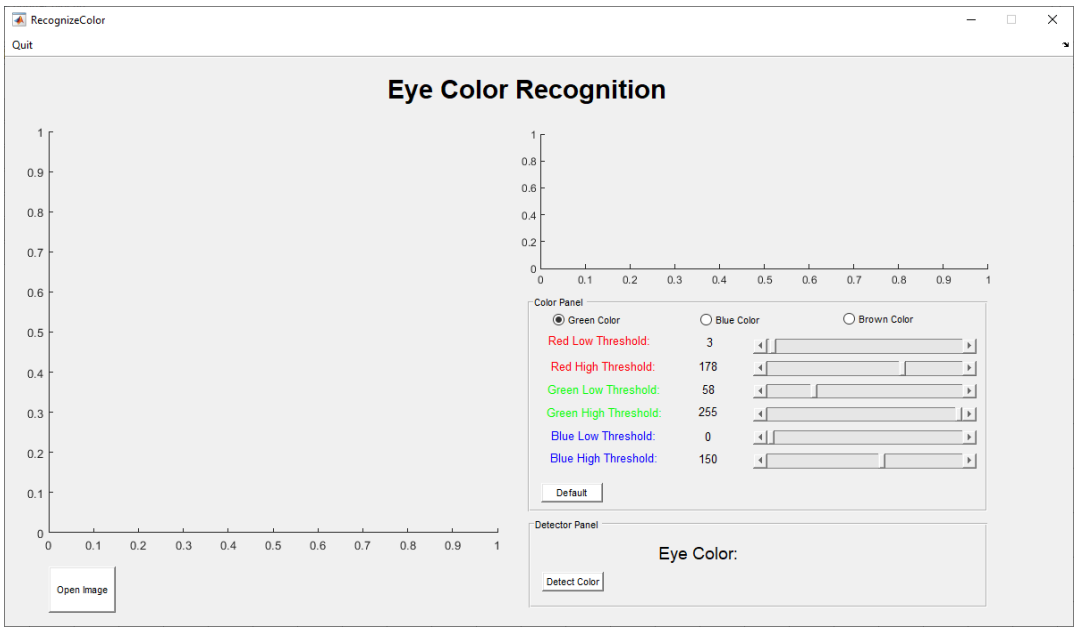
Color Eyes Recognition

Octavio Sales Calvo 902769
Emilio Miralles Rivera 902919

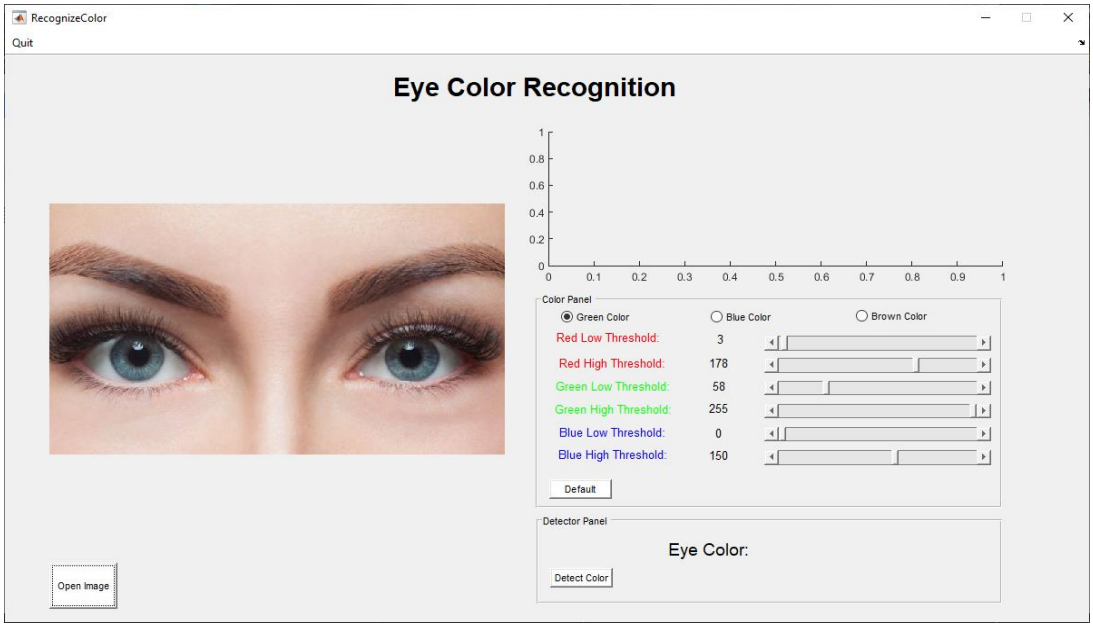
Contents:

1. GUI Application

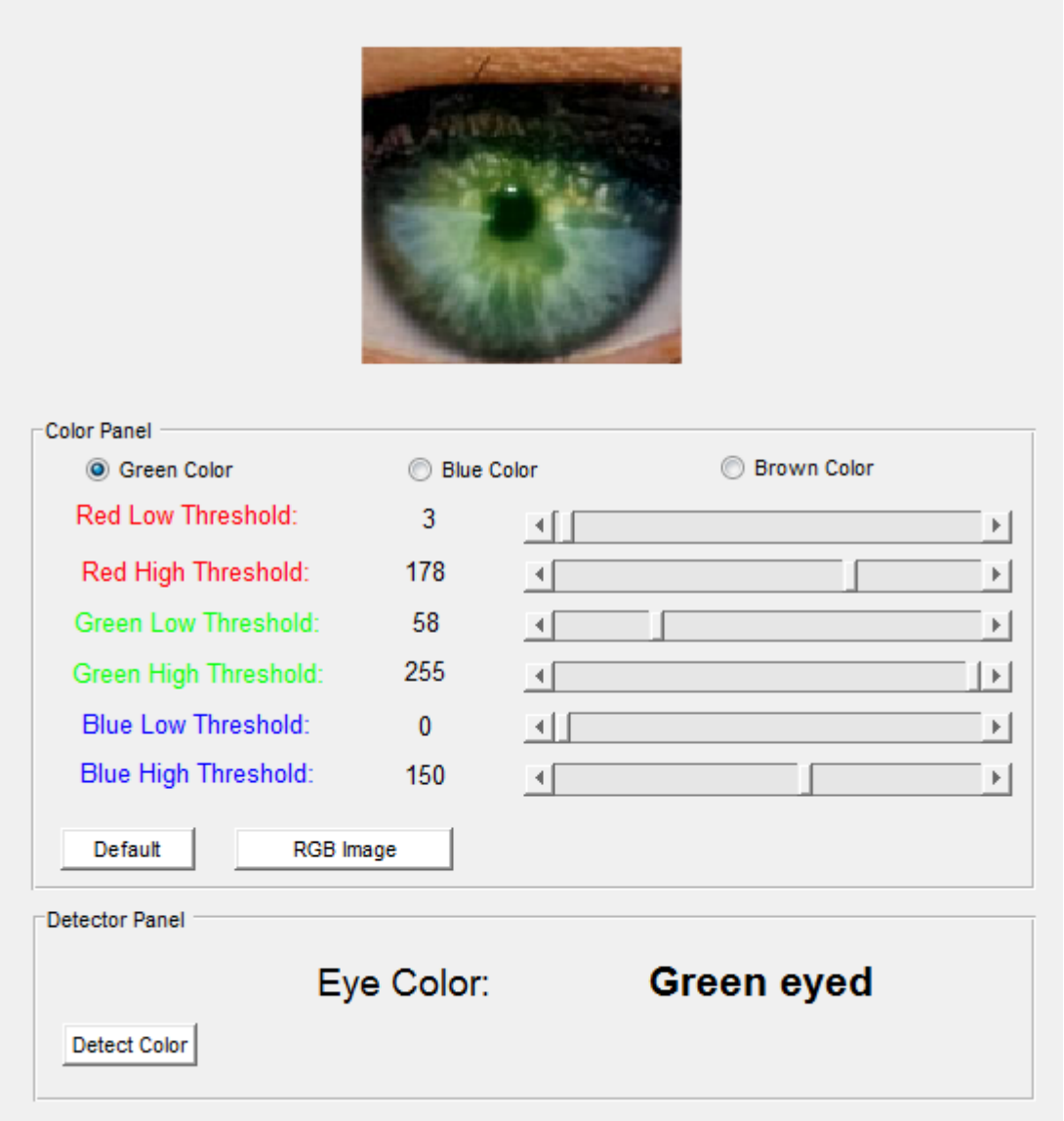
Our Graphical User Interface is intuitive and easy to use. The GUI has several different parts, we will talk about each of them in the following section. The figure below shows Eye Color Recognition GUI in the initial state.



First of all, **Quit menu** has been develop for close the application or even Matlab.
On the left side there are an axes where you can upload the target picture from your Matlab directory. User can upload this image by clicking on the **Open Image button** below the axes. Images can be upload at any time by user.



As the figure below shows, on the right side there are different elements:



On the top of the figure, the application plots the cropped image that fits with the eye's boundaries in the axes.

Under the axes, there is the Color Panel which allows user to set the thresholds of the three color matrix, called RGB matrix, User can easily manage the **low and high thresholds values**. Green **Color radio button** allow User to change the threshold of Green values. Blue color radio button and Brown are implemented in the same way as Green button. The application gives the option to reset the threshold values by clicking on **Default button**. Also has **RGB Image button**, this functionality obtains the RGB intensity profile from one line, which previously is made by the user, over the eye image.

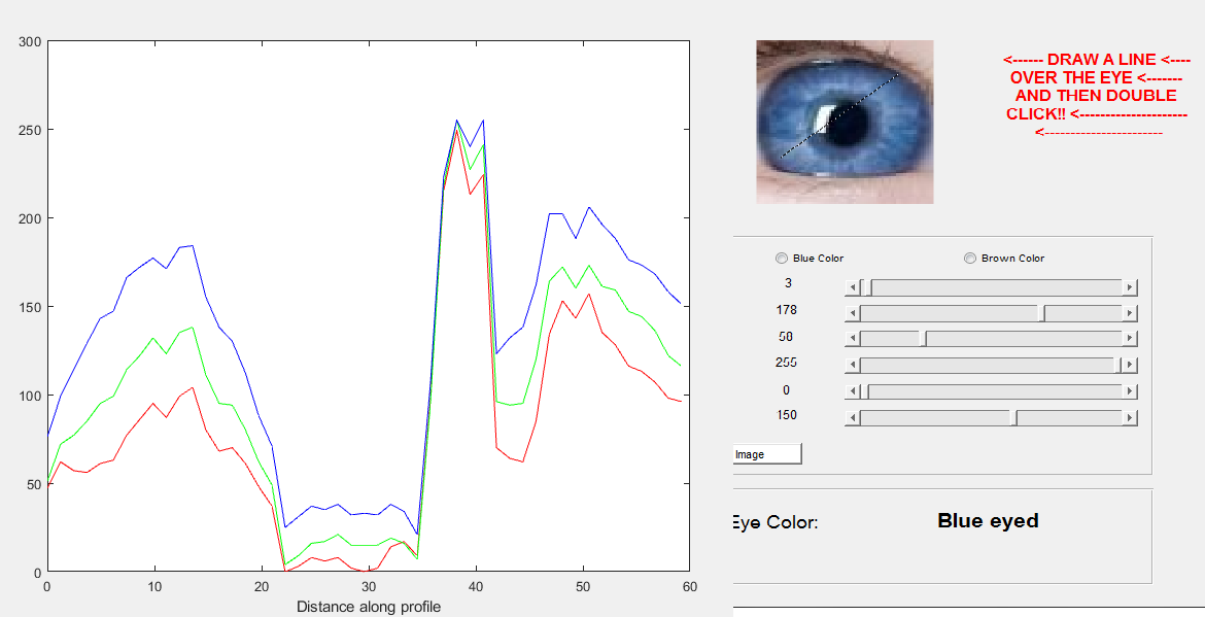


Fig: RGB intensity profile

Finally, on Detector Panel the application displays the eye's color detected from the image. Other application functionality is that plots in new windows **the three eye color maps** of the eye image, where User can understand why the application selects one eye color and the importance of set the thresholds properly. For carry this out, the user has to press **Detect Color button**.

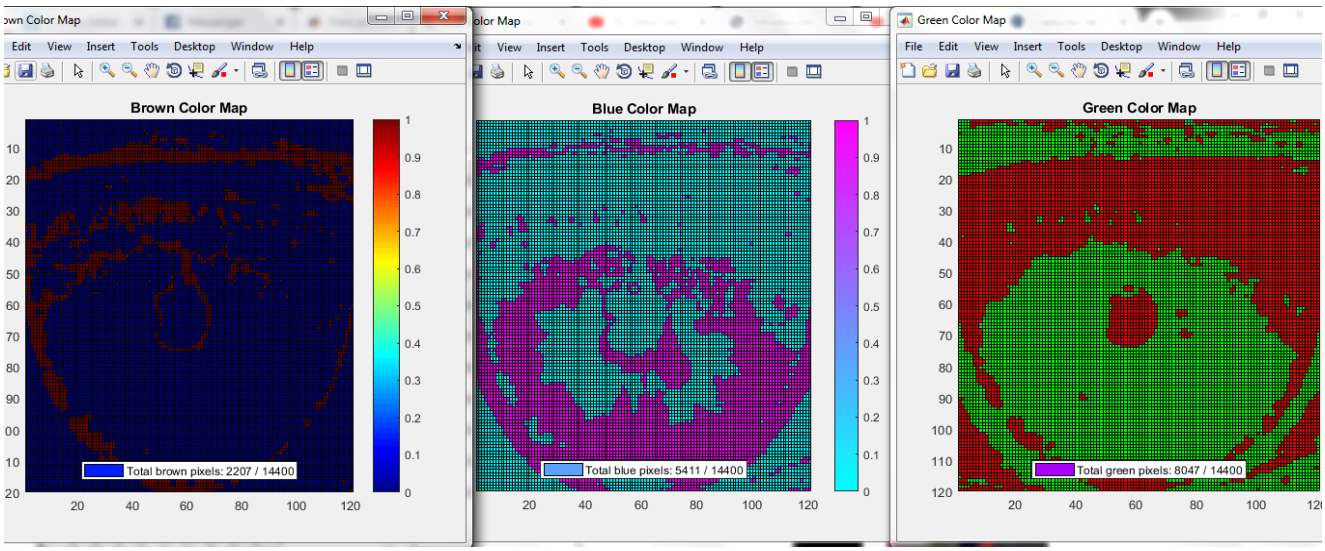


Fig: color maps

2. Algorithms explanation

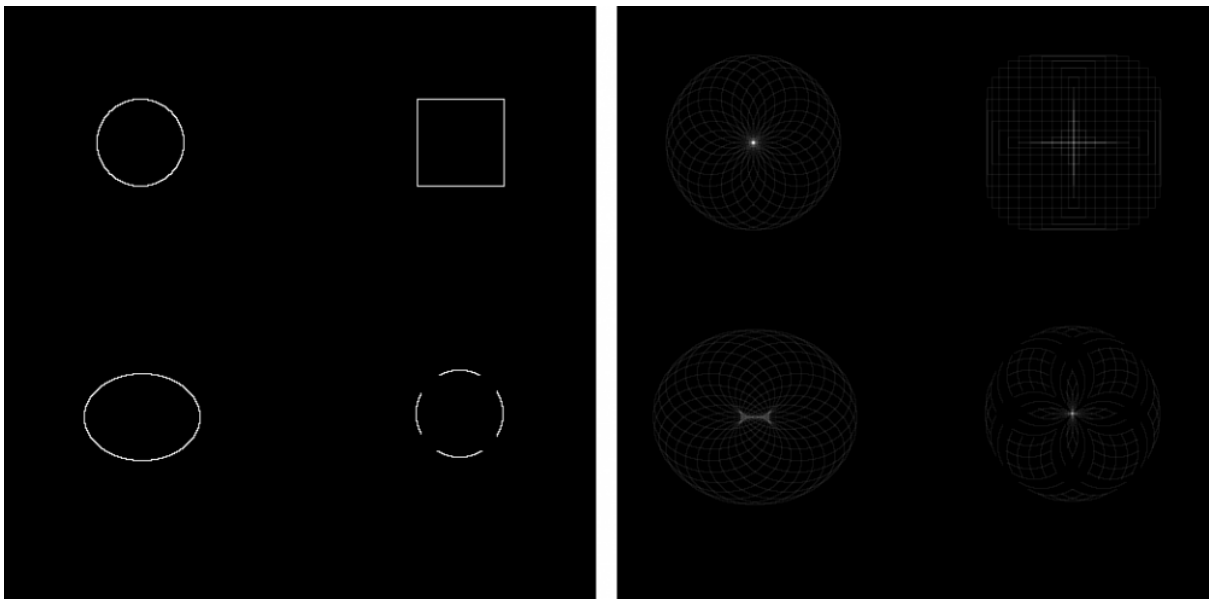
On the application there are two different parts.
The first part, to recognize the color of the eyes, is find the eye's boundaries on the original image uploaded.
The second part, consists of analyze only the part of the image where is one of the eyes. Comparing RGB matrix values of the cropped image and color threshold values the application finds out the color of the eye. The algorithm only has three parameterized colors: green, blue and brown. The application doesn't detect any more eye colors.
Talking deeper about the first part, the application uses a function which is based on Hough transform.
Hough transform is used to detect geometrical forms as circumference which can be represented by mathematical expression. The equation of a circumference is presented below:

$$(x - c_x)^2 + (y - c_y)^2 = r^2$$

where

r - radius of the circumference
c_x -x center coordinate
c_y - y center coordinate

The transform effectively searches for objects with a high degree of radial symmetry, with each degree of symmetry receiving one "vote" in the search space.



Hough circle transform is specific to circular objects. *Left Panel:* This panel shown the input data for the Hough circle transform. The data includes (clockwise from top left) a circle (radius 37 pixels), a square (length 37 pixels), an ellipse (minor axis 37 pixels), and a sectored circle (radius 37 pixels).*Right Panel:* This panel shows the output of a 24 step Hough circle transform. As you can see, the circle and the sectored circle converge to local maxima, while the square and ellipse do not, show the specificity of the transform for circular objects.

The method works by transforming an image around in a circle. Each time a transformed pixel with an intensity greater than zero lands on a Cartesian coordinate, that coordinate

gets one vote. As the image continues to be transformed in a circle of a given radius, if a circle in the image has the same radius, then votes will accumulate at the centroid of this circle. Therefore, by finding the maxima in the transform (points with the highest number of votes) you can find the centroid of circles within the image. A Hough circle transform can also be used to find circles of an unknown radius by searching a 3D transform space, where the the third dimension is the range of radii to be tested.

3. Project assumptions

We are going to enumerate the assumptions done in this project:

- ❖ We assume that both eyes are the same color.
- ❖ We assume that people can only have eyes of three different colors (blue, brown and green).
- ❖ We assume that input pictures of our application have to be face pictures focusing on eyes.

4. Application code

As mentioned before there are two main parts, the **eyes recognition** and the **color detector**:

- **Eyes recognition**:

In this part, first of all, the user must select and upload a picture, jpg format, to the application; to make it possible we use Matlab function: ”[file,dir]=uigetfile();” which allows to open files from disk to Matlab environment, already uploaded the picture the application read the picture using: “img=imread()”. All of this is implemented in “Upload image” button function:

```
% --- Executes on button press in openimagebutton.
function openimagebutton_Callback(hObject, eventdata, handles)
% hObject    handle to openimagebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[file,dir]=uigetfile('*.jpg','Select image to open');
if isequal(file,0)
else
    handles.myImage = imread(fullfile(dir,file));
    myImg=handles.myImage;
    imshow(myImg,'Parent',handles.image);

end
```

In the application it is used the function: “imfindcircles()” which finds circles using circular Hough transform, due to this function the application is able to recognize circles, in this case eyes in the image, getting from it the centers and the radii.The parameters function are the image, the range of radius length, and some properties, in our case: “ObjectPolarity” which indicates whether the circular objects are brighter or darker than the background, we chose ‘Dark’, the other one property is the sensitivity factor that increasing this factor the function will detect more circular objects, including weak and partially obscured circles. According to the circle's radius
range the function looks for from 1/15 to ⅙ image size. After detect the eyes the application crops the image regarding to the center and radius given.

```
Igray = rgb2gray(img);
[centers,radii] = imfindcircles (img, [fix(size(Igray,1)/15) fix(size(Igray,1)/6)],'ObjectPolarity',...
'dark','Sensitivity',0.9);
viscircles(centers,radii);
Icrop = imcrop(img,[centers(1,1)-radii(1) centers(1,2)-radii(1) radii(1)*2 radii(1)*2]);
imshow(Icrop,'Parent',handles.cropImage);
```

- **Color detector:**

The application is developed with two global variables: “colorThreshold” and “cButton”. Where colorThreshold is a RGB matrix that contains the six threshold of each eye’s color detected (three). Variable colorThreshold get the values from the slider callback functions of the GUI. cButton is a flag variable that indicates which radio button is active to modify the color threshold.

After the eyes recognition and threshold already established, the application calls to the “recogniseColor()” function which allows to know the eyes color. this function needs the following parameters: the cropped image (Icrop), the RGB matrix (colorThreshold) and the color flag (cButton). Let's introduce this function:

```
color = recogniseColor(Icrop, colorThreshold, cButton);
```

First of all it creates a RGB matrix by declaring separated arrays of each color, also sets the mask for each color, these masks are implemented as boolean logical way: ‘1’ if the image RGB values are between the thresholds, ‘0’ in the opposite case. After the calculation of the masks, the function combines them by using a logical AND operator, and it makes the addition of all the elements from the array for each color.

```
GREEN_redMask=(r > colorThreshold(1,1) & r < colorThreshold(1,2));
GREEN_greenMask=(g > colorThreshold(1,3) & g < colorThreshold(1,4));
GREEN_blueMask=(b > colorThreshold(1,5) & b < colorThreshold(1,6));

BLUE_redMask=(r > colorThreshold(2,1) & r < colorThreshold(2,2));
BLUE_greenMask=(g > colorThreshold(2,3) & g < colorThreshold(2,4));
BLUE_blueMask=(b > colorThreshold(2,5) & b < colorThreshold(2,6));

BROWN_redMask=(r > colorThreshold(3,1) & r < colorThreshold(3,2));
BROWN_greenMask=(g > colorThreshold(3,3) & g < colorThreshold(3,4));
BROWN_blueMask=(b > colorThreshold(3,5) & b < colorThreshold(3,6));

green= (GREEN_redMask & GREEN_greenMask & GREEN_blueMask);
blue= (BLUE_redMask & BLUE_greenMask & BLUE_blueMask);
brown= (BROWN_redMask & BROWN_greenMask & BROWN_blueMask);
```

Finally, the application compares those additions and decides the color based on which one is the greatest.

```
green= (GREEN_redMask & GREEN_greenMask & GREEN_blueMask);
blue= (BLUE_redMask & BLUE_greenMask & BLUE_blueMask);
brown= (BROWN_redMask & BROWN_greenMask & BROWN_blueMask);


if(sum(sum(green))> sum(sum(blue)) && sum(sum(green)) > sum(sum(brown)))
    color=1;
elseif(sum(sum(blue))> sum(sum(green)) && sum(sum(blue)) > sum(sum(brown)))
    color=2;
elseif(sum(sum(brown)) > sum(sum(green)) && sum(sum(brown)) > sum(sum(blue)))
    color=3;
else
    color=-1;
end
```

- **Other Functionalities:**

To plot the RGB intensity profile the application uses the “improfile” function. And for the three color maps, it uses “pcolor()” and “colormaps” functions.

```
imshow(Icrop,'Parent',handles.cropImage)
set(handles.helpText,'String','<------ DRAW A LINE <---- OVER THE EYE <-------- AND THEN DOUBLE CLICK!!
improfile
set(handles.helpText,'String','');
```

```
num=(sum(sum(brown)));
total=(size(green,1)*size(brown,1));
figure(f2);
pcolor(brown);
colormap jet
axis ij
title('Brown Color Map');
legend([ 'Total brown pixels: ', num2str(num), ' / ', num2str(total)], 'Location', 'south');
colorbar;
```

5. Project difficulties

The most difficult part of the project was find how to recognize the eyes on the pictures.
At the beginning of the project, we try to use a image processing toolbox called Computer Vision Toolbox where it is implemented the Viola-Jones algorithm. Unfortunately, we couldn't solve the problem with this Matlab toolbox because of sometimes the function step used for find out the eyes boundary box didn't work properly. We couldn't make the boundary box around the eyes of the person in the picture.
After this issue, we realized that there was the function imfindcircles which works with Hough transform for detect circles on different images.
Even using this function, the used parameters for recognize the circles were changed several times to solve all the probable cases. We test fourteen images to confirm that the application works properly.

6. References

1. [Computer Vision Toolbox used on the first attempt]
   https://www.mathworks.com/help/vision/getting-started-with-computer-vision-system-toolbox.html?s_tid=CRUX_lftnav

2. [API function imfindcircles]
   https://www.mathworks.com/help/images/ref/imfindcircles.html

3. [Documentation about Hough circle transform]
   https://imagej.net/Hough_Circle_Transform

4. [API function uigetfile]
   https://www.mathworks.com/help/matlab/ref/uigetfile.html

5. [Thresholds for RGB colors]
   https://en.wikipedia.org/wiki/RGB_color_model

6. [API function improfile]
   https://www.mathworks.com/help/images/ref/improfile.html
   https://www.youtube.com/watch?v=TYfKzxBsSeM

7. [API function pcolor]
   https://www.mathworks.com/help/matlab/ref/pcolor.html

8. [API function colormap]
   https://www.mathworks.com/help/matlab/ref/colormap.html