EE475 Homework 2

Aydın Uzun
2015401210

I. INTERPRETATION OF HISTOGRAMS

The most possible matching between images and histograms will be C-2,D-3,A-4 and B-1. First, all of four images are gray scale images, which also means that there are 256 bins at corresponding histograms. C matches with 2, because the range of pixel values is not so wide. Roughly speaking only three quantization levels are visible on C and looking at histogram number 2, there's a perfect mathcing. There are more or less three small ranges confirming our visual judgement. Looking at D one can see the widest range of pixel values on gray scale. Looking at histograms, histogram number 3 is a perfect match for D. Because it has the widest range among all other histograms. Looking at B, one should expect a histogram, which has not a wide range. Because the pixels on B should have similar quantization levels. Our visual judgement says that. Looking at histograms to find a match for image B, histogram number 1 would be my choice. It does not have a wide range. There is a peak at the histogram, as one should expect because most of the pixels on the image have the same quantization level. Besides, A is a perfect match for histogram number 4, because it should have a wide range, but not as wide as D. One can seperate white and black colors easily on A. But the black and white parts of the image should have similar quantization levels between them. Histogram number 4 fulfils our expectation.

II. HOW TO QUANTIZE IMAGES

A.

Spikes up the left or right edge indicate "clipping" of that tone and loss of detail in that area. Clipped areas are often unrecoverable.

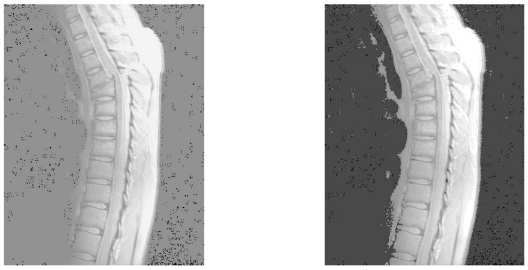
III. HISTOGRAM EQUALIZATION

My histogram equalization histogram is called *myhisteq()*. My function takes as input an uint8 valued image array. If it is not a gray scale image first it converts the image to its gray scale equivalent. Then the algorithm calculates its probability distribution function. After that the cumulative distribution function is calculated. After some other manipulations and roundings uint8 valued output image is created which enhances the contrast. The results of my own histogram equalization function and the buitin *histeq()* are shown on figure 1 and figure 2. On Figure

(a) output of *myhisteq.m* (b) output of *histeq.m*

Fig. 1: equalized *lumbercamp.jpg*

1 one can not see any difference between the outputs of buit-in *histeq()* and the implemented *myhisteq()* functions. But this is not the case on figure 2. There is a gray level difference



(a) output of *myhisteq.m* (b) output of *histeq.m*

Fig. 2: equalized *Gonzalez_Fractured spine.jpg*

between the background colors. But there is not a big difference between the spines seen on figure 2. The outputs of *adapthisteq()*

(a) *lumbercamp.jpg* (b) *Gonzalez_Fractured spine.jpg*

Fig. 3: output of *adapthisteq.m*

function are shown on figure 3. Compared to figure 1 and 2 it's clear that the *adapthisteq()* function performs much better than *histeq()* function. To find the reason behind that one should read the help page of *adapthisteq()* on MATLAB. *adapthisteq()* is a contrast limited adaptive histogram equalization function. Unlike *histeq()*, it operates on small data regions, rather than the entire image. Hence it provides better contrast enhancing. But it has also disadvantages. For example it does not work effectively.

IV. COLOR IMAGE ENHANCEMENT

The first step is to determine the limits over which image intensity values will be extended. These lower and upper limits will be called a and b, respectively. I pick a= 0 and b=255, because for standard 8-bit grayscale pictures, these limits are usually 0 and 255. Next, the histogram of the original image is examined to determine the value limits (lower = c, upper = d) in the unmodified picture. The c and d values are calculated with nested *min()* and *max()* functions respectively. All of these values are recorded into variables in workspace so that one can check and change easily. The rest is easy. I implemented the given equation and calculated the new contrast stretched arrays. The visualisations of all of them can be seen on figure 4 and 5. If the original range covers the full possible set of values, which is my case on *kugu.jpg* seen on figure 4, straightforward contrast stretching will achieve nothing. There's no big difference between 4.a, 4.b and 4.c.

(a) *kugu.jpg*

(b) *contrast stretched kugu.jpg*

(c) *HSV stretched kugu.jpg*

Fig. 4: color image enhancement *kugu.jpg*

I said that, if the original range covers the full possible set of values, straightforward contrast stretching will achieve nothing. But something different happens, when most of the image data is contained within a restricted range, which is my case on seen on figure 5. One should expect two bigger ranges on figure 5. Original values lying outside the bigger range are set to the appropriate limit of extended output range. In our example this effect is the cause of 5.2 looking bluish.
The contrast of an image is a measure of its dynamic range in other words the ”spread” of its histogram. The dynamic range of an image is defined to be the entire range of intensity values contained within an image, or put a simpler way, the maximum pixel value minus the minimum pixel value. Contrast stretching attemppts to improve an image by stretching the range of intensity values it contains to make full use of possible values.

Contrast stretchment in HSV color space results it better enhancements. Because in HSV color space you directly deal with intensity component and leave the color components alone. This is an improvement. Most of time you are not interested in color components.

(a) *lumbercamp.jpg*

(b) *Gonzalez_Fractured spine.jpg*

(c) *Gonzalez_Fractured spine.jpg*

Fig. 5: output of *adapthisteq.m*

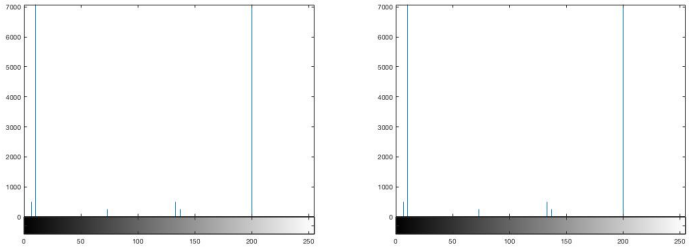
V. Effects of Filtering on Histograms

A.

Histograms of these two different images are same because they both contain same number of black and white pixels. Looking at histogram sketches one can realize that half of the pixels are located at 0. level and half of the pizels are located at 255. level.

B.

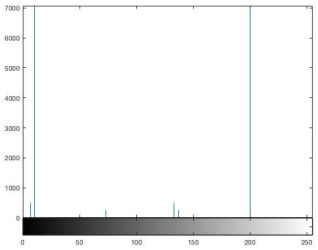
The number of boundary points between the black and white regions is much larger on *Checkerboard_fine.png*. When the images are blurred, the boundary points will give rise to a larger number of different values for *Checkerboard_fine.png*, so the histograms of the two blurred images will be different.



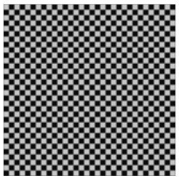
(a) *Checkerboard_gross.png* (b) *Checkerboard_fine.png*

Fig. 6: output of scaled *imhist.m* functions

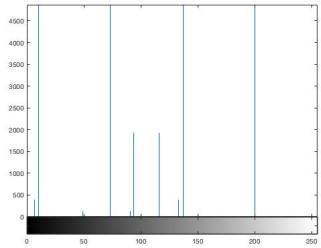
(a) *Checkerboard_gross.png*



(b) histogram of *Checkerboard_gross.png*



(c) *Checkerboard_fine.png*



(d) histogram of *Checkerboard_fine.png*

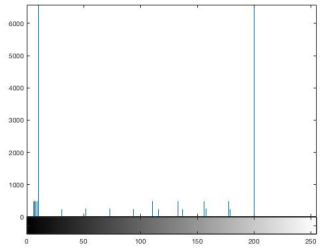
Fig. 7: blurred usinf 3x3 averaging filter

C.

Compared to 3x3 averaging filter, 9x9 box filter results in more blurring. The reasoning is the same as mentioned in b).

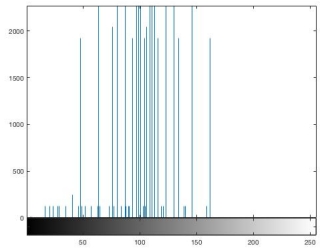
VI. HISTOGRAM DISTANCE

(a) *Checkerboard_gross.png*



(b) histogram of *Checkerboard_gross.png*

(c) *Checkerboard_fine.png*



(d) histogram of *Checkerboard_fine.png*

Fig. 8: blurred using 9x9 averaging filter