Digital Image and Video Processing

Lab 2: 2D Signal Processing

Student Name: **Hugh Jordan**

Student Number: **16321743**

2D Signal Processing

2. Contrast enhancement

2.1. Explain how the contrast stretcher is implemented.

The JFK image was loaded and Matlab, by default, assigned the image the datatype that was native to the file, uint8. The datatype was then cast as a double because some operations are not implemented for uint8. The original image is displayed in Figure 1 and the corresponding histogram of greyscale values of the image is shown in Figure 2. It is evident from Figure 2 that the pixel values do not occupy all the available levels of grey. Because of this, the image values within the range 55 to 200 were stretched to occupy the whole 8-bit greyscale range. This was performed by applying a simple mask such as $mask\ =\ (I\ >=\ t1\ \&\ I\ <=\ t2)$ to perform the following operation:

$$l(h,k) = \begin{cases} 0 & For\ I(h,k) \leq t_1 \\ 1 & For\ t_1 <\ I(h,k) < t_2 \\ 0 & Otherwise \end{cases}$$

This would return 1 when the pixel fell inside t_1 and t_2 and return 0 when it fell outside this range. These values were then stretched across the range 0 -255 by mapping them using the equation $y = mx + c$, where $m = 255/(200 - 55)$ and $c = 55/255$. Similarly, another mask was applied that would return 1 when the pixel fell above t_2 and would return 0 when the pixel fell below t_2. These values were then clipped to 255 using the equation $y = 255x$. The two sets of values were then concatenated together using the equation $max(I_1, I_2)$. This equation would return the larger of the two ranges. The output of this concatenation was the newly contrasted image. This new image is shown in Figure 3 and the corresponding histogram is shown in Figure 4. It is evident from this histogram that the pixel values occupy all the available levels of grey now. This is difficult to identify because there is a high concentration of pixels with the value 0. This concentration was the result of all the values below t_1 that were clipped to 0.

Figure 1. Original Image

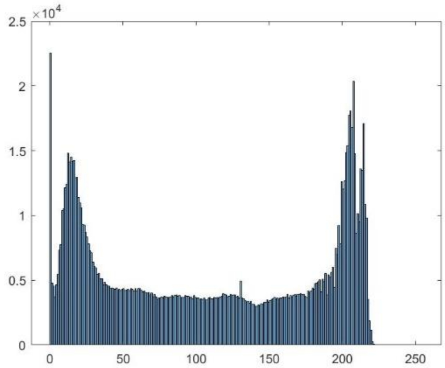

Figure 2. Original Image Histogram

Figure 3. Contrasted Image

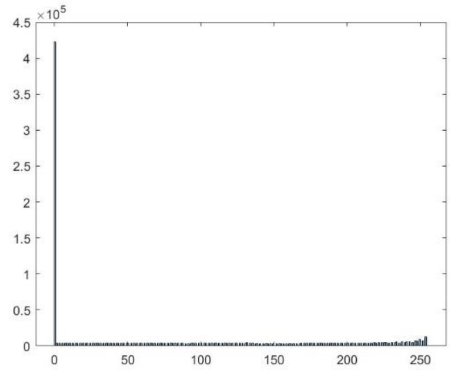

Figure 4. Contrasted Image Histogram

3. The Gaussian Filter

3.1 Explain how the Gaussian Filter is implemented.

The Gaussian filter takes 3 inputs, the variance (σ) of the filter, the size (N) of the filter and a flag which returns a 1D gaussian filter if the value is separable and a 2D filter if the value is combined. The function begins by initialising an array (g) of size N x 1 and iterates through that list assigning each element the array a value using the following equation:

$$g[x] = exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \; where \; \mu = \frac{N+1}{2}$$

The reason the mean (μ) is subtracted from x, is because it has the desirable effect of placing the maximum value of the filter mask at the centre of the array. The function then checks if the flag value is equal to combined. If the flag is, then the transpose of g is cross multiplied by g, resulting in a new array of size N x N. Otherwise, nothing is done, and the size of the array remains N x 1. This means that if an invalid value for the 3rd parameter is entered, then the default behaviour is to return a separable filter. Finally, all the coefficients of the filter are normalised to 1 by dividing each element of the array by the sum of all of the coefficients of the filter. The original image from Section 2 is shown in Figure 5 and the same image with a Gaussian filter applied is shown in Figure 6.

Figure 5. Original Image

Figure 6. Blurred Image

3.2. Explain why it is necessary that all the coefficients of the low pass filter sum to 1.
It is necessary that all the coefficients sum to 1 because if they sum to greater than 1, the image will get brighter after blurring and if they are less than 1, the image will get darker after blurring. This is because this is the equivalent to applying two filters, a Gaussian blurring filter and a brightness filter.

3.3. Changing the value of N to 21 and using the matlab functions tic and toc, record the time that it takes to apply the system to the image 1000 times. The filter should be applied to both the rows and the columns of the image. In your answer include the code you use and write down the time taken.
The time taken to apply the system to the image 1000 times is 11.055983 seconds.

3.4. Change your m-file to use the full 2D implementation of the filter (i.e. not separable). Include your code and write down the time it takes to apply this separable implementation 1000 times as before.
The time taken to apply the system to the image 1000 times is 36.978998 seconds.

3.5. Is there any difference in execution time between these two implementations? Explain your findings.
The execution time of the first implementation is 3.34 times smaller than the execution time of the second implementation. This is because the number of computations can be reduced enormously if the 2D filter is separable. The number of computations for a cascade of a row and column filter with sizes R and C taps is $N \times M \times R + N \times M \times C = N \times M \times [R + C]$ respectively. This means that for a 21×21 filter, the number of computations is reduced by up to a factor of 10.5. This is verified below:

$$\frac{N \times M \times R \times C}{N \times M \times (R + C)} = \frac{R \times C}{(R + C)} = \frac{21 \times 21}{(21 + 21)} = \frac{441}{42} = 10.5$$

4. Unsharp Masking

4.1. Explain how the unsharp masking filter is implemented.
The image was loaded and Matlab, by default, assigned the image the datatype that was native to the file, uint8. The datatype was then cast as a double because some operations are not implemented for uint8. The three colour channels of this image were extracted, and the low pass filter was instantiated. This was accomplished using the Gaussian Filter function from Section 3 with a variance of 2.5^2, a size of 15 and the flag set to separable. This filter was then applied to each of the colour channels. The vertical and horizontal filter masks were applied separately because the number of computations can be reduced enormously by applying the filters separately, thus reducing the total execution time of the filter. The three colour channels were then concatenated back together to give the output from the low pass filter. This output was then subtracted from the original image and multiplied by a gain of 2.5. This gives the fraction of high pass information that should be added back into the image. This fraction is then added back into the image, and the new image is displayed in Figure 12.

4.2. By applying the system and observing the output image in colour, pick a value of f that gives the best-looking result. Write the value of f.

The value 2.5 was initially chosen for f, but this was incrementally reduced by 0.5 until the image appeared both sharper and not that much brighter as the original image. The final value chosen was 0.5. The image that corresponds to each of these incremented values is shown in Figure 8 – 12.

Figure 7. Original Image

Figure 8. f = 0.5

Figure 9. f = 1.0

Figure 10. f = 1.5

Figure 11. f = 2.0

Figure 12. f = 2.5

4.3 Describe the appearance of your output compared with the input.
It is evident from Figures 9 - 12 that the image appears much brighter than the original image. This is because the fraction of high pass information that was added back into the image was too high.

4.4. Justify the use of the selected value.
The value 0.5 was finally chosen because the image appeared both sharper and not much brighter than the original image. This is evident in Figure 8. This is because the fraction of high pass information that was added back into the image was not as high as in Figures 9 – 12.

5. Now process the pool.01.bmp image with these same settings. Is the result as good? Explain your findings.
The result is just as good as the previous result, but the difference is not as distinctive. This is because the original image for the pool.01.bmp image was much sharper than the original image for the sigmedia06907.tif image.

Figure 13. Original Image

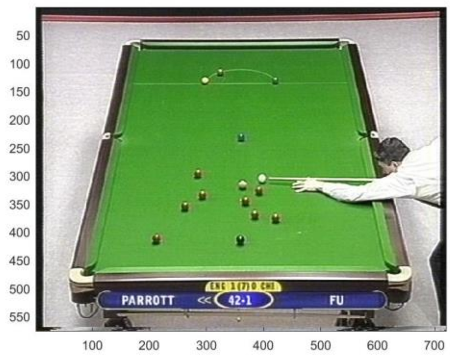

Figure 14. Sharpened Image