Locating An IRIS From Image Using Canny And Hough Transform

Poorvi Bhatt

Abstract: Iris recognition, a relatively new biometric technology, has great advantages, such as variability, stability and security, thus it is the most promising for high security environments. The proposed system here is a simple system design and implemented to find the iris from the image using Hough Transform Algorithm. Canny Edge detector has been used to get edge image to use it as an input to the Hough Transform. To get the general idea of Hough Transform, the Hough Transform for circle is also implemented. RGB value of 3-D accumulator array of peaks of inner circle and outer circle has been performed. And at the end some suggestions are made to improve the system and performance gets discussed.

Index Terms: 3D Accumulator, Biometrics, Canny Edge Detector, Gaussian Filter, Hough Transform, Iris Recognition, MATLAB

INTRODUCTION

Among various biometric technologies, such as finger-prints and face, iris recognition has a relatively short history of use. There are few large-scale experimental evaluations reported in the literature, and essentially none where the image dataset is available to other researchers. One constraint of current iris recognition systems, which is perhaps not widely appreciated, is that they require substantial user cooperation in order to acquire an image of sufficient quality for use. A number of groups have explored iris recognition algorithms and some systems have already been implemented and put into commercial practice by companies such as Iridian Technologies, Inc., whose system is based on the use of Daugman’s algorithm. [1]-[4] A typical iris recognition system generally consists of the following basic modules:

I. image acquisition, iris location, and pre-processing,
II iris texture feature extraction and signature encoding, and
IIIiris signature matching for recognition or verification.

This paper is divided in to four main parts the “Introduction” introduces what is the position of biometrics in today’s word and what is the role of iris recognition in the biometrics world. The “Canny Edge” and “Hough Transform” present the detail approach of the proposed system and discuss issues. At last a conclusion done in “Conclusion” which also includes the future considerations for the improvement of the proposed solution.

2. Canny Edge Detector

2.1 Introduction

Canny edge detection algorithm is known as the most popular algorithm for the edge detection. The main aim of the canny edge detection algorithm in this paper is that, the algorithm should not mark the edge which is not there and it should only mark the real edges.

Hence the error rate is decreased. Another aim is to minimise the distance between the edge pixel detected and the original edge. To reduce the noises from the image it is necessary that the algorithm answer to single image.

2.2 Canny in Proposed Work

In the proposed work canny algorithm has been used to find the feature pixels in the given image. MATLAB edge () function has been used. Matlab Edge () Function takes and intensity or a binary image as its input, and returns binary image of the same size as the size of the input image, with 1’s where the function finds edges in input image and 0’s elsewhere.

2.3 Experiments with Different Methods

Canny is known to be most efficient method for edge detection. But Sobel is also one of the known methods. From the comparison of Sobel and Canny, findings are, both are gradient based edge detectors. However, Canny performs an additional processing, with non-maximum supervision which eliminates possible wide ridges that can result from the Sobel. Furthermore, where Sobel does a simple thresholding, Canny combines the thresholding with contour following to reduce the probability of false contours. And also, Canny was derived as a reasonable approximation to an Optimal Edge Detection, while Sobel was empirically derived. The following figure shows the difference between Sobel method and Canny method practically.

The following figure shows the difference between Prewitt method and Canny method practically.

---

*Poorvi Bhatt is currently working as Sr. Software Engineer at Leidos (Legacy Lockheed Martin IS&GS), Atlanta, GA, USA. Ph-No: 4047353416, Email:poorv@gmail.com*
2.4 Canny Method in Detail
Output = EDGE (Input, 'canny') is the general form the canny method. The Edge Function here will take an intensity image or a binary image as an input. The second argument specifies the method to apply for finding edges from the image. Edge function here will return an array with the values 1's and 0's. 1 for the edge pixel and 0 for the non-edge pixels.

Output = EDGE (Input, 'canny', THRESH)

Here the THRESH argument can me specified as vector and scalar. If THRESH is a two-element vector then first element is low threshold and second element is high threshold. And if THRESH is a scalar then that values is used as a high threshold and 0.4*THRESH is used for the low threshold. If THRESH is not specified then EDGE chooses low and high values automatically.

Output = EDGE (Input,'canny',THRESH SIGMA)

Here SIGMA is the standard deviation of the Gaussian filter. The default SIGMA is 1; the size of the filter is chosen automatically, based on SIGMA. [Output, thresh] = EDGE (Input, 'canny', ....) returns the threshold values as a two-element vector.

3. HOUGH TRANSFORM

3.1 Introduction
Hough Transform is a technique to extract shapes from the image. The classical version of the Hough transform identifies straight lines from the images, but the extended version identifies regular or non-regular shapes from the image. “The transform universally used today was invented by Richard Duda and Peter Hart in 1972, who called it a “generalized Hough transform” after the related 1962 patent of Paul Hough.”[5] In Hough Transform, input image is taken as binary image, the image to which edge detection has been applied. Thus the points which need to be transformed are those which likely to lie on an ‘edge’ in the image. The transform itself is rounded into an arbitrary number of bins, each representing an approximate definition of a possible shape for which transformed is performed e.g. Line, Circle etc... Each feature point in the edge detected image is said to vote for a set of bins corresponding to the shapes that consist the feature point.

By simply incrementing the value stored in each bin for every feature lying on that shape, an array is built up which shows which shapes fit most closely to the data in the image. [6] By finding the bins with the highest value, the most likely shapes can be extracted. The simple way of finding these peaks is by applying some form of threshold, but different techniques may yield better results in different circumstances – determining which shapes are found as well as how many. In this paper, Hough Transform for the circles has been used to locate an iris from the eye.

3.2 Hough Transform for Circles
The extended version of Hough Transform can be used for finding circles from the image. To detect the circle from the image radius and center of the circle will be required. Hence r (radius), Xc(x-coordinate of center), Yc(y-coordinate of center).

The equation of the circle is

\[ r^2 = (x - Xc)^2 + (y - Yc)^2 \]

Where r is radius of circle and (Xc,Yc) is center co-ordinates of the circle. Again in xy-space all featured pixels, obtained after applying the edge detection algorithm, will be processed. Each edge pixel, residing on the circle will be represented as a circle in the parameterized space. And in parameter space, if the circles coincide at say (Xci,Yci) point then, that point is center of the circle in the xy-plane.
Here the circle of fixed radius is described. But if the circle is of unknown size then the third parameter will be the radius of the circle. Then the circle will be of arbitrary size \((X_c, Y_c, r)\). And hence the accumulator will cast votes in three-dimensional accumulator.

The Algorithm for the Hough Transform can be given as
1. The Algorithm for the Hough Transform can be given as
2. Find all the feature points in the image (Using any Edge Detector)
3. For each feature point find the accumulator
4. For each possibility in the accumulator that passes through the feature point.
5. Increment that position in the accumulator
6. Find local maxima in the accumulator.

3.3 Issues with the Hough Transform

**Orientation**
To transform straight line, orientation can be one of the parameter in the accumulator array but for to transform circle orientation is unnecessary. If orientation parameter is included in the accumulator array then the accumulator arrays’ dimension will increase and if it is not included then only specific orientation will be included. According to the objective of this paper, the main focus is on the Circle Hough Transform and orientation is unnecessary in circular transform.

**Smoothing the Accumulator**
A second issue is how to detect peaks in the accumulator array. If it is straight line then it is easy to find the pick by simply picking the largest value in the accumulator and map it back. But if it is for other shapes then relative maxima should be found. This can be done by allowing threshold the maxima that is discovered. The votes in the accumulator array are not just for the sharp line or other shapes that pass through the feature point but also for those that pass close by. This casting or smoothing for nearby possibilities can be done directly during the voting phase of the transform or afterward through a post-processing blurring of the accumulator.

**Grey-level Voting**
The discussion so far has been of binary feature points casting binary votes. A more effective way of doing this is to use the full range of grey-level feature detection and cast votes proportional to the strength of the feature.

3.4 Design and Implementation of the System
The proposed system has used MATLAB for Design and Implementation.

**Implementation of Hough Transform for circles**
Hough Transform for circles works same as the Hough Transform for straight lines. But for lines the parametric space (Hough Space) is 2-Dimensional, whilst for circles the parametric space (Hough Space) is 3-Dimensional. The dimensions of the 3-D parametric space will be radius of the circle, x-coordinate of the center circle and y-coordinate of the center of the circle.

**Algorithm Explanation:**
- Read image and perform image processing necessary.
- Apply canny edge detector.
- Initialize 3-D accumulator array.
- Find radius and Increment 3-D accumulator array.
- Finding Peaks.

This steps is perform in following steps:
Finding peak for center of the circle.
Finding peaks from array of radii.

- As the proposed system is to find iris from the image, the circles looked for are concentric circles. Hence if the center of the circles are found then from the accumulator array radii of that particular center co-ordinates are searched and stored.
- From the stored radii, peaks are found using hill-climbing technique and from the peaks stored smallest peak is considered as a pupil's radius.
- Finding RGB values at different Radii
- Comparing average RGB values for authorization.

Implementation in the detail can be explained as follows.

Step 1. Read Image
In order to process pixel values of the image IMREAD function of MATLAB has been applied. IMREAD function will return an array of image data. If image is gray-scale then array returned will be two-dimensional and if image is color image then array returned will be three-dimensional. Image data here return will be based on the type of the image. In the proposed system image type is taken as JPEG.

Step 2. Perform Canny Edge Detection
In order to perform canny edge detection, image needs to be converted in gray-scale image from RGB image. MATLAB function rgb2gray has applied to perform this task. Rgb2gray function will return a grayscale intensity image. An EDGE function with canny as an argument, of MATLAB has been used to perform canny edge detection on the gray scale intensity image returned by the RGB2GRAY function. EDGE function will return image data with 1's and 0's values. The image data which contains value one is an edge pixel and an image data which contains value zero is a non-edge pixel.

Step 3. Initializing an accumulator array.
As here parametric space (Hough Space) will be 3-Dimensional, accumulator array is also 3-Dimensional array. The 3-Dimensions of the accumulator array can be described as x-coordinate of the center, y-coordinate of the center and radius of the circle. Accumulator array can be created by the spacing entered by the user. The size of the accumulator array will be (MaxRows/Spacing, MaxCols/Spacing, MaxRadius/Spacing). An accumulator of the specified size is initialized and all the values of the array are set to zero.

For each image pixel, it is checked that whether it is an edge pixel or not. If it is an edge pixel then for each (x,y) in accumulator array radius is calculated and accumulator array (xc,yc,r) is incremented.

Step 5. Finding peaks in the accumulator array.
In order to find appropriate circle it is necessary to find peaks in the accumulator array. As in the proposed system, circles needs to be find from the image of the iris it is obvious that all the circles will be concentric. Hence first center of the circle is found and then on that (Xc,Yc) point of the accumulator array radii are searched and stored.

In order to find out center of the circle, accumulator array is sorted and highest peak is found from the sorted array. And in order to find appropriate radii of the circles, for every radius in the accumulator array the values at Xc,Yc Co-ordinate are stored in one dimensional array called r. And then performing hill climbing technique peaks from the one-dimensional array 'r' are found and saved in array called Peaks. The values in 'r' are then matched with values in Peaks and index for the matched value is multiplied by Spacing and will be saved in array called Radii. The array Radii contains all the expected radii of the circles from the image. The smallest value in array Radii will represent radius of the inner circle and largest value in array Radii will represent radius of the outer circle.

Step 6. Finding RGB values at different Radii.
After finding the radius of the inner circle, RGB values at different radii are found and stored. MATLAB's IMPIXEL function has been used to find RGB values at different Radii. IMPIXEL will return Red, Green and Blue value at a given pixel.

Step 7. Comparing Average RGB Values.
After finding RGB values at different positions average of that values are found and on that average values the RGB cube is viewed. This is performed by MATLAB's function VIEW. VIEW set an angle of the view from where an observer sees the 3Dplot. Hence if the average RGB value have more value of red and less value of green and more value of blue then generated RGB Cube can be viewed from the angle where red and blue are emphasized and green is less valuable. As the program runs through it uses MATLAB’s imshow () function to display the image data in the image form. Imshow () function will take image data as an argument and returns an image which is generated from that image data.

Experimental Results & Outputs

Image Selection
In general the issues with the iris recognition system is to capture a high quality image of the iris while remaining noninvasive to the human operator. The following should be point of concern: Desirable to acquire images of the iris with sufficient resolution and sharpness to support recognition. It is important to have good contrast in the interior iris pattern without resorting to a level of illumination that annoys the operator. Noises in the acquired images should be eliminated as much as possible. To initially check the program, JPEG image of circle is created by using the MS-Paint Program, and used as an input. An image of two circle is created using MS-Paint, saved as JPEG and used as a second input. An image of two circle one is outer and one is inner, created using MS-Paint, saved as JPEG and used as a third input. An image of triangle is created using MS-Paint, saved as JPEG and used as a fourth input. Several images of eye is taken from the Google, which is available free. My eye is taken as input. Other people’s eye is taken as an input.

Results
As the program runs through it produces eye image at different stage and it also produces so other details.
Images Produced:
- Canny Edge Image.
- Real Image with circles identified.
- Image of RGB Cube of Average RGB value found.

Information Produced:
- Display image name.
- Canny threshold applied.
- Spacing applied.
- Centre of the circles.
- Radius of inner circle.
- Radius of outer circle.
- RGB value of the pixel at North of the circle with the radius applied.
- RGB value of the pixel at South of the circle with the radius applied.
- RGB value of the pixel at East of the circle with the radius applied.
- RGB value of the pixel at West of the circle with the radius applied.
- Average RGB value of these 4 Points.

### TABLE 1: RESULT DATA

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Name</td>
<td>Canny Edge</td>
</tr>
<tr>
<td>Canny Threshold</td>
<td>Spacing</td>
</tr>
<tr>
<td>Eye7.jpg</td>
<td>0.1</td>
</tr>
<tr>
<td>Eye7-rot.jpg</td>
<td>0.2</td>
</tr>
<tr>
<td>Eye9.jpg</td>
<td>0.5</td>
</tr>
<tr>
<td>Eye2.jpg</td>
<td>0.3</td>
</tr>
<tr>
<td>Eye3.jpg</td>
<td>0.5</td>
</tr>
<tr>
<td>Eye7.jpg</td>
<td>01</td>
</tr>
<tr>
<td>Eye7-rot.jpg</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Findings of the program:
The program is tested under different situations and different image conditions. The results of all the experiments can be concluded as follows.

**Result of canny edge detector matters.**
If canny cannot find appropriate edge pixel from the image then the Hough space may not contain right information to produce an image. Output also depends upon the threshold specified for canny. If threshold is near 0 image will contain more edge pixels and if threshold is near 1 then image will contain less edge pixels.

**Centre of the circles.**
This program is running on the assumption that all the circles that are detected in edge image will have only one center. Hence all the circle should be concentric.

**Spacing.**
Spacing performs crucial role in every results obtained by the program. Spacing defines the hough space. The accumulator array is quantized in order to run the program faster. Higher the value of Spacing, lower the size of an accumulator array and lower the accuracy. In this program the spacing has gone up to 2 points hence it is showing much more accurate results. But pacing 2 occupies more memory in processing.

**CONCLUSION AND FUTURE WORK**
The proposed system have some of the critics need to be discuss.
4.1 Blurred Position of the Center
The problem arises if the center position found as a result of Hough Transform is blurred. This may cause the detection of multiple circles around the center area with slightly different center where only one circles needs to be detected. This problem can be addressed by clustering the binary form Hough Space and detecting the cluster centres. Some techniques that might solve this problem are:
- LBG algorithm
- Neuronal Gas Approaches
- RBF (Radial Based Functions) area filling
- Bayes’ Classification

The result would be a smaller number of circle centres. This speeds up the detection algorithms, give more precise results.

4.2 Limited Accuracy
Another problem is with the limited accuracy of the used digital grids. They are limited in resolution and accuracy of a specific grid value. Both are non-continuous. For example: in case much more than 255 circles are situated within an image the grid value size of the Hough Space had to be increased from 8bit to may be 16 bit or more. Overall this report discusses how Hough Transform can be implemented for Iris Recognition on eye images with one center by finding and comparing RGB values of 3D-array with help of Canny Edge Detector. It also explains its limitations and how future work can overcome those limitations.

END SECTIONS

5.1 Appendices
Web Appendices available with this paper at the Biometrics website on IJSTR Online Library.

5.2 Acknowledgement
I would like to thank Mr. Geoff Dowling (University of Westminster, Harrow, UK) for supervising my project. His help enabled me to transform his suggestions into implementations, and his criticisms into complete project. Also without continuous support from family this could never have been completed.

5.3 Reference


