Digital Image Processing (UE19EC317) Mini Project

Group 9

SRN: PES2UG19EC070

Mansoora S

SRN: PES2UG19EC071

Mehul Joshi

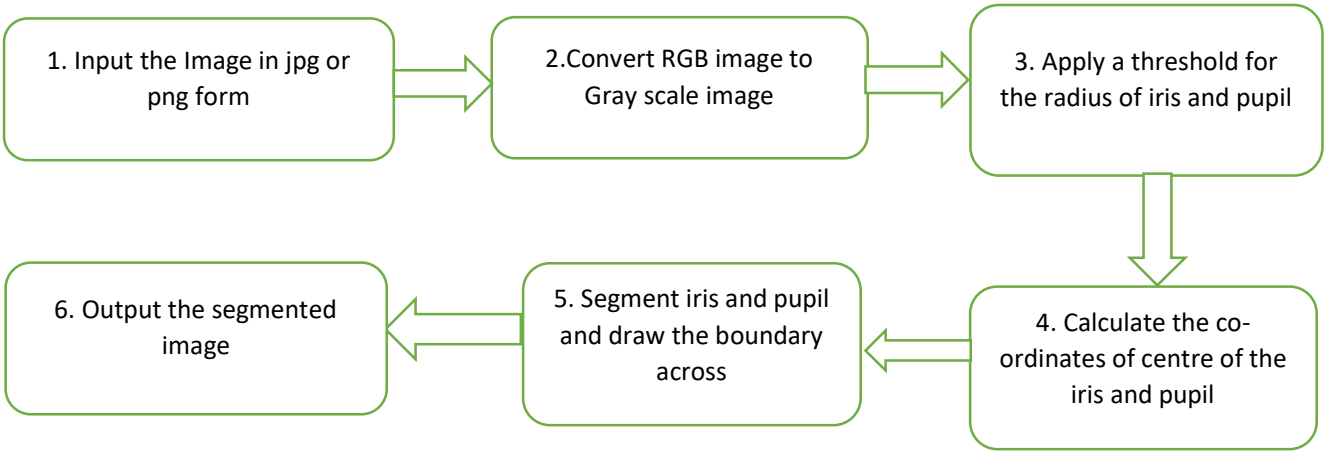
ABSTRACT

Iris segmentation is an essential step in the iris recognition procedure. Most of the iris segmentation algorithms are based on edge information. However, a large number of noisy edge points detected by a normal edge-based detector in an image with specular reflection or other obstacles will mislead the pupillary boundary. We will execute a MATLAB program that differentiates the iris from the pupil based on the intensity variation that exists between the two.

PROBLEM STATEMENT DESCRIPTION

Iris segmentation for different type of test eye images using Matlab to write a program that helps us to differentiate between the human iris and pupil.

Block Diagram



Algorithm and steps:

Step 1: Input the Image in jpg or png form

Step 2: Convert RGB image to Gray scale image

Step 3: Apply a threshold for the radius of iris and pupil

Step 4: Calculate the co-ordinates of centre of the iris and pupil

Step 5: Segment iris and pupil and draw the boundary across

Step 6: Output the segmented image

Matlab Code:

%Main program

```
clc;
clear all;
I=imread('eye.jpg');
[m,n]=size(I);
%if the size of the image is small resize it
if m<256 | n<512
    A=imresize(I,[256,512]);
else
    A=I;
end
eye=rgb2gray(A);
[ci,cp,out]=thresh(eye,60,600);
figure;
subplot(1,2,1),imshow(A);
title('original image');
subplot(1,2,2),imshow(out);
title('segmented image');
```

%thresh.m function

```
%function to search for the centre coordinates of the pupil
and the iris along with their radii
%checking if the selected points(by thresholding)
%correspond to a local minimum in their immediate(3*s)
neighbourhood
%these points serve as the possible centre coordinates for the
iris.
%Once the iris has been detected;the pupil's centre
coordinates
```

```
%are found by searching a 10*10 neighbourhood around the iris
centre and varying the radius
%until a maximum is found
%INPUTS:
%I:image to be segmented
%rmin ,rmax:the minimum and maximum values of the iris radius
%OUTPUTS:
%cp:the parametrs[xc,yc,r] of the pupilary boundary
%ci:the parametrs[xc,yc,r] of the limbic boundary
%out:the segmented image
function [ci,cp,out]=thresh(I,rmin,rmax)
scale=1;
rmin=rmin*scale;
rmax=rmax*scale;
%scales all the parameters to the required scale
I=im2double(I);
%the image is converted to double
pimage=I;
I=imresize(I,scale);
I=imcomplement(imfill(imcomplement(I),'holes'));
%this process removes specular reflections by using the
morphological operation 'imfill'
rows=size(I,1);
cols=size(I,2);
[X,Y]=find(I<0.5);
%Generates a column vector of the image elements
%that have been selected by tresholding;one for x coordinate
and one for y
s=size(X,1);
for k=1:s
    if (X(k)>rmin)&(Y(k)>rmin)&(X(k)<=(rows-
rmin))&(Y(k)<(cols-rmin))
            A=I((X(k)-1):(X(k)+1),(Y(k)-1):(Y(k)+1));
            M=min(min(A));
            %this process scans the neighbourhcood of the
selected pixel
            %to check if it is a local minimum
           if I(X(k),Y(k))~=M
              X(k)=NaN;
              Y(k)=NaN;
           end
    end
end
v=find(isnan(X));
X(v)=[];
Y(v)=[];
%deletes all pixels that are NOT local minima(that have been
set to NaN)
index=find((X<=rmin)|(Y<=rmin)|(X>(rows-rmin))|(Y>(cols-
rmin)));
X(index)=[];
```

```
Y(index)=[];
%This process deletes all pixels that are so close to the
border
%that they could not possibly be the centre coordinates.
N=size(X,1);
%recompute the size after deleting unnecessary elements
maxb=zeros(rows,cols);
maxrad=zeros(rows,cols);
%defines two arrays maxb and maxrad to store the maximum value
of blur
%for each of the selected centre points and the corresponding
radius
for j=1:N

[b,r,blur]=partiald(I,[X(j),Y(j)],rmin,rmax,'inf',600,'iris');
%coarse search
    maxb(X(j),Y(j))=b;
    maxrad(X(j),Y(j))=r;
end
[x,y]=find(maxb==max(max(maxb)));
ci=search(I,rmin,rmax,x,y,'iris');%fine search
%finds the maximum value of blur by scanning all the centre
coordinates
ci=ci/scale;
%the function search searches for the centre of the pupil and
its radius
%by scanning a 10*10 window around the iris centre for
establishing
%the pupil's centre and hence its radius
cp=search(I,round(0.1*r),round(0.8*r),ci(1)*scale,ci(2)*scale,
'pupil');%biological limits on the relative sizes of the iris
and pupil
cp=cp/scale;
%displaying the segmented image
out=drawcircle(pimage,[ci(1),ci(2)],ci(3),600);
out=drawcircle(out,[cp(1),cp(2)],cp(3),600);
```

%line_int.m function

```
%function to calculate the normalised line integral around a
circular contour
%A polygon of large number of sides approximates a circle and
hence is used
%here to calculate the line integral by summation
%INPUTS:
%1.I:Image to be processed
%2.C(x,y):Centre coordinates of the circumcircle
%Coordinate system :
%origin of coordinates is at the top left corner
%positive x axis points vertically down
%and positive y axis horizontally and to the right
```

```
%3.n:number of sides
%4.r:radius of circumcircle
%5.part:To indicate wheter search is for iris or pupil
%if the search is for the pupil,the function uses the entire
circle(polygon) for computing L
%for the iris only the lateral portions are used to mitigate
the effect of occlusions
%that might occur at the top and/or at the bottom
%OUTPUT:
%L:the line integral divided by circumference
function [L]=lineint1(I,C,r,n,part)
theta=(2*pi)/n;% angle subtended at the centre by the sides
%orient one of the radii to lie along the y axis
%positive angle is ccw
rows=size(I,1);
cols=size(I,2);
angle=theta:theta:2*pi;
x=C(1)-r*sin(angle);
y=C(2)+r*cos(angle);
if (any(x>=rows)|any(y>=cols)|any(x<=1)|any(y<=1))
    L=0;
    return
    %This process returns L=0 for any circle that does not fit
inside the image
end
%lines 34 to 42 compute the whole line integral
if (strcmp(part,'pupil')==1)
          s=0;
          for i=1:n
          val=I(round(x(i)),round(y(i)));
          s=s+val;
          end

          L=s/n;
end
%lines 44 onwards compute the lateral line integral(to prevent
occlusion affecting the results,the pixel average is taken
only along the lateral portions)
if(strcmp(part,'iris')==1)
          s=0;
            for i=1:round((n/8))
          val=I(round(x(i)),round(y(i)));
          s=s+val;
          end

          for i=(round(3*n/8))+1:round((5*n/8))
          val=I(round(x(i)),round(y(i)));
          s=s+val;
          end

          for i=round((7*n/8))+1:(n)
```

```
            val=I(round(x(i)),round(y(i)));
            s=s+val;
               end
               L=(2*s)/n;
end
```

%partiald.m function

```
%function to find the partial derivative
%calculates the partial derivative of the normailzed line
integral
%holding the centre coordinates constant
%and then smooths it by a gaussian of appropriate sigma
%%rmin and rmax are the minimum and maximum values of radii
expected
%function also returns the maximum value of blur and the
corresponding radius
%It also returns the finite differnce vector blur
%INPUTS:
%I;input image
%C:centre coordinates
%rmin,rmax:minimum and maximum radius values
%n:number of sides of the polygon(for lineint)
%part:specifies whether it is searching for the iris or pupil
%sigma:standard deviation of the gaussian
%OUTPUTS:
%blur:the finite differences vector
%r:radius at maximum value of 'blur'
%b:maximum value of 'blur'
function [b,r,blur]=partiald(I,C,rmin,rmax,sigma,n,part)
R=rmin:rmax;
count=size(R,2);
for k=1:count
[L(k)]=lineint(I,C,R(k),n,part);%computing the normalized line
integral for each radius
if L(k)==0%if L(k)==0(this case occurs iff the radius takes
the circle out of the image)
    %In this case,L is deleted as shown below and no more
radii are taken for computation
    %(for that particular centre point).This is accomplished
using the break statement
     L(k)=[];
    break;
end
end
D=diff(L);
D=[0 D];
%append one element at the beginning to make it an n vector
%Partial derivative at rmin is assumed to be zero
if strcmp(sigma,'inf')==1%the limiting case of the gaussian
with sigma infinity
f=ones(1,7)/7;
else
```

```
f=fspecial('gaussian',[1,5],sigma);%generates a 5 member 1-D
gaussian
end
blur=convn(D,f,'same');%Smooths the D vecor by 1-D convolution
%'same' indicates that size(blur) equals size(D)
blur=abs(blur);
[b,i]=max(blur);
r=R(i);
b=blur(i);
%calculates the blurred partial derivative
```

%search.m

```
% %function to detect the pupil  boundary
%it searches a certain subset of the image
%with a given radius range(rmin,rmax)
%around a 10*10 neighbourhood of the point x,y given as input
%INPUTS:
%im:image to be processed
%rmin:minimum radius
%rmax:maximum radius
%x:x-coordinate of centre point
%y:y-coordinate of centre point
%OUTPUT:
%cp:centre coordinates of pupil(x,y) followed by radius
function [cp]=search(im,rmin,rmax,x,y,option)
rows=size(im,1);
cols=size(im,2);
sigma=0.5;%(standard deviation of Gaussian)
R=rmin:rmax;
maxrad=zeros(rows,cols);
maxb=zeros(rows,cols);
for i=(x-5):(x+5)
for j=(y-5):(y+5)

[b,r,blur]=partiald(im,[i,j],rmin,rmax,0.5,600,option);
        maxrad(i,j)=r;
        maxb(i,j)=b;
    end
end
B=max(max(maxb));
[X,Y]=find(maxb==B);
radius=maxrad(X,Y);
cp=[X,Y,radius];
```

%draw_circle.m

```
%function to generate the pixels on the boundary of a regular
polygon of n sides
```

```
%the polygon approximates a circle of radius r and is used to
draw the circle
%INPUTS:
%1.I:Image to be processed
%2.C(x,y):Centre coordinates of the circumcircle
%Coordinate system :
%origin of coordinates is at the top left corner
%positive x axis points vertically down
%and positive y axis horizontally and to the right
%3.n:number of sides
%4.r:radius of circumcircle
%OUTPUT:
%O:Image with circle
function [O]=drawcircle1(I,C,r,n)
if nargin==3
    n=600;
end
theta=(2*pi)/n;% angle subtended at the centre by the sides
%orient one of the radii to lie along the y axis
%positive angle ic ccw
rows=size(I,1);
cols=size(I,2);
angle=theta:theta:2*pi;
%to improve contrast and help in detection
x=C(1)-r*sin(angle);%the negative sign occurs because of the
particular choice of coordinate system
y=C(2)+r*cos(angle);
if any(x>=rows)|any(y>=cols)|any(x<=1)|any(y<=1)%if circle is
out of bounds return image itself
    O=I;
    return
end
for i=1:n
I(round(x(i)),round(y(i)))=1;
end
O=I;
```

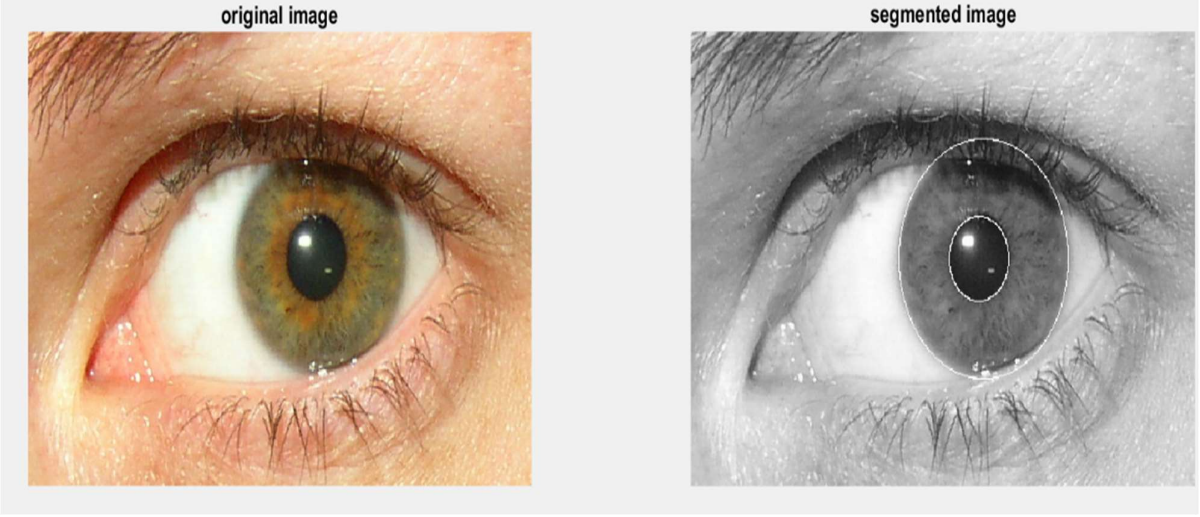
Results:

Output: 1. Fully opened eyes

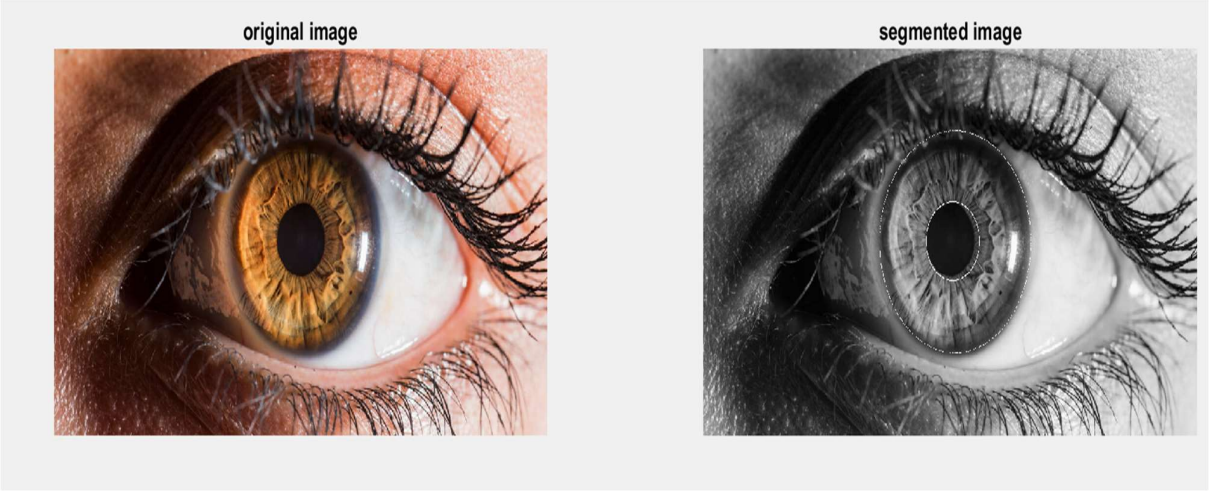
1.


original image
segmented image

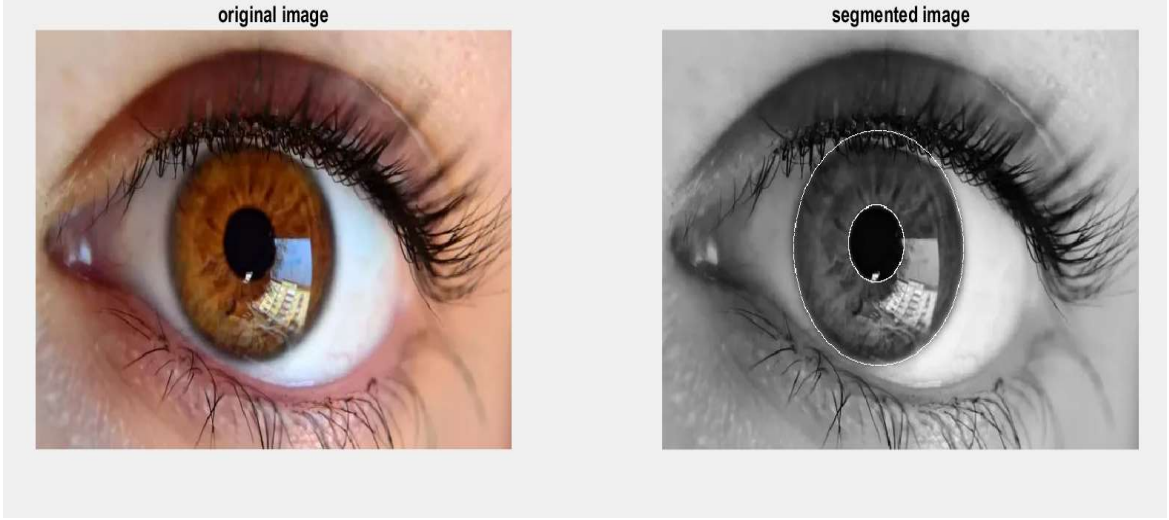
2.


original image
segmented image

3.

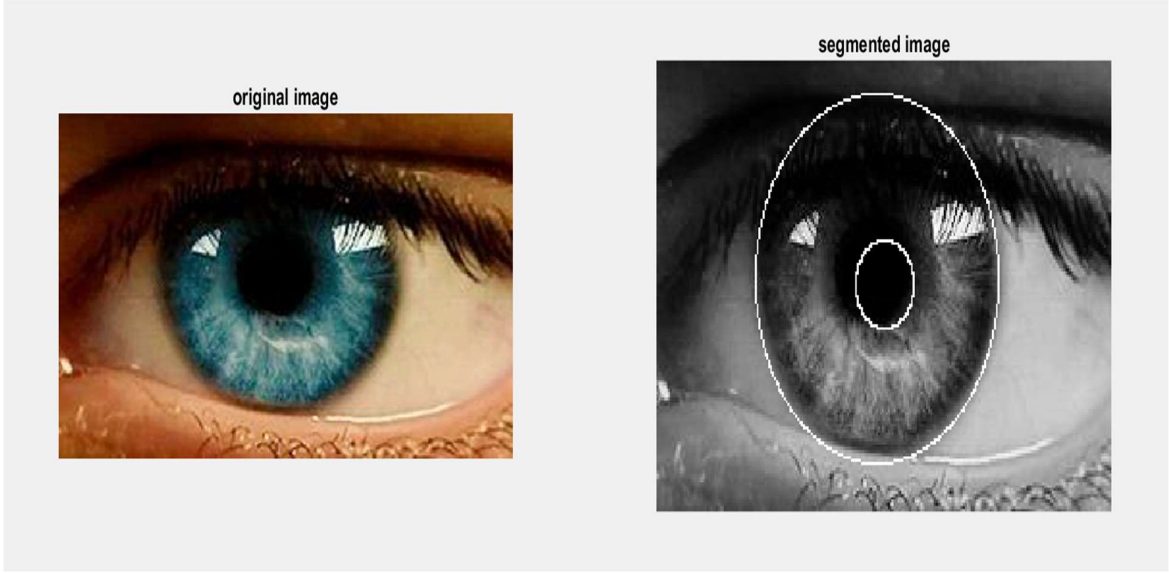

original image
segmented image

4.



2. 75% open eyes: Need to change the threshold for rmin and rmax accordingly since the iris is not properly displayed

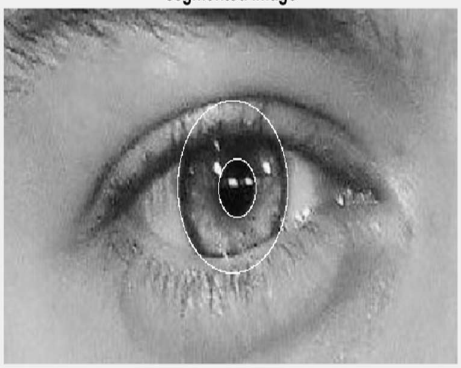
1.



2.

original image

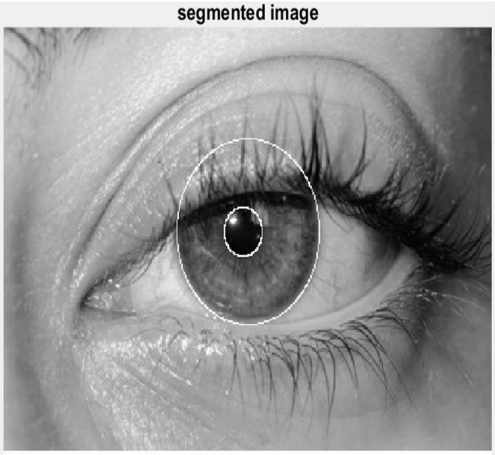
segmented image



3.

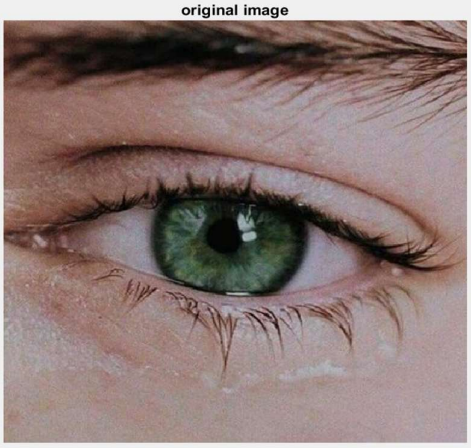
original image

segmented image



4.

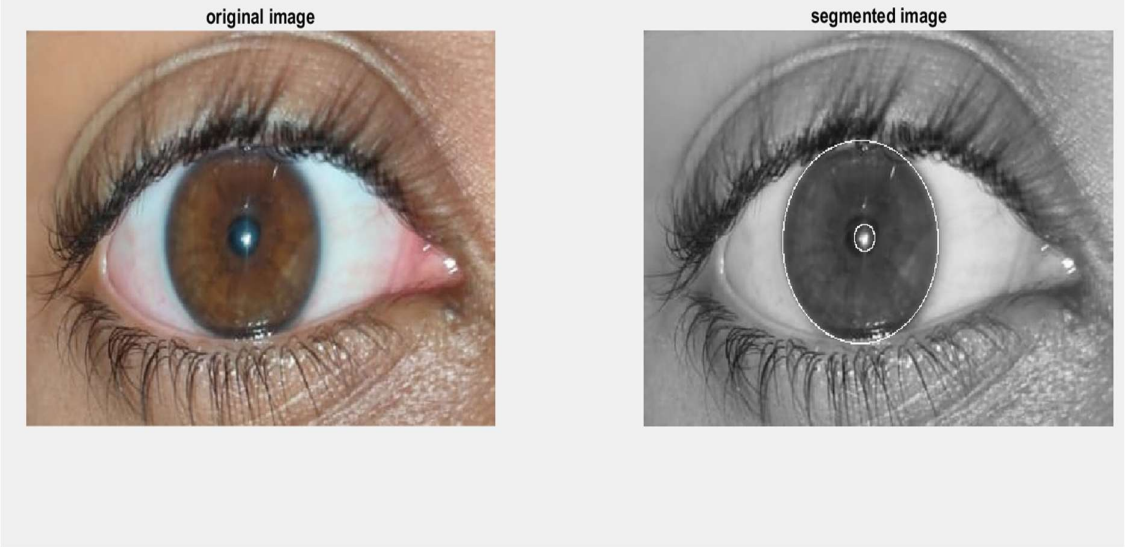
original image



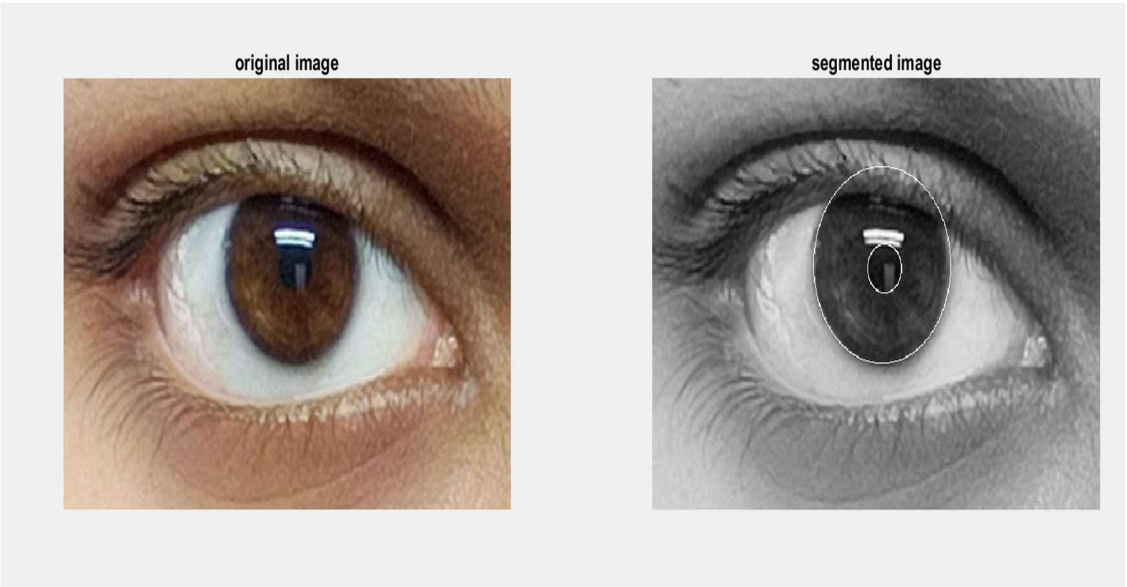
segmented image

3. Live captured Fully opened eyes

1.

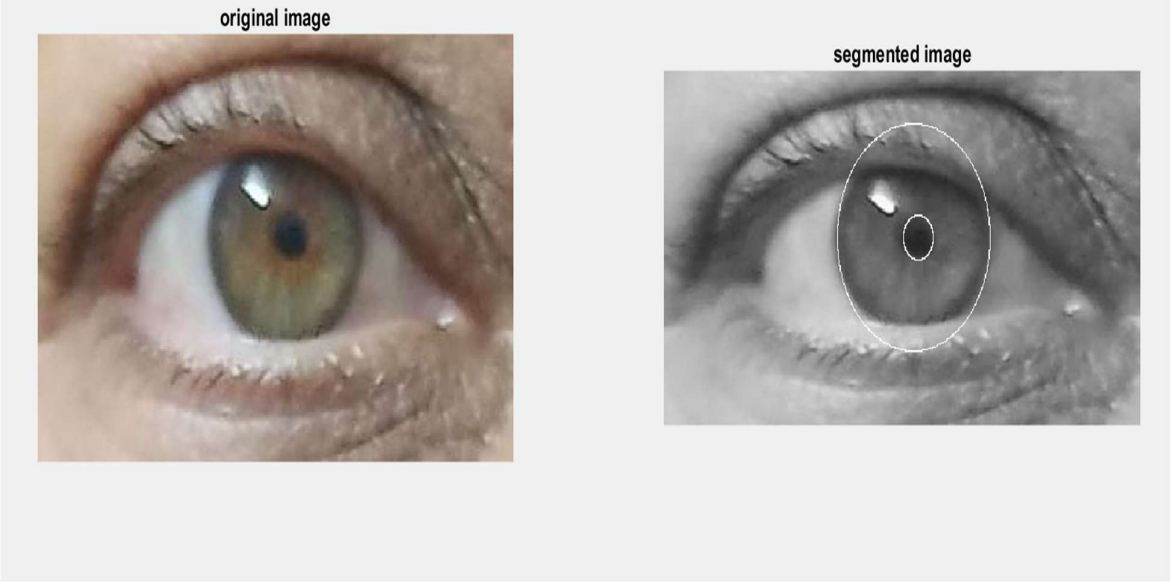


2.

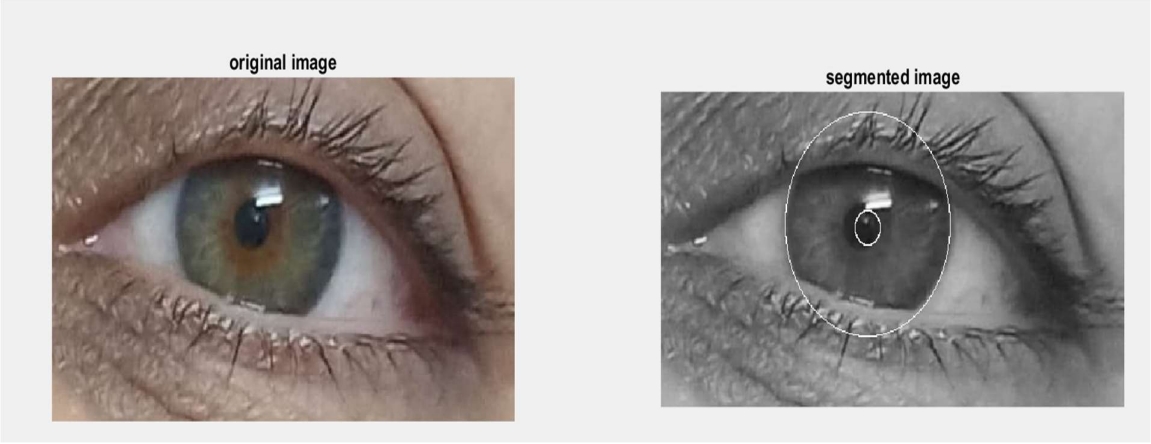


4. Live captured 75% opened eyes

1.

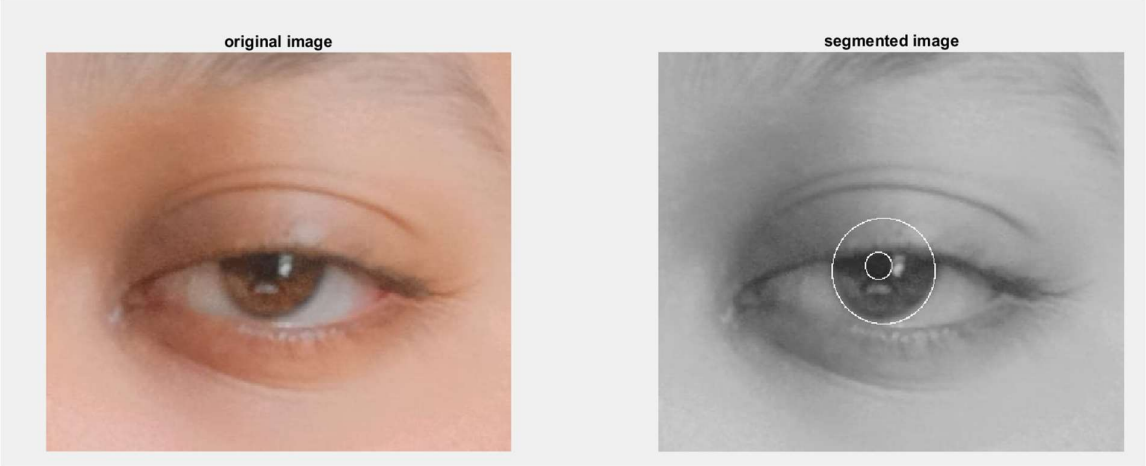


2.

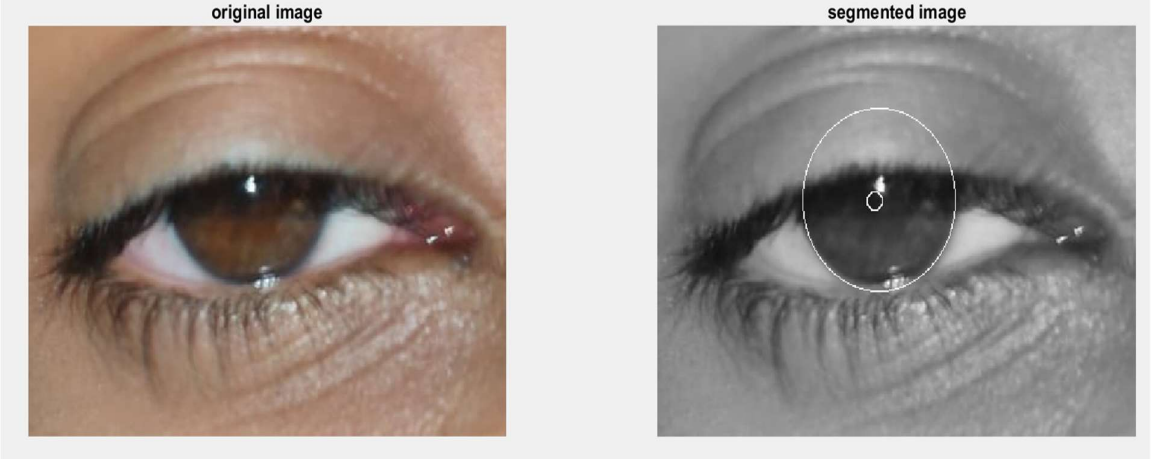


5. Live captured 50% opened eyes

1.



2.



Inferential Discussion:

Iris segmentation is the first part of Iris recognition which is one the most accurate bio metric recognition technique used.

In particular, a critical step of the recognition process is the segmentation of the iris pattern in the input face/eye image. This process has to deal with the fact that the iris region of the eye is a relatively small area, wet and constantly in motion due to involuntary eye movements. Moreover, eyelids, eyelashes and reflections are occlusions of the iris pattern that can cause errors in the segmentation process. As a result, an incorrect segmentation can produce erroneous biometric recognitions and seriously reduce the final accuracy of the system.

Conclusions:

The above program is used for segmenting the Iris and the Pupil in the given eye image.
Testing the algorithm for different formats of images and also different test eye images with fully opened and 75% opened and 50% opened eye images gives the following results.