# EE475 Homework 1

Aydın Uzun
2015401210

## I. IMAGE READ & WRITE; HISTOGRAM PLOT

The file *COLOR_Birds.bmp* is read by function *imread()* and stored in array **I** and displayed by *imshow()*. Then *imwrite()* function is used to store it with the filename extension ".png". Stored image with extension *.png* is shown on figure 1. Then *iminfo()* function is used to see the contents of the written file. Its output is a 1x1 struct with 38 fields. Most of the fields are empty. The most important information one can get is the following: *Filesize = 563345, Format = 'png',, Width = 768, Height = 512, Bitdepth = 24, Colortype = 'truecolor'* . Besides, the information about *Filename* and *FileModDate* can be different for different users.

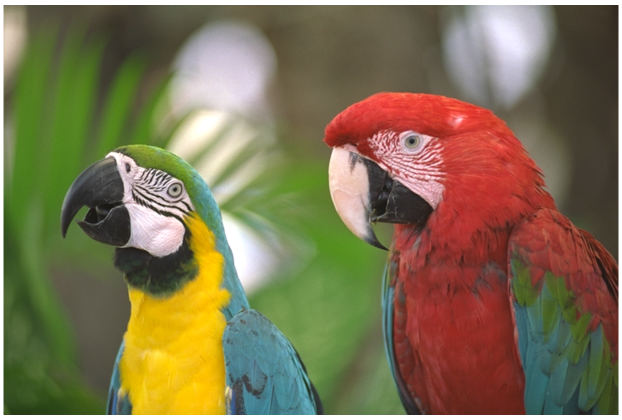


Fig. 1: COLOR_Birds.png

### *A.*

**I** is a 512x768x3 uint8 valued array. First, R,G and B components are extracted into 3 different 512x768 arrays. To take the dimensions of arrays *size()* function is used. Then an equal sized new array called *null* is created and concatenated with R,G,B components using function *cat()*. The new arrays which contains only R,G,B components each and the older one are displayed on figure 2.

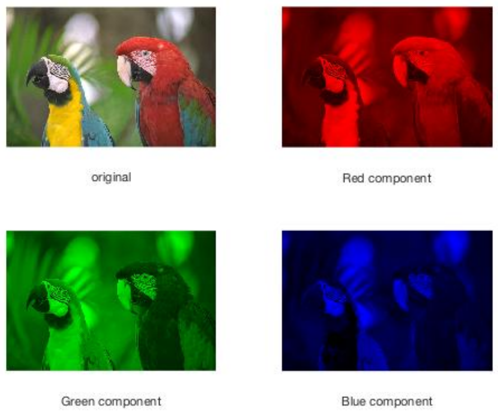


Fig. 2: RGB color components

### *B.*

The average of extracted R,G and B components is taken to achieve the demanded image. The average array is shown with *imshow()* function and saved with *imwrite()* function to a file called *GRAY_Birds.bmp*. *GRAY_Birds.bmp* is shown on figure 3.

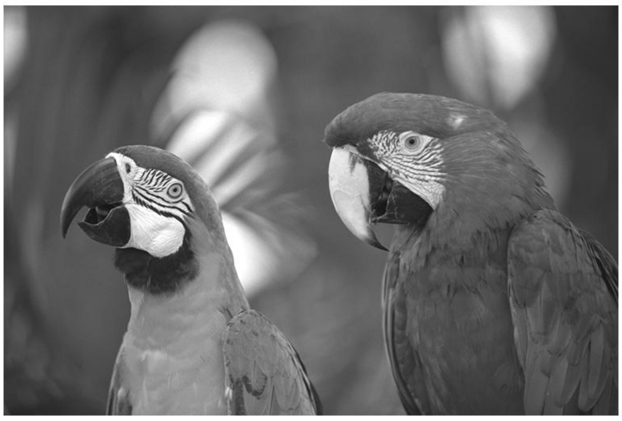


Fig. 3: GRAY_Birds.bmp

### *C.*

*imhist()* function is used to plot histograms of the R,G,B and gray-level components. The plots are shown on figure 4. One can conclude from the figure 4 that number of pixels ,whose red color values and green values are high, is higher than the number of pixels whose blue color values are high. This means that one can observe red and green more than blue. This is true, because blue can be seen only on some details. However, green is the backgorund color and red is almost entire color of one bird. One more conclusion will be about the histogram of gray level component. The histogram of gray level component is the mean of other three components. As expected white and black points on the original image are white and black respectively. The colorful pixels are scaled in between.

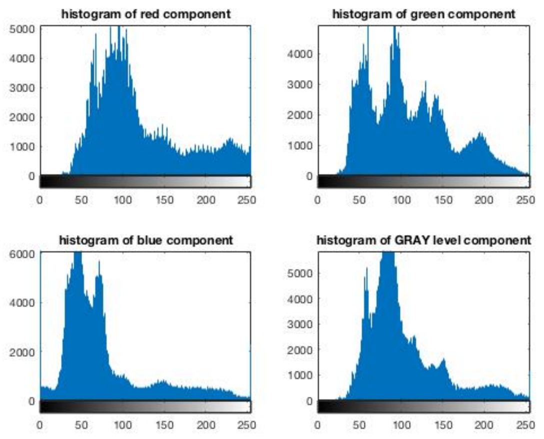
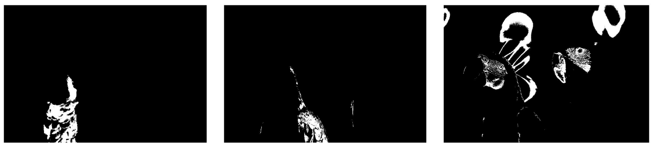


Fig. 4: histograms

### *D.*

The R,G,B color values with given conditions indicate clearly yellow content. Weak presence of the blue component corresponds to yellow. So one can expect, the yellow points on the original image correspond to logical 1, in other words 255 in binary image, in other words white in color. Comparing Figure 2 and Figure 5.a this is true.



(a) d (b) e (c) f

Fig. 5: binary images corresponding to parts d,e and f

### *E.*

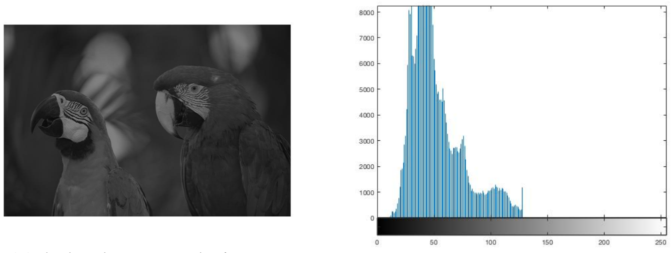
Combination of blue and green,what the given condition says, indicates cyan. Comparing Figure 2 and Figure 5.b one can confirm this claim.

### *F.*

Absence of green corresponds to magenta. But the given condition does not state the absence of green exactly. Therefore one should not expect that white points on the binary image correspond to magenta on original image.

### *G.*

The histogram of halved gray scale image looks like the histogram of gray level component shown on Figure 4. Although they are similar in shape, the pixel values are halved. The maximum gray level value is not 255 anymore causing a decrease in brightness of the image comparing with Figure 3.



(a) halved gray scale image

(b) halved gray scale histogram

Fig. 6: halved gray scale

### *H.*

According to my visual judgement the reddest point on the image can be found around the head of the bird standing on the right side. To check my assumption, my algorithm to find the reddest pixel is to find a pixel with maximum R component and minimum G and B components. My conditions for such a point is R¿220, G¡50 and B¡50. Only two pixels provide these conditions , which have the following coordinates [133,553] and [135,587]. These two pixels are located on the head of bird standing on the right side.

## II. Average Optical Density

### *A.*

Using the given algorithm the average optical density values are calculated using functions *sum()* and *size()*.

$$AODofsubject06.centerlight = 189.8649 \tag{1}$$

$$AODofsubject06.noglasses = 166.6666 \tag{2}$$

$$AODofBaby.png = 66.3749 \tag{3}$$

### *B.*

Face images cover the whole 0-255 range better than baby image. It is an indicator for contrast of the image. One can easily say that baby image has a lower contrast comparing with face images.

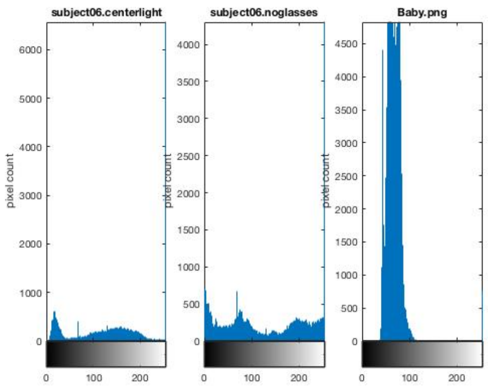


Fig. 7: histograms of face images

### *C.*

c) I found the constants c empirical. The constants are the following.

$$cforsubject06.centerlight = -100 \tag{4}$$

$$cforsubject06.noglasses = -78 \tag{5}$$

$$cforBaby.png = +34 \tag{6}$$

Comparing the histograms of adjusted images seen on Figure 9 and original images seen on Figure 7 one can conclude that what we do by adding constants is just shifting of histograms. Negative c means shifting to left, positive c means shifting to right. However, there are some data losses due to overflow and overflow. For example for image *subject06.centerlight* the maximum gray level value is 155. This also means that the brightness of the image were decreased due to negative c. On the other hand, positive c means an increase in brightness, which is the case on image *Baby.png*.

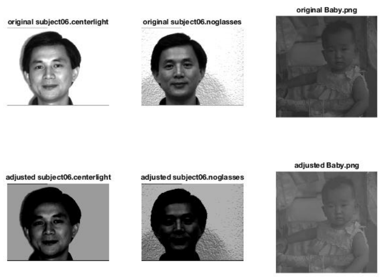

original subject06.centerlight
original subject06.noglasses
original Baby.png
adjusted subject06.centerlight
adjusted subject06.noglasses
adjusted Baby.png

Fig. 8: Original image and adjusted images

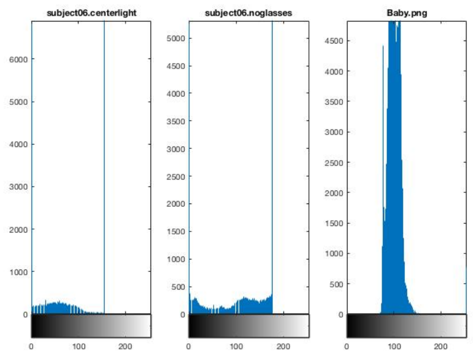


Fig. 9: adjusted histograms

*D.*

By multplying the image arrays by a constant a, we scale its pixel values. It's expected that the histograms of multiplied image and original image are very similar in shape, if a is smaller than 1. When a is bigger than 1, overflow is an issue to consider and may cause huge data losses. The pixel range of multplied images will be very different. Because multiplying an image array by 0.5, the maximum pixel value is decreased to 127. This effect can be seen on Figure 10. The brightness of images multplied by 0.5 are decreased. On the other hand, the brightness of images multiplied by 2 is increased. One more comment on the changed properties of images will be about their pixel range. The pixel range of images multplied by 0.5 was decreased clearly causing a decrease in contrast of images.

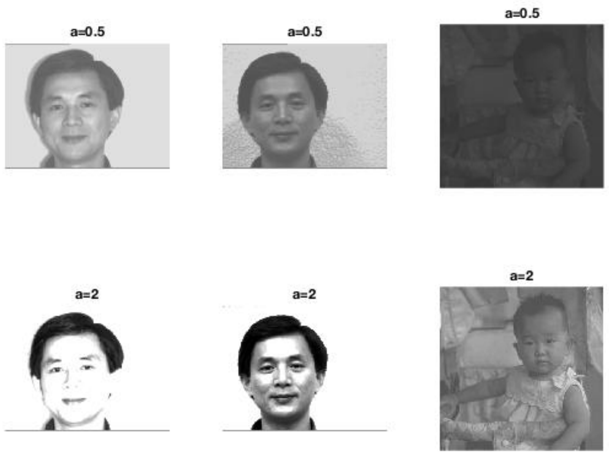


Fig. 10: adjusted images for different values of a

## III. Get a Feeling About the Amount of Pixels

*A.*

The data processing rate can be calculated with the formula

$$dpr(bps) = fr(frames/sec) \times br\_in\_bpf(bit/frames) \tag{7}$$

$$br\_in\_bpf(bit/frames) = sof \times 3 \times \log_2(ql) \tag{8}$$

The abbreviations used in formula are following: dpr = *data processing rate*, fr= *frame rate*, br_in_bpf=*bit rate in bit per frame*, sof=*size of frame*, ql = *quantization levels*. Besides, 3 stands for RGB. Using the given formula the data processing rate can be calculated as 184.32 Mbps.

*B.*

Image processing rate can be calculated with the formulas

$$dpr(bps) = fr(frames/sec) \times br\_in\_bpf(bit/frames) \tag{9}$$

$$br\_in\_bpf(bit/frames) = sof \times \log_2(ql) \tag{10}$$

$$sof = 256 \times 256 \times 256 \tag{11}$$

$$ql = 65536 \tag{12}$$

$$fr = 10 frame/sec \tag{13}$$

Using the given formula the image processing rate can be calculated as 2.6844 Gbps.

*C.*

The amount of data to be processed per day can be calculated using the following formulas.

$$size = 256 \times 256 \tag{14}$$

$$quantization\ levels = 16 \tag{15}$$

$$total\ bit\ number\ at\ each\ frame = size \times quantizationlevels \tag{16}$$

$$seconds\ in\ a\ day \ = \ 24 \times 60 \times 60 \tag{17}$$

$$aodppd \ = \ seconds\ in\ a\ day \ \times 1000 \times 24 \times tbnaef \tag{18}$$

The abbreviations used in formulas are following: aodppd = *amount of data processed per day*, tbnaef = *total bit number at each frame*. Using these formulas the amount of data processed per day can be calculated as approximately 271.75 TB.

*D.*

The storage needed can be calculated with the following formulas

$$storage = number\ of\ frames \times RGB\ components(bits) \times color\ depth \tag{19}$$

$$color\ depth = 12 \tag{20}$$

$$time\ for\ each\ frame = 1/25\ sec \tag{21}$$

$$numberofframes = twohoursinec \times time\ for\ each\ frame \tag{22}$$

$$RGB\ components = 2048 \times 1080 \times 3 \tag{23}$$

Using the given formula storage needed can be calculated as 1791.6 GB.

## IV. Pixels Varieties

*A.*

The answer to this question actually depends on your definition of birghtness. In order to detect which pixels emit the most amount of light, I find the pixels where the addition of R,G and B values is maximum. The maximum of addition of RGB values is 765. Among 1000x1500 pixels there are only 85 pixels which have this maximum value. According to my definition these pixels emit the most amount of light. Most of these pixels are located in the centre of the picture, as one should expect. One more algorithm to find pixels that emit most amount of light will be convert this image to gray scale image with function *rgb2gray()* and then find the points which have maximum value on gray scale.

### *B.*

One should expect that the brightest pixel is the closest point to range camera, and the darkest pixel is the farthest point to range camera. Making this interpretation the rest will be the same as previous question. But one important thing to take into account before using the algorithm mentioned in the previous question is the useless part of the image. The last 7-8 rows are just some numbers in the camera view. It has nothing to do with the distance of images. Therefore you should get rid of these pixels. I set all of them's RGB values to some number in the middle which might be 100. Doing this I guarantee that they don't harm my calculations. After this improvement I calculated the maximum RGB sum as 648 corresponding to pixel [118,169]. This point is the closest point to camera. [118,169] confirms also my visual judgement. The minimum RGB sum is 201 corresponding to [51,130]. This point is the farthest point to camera. But this point does not confirm my visual judgement.

### *C.*

If the image represents material density, the opaquest point will be the densest part and the least opaque point will be the most transparent part. From this view the rest will be the same as the previous questions. But to take into account is the region of interest. The least opaque (most transparent) point of skull image is point [64,25]. It is a point standing on left edge of skull image. When I use the same algorithm to detect the opaquest point, I get 8228 points which is too much. Most of these points do not stay on thr region of interest. Even in the region of interest there are many opaqest points. To see this fact better I convert the image into binary image. Doing this I detect easily that the most opaque points stay on the spherelike in the region of interest seen on figure 11. The least opaque (most transparent) points of breast image are points [220,71],[219,73],[214,110]. These points confirm my visual judgement. When I use the same algorithm to detect the opaquest point, I get 23033 points which is too much. This is the same problem like the problem at skull image. To see the opaquest points better I convert the brest image to binary image seen on figure 12. The most opaquest points stay on the edge of the breast.

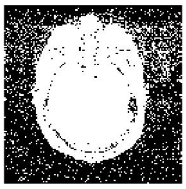


Fig. 11: binary image of skull used in order to detect opaquest points

Fig. 12: binary image of breast used in order to detect opaquest points

### *D.*

For a thermal image, the bright points indicate hot regions, and the dark points indicate cold regions. . From this view the rest will be the same as the previous questions. The hottest point on the thermal image will be the point [72,47], standing on the head of third person from above. However there are 267 coldest points. To see these points better I convert the image to a binary image. On figure 13, white points correspond to colder regions.

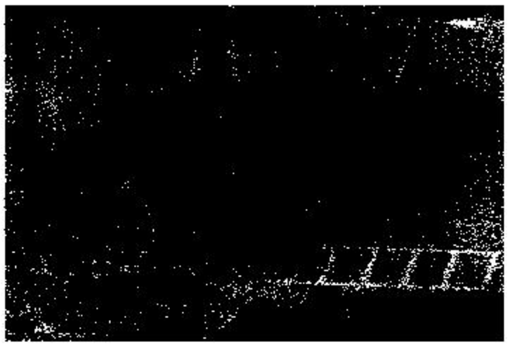


Fig. 13: binary image of thermal image used in order to detect coldest points

I conclude that the rails on the image are the coldest points on the image.

### *E.*

First, I take the difference between two taxi images. I expect that when the difference in gray scale is high between pixels of two images, the pixels move faster than other points. In order to avoid underflow I take the absolute value of the difference array. The new image can be seen on figure 14. In my opinion, white points move faster. The weak spots of this algorithm come to exist when the gray scale level difference between two different points of same taxi is high, which will be the case on white taxis in gray scale.

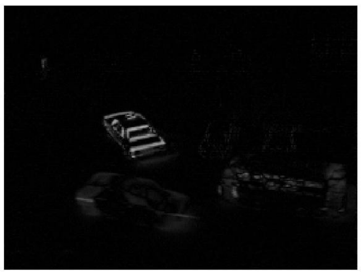


Fig. 14: absolute difference image used in order to detect fastest changing points

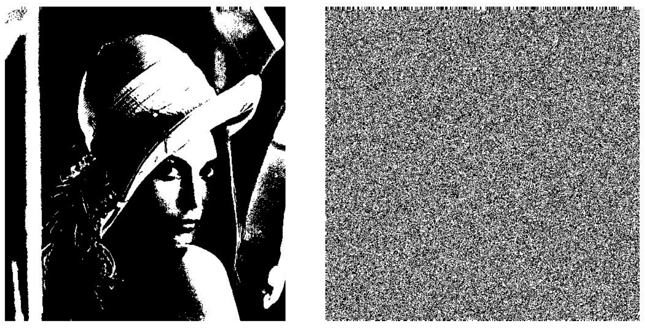
## V. IMAGE QUANTIZATION

### *A.*

The MSB binarized lena can be seen on Figure 15.a. It's clear that lena is still perceptible. The algoirthm to MSB binarize the image is simple, I only used nested for loops.

## *B.*

The LSB binarized lena can be seen on Figure 15.b. It's clear that lena is not perceptible anymore. All the information about lena has gone. The reason for the information loss is clear. Imagine two pixels side by side having gray scale values 140 and 141. Now one of these two pixels is shifted to 0 and the other one is shifted to 255.



(a) binary MSB plane image (b) binary LSB plane image

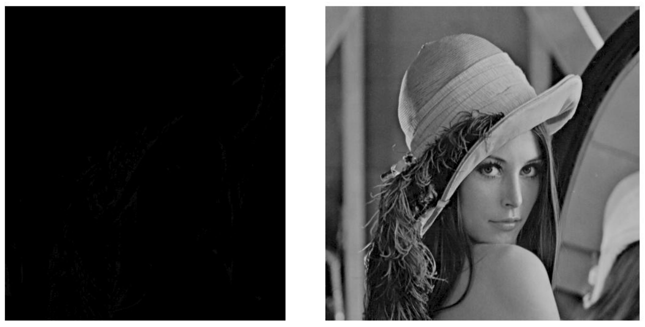
Fig. 15: Lena with different binary plane images

## *C.*

In figure 16.a one can see the effect of setting bits of order 5-8 to zero. It's just a dark image. There's a huge amount of data loss. Because now the quantization levels between 16 and 255 are not visible just dark. One can see only 15 quantization levels which is very small amount of data. By the way these quantization levels represent darker parts confirming our visual judgement.

## *D.*

In figure 17.b one can see the effect of setting bits of order 1-4 to zero. Only 15 quantization levels are discarded. As expected, there is not too much data loss. Lena is still perceptible.



(a) bits of order 5-8 zero (b) bits of order 1-4 zero

Fig. 16: Lena with different binary plane images