Extraction of Circle from Digital Images

Nitasha
M.Tech. Student
UCoE, Punjabi University,
Patiala, India

Reecha Sharma
Assistant Professor
UCoE, Punjabi University,
Patiala, India

ABSTRACT

As we know, we wouldn't be able to drive our cars. Because wheels are circles, so are steering wheels. 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 circle using two algorithms. First one is Canny Edge Detection Algorithm with the help of this algorithm we find out edge detected digital image. Second is Freeman Chain Code Algorithm for contour tracing or find out the arcs which are part of circle or not. All this is done in MATLAB R2010a.

Keywords

Circle Detection, Canny Edge Detection Algorithm, Freeman Chain Code, Edge Detection.

1. INTRODUCTION

Digital image processing is the upcoming field. Digital image processing is roughly divided into three categories. Image Enhancement, Image compression, and Image Segmentation. Image segmentation subdivides an image into its constituents regions and objects. Segmentation algorithms are based on one of the basic properties of intensity values:

1. Discontinuity
2. Similarity

In the first category, the approach is to partition an image based on abrupt changes in intensity, such as edges [2]. The principle approach in the second category is based on partitioning an image into regions that are similar according to a set of predefined criteria. Extracting the edges of objects and identifying basic shapes in an image is of importance in industrial applications of image processing. Edge detection is very useful in a number of contexts. Edge characterizes object boundaries and are, therefore, useful for segmentation, registration and identification of objects in scenes. The output of edge detection should be an edge image, in which the value of each pixel reflects how strong the corresponding pixel in the original image meets the requirements of being an edge pixel. Edge detection refers to the process of identifying and locating sharp discontinuities in an image. The discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene.

Discontinuities can be classified as:

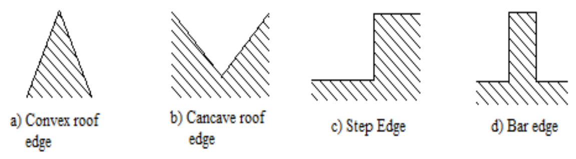


Figure 1 Classification of Discontinuities

1. **Gradient Discontinuity-** where the gradient of the pixel value changes across a line.

- Roof edges
- Ramp edges
- Convex edges
- Concave edges

Ramp edges have the same signs in the gradient components on either side of the discontinuity, while roof edge has opposite signs in the gradient components.

2. Jump or Step Discontinuity- where pixel values themselves change suddenly across some line. The step edge defines a perfect transition from one segment to another. In case of step edge, image intensity abruptly changes form one value to one side of the discontinuity to a different value on the opposite side as shown in figure 1c.

3. Bar Discontinuity- where pixel values rapidly increase then decrease again (or vice versa) across some line as shown in figure 1d.

2. EDGE DETECTION TECHNIQUES

There are various edge detection techniques that are shown below:

- Simplest Operator
- Sobel Operator
- Robert's Cross Operator
- Prewiit's Operator
- Canny Edge Detection Algorithm

In this paper, we explain Canny Edge Detection Algorithm to obtained edge detected digital image. Block diagram of Canny Edge Detected Algorithm is shown in figure 2. It is a multi step process to detect a wide range of edge in digital images [3].

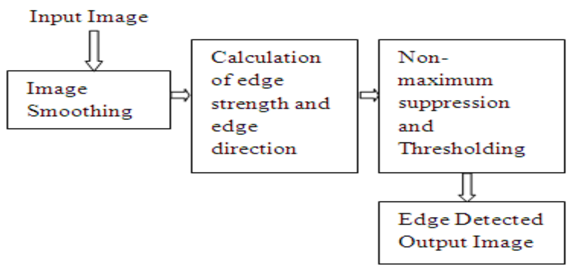

Input Image
Image Smoothing
Calculation of edge strength and edge direction
Non-maximum suppression and Thresholding
Edge Detected Output Image

Figure 2 Block diagram of Canny Edge Detection Algorithm

The edges in an image are expected to be the boundaries of objects that tend to produce sudden changes in the image intensity [3].

3. EXTRACTION OF CIRCLE FROM DIGITAL IMAGES

For a digital image, four 3x3 window operators are used to get two pixel directions, Horizontal (A_x) and Vertical (A_y), for each pixel, taking into account the neighbors of the pixel.

The kernels used to find Horizontal gradient orientation are:

A_{x1} =

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

A_{x2} =

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

Similarly, Kernels are used to find the vertical gradient orientations are:

A_{y1} =

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

A_{y2} =

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

Where X is the pixel under consideration. A_{x1}, A_{x2}, A_{y1}, and A_{y2} kernals are used to get horizontal and vertical pixels directions and pixels edge strength [6].

Calculation of Pixel direction and Edge strength

To calculate pixel direction, we have to find out horizontal and vertical orientation, and to find out edge strength, we have to find out horizontal and vertical gradient magnitude. Horizontal orientation and Horizontal gradient magnitude are calculated as follow:

Horizontal Kernals and Image is convolved to get a set of values B_{x1} and B_{x2}. These two sets of values are used to determine the horizontal orientation of the pixel point (x, y) of the image I.

$$B_{x1} = I * A_{x1}$$

$$B_{x2} = I * A_{x2}$$

If B_{x1} (x, y) □ B_{x2}(x, y) then the orientation A_h_ori is $\longrightarrow$

If B_{x1}(x, y) □ B_{x2}(x, y) then the orientation A_h_ori is $\longleftarrow$

If B_{x1} (x, y) = B_{x2} (x, y) then the orientation is zero

The horizontal magnitude is obtaind by:

$$B\_h\_mag = |B_{x1}(x, y) - B_{x2}(x, y)|$$

Similarly we can determine vertical orientation of the pixel point (x, y) of the image I.

$$B_{y1} = I * A_{y1}$$

$$B_{y2} = I * A_{y2}$$

If B_{y1} (x, y) □ B_{y2} (x, y) then the orientation A_v_ori is $\downarrow$

If B_{y1} (x, y) □ B_{y2} (x, y) then the orientation A_v_ori is $\uparrow$

If B_{y1}(x, y) = B_{y2} (x, y) then the orientation is zero

The vertical magnitude is obtained by:

$$B\_v\_mag = |B_{y1}(x, y) - B_{y2}(x, y)|$$

Now the Edge Strength is calculated as:

$$B\_mag = |B\_h\_mag + B\_v\_mag|$$

The pre-defined conditions to get the pixel direction are shown in table 1 [6].

Table 1 Conditions to determine the pixel directions

No	Horizontal orientation	Vertical orientation	Conditions	Range	Result
1	→	↑ Or ↓	**B_h_mag** □ **B_v_mag**	14^0-344^0	→ 1
2	→	↓	**B_h_mag** □ **B_v_mag** **B_v_mag** □ **B_h_mah**	284^0 – 346^0	2 ↘
3	→ Or ←	↓	**B_v_mag** □ **B_h_mag**	256^0 – 284^0	3 ↓
4	←	↓	**B_h_mag** □ **B_v_mag** **B_v_mag** □ **B_h_mag**	194^0-256^0	4 ↙
5	←	↓ or ↑	**B_h_mag** □ **B_v_mag**	164^0-194^0	5 ←
6	←	↑	**B_h_mag** □ **B_v_mag** **B_v_mag** □ **B_h_mag**	76^0 - 164^0	6 ↖
7	→ or ←	↑	**B_v_mag** □ **B_h_mag**	76^0-104^0	7 ↑
8	→	↑	**B_v_mag** □ **B_h_mag** **B_h_mag** □ **B_v_mag**	14^0-104^0	8 ↗

Non-maximum suppression

Non-maximum suppression is a process for marking all pixels whose intensity is not maximal as zero within a certain local neighborhood. This local neighborhood can be a linear window at different directions [4].

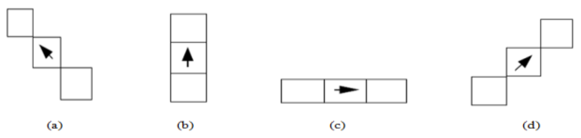


Figure 3 shows four examples of linear windows at angles of 0°, 45°, 90°, and 135°.

Hysteresis thresholding requires two thresholds, an upper and a lower threshold. The process starts when an edge point from non-maximum suppression is found to exceed the upper threshold. The neighbors of the point are then searched to determine whether or not they exceed the lower threshold. Any neighbor that exceeds the lower threshold is labeled as an edge point and its neighbors are then searched to determine whether or not they exceed the lower threshold. In this manner, the first edge point found (the one that exceeded the upper threshold) becomes a seed point for a search. Its neighbors, in turn, become seed points if they exceed the lower threshold, and so the search extends, along branches arising from neighbors that exceeded the lower threshold. For each branch, the search terminates at points that have no neighbors above the lower threshold.

Freeman's Chain Code for Contour Tracing

Chain code is a list of codes ranging from 0 to 7 in clockwise direction [5]. These codes represent the direction of the next pixel connected in 3x3 windows, as shown in the Table 2. For example, if a current pixel in an image is located at coordinate (x,y), the coordinate of the next pixel based on the chain code is given by Table 2. The coordinates of the next pixel are calculated based on addition and subtraction of column and row by 1, depending on the value of the chain code, as shown in Table 3.

Table 2 Relation of Pixel and Chain Code with Current Pixel

	Column-1	column	Column+1
Row-1	**6**	**7**	**8**
Row	**5**	**Current pixel**	**1**
Row+1	**4**	**3**	**2**

Table 3 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**

How Arcs are find out by contour tracing:

After the edge detection, linking algorithms are designed to assemble edge pixels into meaningful edges and/or region boundaries [3]. 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***

$$|\textbf{angle}(x,y) - \textbf{angle}(x_0 - y_0)| \; □ \; \textbf{A}$$

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***

$|\textbf{magnitude}(x,y) - \textbf{magnitude}(x_0, y_0)| \; □ \; \textbf{E}$ Where E is the positive threshold and A is positive angle threshold. Pixels are linked with each other if both magnitude and direction criteria are satisfied [7]. By doing this arcs are find out. Next step is to determine these arcs are to be part of the circle or not. After all these step circle looks like this as shown in figure no 4.

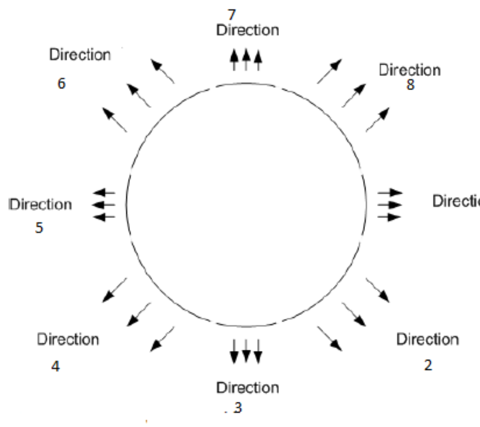


Figure 4 Arcs of the circle with edge directions

We observe from the figure 4 that, major part of the circle formed by 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.

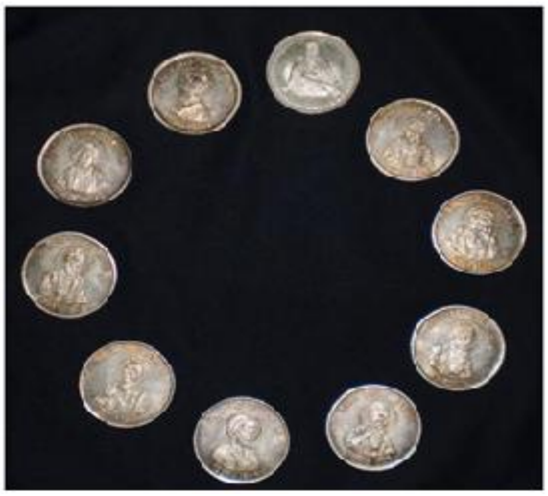
"The condition for considering the arc as part of a potential circle is that there should be at least 3 arcs present with pixel directions 2, 4, 6 or 8 which are connected to each other by arcs with pixel directions 3, 5, 7 or 1".

So to find the potential circle we start scanning the image to find the arc with pixel direction 2 or 8. Once the arc with pixel direction 8 is found, we scan at the end point of the arc using the block of size 3x3, for a pixel part of an arc with pixel direction 1, as shown in Figure 6.6. Once the arc with pixel direction 1 is found, we scan the other end of the arc, using the block of size 3x3, for a pixel part of an arc with gradient direction 2. Once the arc with pixel direction 2 is found, we scan other end of the arc, using the block of size 3x3, for a pixel part of an arc with pixel direction 3. Once the arc with pixel direction 3 is found, we scan other end of the arc, using the block of size 3x3, for a pixel part of an arc with pixel direction 4. Once the arc with pixel direction 4 is found, three major arcs are found, which satisfies the condition of being part of the circle.

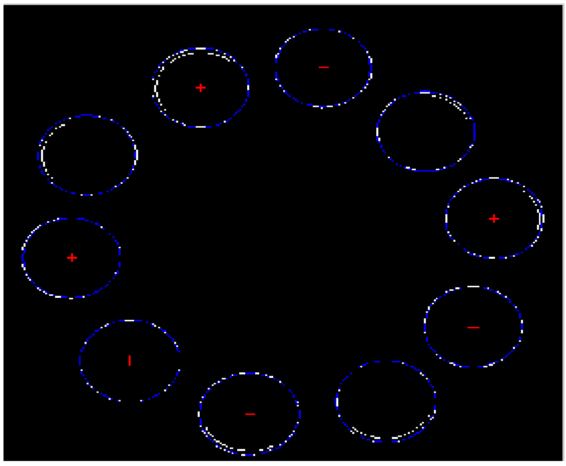
Once accurate arcs are find out they are draw on the circle with blue color. By doing this, circle is detected from the images.

4. RESULT AND FUTURE SCOPE

Original Image



After Circle Detection



As in this research we have done our work on isolated circular object, in further studies we can focus on different circular object problems like--- if objects are overlapping, there are co-centric circles or due to image distortion the circular object becomes elliptical.

5. REFERENCES

[1] J.Canny, “A computational approach to edge detection,” IEEE Trans. Pattern Anal. and Mach. Intell., vol. 8, pp. 679-714, Nov. 1986.

[2] B.Chanda and D.Dutta Majumder, “Digital Image Processing and Analysis”. Prentice Hall 2003.

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

[4] C. Sun and P. Vallotton, “Fast linear feature detection using multiple directional nonmaximum suppression,” in Proc. Int. Conf. Pattern Recognition, Hong Kong, China, 2006, pp. 288-291.

[5] R S Vaddi[1], L N P Boggavarapu[1], H D Vankayalapati[2], K. R. Anne[1], “Contour Detection Using Freeman Chain Code And Approximation Method For The Real Time Object Detection”, [1]Department of Information Technology, V R Siddhartha Engineering College, Kanuru, Vijayawada, India.[2]Department of Computer Science & Engineering, V R Siddhartha Engineering College, Vijayawada, India.

[6] D. Forsyth and J. Ponce, Computer Vision: A Modern Approach. Upper Saddle River, NJ: Prentice Hall, 2003.

[7] M.Nixon and A.S.Aguado, Feature Extraction & Image Processing. Burlington, MA: Academic Press, 2008.

[8] Sabina Priyadarshini and Gadadhar Sahoo , “ A New Edge Detection Method based on Additions and Divisions”, Department of Information Technology Birla Institute of Technology Mesra, Ranchi, International Journal of Computer Applications (0975 – 8887) Volume 9– No.10, November 2010.

[9] G.M.Schuster and A.K.Katsaggelos, “Robust circle detection using a weighted MSE estimator,” in Proc. Inter. Conf. Image Processing, Singapore, 2004, pp. 2111-2114.