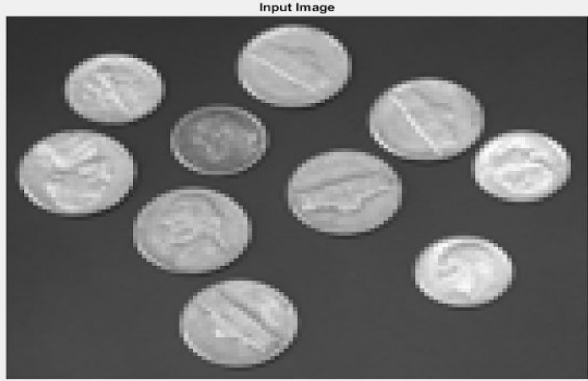
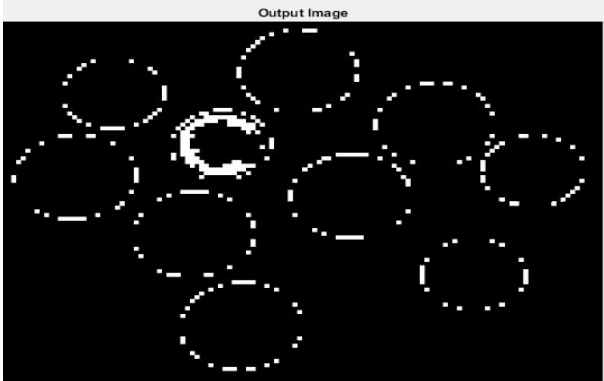
Parameters: **temp** -> input image pixel
T -> Threshold-**80**

Observation:

We can see that, through global thresholding, we can segment the image into two mutually exclusive parts. The way to separate them perfectly (like in the coins.png), is to look at the histogram of the image, and if the image's intensity distribution is such that there is a good separation between the dark and light pixels, we can take a point at that region as our Threshold, T (Hence, we have taken 80), and using that, we get a white region in where the coins were supposed to be, and dark region everywhere else.

B.2. Band Thresholding


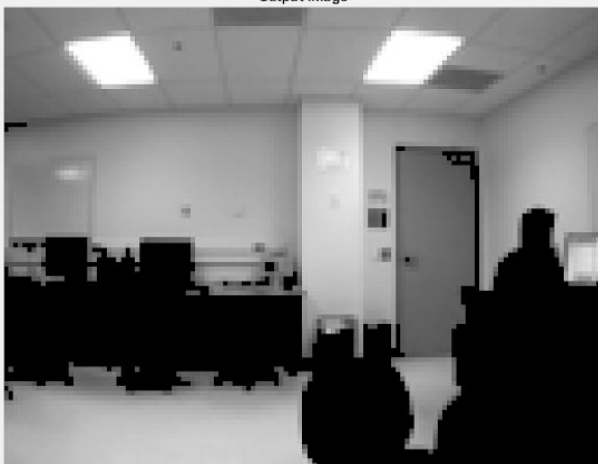
Input Image	Output Image
	
Transform : Band Thresholding	
Parameters: temp -> input image pixel L -> Lower Threshold- 80 H -> Upper Threshold- 120	

Input Image	Output Image
	
Transform : Band Thresholding	
Parameters: temp -> input image pixel L -> Lower Threshold- 80 H -> Upper Threshold- 120	

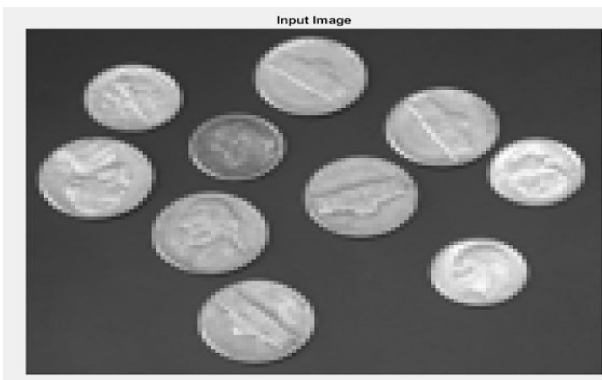
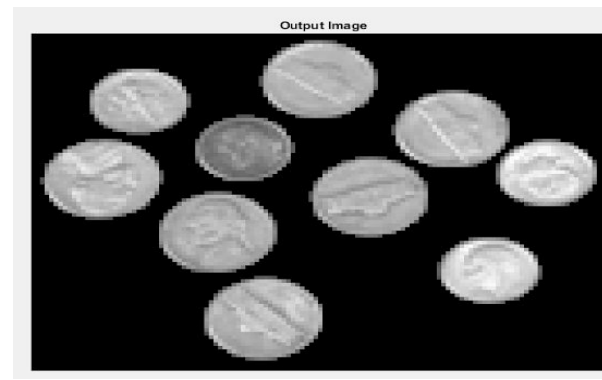
Observation:

We can observe that, through Band Thresholding, we can fix a band of pixels to keep, and the rest can be removed. Here, in the above two cases, a threshold between 80 and 120 is taken, thus any pixel having an intensity value in-between them are returned as white, otherwise black. This thresholding method helps to find objects/ regions in an image.

B.3 Semi Thresholding

Input Image	Output Image
	

Transform : Semi Thresholding
Parameters: temp -> input image pixel T -> Threshold-80

Input Image	Output Image
	
Transform : Semi Thresholding	
Parameters: temp -> input image pixel T -> Threshold-80	

Observation:

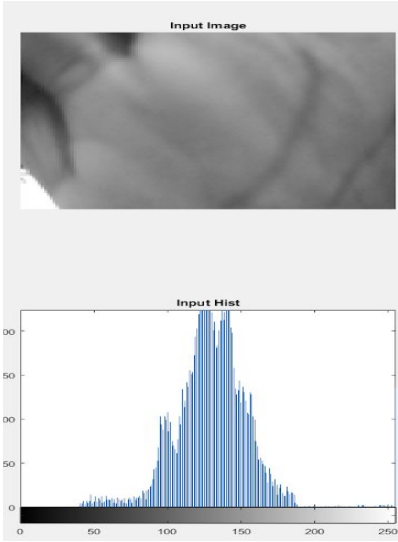
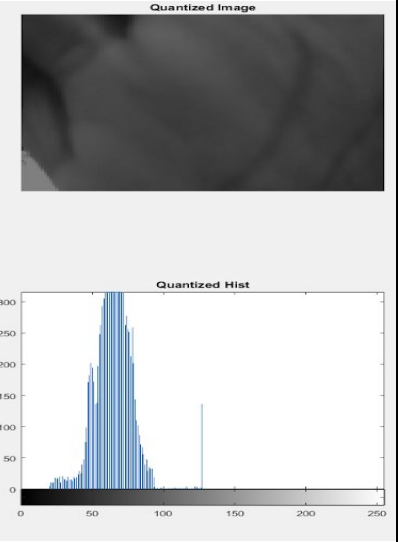
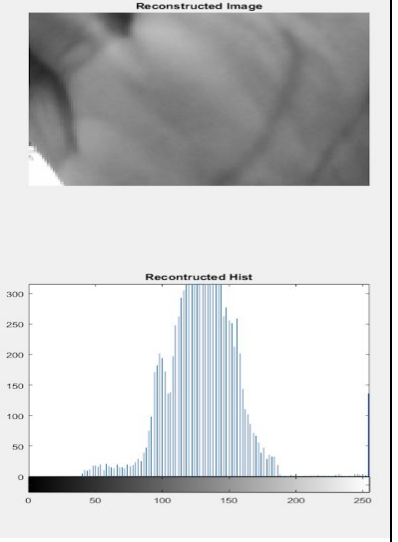
We observe through Semi-thresholding that, we keep the pixels above a certain threshold and remove the rest, ie, make them dark.

So, in the above images, all those pixels having an intensity value greater than 80 are kept, while the rest are put to 0.

Thus, this method helps to obtain objects from an image whose background or other information are not relevant.

C. Grayscale Quantization

ShiftFactor-1

Input Image	Output Image (Quantized)	Reconstructed Image
 <p>The input image is a grayscale photograph of a person's face. Below it is a histogram titled 'Input Hist' showing a smooth, bell-shaped distribution of pixel intensities across the range 0 to 255.</p>	 <p>The quantized image shows the same face but with visible horizontal banding artifacts. Below it is a histogram titled 'Quantized Hist' showing a distribution where the pixel values are concentrated in discrete steps, with a prominent peak around 128.</p>	 <p>The reconstructed image is visually very similar to the original input image. Below it is a histogram titled 'Reconstructed Hist' showing a distribution that is nearly identical to the original input histogram, though with some minor differences in the tails.</p>
Transform : Grayscale Quantization		
Parameters: temp -> input image pixel ShiftFactor -> (1)-> Amount of right-shift for every pixel, ie, $\text{temp}/2^{\text{ShiftFactor}}$		

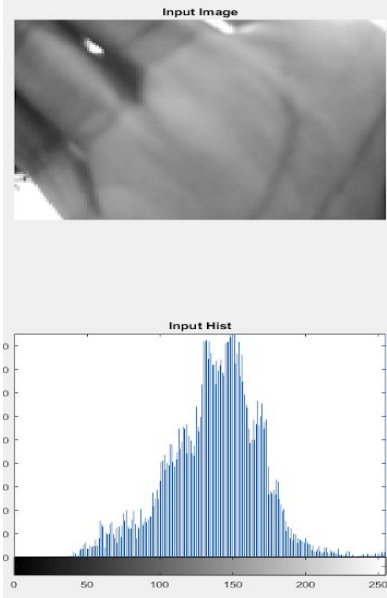
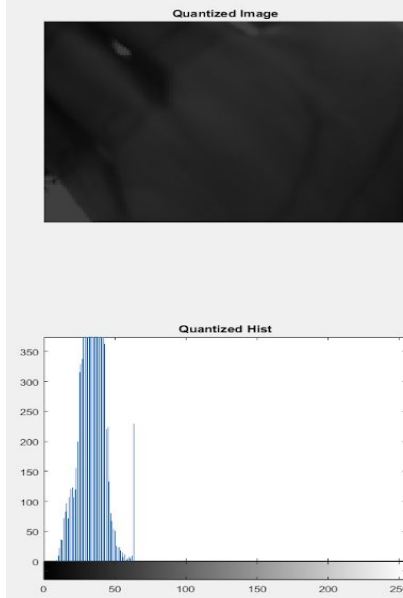
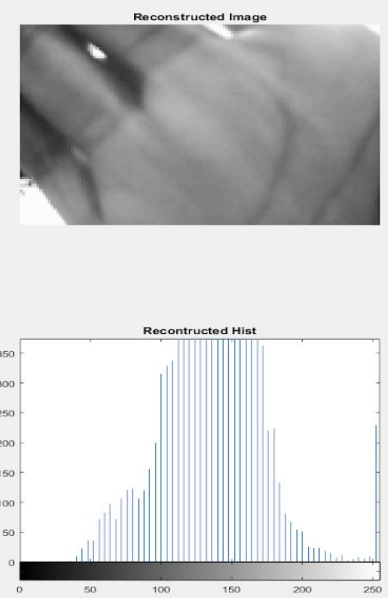
Observation:

We can see that, through quantization by a factor of 2, ie, we right-shift every pixel by 1, we reduce the dynamic range by half, ie, 128, and thus, we see the histogram of the quantized image shifted to the left, and the highest pixel being at 128.

In the reconstructed image, we don't notice a lot of difference from the original image, but when we look at the histogram, we see few values are missing.

This is because, as we are quantizing, we divide by 2, and the resultant number doesn't stay as float, but gets rounded-off. Then, when we reconstruct, even though we multiply by the same factor, the value would not be the same as the original pixel intensity. Hence we see values missing from the histogram.

ShiftFactor-2

Input Image	Output Image (Quantized)	Reconstructed Image
 <p>The input image is a grayscale photograph of a hand. Below it is the 'Input Hist' histogram, which shows a continuous distribution of pixel intensities across the range 0 to 255, with a peak around 150.</p>	 <p>The quantized image is significantly darker than the input. Below it is the 'Quantized Hist' histogram, which shows a very narrow distribution of pixel intensities, mostly concentrated between 0 and 64, indicating a loss of dynamic range.</p>	 <p>The reconstructed image is visually similar to the input image. Below it is the 'Reconstructed Hist' histogram, which shows a distribution of pixel intensities that is broader than the quantized histogram but still has gaps, indicating some loss of information during reconstruction.</p>
Transform : Grayscale Quantization		
Parameters: temp -> input image pixel ShiftFactor -> (2)-> Amount of right-shift for every pixel, ie, $\text{temp}/2^{\text{ShiftFactor}}$		

Observation:


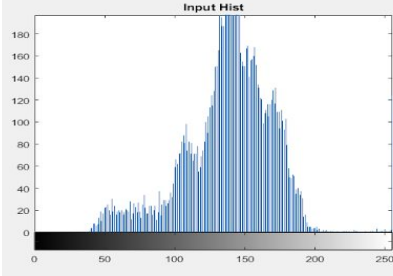

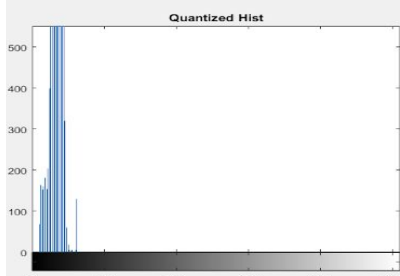

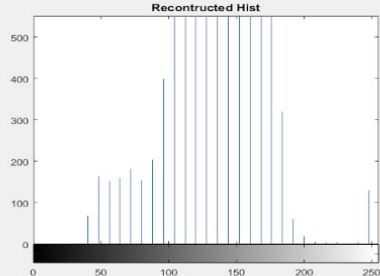
We can see that, through quantization by a factor of 4, ie, we right-shift every pixel by two times, we reduce the dynamic range by four times, ie, 64, and thus, we see the histogram of the quantized image shifted to the left, and the highest pixel being at 64.

In the reconstructed image, we don't notice a lot of difference from the original image, but when we look at the histogram, we see few values are missing.

This is because, as we are quantizing, we divide by 4, and the resultant number doesn't stay as float, but gets rounded-off. Then, when we reconstruct, even though we multiply by the same factor, the value would not be the same as the original pixel intensity. Hence we see values missing from the histogram.

We also see more values are missing than in the previous case, because, since by dividing by 4, we have more cases of remainders, hence more error due to quantization.

ShiftFactor-3

Input Image	Output Image (Quantized)	Reconstructed Image
 	 	 
Transform : Grayscale Quantization		
Parameters: temp -> input image pixel ShiftFactor -> (3)-> Amount of right-shift for every pixel, ie, $\text{temp}/2^{\text{ShiftFactor}}$		

Observation:

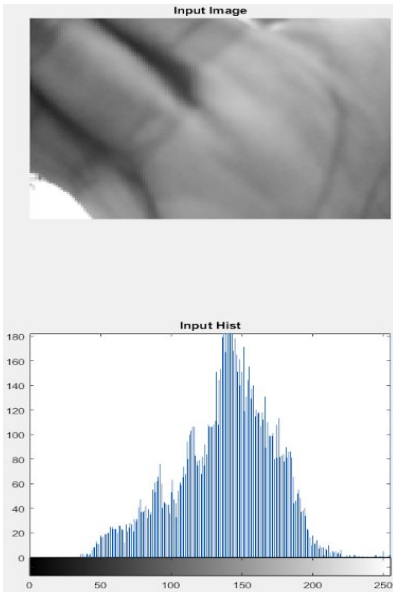
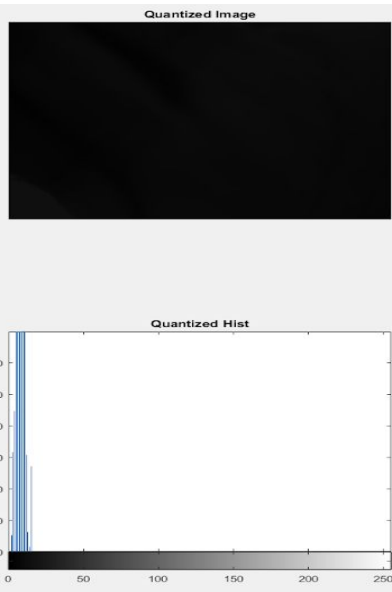
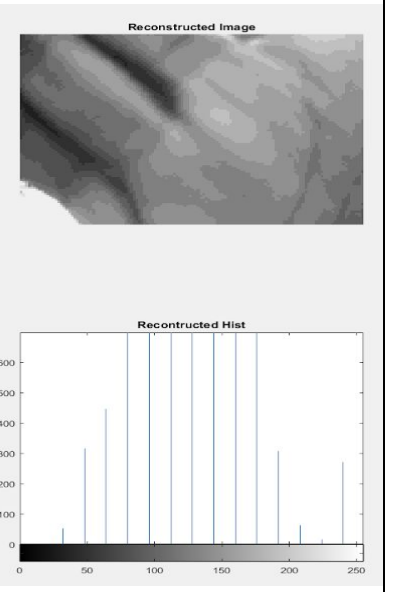
We can see that, through quantization by a factor of 8, ie, we right-shift every pixel by three times, we reduce the dynamic range by eight times, ie, 32, and thus, we see the histogram of the quantized image shifted to the left, and the highest pixel being at 32.

In the reconstructed image, we don't notice a lot of difference from the original image, but when we look at the histogram, we see few values are missing.

This is because, as we are quantizing, we divide by 8, and the resultant number doesn't stay as float, but gets rounded-off. Then, when we reconstruct, even though we multiply by the same factor, the value would not be the same as the original pixel intensity. Hence we see values missing from the histogram.

We also see more values are missing than in the previous case, because, since by dividing by 8, we have more cases of remainders, hence more error due to quantization.

ShiftFactor-4

Input Image	Output Image (Quantized)	Reconstructed Image
		
Transform : Grayscale Quantization		
Parameters: temp -> input image pixel ShiftFactor -> (4)-> Amount of right-shift for every pixel, ie, $\text{temp}/2^{\text{ShiftFactor}}$		

Observation:

We can see that, through quantization by a factor of 16, ie, we right-shift every pixel by four times, we reduce the dynamic range by sixteen times, ie, 16, and thus, we see the histogram of the quantized image shifted to the left, and the highest pixel being at 16.

In the reconstructed image, we don't notice a lot of difference from the original image, but when we look at the histogram, we see few values are missing.

This is because, as we are quantizing, we divide by 16, and the resultant number doesn't stay as float, but gets rounded-off. Then, when we reconstruct, even though we multiply by the same factor, the value would not be the same as the original pixel intensity. Hence we see values missing from the histogram.

We also see more values are missing than in the previous case, because, since by dividing by 16, we have more cases of remainders, hence more error due to quantization.

For each case (128, 64, 32 and 16) capture results as shown in Table 1. Answer the following: 1) Do you observe any values greater than the maximum chosen level (128, 64, 32 or 16) in the histogram plot of the quantized image? What is the new dynamic range?

2) Do you observe any missing values in the histogram of the reconstructed image at locations where there were values in the original histogram? Comment.

1) There will not be any value greater than the maximum chosen levels in the respective cases of ShiftFactors as we are dividing the maximum value, ie, 256 by $2^{\text{ShiftFactor}}$.

The new dynamic range of the image is

Dynamic Range of the new image:

$$\frac{1}{2^{\text{ShiftFactor}}} (\text{Highest Pixel Value of the original image} - \text{Lowest Pixel Value of the original image})$$

Therefore, in our case, the Dynamic range(s) of the maximum level(s) chosen are- **128,64,32 and 16** respectively (As the maximum pixel intensity value of the input image is 256 and the minimum pixel intensity value is 0)



2) For the reconstructed image, we do observe missing values when compared to the original histogram. This is because, when we quantize the image according to lower levels, it involves dividing by a factor of 2. As we quantize the image from 128 levels to 16 levels, the factor of 2 with which we divide the pixel values increases, ie, from 2 to 16. As we divide the pixel values by these factors, the modified pixel values get rounded-off to the nearest integer, and when we are reconstructing, when we multiply the modified pixel value by the same factor, the reconstructed pixel value comes different than the original pixel intensity at that location.

Thus as we increase our shiftfactor, the quantization error increases as there will be a bigger range present in the remainder as we go from 2 to 16 ,when we divide the pixel values by that factor, and thus values will be missed due to round-up/ round-down errors.

Therefore we see missed values in the reconstructed histogram of the image, and the missing values increase as we increase our quantization factor.

D. Grayscale Transformation

D.1 Negative Transformation



Input Image	Output Image
	
Transform : Negative Transformation	
Parameters: temp -> input image pixel	

Observations:



We see that the output image is a negative of the original image as we mapped the original image according to the equation $y=255-x$. Hence all the white pixels become dark and all the dark pixels become white.

D.2 Transformation 2 - Brightness/ Contrast Transformation



Case (i) $A < B$

Input Image	Output Image
	
Transform : Transformation 2 - Contrast Transformation	
Parameters: temp -> input image pixel M -> Transformation Constant 'A'-70 N -> Transformation Constant 'B'-110	

Case (ii) $A = B$

Input Image	Output Image
	
Transform : Transformation 2 - Same Transformation	
Parameters: temp -> input image pixel M -> Transformation Constant 'A'-110 N -> Transformation Constant 'B'-110	

Case (iii) $A > B$

Input Image	Output Image
	
Transform : Transformation 2 - Brightness Transformation	
Parameters: temp -> input image pixel M -> Transformation Constant 'A'- 110 N -> Transformation Constant 'B'- 70	

Observations

For case (i) $A < B$

From the graph of the transformation, we can see that, if B is greater than A, more lower intensity values get mapped to a smaller range of lower intensity values, and large range of higher intensity values get mapped to a smaller range of high intensity values, and the intensity range between B and $255-B$ gets mapped to a wider range of intensity values.

Simply put, the extreme dark pixels, or those close to dark, become darker; the extreme white pixels, or those close to extreme white become whiter, and the intensities in-between, become slightly lighter as the graph spreads to a bigger range for those in-between B and $255-B$ intensity values.

Thus there is more **contrast** present here due to the extreme pixel intensities and brightness in-between.

For case (ii) $A = B$

From the graph of the transformation, we can see that, if B is equal to A, then we can see the graph becomes linear, ie, $y=x$, therefore there will be a linear relationship between the corresponding pixel intensities.

Thus all the pixel intensities will remain the same as they will be mapped to the same values, hence the image will **remain as it is**.

For case (i) $A > B$



From the graph of the transformation, we can see that, if B is less than A , smaller range of lower intensity values get mapped to a bigger range of lower intensity values, and smaller range of higher intensity values get mapped to a larger range of high intensity values as well, and the intensity range between B and $255-B$ gets mapped to a shorter range of intensity values.

Simply put, the extreme dark pixels, become less darker; the extreme white pixels, become less whiter, and the intensities in-between, become slightly more whiter/darker depending upon the pixel value as the graph spreads to a smaller range for those in-between B and $255-B$ intensity values, thus there is an overall brightness in the image



Here we can view some- sort of **brightness** present in the image.

D.3. Transformation 3 - Brightness/ Contrast Transformation



Case (i) $A < B$

Input Image	Output Image
	
Transform : Transformation 3 - Contrast Transformation	
Parameters: temp -> input image pixel M -> Transformation Constant 'A'-70 N -> Transformation Constant 'B'-110	

Case (ii) $A = B$

Input Image	Output Image
	
Transform : Transformation 3 - Same Transformation	
Parameters: temp -> input image pixel M -> Transformation Constant 'A'-110 N -> Transformation Constant 'B'-110	

Case (iii) $A > B$

Input Image	Output Image
	
Transform : Transformation 3 - Brightness/ Contrast Transformation	
Parameters: temp -> input image pixel M -> Transformation Constant 'A'-110 N -> Transformation Constant 'B'-70	

Observations

For case (i) $A < B$

From the graph of the transformation, we can see that, if B is greater than A , more lower intensity values get mapped to a smaller range of lower intensity values, and therefore the rest of the pixel intensities get mapped to a larger range of intensities.

Simply put, the dark pixels, or those close to dark, become darker; and the rest of the pixels become less white.

Thus we can see some sort of **contrast** in the image in this case

For case (ii) $A = B$

From the graph of the transformation, we can see that, if B is equal to A , then we can see the graph becomes linear, ie, $y=x$, therefore there will be a linear relationship between the corresponding pixel intensities.

Thus all the pixel intensities will remain the same as they will be mapped to the same values, hence the output image will remain the **same** as the input.

For case (i) $A > B$

From the graph of the transformation, we can see that, if B is less than A , smaller range of lower intensity values get mapped to a bigger range of intensity values, and therefore the rest of the pixel intensities get mapped to a smaller range of intensities.

Simply put, the dark pixels, become less darker; and the rest of the pixels become more lighter. Here we can view some- sort of **brightness** in the image.

Lab Evaluation

Describe any difficulties encountered while performing lab

- ☐ Histogram equalization using Assembly was a bit tricky. Before I encountered that, I was under the impression that all pointer values get to only $R2$ during the return of a function, but then realized that $R2$ just points to an address, so it doesn't matter to which register the return address points to, as long as that register has the right address location.
- ☐ Division was a bit tricky to figure out without the help of the TA's example involving the division of a 32-bit integer with a 16-bit integer.
- ☐ The transformation functions in Assembly was again made easy by the example provided by the TA. Multiplication and then division was stressed rather than the opposite due to the possibility of getting 0.

List any errors or any unclear statements found in lab manual

- ❑ Every objective was clearly explained and pseudo-code for histogram equalization was given with proper explanation.

Give suggestions for improving this lab

- ❑ It was a challenging lab which involved learning Assembly with DSP56858E, which proved to be extremely useful. So there isn't any reason to improve this lab as it covered the two basic requirements for any project- Curiosity to learn new things & Challenge one's knowledge into learning more.

Describe what you learnt from this lab

- ❑ Histogram Equalization - Learnt to program it in both C and Hybrid. The program had to be memory efficient as well, since the execution time of the program should sync up with the time taken to read a frame from the PC to the board.
- ❑ Segmentation- Covered 3 thresholding techniques- Global, Band and Semi. Programmed all in both C and Hybrid.
- ❑ Grayscale Quantization- Performed grayscale quantization to reduce the overall storage bits, and therefore reduce time taken to transmit the image, and then reconstructing the image back to its original size. Eventhough, as we increase our quantization shiftfactor, the reconstructed image starts to degrade, it still is recognizable, which stands to reason that we can use it for some image processing pipelining methods for image recognition as we don't require the full resolution of the image to recognize it.
- ❑ Grayscale Transformations- Implemented 3 transformations pertaining to negative, brightness and contrast transforms on the image. Using different line equations, we can change the brightness, contrast or make the image negative, and this was in both C and Hybrid
- ❑ Also, in all the above cases, learnt to take an image from a webcam of the PC from MATLAB, send it to the board for processing, then send it back to the PC after being processed and display the images via MATLAB.
- ❑ Eventhough in the beginning Assembly takes more time to code than C, we can see that it involves very less storage capacity, almost 50% reduction of bytes than C, which makes it more faster than implementing with C.

Indicate what you liked most and what you liked least

- ❑ Eventhough at the start, Assembly was quite challenging, but I have grown to like it and it started being more and more intuitive.
- ❑ The transformation functions were interesting to implement, and to figure out how to change the brightness or contrast of the image
- ❑ There was nothing to dislike in this project, everything was interesting and the time spent doing this was worth it.