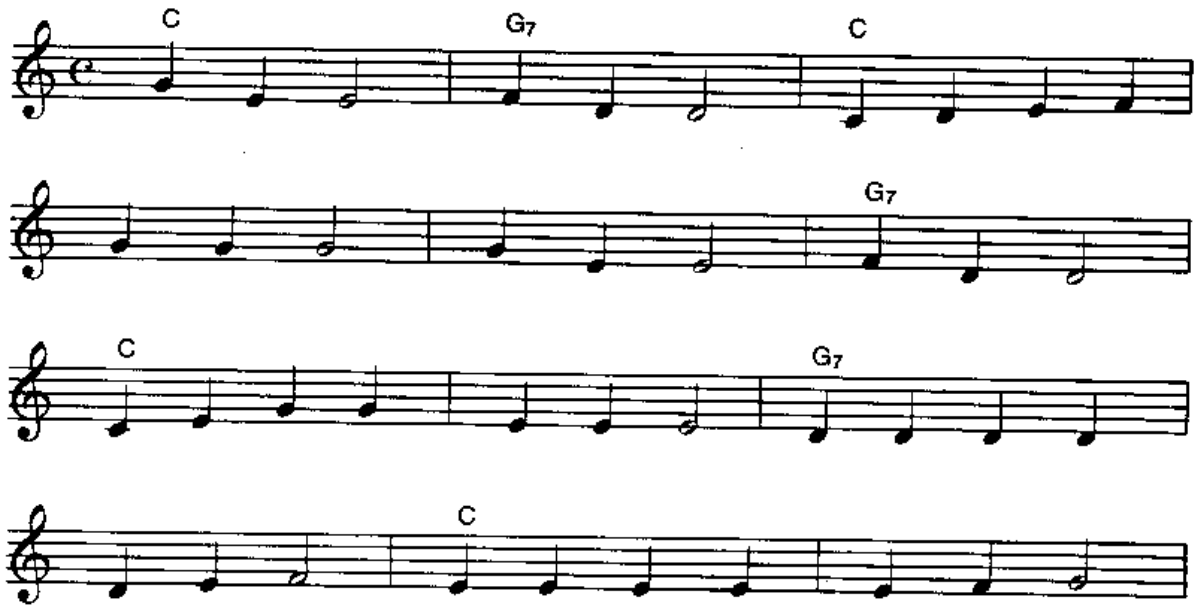
EE 417 Take Home Final Exam Report

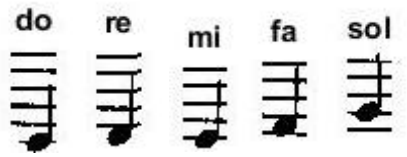
Dilara Ademoğlu – 20620

Question 1

This is the original image:



The original image was not in good condition. So I applied ***imrotate*** and rotated it by 0.6 degrees. Then I applied a gaussian filter with threshold 0.2. After that I manually detected the notes and extracted them by a 54x22 matrix. I have written a function called **storedata** for this. Results are:



I have used these 5 notes as my templates.

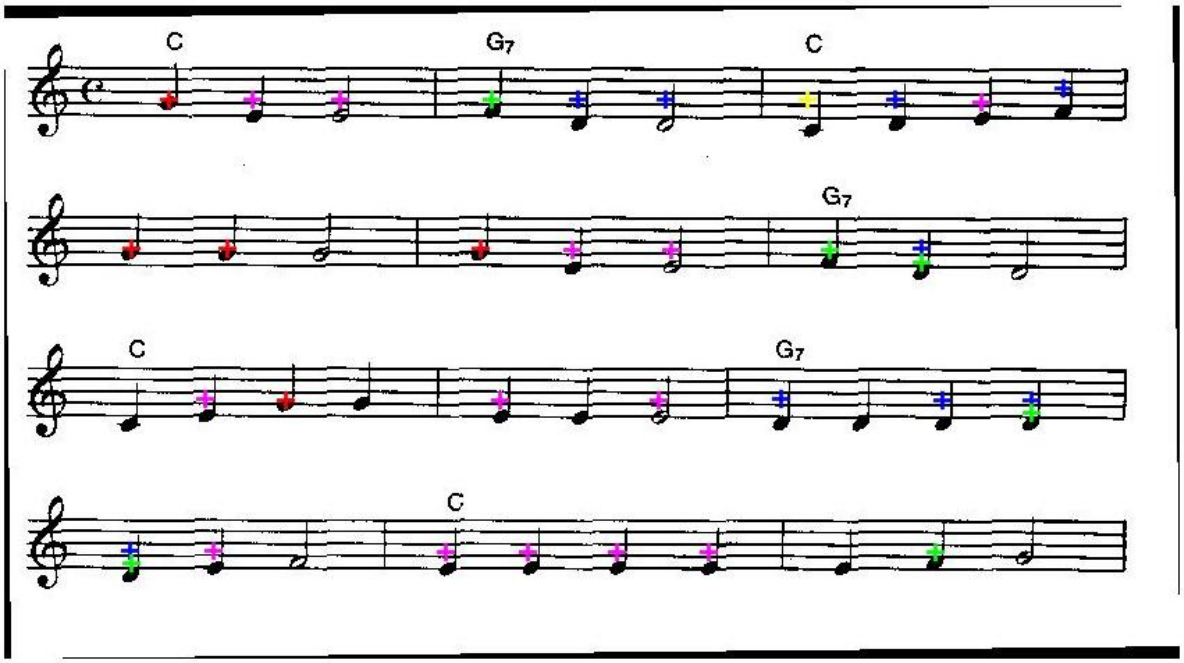
I wrote a function called **detect**, which returns a struct which I stored the coordinates of the matched notes, number of notes found and the white pixels of the difference image which I have got inside the function.

Inside two for loops I visit every pixel, and for each pixel I scan an area the same size as my template notes. For each pixel I take a difference of the two matrices. As a result I get a binary image. I count the number of white pixels I got in the difference image by using a function I wrote, I called it **numdark**. Numdark function is very simple, it visits each pixel in that matrix and increments the value if the pixel

value is 1. The example difference images are below, first one is when a note is detected, and the second one is when the note is not detected.

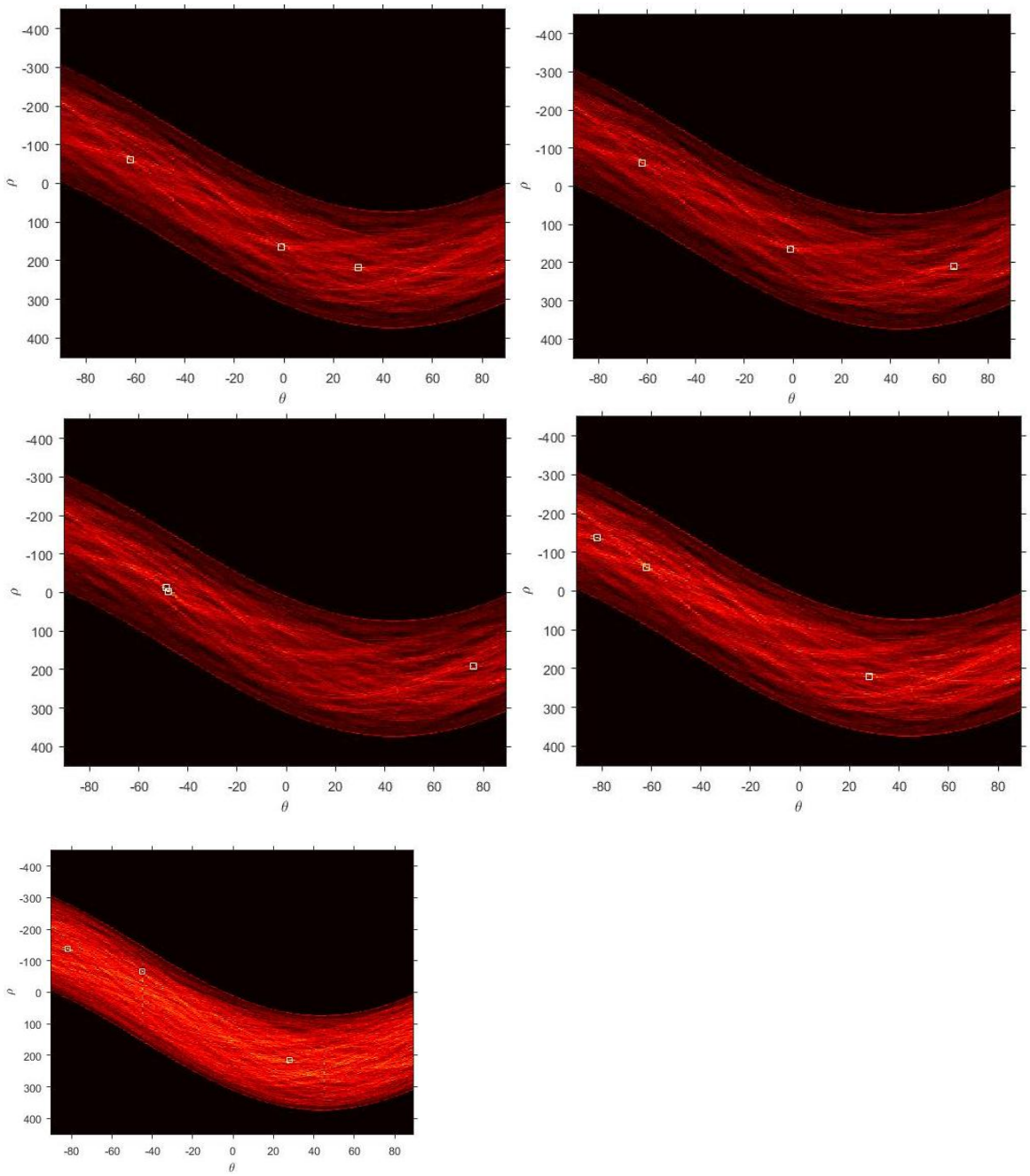
After detecting the number of white pixels in the difference image, I have an if function to see if they are similar. Inside the if function I compare the number of white pixels with a threshold value I manually decided. If the number of white pixels is smaller than the threshold value it is our note. Inside the if function I have another if condition, in this condition I check if the found note's coordinates are next to the next found note. Since I move pixel by pixel this is necessary. If it is not, then I add the note to the array and increment the number of notes found by 1. After that I add the array to the end of my matrix.

I use the x and y values from the matrix that was returned from the detect function to ***plot*** them on the original image. Yellow represents Do, blue represents Re, purple represents Mi, green represents Fa, and red represents Sol. The output is shown below:

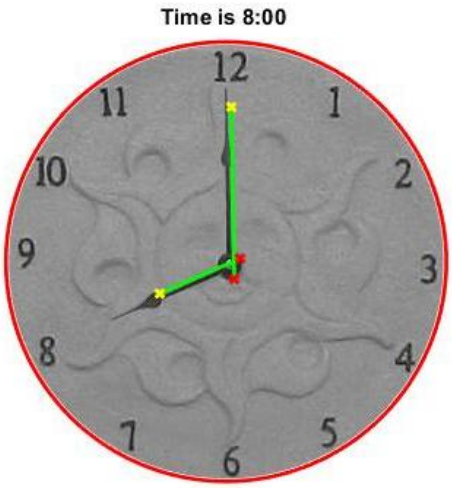


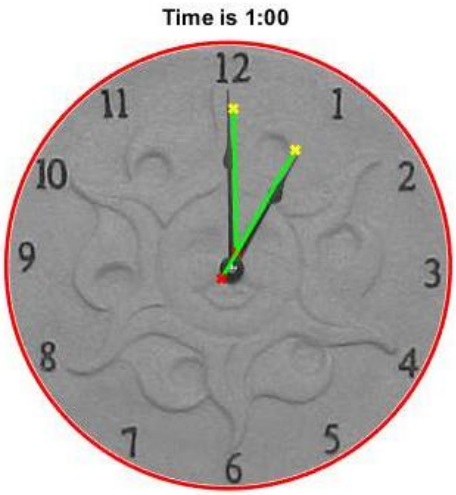
Question 2

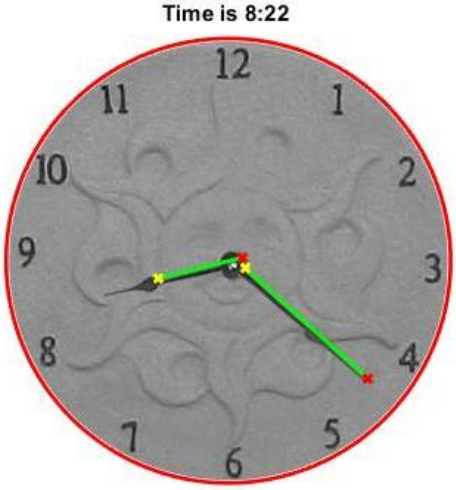
After reading the image, I have used ***rgb2gray***, then I have used gaussian filter with 2.5 threshold. I found the binary image with canny edge function. After that in order to detect lines I used Hough method. I used ***houghpeaks*** with parameter **3** to find 3 peaks. The images I got are this for each of the test images:

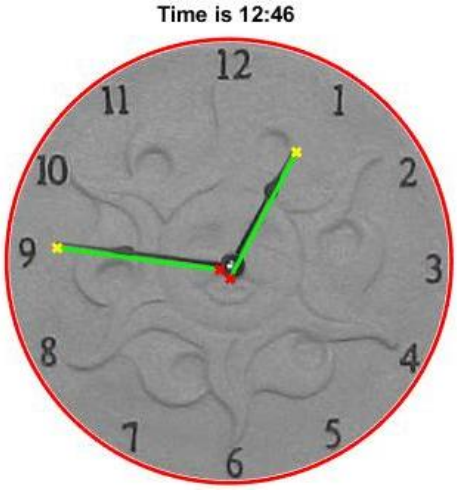


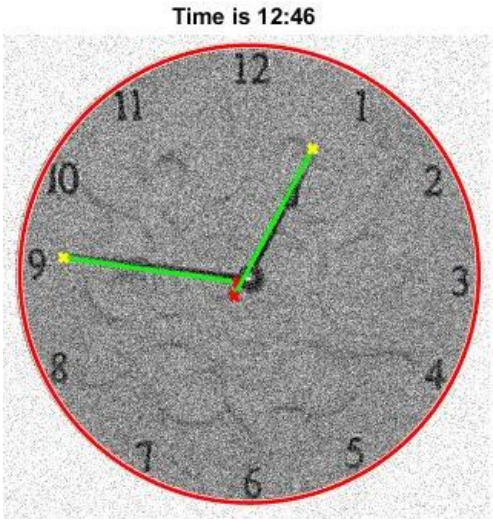
After finding the peaks I used them to detect the lines. I have set '**Fillgap**' value to **10** and '**MinLength**' value to **50**, in order to get rid of small/short lines. After finding the lines I have plotted them with **plot** function. After that I have find the outer circle with '**Sensitivity**' value **0.98**(default is 0.89) and '**ObjectPolarity**' value as **'dark'**, since we are trying to detect a circle whose background is bright. Then I have used ***viscircles*** to display the circle on the image. These are the results:











Later, I have extracted the theta values of both lines to find the angles. I have written 2 functions, one of them to detect hour and the other to detect minute. For both functions I sent ***theta*** value (I called it angle)

and ***point 1***(which are the points shown with yellow) extracted from the line structure and the ***center*** of the circle.

In the function **oclock**, since theta only shows the angel, not direction, I had to decide which part of the clock the arms are. In order to do that, I have compared the point 1 with center, if point 1's x value is smaller than center's x value of my hour hand is between 6 and 12, else it is between 12 and 6. Then in another if condition I have assigned a range up to 30 and compared angle with these in each if/elseif, it continued until 180, since this was the maximum theta value. The first if was like this:

```
if(angle>=0 && angle<30)
```

According to that I have the output of the function, which is the hour.

For **findminute** function with same parameters for the minute hand, I did the comparison with center and point 1. I have divided the angle to 6 since my angle value goes up to 180 and I only want up to 30. After that, I have either subtracted this value from 60 or 30, depending the result of the comparison with if function. The output is minute.

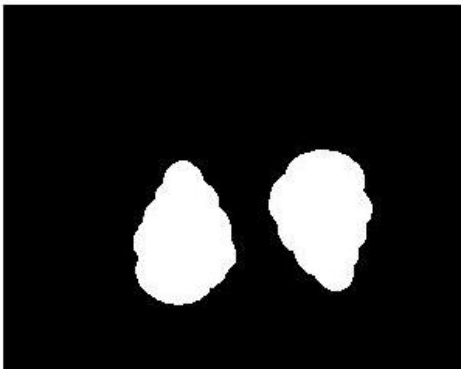
Lastly I have created an array which says ["Time is " hour ":" minute]. It takes the output values of oclock and findminute function. Then I displayed this array as a title of the picture.

Question 3

We have 9 images for this question, t1, t2, t3, t4, t5, t6, te1, te2, te3, can be seen respectively:

I have read all 9 images with ***imread***. Then I have applied segmentation with three functions I wrote, **binarize**, **getImg** and **detectchoco**.

In getImg function I have first cropped the image to detect chocolates better, with ***imcrop***. Then I have applied the binarize function I have wrote. In this function I have applied ***rgb2gray, imbinarize*** with a certain threshold value which I got as a parameter**,** ***imcomplement, imfill*** with parameter **'holes'** to the image respectively. I used ***strel*** with parameters **'disk'** and value 5, then used ***imopen*** with the output of strel funciton and the binary image. This output is returned from the binarize function. An example for the output **of "binarize**" function can be seen below, binarize applied to t1.jpg:



After retrieving the binary image, I used ***bwlabel*** to get label. Then applied this for loop:

```
for j=1:max(max(label))
    [row,col]=find(label==j);
    target=uint8(zeros([(max(row)-min(row)+2)  (max(col)-min(col)+2)]));
    sy=min(col)-1;
    sx=min(row)-1;
    for i=1:size(row,1)
        x=row(i,1)-sx;
        y=col(i,1)-sy;
        target(x,y)=t1(row(i,1),col(i,1));
    end
    store(k).target=target; %store is the struct to store target images
    k=k+1; %go for the next struct
end
```

max(max(label)) is number of objects in the image. Target is the image I want to get; it is initially created as zeros for each iteration. Inside the loop I create target image pixel by pixel and then I store it in the struct called store before going for the next label.

After getting all the chocolates separately from the test images, I apply detectchoco function and create 2 different structs which I called choco and test. I apply detectchoco 6 times to the structs I got from the getImg function. Since they have 2 of the same chocolates in the same image, it is not necessary to apply all 12 images. Detectchoco has 2 parameters, which are the struct of images I got from getImg function and an integer to decide which image from the struct I want. Example: im1(1), im1 is the struct 1 is the first line of the struct.

Inside detect choco function, I have gotten a **cc** value using ***bwconncomp*** function with parameter 8, which is calculating the 8-neighbor connectivity. After getting the cc, I sent it to ***regionprops*** function with parameters '**Area', 'Perimeter', 'MajorAxisLength' and 'MinorAxisLength'**. Inside a for loop I have stored these parameter. I have first tried to compare areas but then decided this wouldn't be very efficient. So I have decided to add another parameter I called Div, to the struct. Div is the division output of MajorAxisLength and MinorAxisLength. Since the main difference of these chocolates is their shapes, I thought this should be reasonable, and I was right.

After getting the output struct from detectchoco function, I have created 2 for loops, one going to 13 and the other to 6, since I have 6 types of chocolates and 13 chocolates to detect. Inside these 2 loops I have created 2 13x5 matrices. One of them to store the difference of the division results, the other to store differences of the parameters, both shown below, respectively.

label | a | c

13x6 double

	1	2	3	4	5	6
1	0.1422	0.0794	0.1518	0.0579	0.2970	0.4236
2	0.0354	0.0983	0.3294	0.2355	0.4747	0.2460
3	0.3005	0.2377	0.0065	0.1005	0.1387	0.5820
4	0.3297	0.3925	0.6237	0.5298	0.7689	0.0482
5	0.4160	0.4789	0.7100	0.6161	0.8553	0.1346
6	0.1818	0.1190	0.1121	0.0182	0.2574	0.4633
7	0.0098	0.0530	0.2842	0.1902	0.4294	0.2913
8	0.4281	0.3652	0.1341	0.2280	0.0111	0.7095
9	0.2333	0.2961	0.5273	0.4334	0.6725	0.0482
10	0.5010	0.5639	0.7950	0.7011	0.9403	0.2196
11	0.3069	0.2440	0.0129	0.1068	0.1324	0.5883
12	0.4294	0.3665	0.1354	0.2293	0.0099	0.7108
13	0.3336	0.2707	0.0396	0.1335	0.1057	0.6150

label | a | c

13x6 double

	1	2	3	4	5	6
1	72.7200	16.3340	44.3940	47.1000	88.5000	32.0240
2	16.4980	39.8880	11.8280	9.1220	32.2780	24.1980
3	31.2520	25.1340	2.9260	5.6320	47.0320	9.4440
4	39.3220	17.0640	10.9960	13.7020	55.1020	1.3740
5	58.7580	2.3720	30.4320	33.1380	74.5380	18.0620
6	34.2460	22.1400	5.9200	8.6260	50.0260	6.4500
7	15.2000	41.1860	13.1260	10.4200	30.9800	25.4960
8	2.4980	58.8840	30.8240	28.1180	13.2820	43.1940
9	42.5040	13.8820	14.1780	16.8840	58.2840	1.8080
10	52.6800	3.7060	24.3540	27.0600	68.4600	11.9840
11	27.3820	29.0040	0.9440	1.7620	43.1620	13.3140
12	7.6340	64.0200	35.9600	33.2540	8.1460	48.3300
13	56.9160	0.5300	28.5900	31.2960	72.6960	16.2200

After observing the matrices, I have created an if condition with thresholds I decided manually. I got accurate results when I set the threshold to 0.09 for the first matrix and between 0 and 20 for the second matrix.

Inside this if condition I have stored the difference values I maintained from the matrices and the chocolate numbers from test images. Then I displayed the matching ones in a ***subplot*** and also in the console.

The output is displayed in the command window as below:

Editor - fin_q3.m

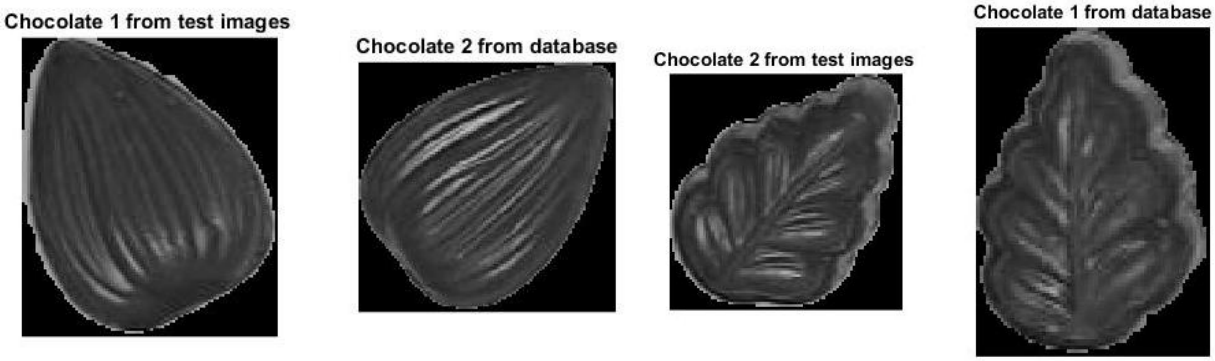
final_q1.m | final_q2.m | fin_q3.m | final_q4.m

```
44 - for i=1:13
45 -     for j=1:6
46 -         a(i,j)=abs(test(i).Div-choco(j).Div);
47 -         c(i,j)=abs(test(i).Perimeter-choco(j).Perimeter);
48 -         if(a(i,j)<0.09 && c(i,j)<20 && c(i,j)>1)
49 -             b(k).a=a(i,j);
50 -             b(k).b=i;
51 -             b(k).c=j;
52 -             b(k).d=c(i,j);
53 -             k=k+1;
54 -             str1=["Chocolate " i " from test images "];
55 -             str2=["Chocolate " j " from database"];
56 -             str2=str2(1,1)+str2(1,2)+str2(1,3);
57 -             str1=str1(1,1)+str1(1,2)+str1(1,3);
58 -             str=str1 + "is detected as " +str2;
59 -             disp(str);
60 -             figure;
61 -             subplot(1,2,1)
```

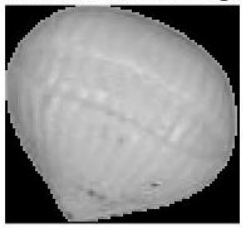
Command Window

```
Chocolate 1 from test images is detected as Chocolate 2 from database
Chocolate 2 from test images is detected as Chocolate 1 from database
Chocolate 3 from test images is detected as Chocolate 3 from database
Chocolate 4 from test images is detected as Chocolate 6 from database
Chocolate 6 from test images is detected as Chocolate 4 from database
Chocolate 7 from test images is detected as Chocolate 1 from database
Chocolate 8 from test images is detected as Chocolate 5 from database
Chocolate 9 from test images is detected as Chocolate 6 from database
Chocolate 12 from test images is detected as Chocolate 5 from database
fx >>
```

Images below are the results, left ones are from the t1, t2, t3, t4, t5, t6 and right ones are from te1, te2, te3:



Chocolate 3 from test images



Chocolate 3 from database

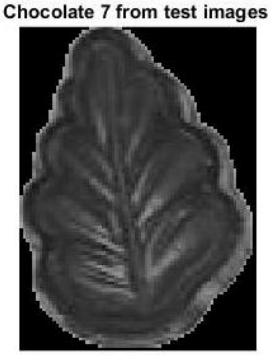
Chocolate 4 from test images

Chocolate 6 from database

Chocolate 6 from test images

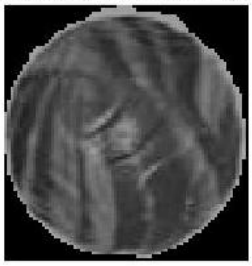
Chocolate 4 from database

Chocolate 7 from test images

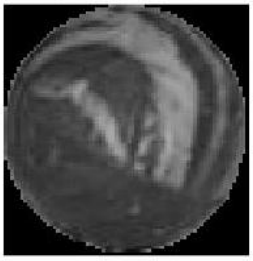


Chocolate 1 from database

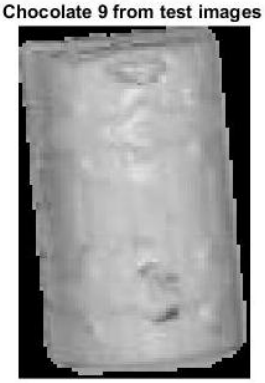
Chocolate 8 from test images



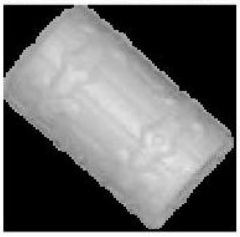
Chocolate 5 from database



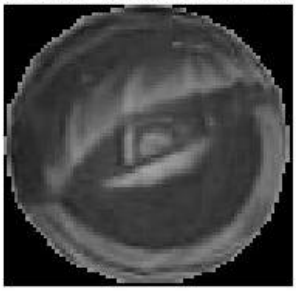
Chocolate 9 from test images



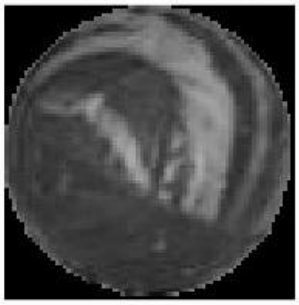
Chocolate 6 from database



Chocolate 12 from test images

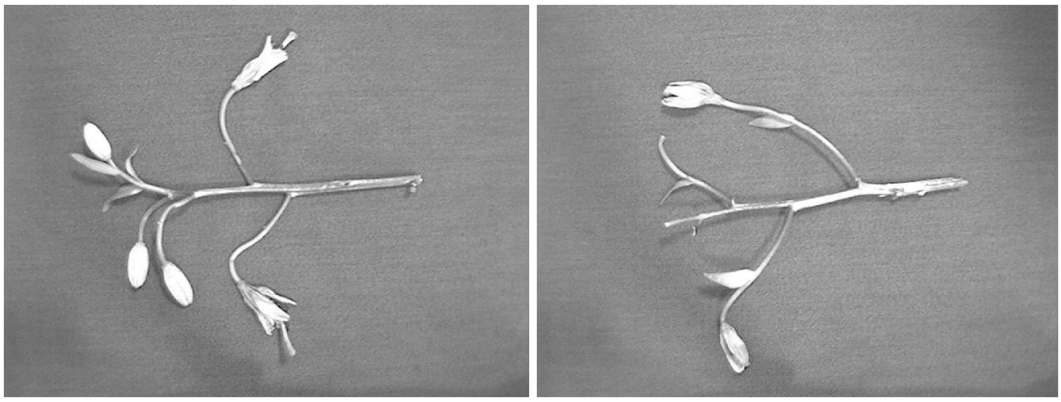


Chocolate 5 from database



Question 4

Initial test images are given as:

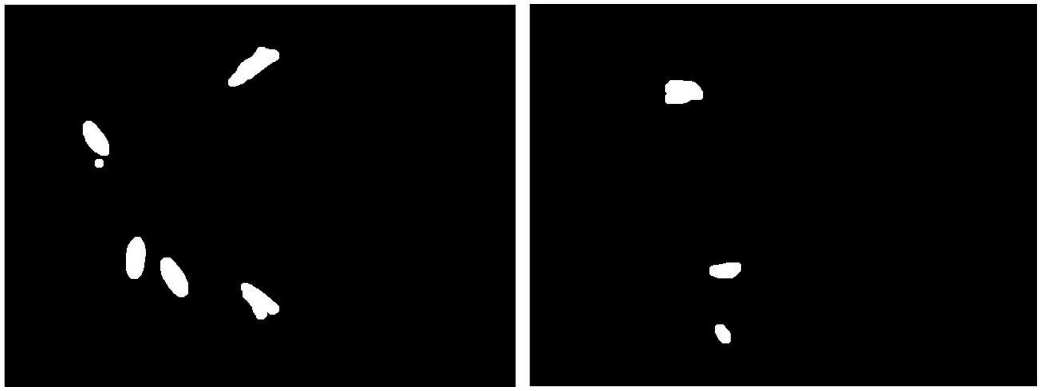


I have used the getImg and binarize functions from the previous question. I have modified the getImg function to return also the number of objects detected so that I can display the number of leaves detected.

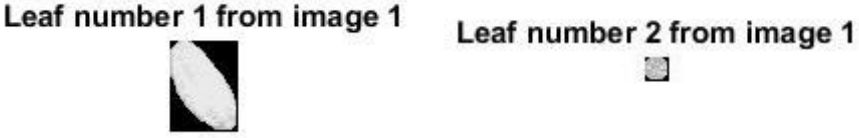
getImg function now takes 4 parameters, the image, a threshold value for binarize function, an integer s, which will be used in ***strel*** function and an integer a to define which image is which. Binarize function is again inside getImg function, and it get two of the parameters: threshold and integer s. In binarize function, I don't get imcomplement anymore since my background is darker than the objects.

I have given threshold value as 0.6 for both of them and strel function as 6 and 7 respectively.

The binary images from binarize function are below, white pixels are the locations of the leaves:



The leaves found are:



Leaf number 3 from image 1

Leaf number 4 from image 1

Leaf number 5 from image 1

Leaf number 6 from image 1

Leaf number 1 from image 2

Leaf number 2 from image 2

Leaf number 3 from image 2

Total number of leaves displayed in the command window:

Editor - C:\Users\Dilara\Documents\MATLAB\final_q4.m

final_q1.m | final_q2.m | fin_q3.m | final_q4.m

```
clc;clear all;
I1=imread('Resim1.png');
I2=imread('Resim2.png');
I2=imgaussfilt(I2,2);
% figure;imshow(I1);
% figure;imshow(I2);

[im1,n1]=getImg(I1,0.6,6,1);
[im2,n2]=getImg(I2,0.6,7,2);

function[store,k]=getImg(t,thres,s,a)
tlbw=binarize(t,thres,s);
% figure;imshow(tlbw);
label=bwlabel(tlbw);
k=1;
for j=1:max(max(label))
    [row,col]=find(label==j);
    len=max(row)-min(row)+2;
    breadth=max(col)-min(col)+2;
    target=uint8(zeros([len breadth]));
    sy=min(col)-1;
```

```
There are 6 leaves in image 1
There are 3 leaves in image 2
fx >>
```

The problem with this code is it cannot detect the leaves which are under the stems.