Iris Recognition Using Canny Edge Detection and Freeman Chain Code Algorithm

[1]Nitasha Singla, [2]Amarjeet Kaur, [3]Anjana Sharma
[1,2,3]Chandigarh Engineering College, Landran, Mohali, Punjab, India

Abstract
Circle detection is very important for many applications. Planets, movement of planets, even cars, scooters, buses we would not be able to drive because wheels are circle when we play sports, we use balls- which are spheres, or three dimensional representations of circles. So circle plays very important role in our life. In this paper, we detect centre of iris using canny edge detection and Freeman chain code algorithm.

Keywords
Iris, Canny edge detection algorithm, Freeman Chain Code

I. Introduction

As we know iris is circular in nature. Iris recognition system mainly includes eye image capturing, image pre-processing and edge detection through iris region segmentation, feature extraction and pattern matching. Among them edge detection is one of the major part in iris recognition system. Edge detection technique detected pupil boundary detection accurately and easier. A circular canny edge detection method is used to look for a circle in the image which has maximum gray level difference with its neighbor [1]. Circular hough transform uses different approaches which are computationally complex [2]. In the subsequent sections we are presenting the iris detection using canny edge detection and freeman chain code algorithm.

II. Canny Edge Detection

There are various edge detection algorithms like: Sobel operator, Priwitt's operator, Robert's cross operator and canny edge detection algorithm. Canny edge detection algorithm is best out of these because canny edge detection algorithm uses a multi-stage process to detect a wide range of edges from the images [3].

A. Convert Colored Image to Grayscale: There is no reason why you could not do canny edge detection on a color image, but I encourage you to first convert the image to grayscale using some sort of RGB → Grayscale.

B. Image Smoothing

In this step noise is removed from the image using Gaussian filter. Filtering is done using a simple mask, it is used exclusively in the Canny algorithm. The Gaussian smoothing can be performed using standard convolution methods. A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. The larger the width of the Gaussian mask, the lower is the detector's sensitivity to noise. The localization error in the detected edges also increases slightly as the Gaussian width is increased. Image smoothing is done primarily to suppress noise and to get a continuous edge contour during the non-maximum suppression process. The output thus obtained is a blurred intermediate image.

C. Compute Edge Strength and Direction of Edges

A. Edge Strength calculation

In this stage we use sobel's operator. The Sobel operator is a discrete differential operator that generates a gradient image. Horizontal and vertical Sobel operators that are used to calculate the horizontal and vertical gradients D_x and D_y, respectively, are shown below:

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

To obtain the gradient image, a smoothened image from the first stage is convolved with the horizontal and vertical Sobel operators.

$R_x = (I*D_x)$ (1)

$R_y = (I*D_y)$ (2)

In (1) and (2), I is the image obtained after Image smoothing; R_x and R_y are images with pixel values that are the magnitude of the horizontal and vertical gradient, respectively. Images R_x and R_y from equations (1) and (2) are used in equation (3) to obtain the "edge strength" of a pixel in an image. The edge strength is

$R = \sqrt{R_x^2 + R_y^2}$ (3)

B. Edge Direction Calculation:

Edge Direction is Determined Using this Formula:

$B=\tan^{-1}(R_y/R_x)$ (4)

The edge directions obtained from equation (4) are rounded off to one of four angles--0 degree, 45 degree, 90 degree or 135 degree--before using it in next step.

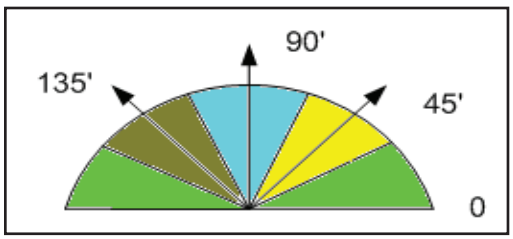


Fig. 1: Edge Directions are Rounded off to one of the Four Angles to be Used in Step.

D. Non Maximum Suppression

In the last step the edges it finds can be either very thick or very narrow depending on the intensity across the edge and how much the image was blurred first. One would like to have edges that are only one pixel wide. This step keeps only those pixels on an edge with the highest edge strength [4].

Three pixels in a 3 × 3 around pixel (x, y) are examined.
If A = 0^0, then the pixels (x + 1, y), (x, y), and (x − 1, y) are examined.
If A= 90^0, then the pixels (x, y + 1), (x, y), and (x, y − 1) are examined.
If A= 45^0, then the pixels (x + 1, y + 1), (x, y), and (x − 1, y − 1) are examined.
If A= 135^0, then the pixels (x + 1, y − 1), (x, y), and (x − 1, y + 1) are examined.
If pixel (x, y) has the highest edge strength of the three pixels examined, it is kept as an edge otherwise should not be classified as an edge pixel.

E. Hysteresis Thresholding

Thresholding with hysteresis is the last stage in canny edge detection, which is used to eliminate spurious points and non-edge pixels from the results of non-maximum suppression. Some of the edges detected by Steps 1–3 will not actually be valid, but will just be noise. We would like to filter this noise out.
Thus all the edges in an image are detected using the canny edge detection Algorithm.

III. Freeman Chain Code Algorithm

After Non-maximum suppression and thresholding, we have thin edges for all the possible object boundaries in the image. These thin edges are contour traced to detect the potential circle's edge pixels. One of the well known algorithms for contour tracing is Freeman Chain code [5].
This code is a list of codes ranging from 0 to 7 in clockwise direction which represents the direction of the next pixel connected in 3*3 windows as shown in Table 1. The coordinate of the next pixel is calculated based on the addition and subtraction of columns and rows by 1, depending on the value of the code.

Table 1: Relation of Pixel and Chain Code with Current Pixel

	Column-1	Column	Column+1
Row-1	5	6	7
Row	4	Current Pixel	0
Row+1	3	2	1

Table 2: Coordinates of Next Pixel Calculated Based on the Chain Code for Current Pixel (x,y)

Code	Next Row	Next Column
0	x	y
1	x + 1	y + 1
2	x + 1	y
3	x + 1	y – 1
4	x	y – 1
5	x -1	y – 1
6	x – 1	y
7	x – 1	y +1

A. Finding Arcs by Contour Tracing

After the edge detection, linking algorithms are designed to assemble edge pixels into meaningful edges and/or region boundaries [5]. One of the simplest approach for linking the edge points is to analyze the characteristics of the pixels in a small neighborhood about every point (x,y) that has been declared an edge point by using method that we have discussed earlier. All the points that are similar according to predefined criteria are linked, forming an edge of pixels that share common properties according to the predefined criteria.
There are two principal properties to track the edges which form the boundaries of the object. One is based on the edge strength and other one is based on the pixel direction. Based on the pixel direction, property states,"An edge pixel at (x_0, y_0) in the predefined neighborhood of (x,y) has an angle similar to the pixel at (x,y) if

$$\left| \text{angle}(x,y)-\text{angle}(x_0-y_0) \right| < B \qquad (5)$$

Based on the Edge strength, property states, "An edge pixel at (x_0, y_0) in the predefined neighborhood of (x,y) has an magnitude similar to the pixel at (x,y) if

$$\left| \text{magnitude}(x,y) - \text{magnitude}(x_0, y_0) \right| < R \qquad (6)$$

Where R is the positive threshold and B is positive angle threshold. Pixels are linked with each other if both magnitude and direction criteria are satisfied.

The image is scanned to find a pixel with any pixel direction [5]. For example, let us consider that the pixel direction we are looking for is located at (x,y) with the pixel direction 4, as shown in fig. 2. Now we draw the normal perpendicular to the direction of pixel under consideration that is 4. We examine pixel (x-1, y-1) and (x+1, y+1) for larger gradient magnitude. From both these pixels, which pixel has higher gradient that pixel is given higher priority and second one is leave. So in the diagram pixel (x-1, y-1) has higher gradient so it is given first priority. We know that pixel (x-1, y+1) and (x+1, y-1) is discarded as, if these pixels had pixel direction 4 then this would have been made zero during the NMS stage. Now pixel (x, y-1) and (x-1, y) both have equal probability, hence it will not matter which is given higher priority. Here we are going to give pixel (x-1, y) second priority and pixel (x, y-1) the third priority.

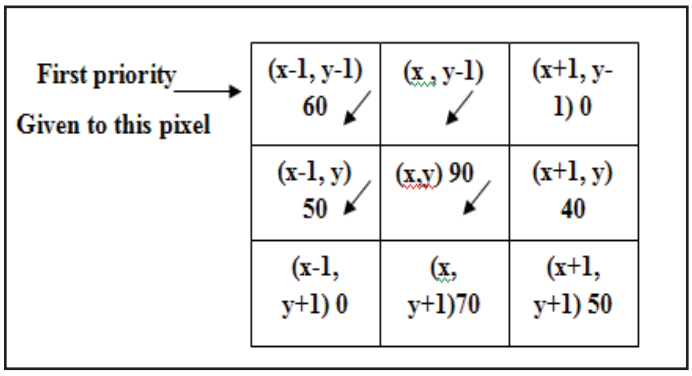


Fig. 2:

Hence, when (x,y) pixel is found having pixel direction 4, we first check if pixel (x-1,y-1) has the same pixel direction. If yes, then we update x to x-1 and y to y-1 and repeat the same process till we find neighboring pixels with a pixel direction other than 4. We consider this arc only if the arc length is at least 10 pixels; otherwise the arc is ignored and the scan for pixels with direction 4 is continued. This arc has angle in the range of 1960 - 2560.
Similarly, Contour tracing for pixel with direction 1, 2, 3, 5, 6, 7 or 8 is done. By doing this we get arcs at different-different direction or we can say that in different angles.

B. Arcs are to be part of circle or not

At the end of all these steps, Circle will looks like this as shown below.

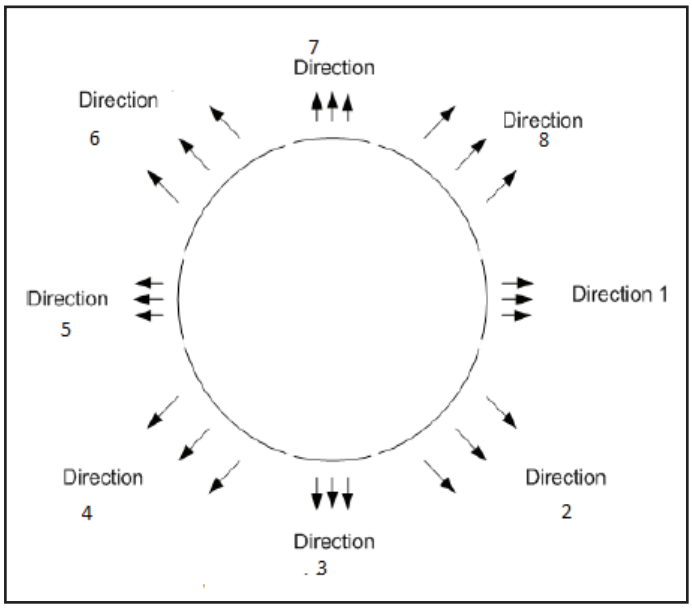


Fig. 2: Arcs of the Circle With Edge Directions

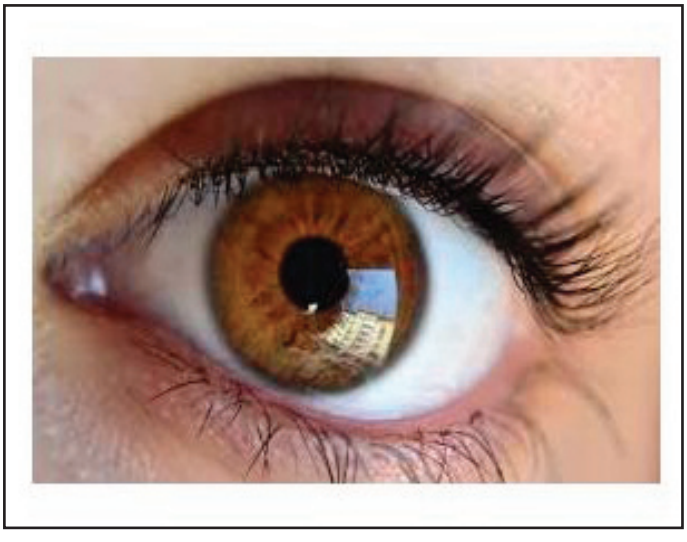
As we see that, circle consist of 4 major arc with pixel direction 2,4,6 and 8 with arc with pixel direction 1, 3, 5, and 7. Pixel Direction 1, 3, 5 and 7 acts as a connector.

"The condition for considering the arc as part of a potential circle is that there should be at least 3 arcs present with pixel directions 2, 4, 6 or 8 which are connected to each other by arcs with pixel directions 3, 5, 7 or 1".

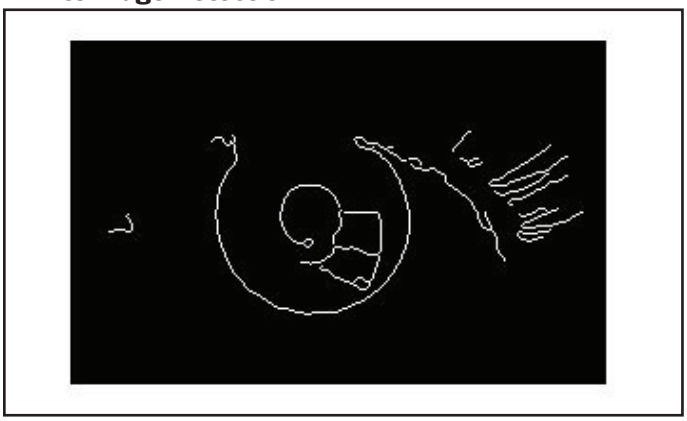
To find circle we scan the image to find arc with pixel direction 2 or 8. Once the arc with pixel direction 2 is found, we again scan the image at the end of the arc using the block of size 3*3, for a pixel part of the arc with pixel direction 3, as shown in figure 2. Similarly we repeat same process until we did not found 3 major arcs. Once accurate arcs are find out they are draw on the circle with blue color. By doing this, circle or we can say iris is detected from the image.

IV. Result and Future Scope

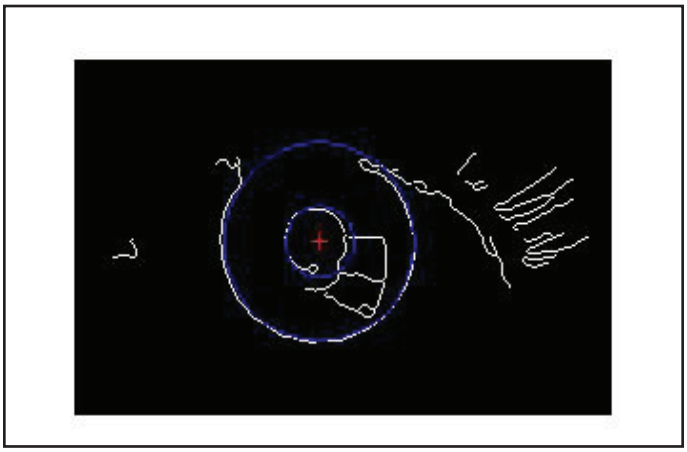
A. Original Image



B. After Edge Detection



C. After Iris Detection



As in this paper, we detect iris which is circular in nature, by using this method we can find different different circular objects. Even we can differentiate between 'O' and '0' by using this method.

References

[1] Daugman J,"How iris recognition works, Circuits and Systems for Video Technology"? IEEE Transactions on, 14 21, 2004.

[2] Wildes R P,"Iris recognition: An emerging biometric technology", Proceedings of the IEEE, 85 1348, 1997.

[3] R.C.Gonzalez, R. E. Woods,"Digital Image Processing", 3rd ed. Prentice Hall, 2009.

[4] B.Chanda, D.Dutta Majumder,"Digital Image Processing and Analysis", Prentice Hall 2003.

[5] Nitasha Singla, Reecha Sharma,"Extraction of circle from digital images", International Journal of Computer Applications & Information Technology, 2012.

[6] Bhawna Chouhan, Dr. (Mrs.) Shailja Shukla,"Iris Recognition System using canny edge detection for Biometric Identification", International Journal of Engineering Science and Technology (IJEST).