Assignment 1

DIGITAL IMAGE AND VIDEO PROCESSING

Arnav Malhotra | 17317424 | 12/02/2019

Q1. Create an .m script which loads a raw image and then assigns that image to the matrix pic. Since the image is just raw, 8 bits per pel in a single file with no header, you have to know beforehand what size it is. That is a pain in general since pictures can be any size. This is the reason why most image formats in common use e.g. .bmp, .tif, .jpeg, have header information which included the size of the image. Here is an example that you can use for loading the lenna.256 image. Note that in Matlab the % symbol is used to indicate a comment used to explain the code.

A1.

```
hres = 256; vres = 256;
name = ['D:\Docs\Study\Digital Image and
Video\Processing\image_stills\lenna.256'];
fin = fopen(name,'rb'); pic = double(fread(fin,[256 256])');
figure(1);image(pic);colormap(gray(256));shg;
```

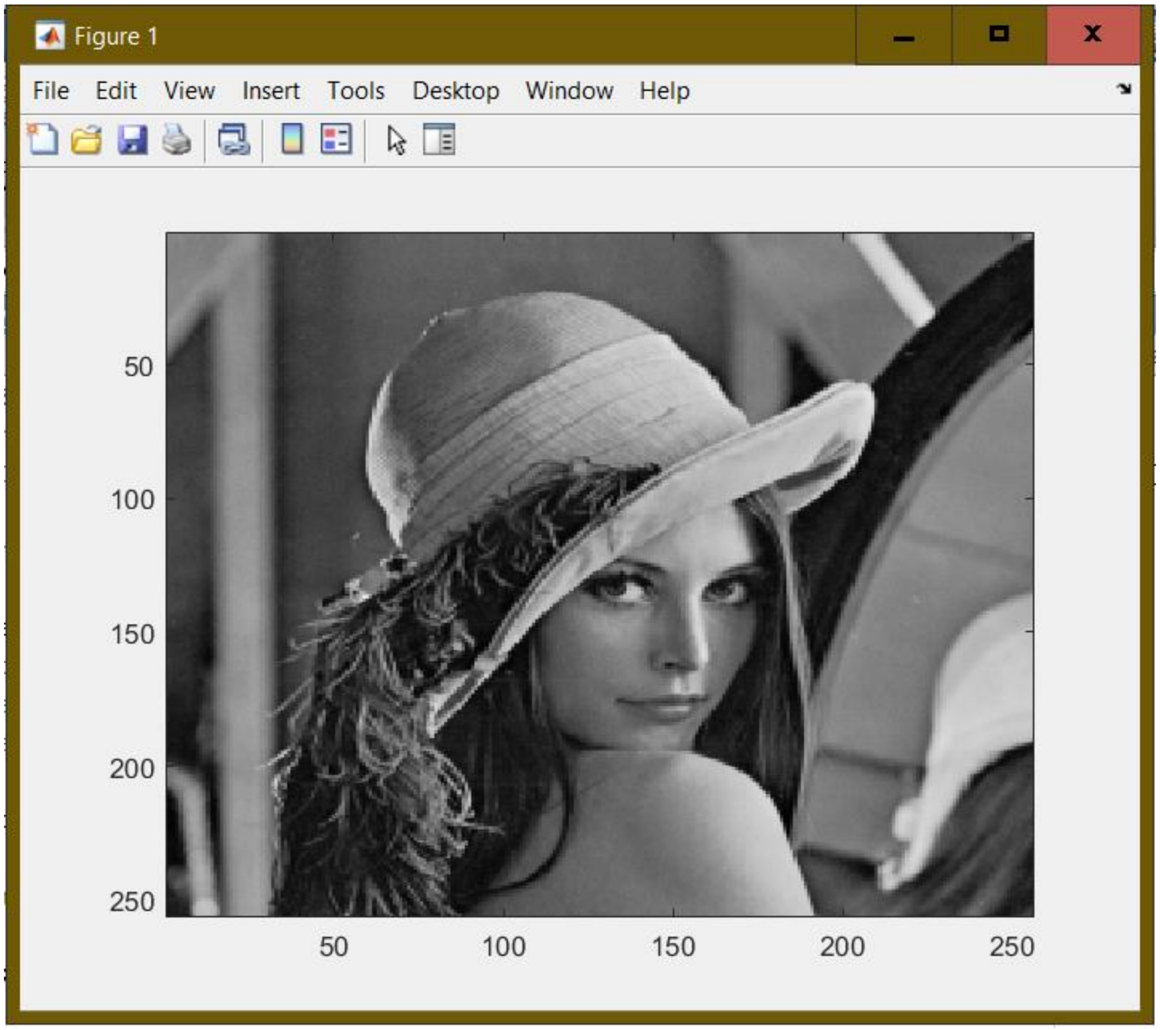

Figure 1: Reading a raw image

Q2. Add a line in your .m file which adds 128 to the image using “newpic = pic+128;”. Now display the resulting picture in Figure window 2. Explain what you see compared to pic. Why has much of the picture turned white?

A2. By adding 128 to “pic”, we have essentially added 128 to value of each pixel of the grayscale image which is between 0 and 255. Since 0 represents black and 255 represents white, adding 128 makes the image whiter (as shown in Figure 2).

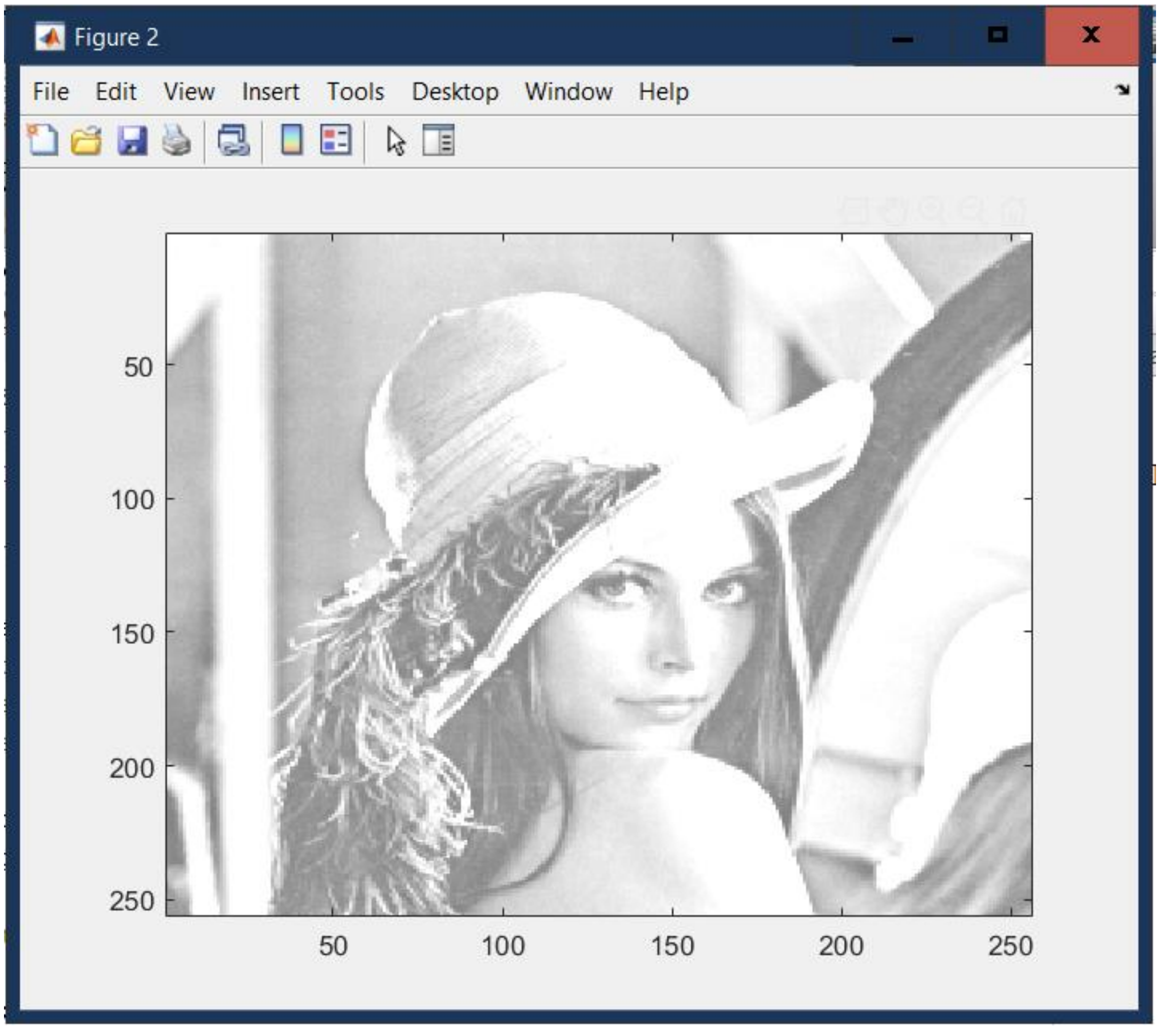

Figure 2: Making image whiter by a certain value

Q3. Add a line in your .m file which subtracts 128 from the image using “newpic = pic-128;”. Now display the resulting picture in Figure window 3. Explain what you see compared to pic. Why has much of the picture turned black?

A3. Just like in Q2, by subtracting 128 from “pic”, we have essentially subtracted 128 from value of each pixel of the grayscale image which is between 0 and 255. Since 0 represents black and 255 represents white, subtracting 128 makes the image darker (as shown in Figure 3).

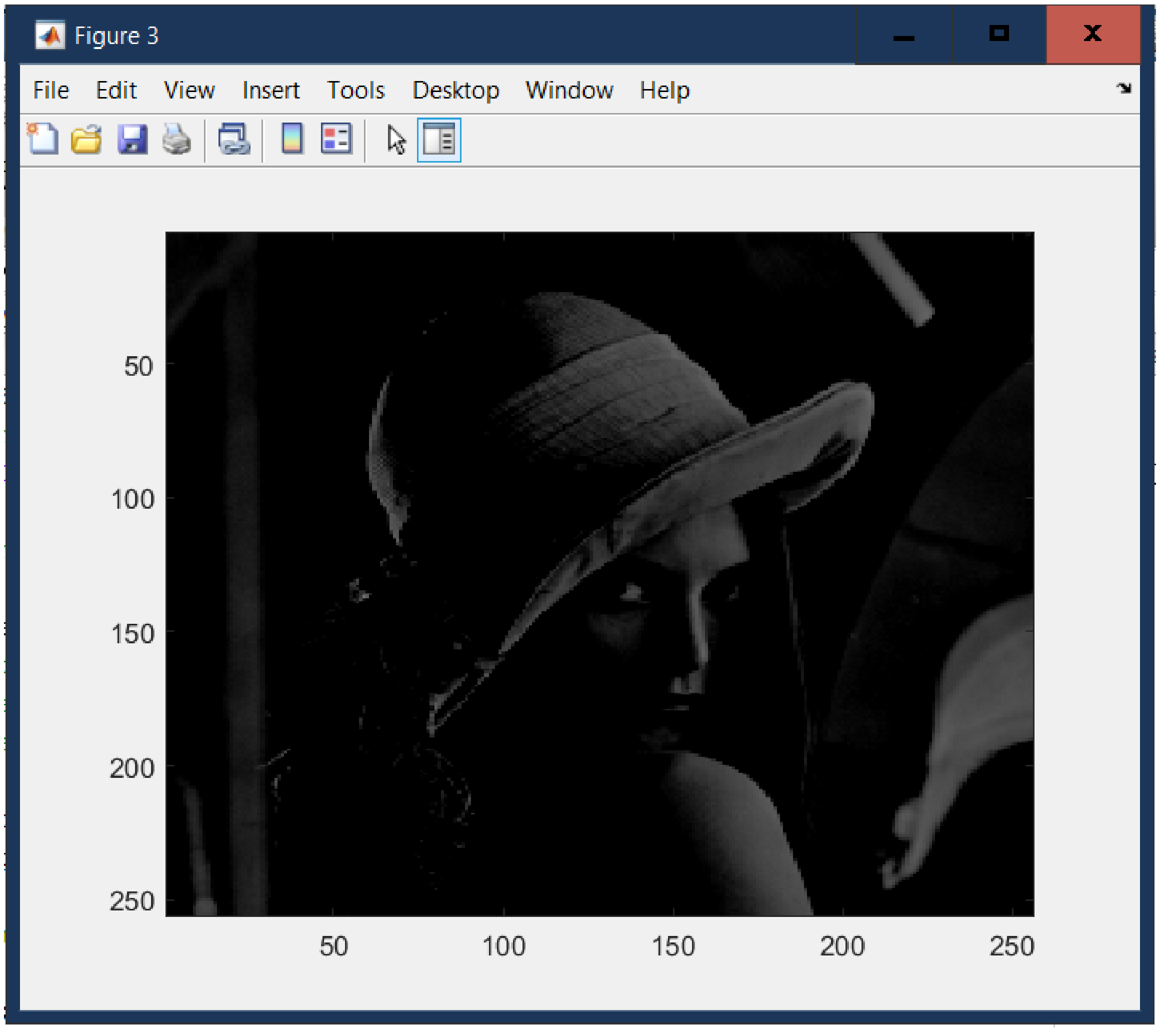

Figure 3: Making image darker by a certain value

Q4. Write a Matlab script that loads an image from a user defined filename, and generate the 2D histogram of the image. Use the picture sigmedia06907.tif, and assign it to "pic". Note that the inbuilt Matlab function imread can read in all common image formats like *bmp, jpeg, tiff etc*. Also note that pic now has 3 color planes, as Matlab assigns the Red pixels to "pic(:,:,1)". Green and Blue get assigned to the second and third planes respectively. We are only working on gray scale images here so the .m file will need to convert this RGB image to grayscale using the Matlab function rgb2gray. Use help rgb2gray to see how to use it. Type "size(pic)" in Matlab and explain the output.

A4.

```
%define the picture name
name = imread('D:\Docs\Study\Digital Image and Video
Processing\image_stills\4s1\lab1\Sigmedia06907.tif');
pic = rgb2gray(name);
figure(1);imshow(pic);shg;
figure(2);hist(pic(:),[0:255]);
```

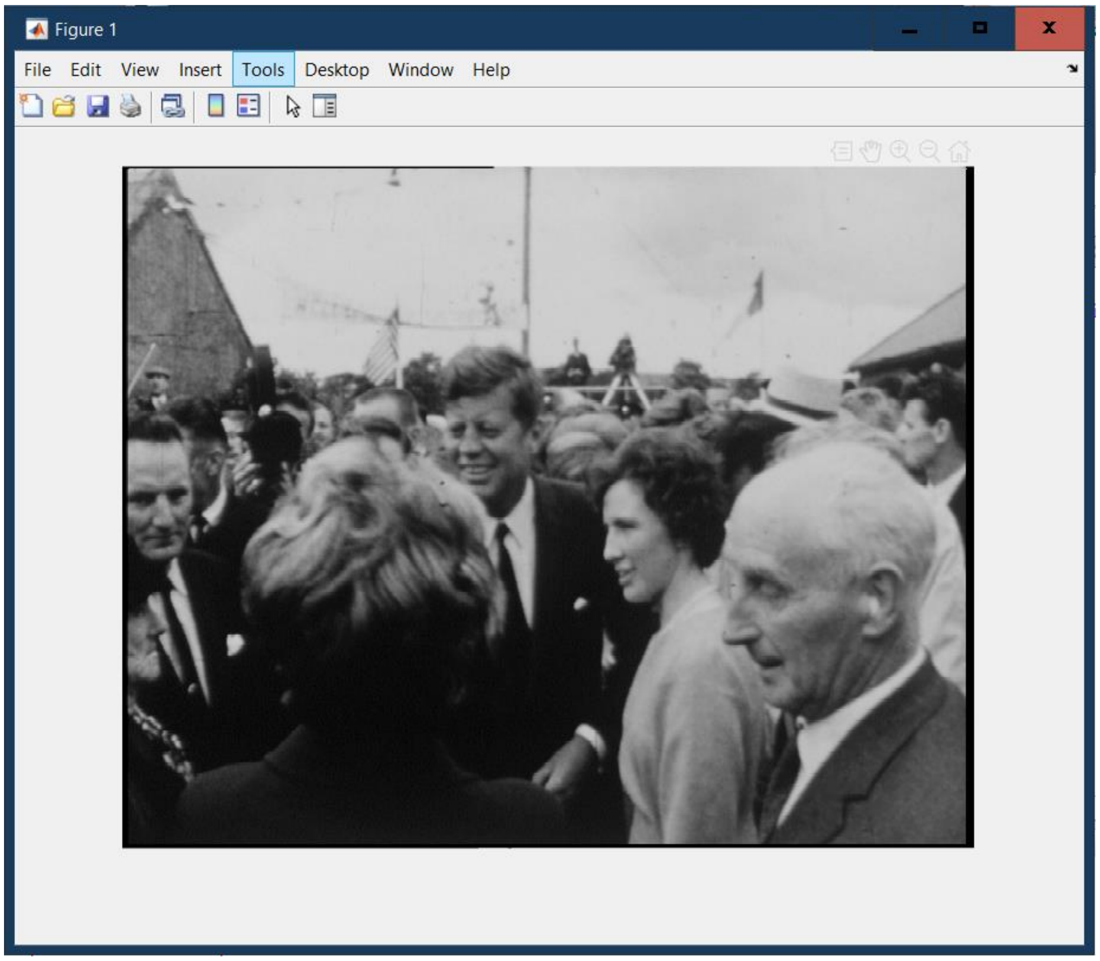

Figure 4: Reading a .tif image and converting it to greyscale

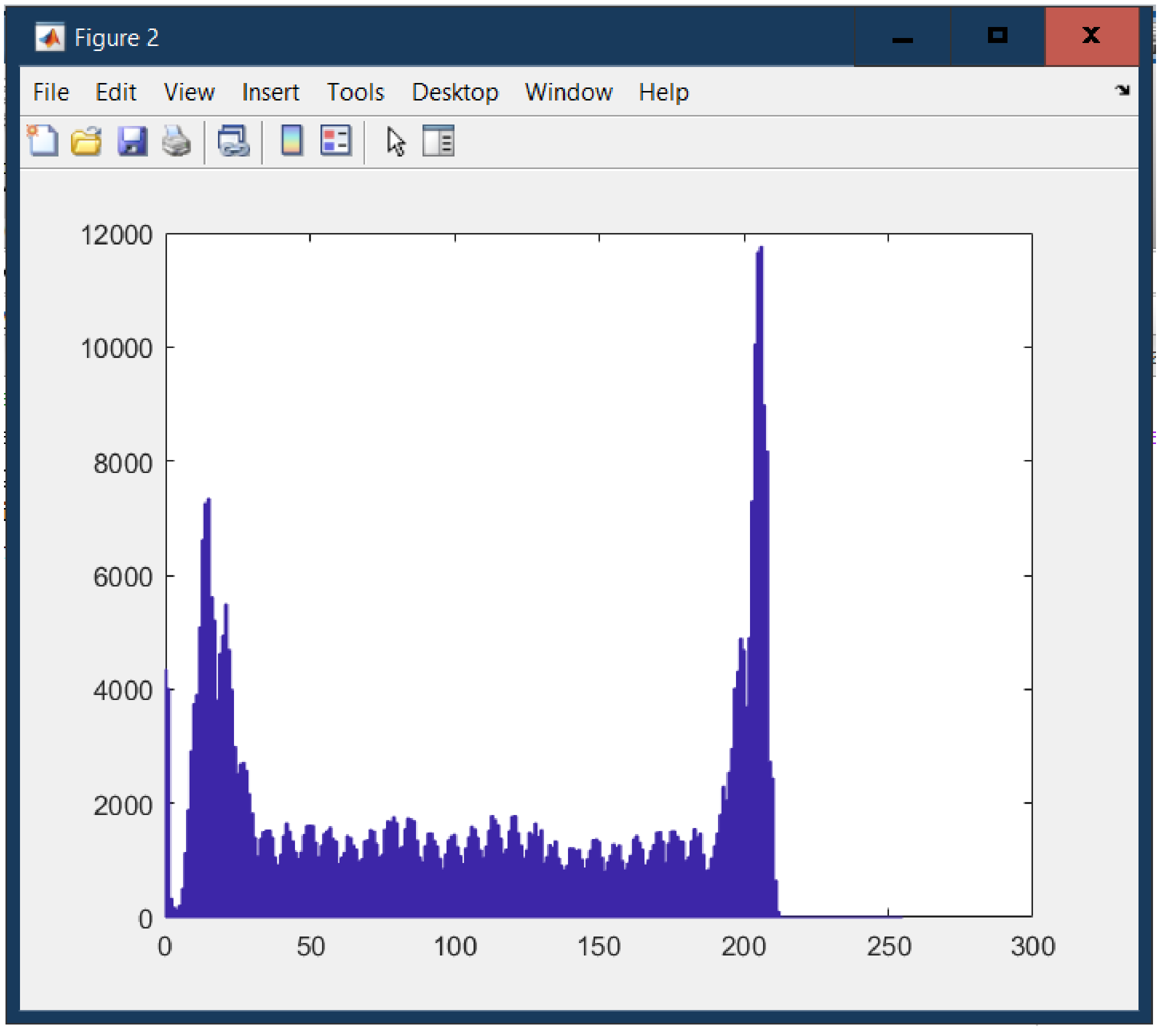

Figure 5: Histogram of Fig. 4

Using "size(pic)" command gives the following output-

```
ans =

   256   256
```

This represents the size of the image in terms of number of pixels along vertical axis and horizontal axis respectively. The convention seems different than typical (x,y) format as the values of the pixels are stored in matrices.

Q5. Change your script above to load the "pool.01.bmp" image. Use imread and your file should also work out the size of the image with [vres,hres]=size(picture).

A5.

```
%define the picture name
pic = imread('D:\Docs\Study\Digital Image and Video
Processing\image_stills\4s1\lab1\pool.01.bmp');
[vres,hres] = size(pic);
figure(1);imshow(pic);shg;
figure(2);hist(pic(:,:,1),[0:255]);
figure(3);hist(pic(:,:,2),[0:255]);
figure(4);hist(pic(:,:,3),[0:255]);
```

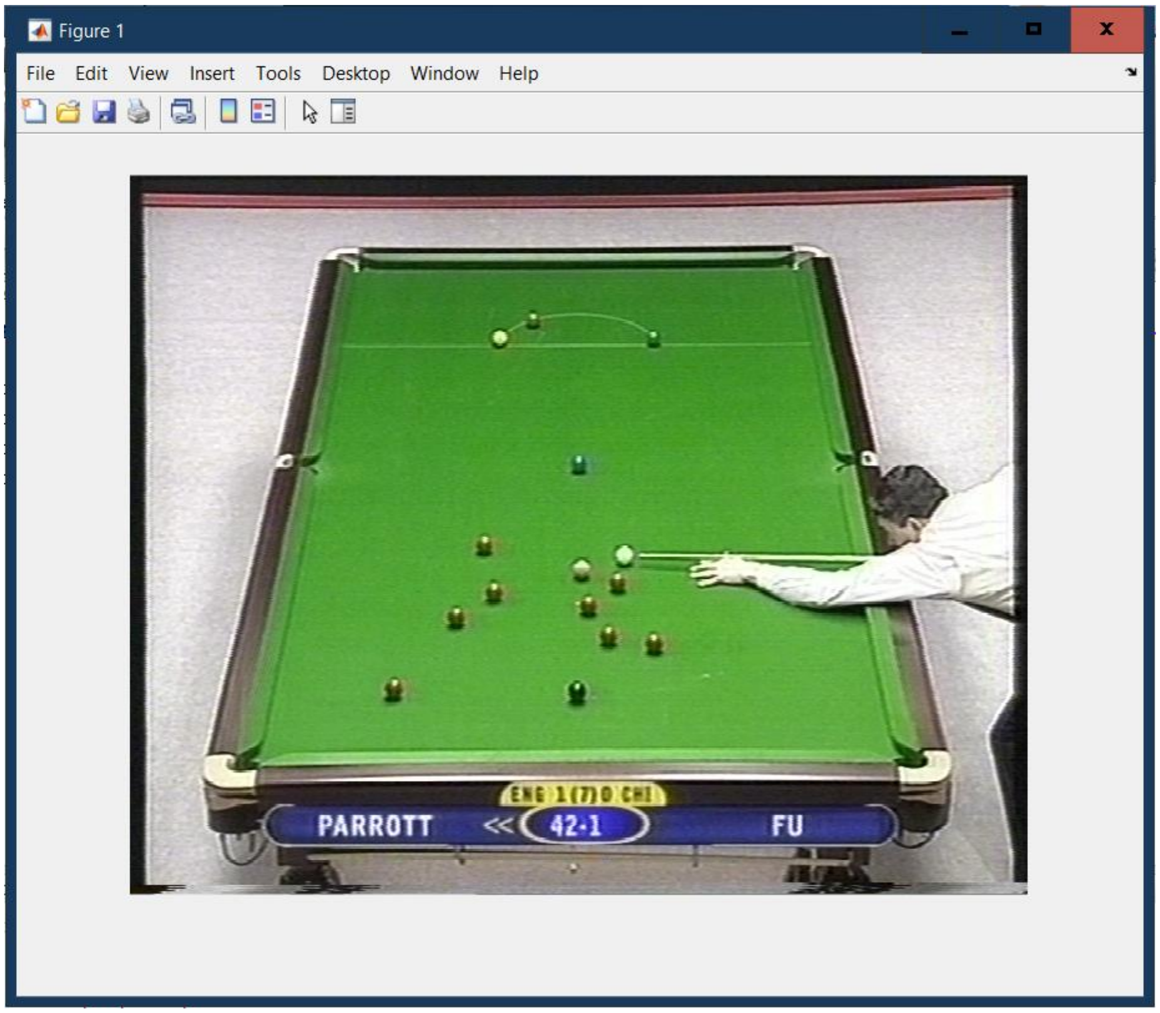

Figure 4: Reading a .bmp image

Q6. Plot the histograms of the R, G, B components of the image in three different figure windows. Recall that pic(:,:,1) is the first colour plane, pic(:,:,2) is the second. Put images of the 3 histograms into your report. Explain how the table region of the image manifests in the three histogram spaces.

A6. The following figures represent the histograms for all 3 channels of the image "pool.01.bmp"

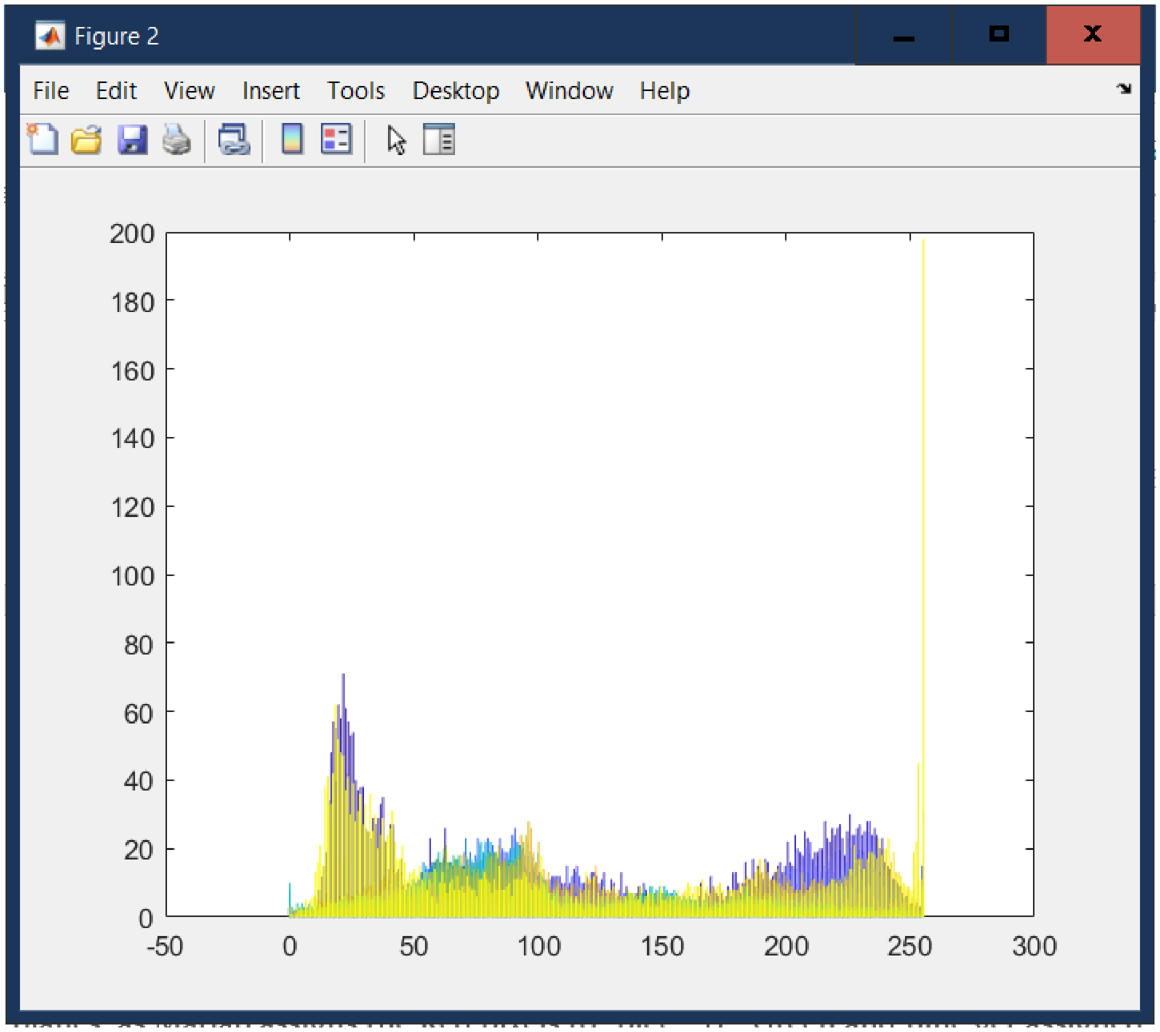

Figure 7: Red channel histogram of "pool.01.bmp"

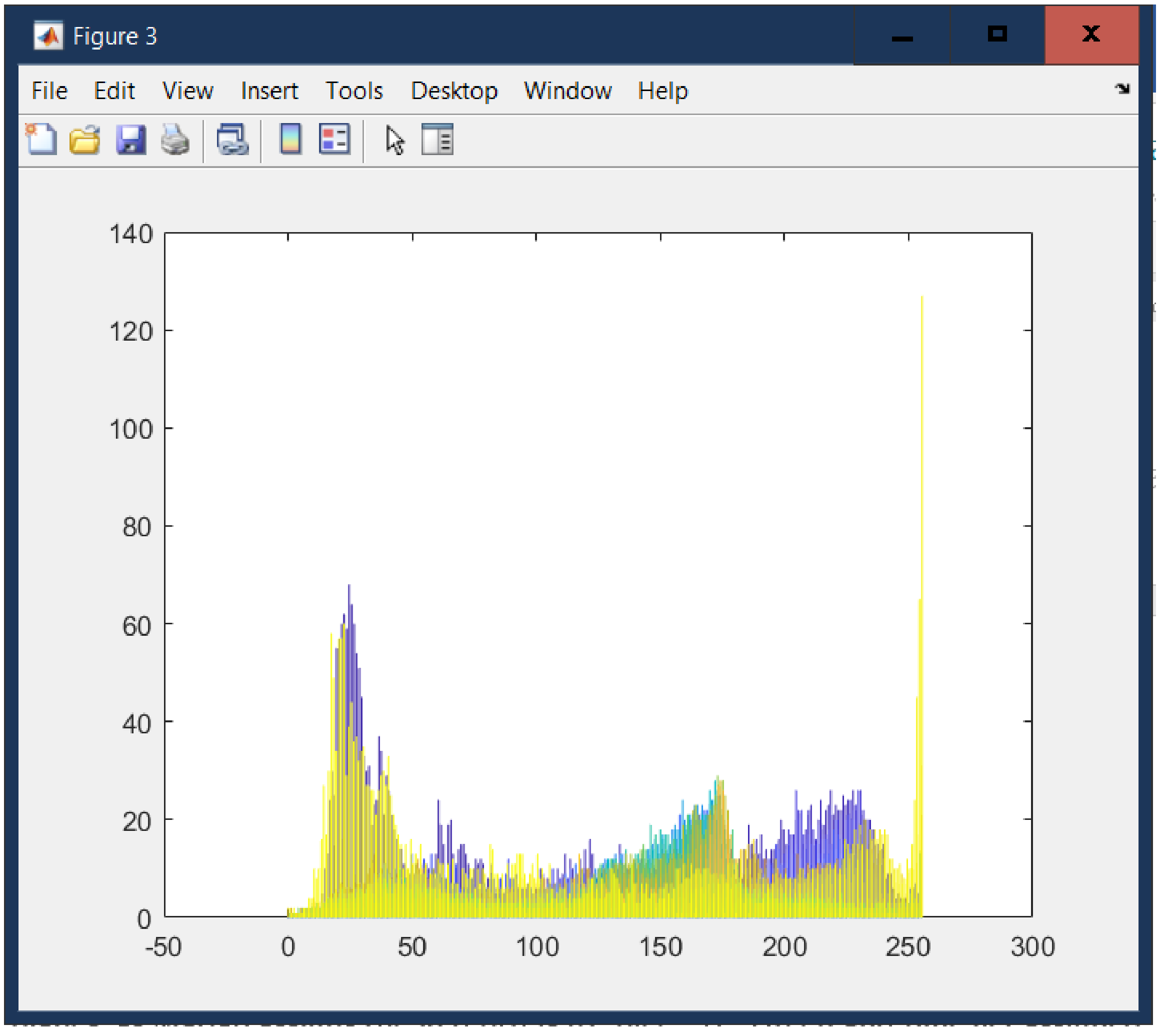

Figure 8: Green channel histogram of "pool.01.bmp"

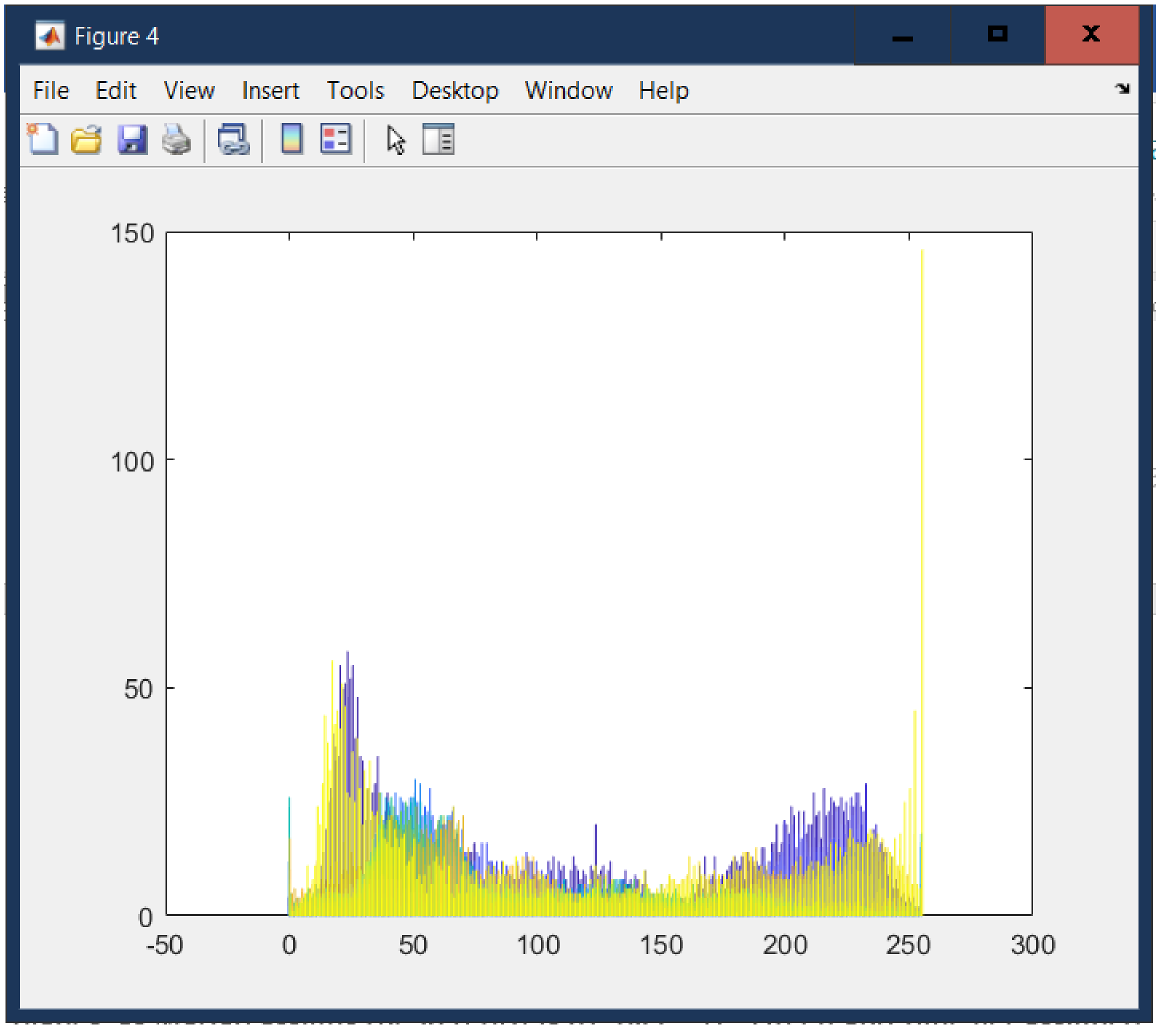

Figure 9: Blue channel histogram of "pool.01.bmp"

Since the table is green, we can say for sure that whichever high values of the histogram stand out in green channel when compared to red and blue would give us the values for the table in the green channel. On the other hand, the high number of values towards the lower end in case of red and blue channels can help identify the values for the pool table.

Q7. Write a script to read in the image and using your own judgement choose a suitable value of *t1* and *t2* for thresholding the intensity of the image so that mask contains pixels that are 1 where the nuclei exist. Satisfy yourself that you have achieved an acceptable segmentation of the image.

A7.

```
pic = imread('D:\Docs\Study\Digital Image and Video
Processing\image_stills\4s1\lab1\nuclei.02.png');
[vres,hres] = size(pic);
figure(1);hist(pic(:),[0:255]);
t1 = 43;
t2 = 255;

mask = pic(:,:,1);
mask = double(mask);
mask(mask < t1) = 0;
mask(mask < t2 & mask > t1) = 1;
mask(mask > t2) = 0;

figure;imshow(pic)
figure;imshow(mask)
```

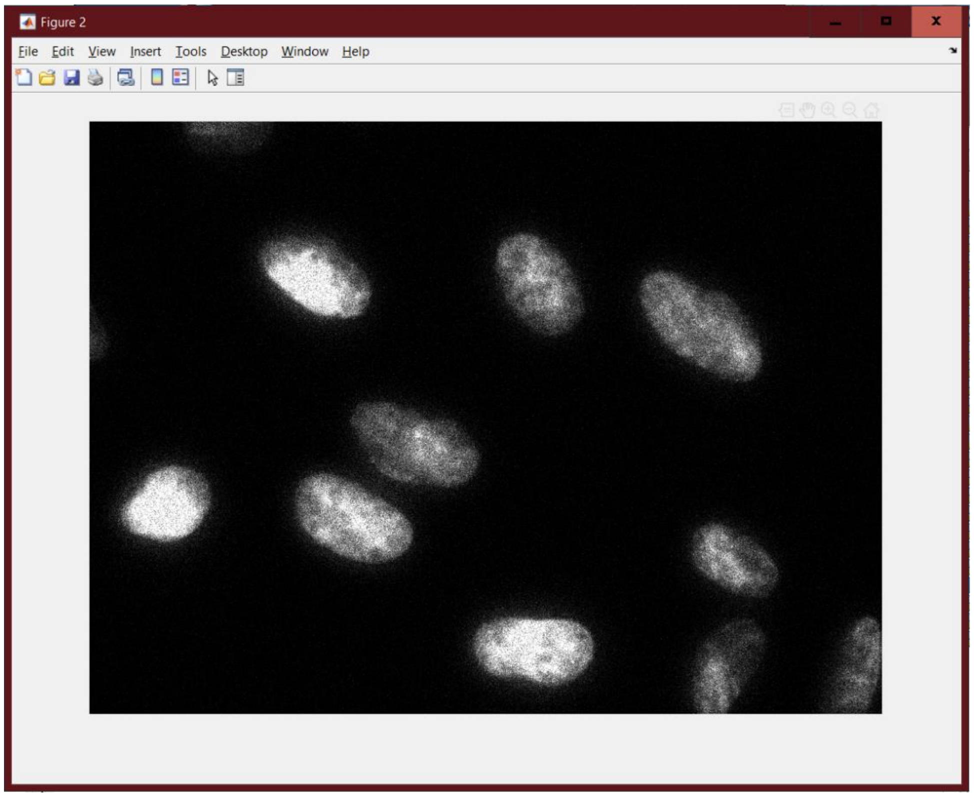

Figure 10: Reading .png image

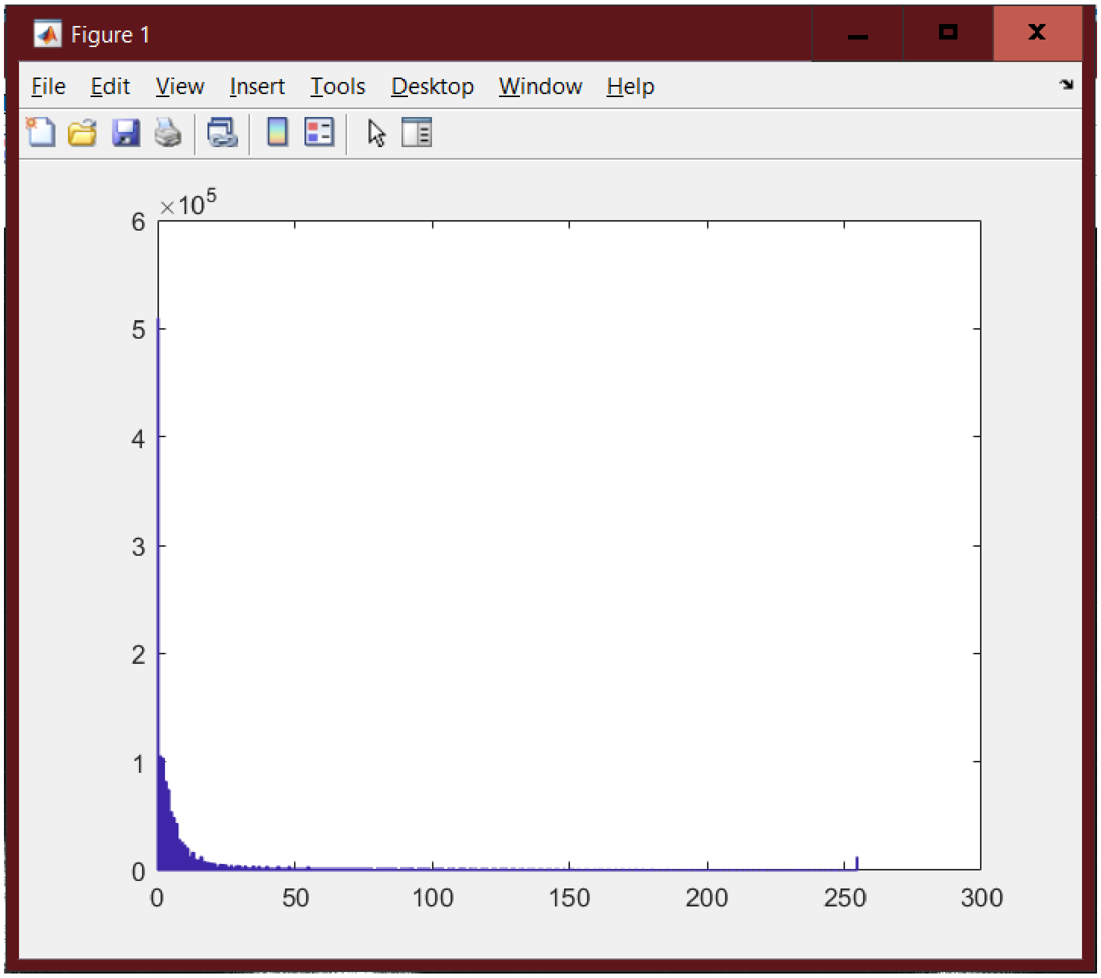

Figure 11: Histogram of Fig. 10

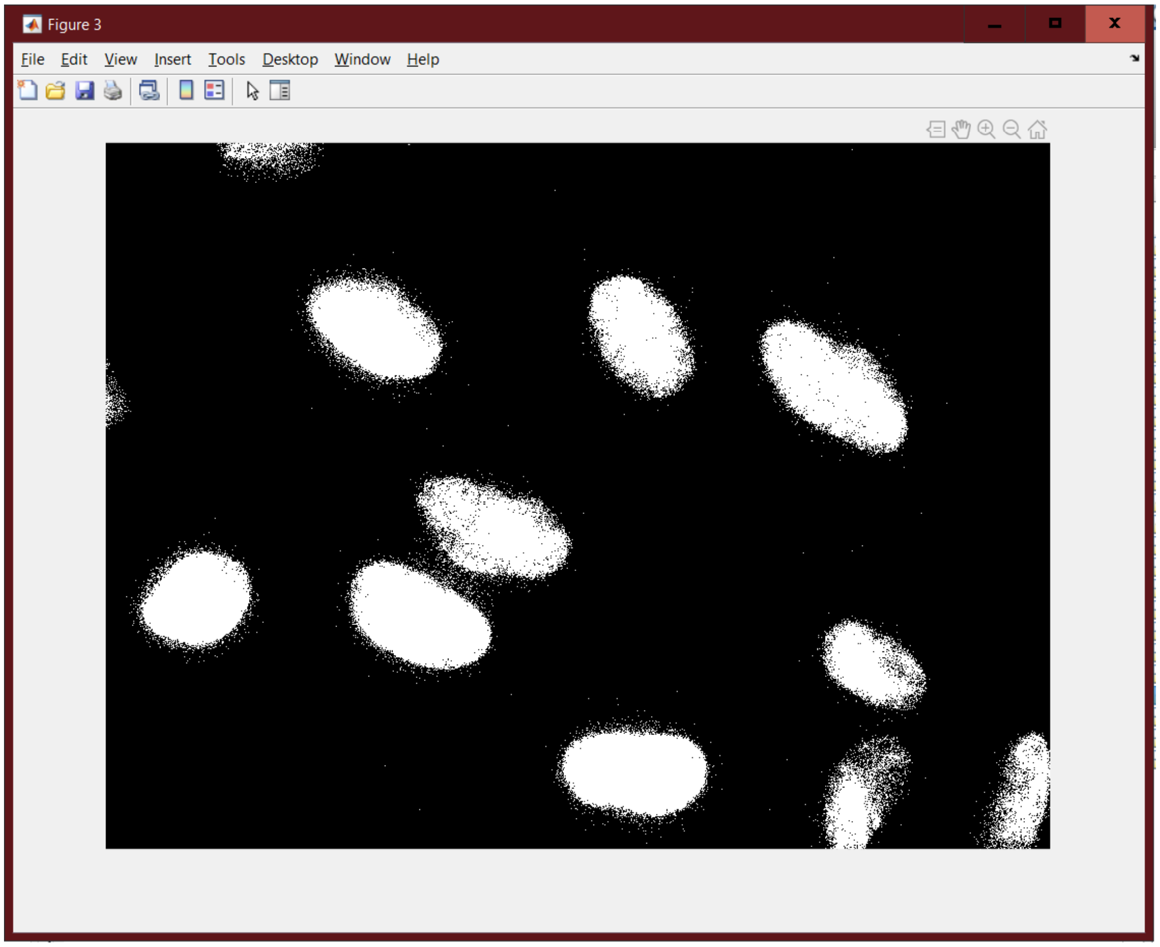

Figure 12: Fig. 10 after Thresholding

Q8. By examining the histograms of colour planes that you estimated in Section 5.1, write a script that segments the table in the image pool.02.bmp by applying thresholds to each of the 3 colour channels. Include in your report, an image showing the segmentation that you achieve. Describe how well your *segmentation mask* matches the actual limits of the table. Specifically mention where the algorithm works and where it does not. Write down the values of the thresholds that you use for each channel. Explain why applying thresholds on all 3 colour channels improves segmentation compared to performing the segmentation on the green channel alone.

A8.

```
pic = imread('D:\Docs\Study\Digital Image and Video
Processing\image_stills\4s1\lab1\pool.02.bmp');
figure(1);imshow(pic)
%pic = double(pic);

%threshold values for r, g and b planes resp.
rLow = 45;
rHigh = 120;
bLow = 25;
bHigh = 70;
gLow = 125;
gHigh = 240;

%red plane mask
red_plane = pic(:,:,1);
red_plane = double(red_plane);

red_plane(red_plane<rLow) = 0;
red_plane(red_plane<rHigh & red_plane>rLow) = 1;
red_plane(red_plane>rHigh) = 0;

%green plane mask
green_plane = pic(:,:,2);
green_plane = double(green_plane);

green_plane(green_plane<gLow) = 0;
green_plane(green_plane<gHigh & green_plane>gLow) = 1;
green_plane(green_plane>gHigh) = 0;

%blue plane mask
blue_plane = pic(:,:,3);
blue_plane = double(blue_plane);

blue_plane(blue_plane<bLow) = 0;
blue_plane(blue_plane<bHigh & blue_plane>bLow) = 1;
blue_plane(blue_plane>bHigh) = 0;
```

```
%combining all three masks
[vres hres channel] = size(pic)
size_mat = (red_plane & green_plane & blue_plane);

%figure;hist(pic(:,:,1),[0:255]);
%figure;hist(pic(:,:,2),[0:255]);
%figure;hist(pic(:,:,3),[0:255]);

%repmat(size_mat, [1,1,3]);
newpic = bsxfun(@times, pic, cast(size_mat, class(pic)));
figure(2); imshow(newpic);
```

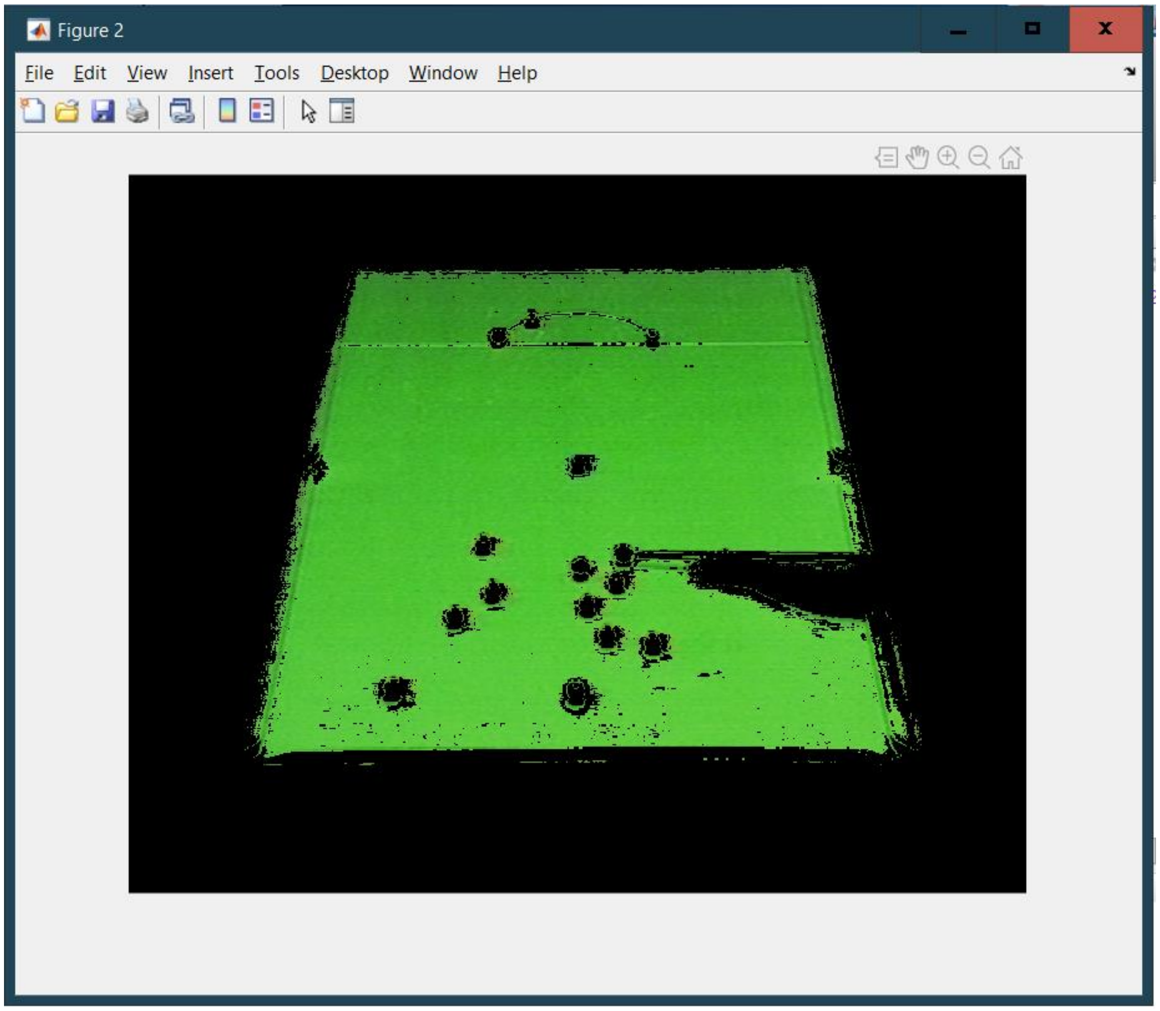

Figure 13: Fig. 4 after Image Segmentation

As is evident from Fig. 12 above, the segmentation mask used works very well although there are some artefacts towards the nearer end of the pool table.

The algorithm works fine for slightly darker shades of green but has little problem when it comes to the brightest parts. The threshold values used for each channel are as follows:

1) Red -	Low	= 45
	High	= 120
2) Blue -	Low	= 25
	High	= 70
3) Green -	Low	= 125
	High	= 240

Using all three channels instead of just G channel (or any single channel) helps remove artefacts from segmentation. This even works in case of other formats of channels such as HSV or YUV. For eg. The following is a result of using Hue alone vs Hue and Saturation when it comes to HSV.

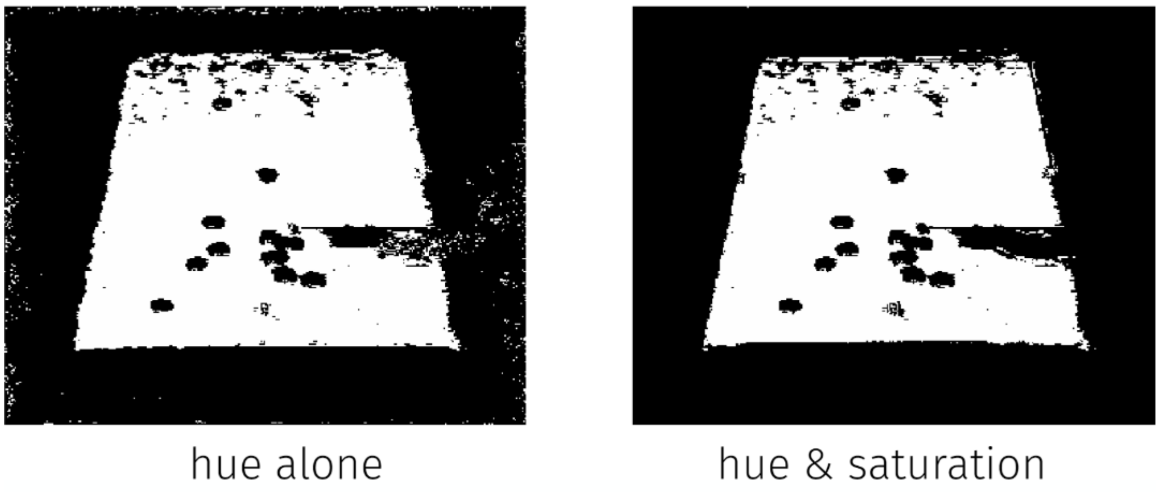

Figure 14: Using single vs multiple channels for Image Segmentation