Enrique J. Gonzalez Mendez

INEL 6080 Computer Vision Project 2020

Prof. M. Toledo

Eye Gaze Detection

Approach:

The initial approach is represented in Figure 1. The face region detection part was going to utilize the viola-jones algorithm [2] to detect the face, but we later discover that we can go directly to the eye region using the same algorithm, thanks to it being trained to detect the eyes as well. After taking the eye region, we crop this part of the image for further transformations and identification of the iris. After acquiring that image, we do some morphological operations and segmentation techniques to get an accurate area representation of the iris and with that we can calculate the center of it by using regionprops function that MATLAB provides. With all this we can now get the center of the iris quite easily without doing much computational expensive operations like the Hough transform.

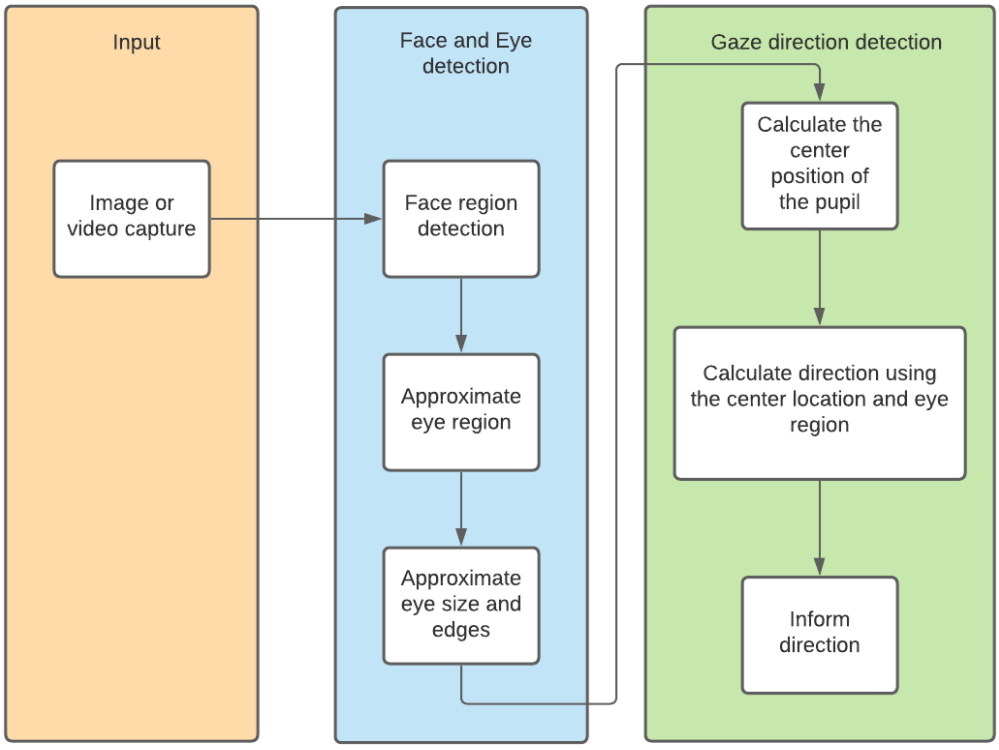

Figure 1: Initial System Block Diagram

Interesting obstacles or findings:

- The imfindcircle function could not detect any round figures in the image, even when adjusting the sensitivity and the object polarity. To solve this, we use the regionprops function which gives properties of image regions and with this we can instead search for the iris ourselves using these properties. It also makes our algorithm more robust thanks to it not having to depend on the imfindcircle function which is very sensitive with respect to the contrast of images and other things.

- With a threshold of 0.1 using the imbinarize function we filter almost everything, only the pupil and some eye leashes get into the final image of the test images.
- We need to note that the operations that we do on every frame cannot be to extensive due to the nature of a live video. The frame is only going to be there according to the rate of acquisition, commonly that is 30 frames per second, which means that every frame will be there roughly 33.33ms to do actual operations on it. That limits us in a way that we cannot do to many operations or the algorithm will then not have enough time to do its thing. This means that we should work, every time that we can, in only limited areas in the picture. Using the viola-jones algorithm we can reduce the region that we need to work on thanks to it being able to identify the eye region.

Requirements:

- MATLAB 2020b
- Image Processing Toolkit: Contains standard algorithms for image segmentation, noise reduction and morphological transformations.
- Computer Vision Toolkit: has viola-jones algorithm already train, and we use it to detect the eye region.
- Image Acquisition Toolkit: for live video setup and to get every frame as an image to do all the operations that we want.

Important Functions

- bwmorph: does morphological operations to the image. We use it to dilate and erode and fill parts of the images when looking at the eye region.
- imbinarize: filers the image using automatic thresholding, it is recommended in MATLAB instead of im2bw. We can also input the threshold manually.
- imCrop: crops the given image with the rectangle given has a second parameter. We use this function to reduce the area of the image where we are going to be making morphological operations, saving computation time on our process, and making the identification of the iris a lot easier by not having to consider other parts in the image.
- Regionprops: gets the images properties like area, centroids, and bounding box. This are essential to detect the iris region and the center of them.

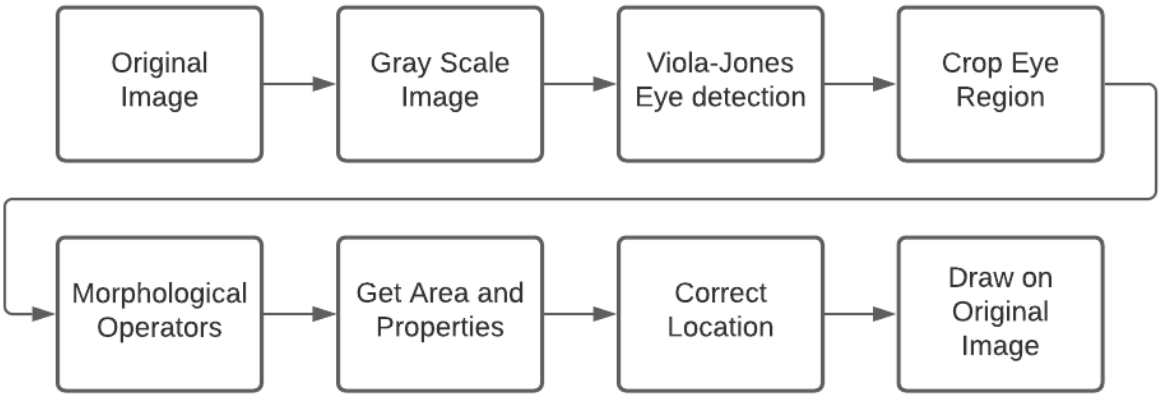

Figure 2: Overall Detection Algorithm

Figure 3: Pupil detected

Figure 4: Both center of the eyes detected

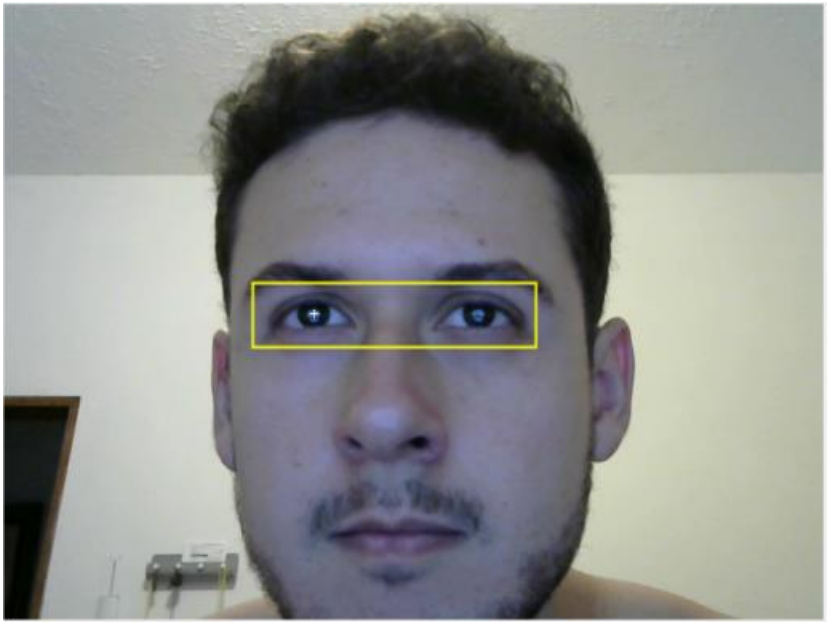

Figure 5: Localizing center of iris in live video

Code:

We first setup the necessary objects such as the eye detector and the camera. We grab a snapshot to then set the frame size and the video player size. After this we setup the variables for the frame count and a Boolean variable that will help stop the script. After starting the camera and the eye detector, we grab a new snapshot of the video, turn it to a grayscale image, detect the eye region with viola-jones and then we crop the eye region for further processing.

```
%-------------------------------------------------------------------------
% Create the face detector object.
eyeDetector = vision.CascadeObjectDetector('EyePairSmall');

% Create the webcam object.
cam = webcam();

% Capture one frame to get its size.
videoFrame = snapshot(cam);
frameSize = size(videoFrame);

% Create the video player object.
videoPlayer = vision.VideoPlayer('Position', [100 100 [frameSize(2), frameSize(1)]+30]);

runLoop = true;
frameCount = 0;
%-------------------------------------------------------------------------
```

Figure 6: Setup code

```
while runLoop && frameCount < 400

    % Get the next frame.
    videoFrame = snapshot(cam);
    videoFrameGray = rgb2gray(videoFrame);
    frameCount = frameCount + 1;

    % Detection mode.
    bbox = eyeDetector.step(videoFrameGray);

    % Crop image after detecting eye region
    s = length(bbox(:,4));
    I2 = imcrop(videoFrameGray,bbox(s,:));
    grayImage = I2;
    %-------------------------------------------------------------------------
```

Figure 7: Capturing image and converting it to grayscale

After acquiring the frame that represents the eye region, we binarize the image with a 0.1 threshold to only have the iris area filer through. We then use a closing and an opening to first fill small holes and then elimination noise from the image. We then again try to remove unconnected components from the image and then we fill any holes that might be inside connected components. We then utilize the regionsprops function to get the properties of the image ,this includes the area, centroids, and the bounding box, which we will utilize all of them to draw the approximation of the center of the iris. We then pass to correct the location given from the crop image and pass it to the original image coordinates. We can see this in Figure 9. After this we enter inserting the shapes that are aquire and puttin them in the videoframe for the user to see. In Figure 14 detected we can see the result after correcting and marking the area of interest in the user's frame, but we still only have one of the irises.

```
% Turn to binary image to filter the skin
newImage = ~imbinarize(grayImage,0.10);
newImage = bwmorph(newImage,'close');
newImage = bwmorph(newImage,'open');
newImage = bwareaopen(newImage,200);
bwI = imfill(newImage,'holes');
% Morph operations done----------------------------------------------------
% Get areas and properties
stats = regionprops(bwI);
N = size(stats,1);
if N < 1 || isempty(stats)
    eyesNotDetected = []; % eyes not found
    continue
end
areas =  stats.Area;
centroids = stats.Centroid;
[areaMax, iMax]=max(areas);

```

Figure 8: Morphological operations

```
%-------------------------------------------------------------------------
% Get correct location on original image
stats(iMax).BoundingBox(1) = stats(iMax).BoundingBox(1) + bbox(1);
stats(iMax).BoundingBox(2) = stats(iMax).BoundingBox(2) + bbox(2);

%Get eye center

center=round(stats(iMax).Centroid);
X=center(1) + bbox(1);
Y=center(2) + bbox(2);

```

Figure 9: Correct locations

After successfully acquiring one of the irises, we explore other ways to get the second one. The first approach was to crop the eye region sub image into two separate images, left eye image and right eye image, and then run the algorithm on both separate images. This approach did not work well on a live video, so it was discarded. The approach we took was to stay with one eye region image but to try and get two centroids and two areas that each one corresponds to the irises. In Figure 10 we show how we set up the regionprops function to get both irises successfully. The resulting image is seen in Figure15.

```
24 -    centroids = stats.Centroid;
25 -    [areaMax1, iMax1] = max([stats.Area])
26 -    stats(iMax1).Area = 0;
27 -    [areaMax2, iMax2] = max([stats.Area])
28
29
30
31      %-------------------------------------------------------------------------
32      % Get correct location on original image
33 -    stats(iMax1).BoundingBox(1) = stats(iMax1).BoundingBox(1);
34 -    stats(iMax1).BoundingBox(2) = stats(iMax1).BoundingBox(2);
35 -    stats(iMax2).BoundingBox(1) = stats(iMax1).BoundingBox(1);
36 -    stats(iMax2).BoundingBox(2) = stats(iMax1).BoundingBox(2);
37
38
39      %Get eye center
40 -    center1=round(stats(iMax1).Centroid);
41 -    X1=center1(1);
42 -    Y1=center1(2);
43 -    center2=round(stats(iMax2).Centroid);
44 -    X2=center2(1);
45 -    Y2=center2(2);
46
```

Figure 10: Modified code to detect both irises

```
63 -        if ~isempty(bbox)
64
65
66              % Convert the rectangle represented as [x, y, w, h] into an
67              % M-by-2 matrix of [x,y] coordinates of the four corners. This
68              % is needed to be able to transform the bounding box to display
69              % the orientation of the eyes.
70 -            bboxPoints = bbox2points(bbox(1, :));
71              % Convert the box corners into the [x1 y1 x2 y2 x3 y3 x4 y4]
72              % format required by insertShape.
73 -            bboxPolygon = reshape(bboxPoints', 1, []);
74              % Check for Iris location in between the already detected eye
75              % Display a bounding box around the detected eyes.
76 -            videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon, 'LineWidth', 3);
77              % Display Iris shape
78 -            videoFrame = insertShape(videoFrame, 'Circle', [X,Y,stats(iMax).BoundingBox(3)/2], 'LineWidth', 2);
79              % Display detected corners.
80 -            videoFrame = insertMarker(videoFrame, [X,Y], '+', 'Color', 'white');
81 -        end
```

Figure 11: Mark areas on video frame

```
82
83          % Display the annotated video frame using the video player object.
84 -        step(videoPlayer, videoFrame);
85
86          % Check whether the video player window has been closed.
87 -        runLoop = isOpen(videoPlayer);
88 -    end
89
```

Figure 12: Display the video with mark areas

```
90      % Clean up.
91 -    clear cam;
92 -    release(videoPlayer);
93 -    release(eyeDetector);
```

Figure 13: Clean up code

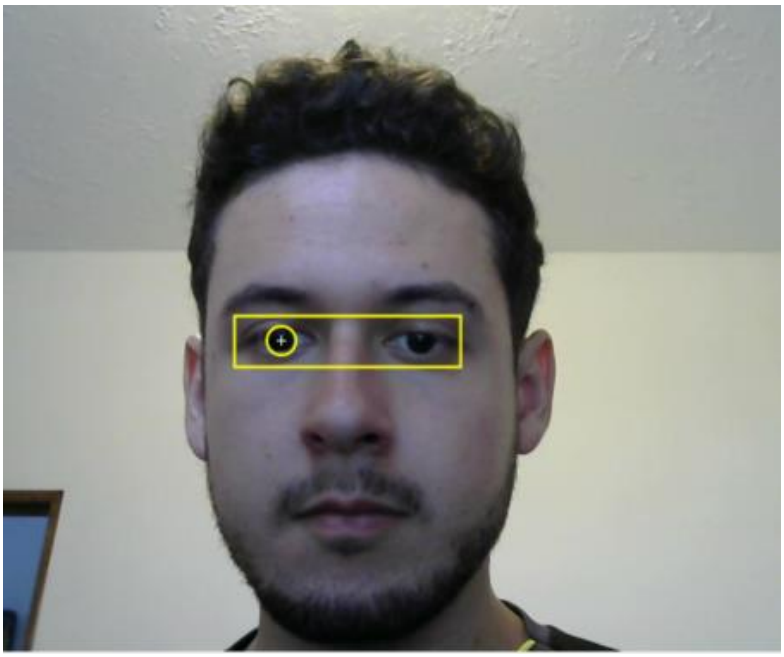

Figure 14: Iris area detected

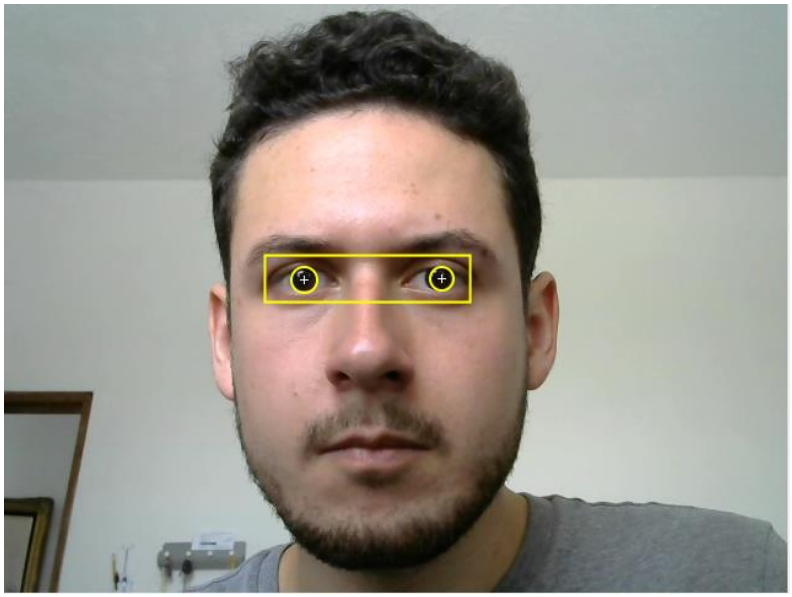

Figure 15: Both irises detected

Future work:

Here we could see that it us easily feasible the detection of the center of the iris. For future work we can add a camera calibration algorithm to position the user with respect to the camera more clearly and to suggest that the user is seen the screen and not somewhere else. Additionally, other uses can include test that need to consider the users eye gaze, a lot of research is done in terms of psychological test in this area. We can also utilize this for simple VR headsets. One thing to notice is that the algorithm does not work well if the target is too close or too far away from the camera. The measure distance for it to work appropriately is in the range of about 1’6’’ feet to 2’0’’ feet. Other areas to work on the future is to optimize the code for it to run more smoothly given that it should work on a live video and to calculate the gaze direction.

References:

[1] Hassaballah, M. & Murakami, Kenji & Ido, Shun. (2011). An Automatic Eye Detection Method for Gray Intensity Facial Images. International Journal of Computer Science Issues. 8. 272-282.

[2] Viola, Paul & Jones, Michael. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE Conf Comput Vis Pattern Recognit. 1. I-511. 10.1109/CVPR.2001.990517.

[3]Mustafa UCAK (2007), Face & eye detection. File Exchange in MathWorks. https://www.mathworks.com/matlabcentral/fileexchange/13716-face-eye-detection

[4] Detect and Track Face MATLAB example. https://www.mathworks.com/help/vision/ug/detect-and-track-face.html;jsessionid=c295b4e92294c404e0728051de87

[5] MATLAB Documentation in general