Circular Hough Transform

Simon Just Kjeldgaard Pedersen

Aalborg University, Vision, Graphics, and Interactive Systems,
November 2007

1 Introduction

A commonly faced problem in computer vision is to determine the location, number or orientation of a particular object in an image. One problem could for instance be to determine the straight roads on an aerial photo, this problem can be solved using Hough transform for lines. Often the objects of interest have other shapes than lines, it could be parables, circles or ellipses or any other arbitrary shape. The general Hough transform can be used on any kind of shape, although the complexity of the transformation increase with the number of parameters needed to describing the shape. In the following we will look at the Circular Hough Transform (CHT).

2 Parametric Representation

The Hough transform can be described as a transformation of a point in the x,y-plane to the parameter space. The parameter space is defined according to the shape of the object of interest.

A straight line passing through the points (x_1, y_1) and (x_2, y_2) can in the x,y-plan be described by:

$$y = ax + b \tag{1}$$

This is the equation for a straight line in the cartesian coordinate system, where a and b represent the parameters of the line. The Hough transform for lines does not using this representation of lines, since lines perpendicular to the x-axis will have an a-value of infinity. This will force the parameter space a, b to have infinite size. Instead a line is represented by its normal which can be represented by an angel θ and a length ρ.

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{2}$$

The parameter space can now spanned by θ and ρ, where θ will have a finite size, depending on the

resolution used for θ. The distance to the line ρ will have a maximum size of two times the diagonal length of the image. [3]

The circle is actually simpler to represent in parameter space, compared to the line, since the parameters of the circle can be directly transfer to the parameter space. The equation of a circle is

$$r^2 = (x-a)^2 + (y-b)^2 \tag{3}$$

As it can be seen the circle got three parameters, r, a and b. Where a and b are the center of the circle in the x and y direction respectively and where r is the radius. The parametric representation of the circle is

$$\begin{aligned} x &= a + r\cos(\theta) \\ y &= b + r\sin(\theta) \end{aligned} \tag{4}$$

Thus the parameter space for a circle will belong to R^3 whereas the line only belonged to R^2. As the number of parameters needed to describe the shape increases as well as the dimension of the parameter spaceR increases so do the complexity of the Hough transform. Therefor is the Hough transform in general only considered for simple shapes with parameters belonging to R^2 or at most R^3. In order to

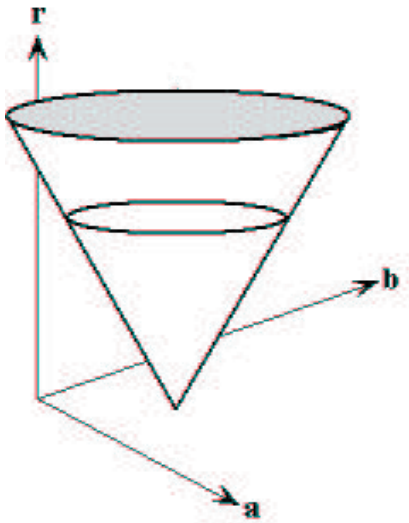


Figure 1: The parameter space used for CHT [2].

simplify the parametric representation of the circle, the radius can be held as a constant or limited to number of known radii.

3 Accumulator

The process of finding circles in an image using CHT is:
First we find all edges in the image. This step has nothing to do with Hough Transform and any edge detection technique of your desire can be used. It could be Canny, Sobel or Morphological operations. [3][2]

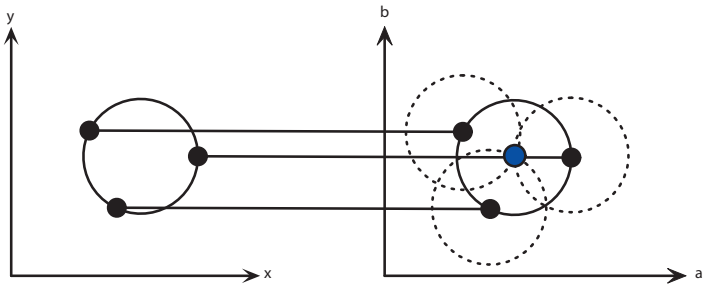
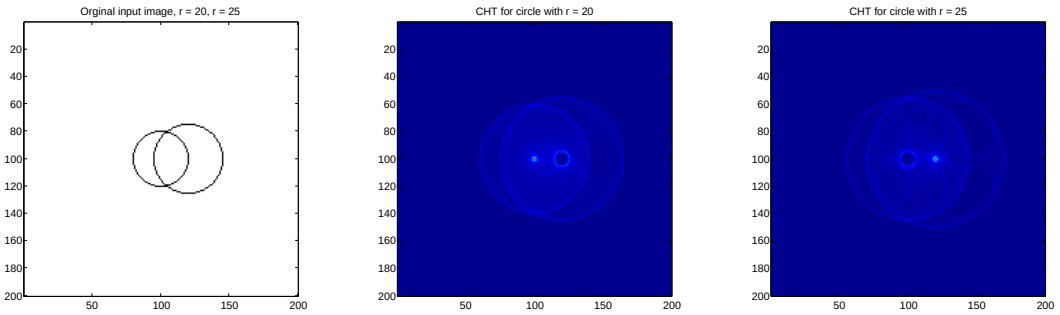


Figure 2: A Circular Hough transform from the x,y-space (left) to the parameter space (right), this example is for a constant radius [2].

At each edge point we draw a circle with center in the point with the desired radius. This circle is drawn in the parameter space, such that our x axis is the a - value and the y axis is the b value while the z axis is the radii. At the coordinates which belong to the perimeter of the drawn circle we increment the value in our accumulator matrix which essentially has the same size as the parameter space. In this way we sweep over every edge point in the input image drawing circles with the desired radii and incrementing the values in our accumulator. When every edge point and every desired radius is used, we can turn our attention to the accumulator. The accumulator will now contain numbers corresponding to the number of circles passing through the individual coordinates. Thus the highest numbers (selected in an intelligent way, in relation to the radius) correspond to the center of the circles in the image. [4][6]



(a) The input image. (b) Hough transform where r = 20. (c) Hough transform where r = 25.

Figure 3: An example of how the accumulator of the CHT looks like with real world data. This example shows two CHTs for different radii.

4 Algorithm

The algorithm for Circular Hough Transformation can be summarized to [5]:

1. Find edges
2. //HOUGH BEGIN
3. For each edge point

   Draw a circle with center in the edge point with radius r and increment all coordinates that the perimeter of the circle passes through in the accumulator.
4. Find one or several maxima in the accumulator
5. //HOUGH END
6. Map the found parameters (r,a,b) corresponding to the maxima back to the original image

5 Implementation

With the algorithm outlined in Section 4 is it pretty straight forward to implement a working version of CHT. But several aspect should be considered.

5.1 How to Store Data

The accumulator array which is three dimensional, if the radius is not held constant, can quite fast grow large. Its size is depended on the number of different radii and especially the image size. The computational cost of calculating all circles for each edge points increases with the number of edge points which is usually a function of image size. The overall computation time of CHT can therefore quickly reach an infeasible amount of time with large images with many edge points.

5.2 How to Draw Circle in Discrete Space

A circle can be drawn in discrete space directly from the equations in (4), but one problem arises. How is the discrete values or the resolution of θ selected. One solution is to use a high resolution of θ, and then round the values off, but this is likely to result in massive overdraw (an edge pixel being drawn more than once) or lack of pixels if the radius is large. The rounding of $\sin, \cos$ should be carried out after the values have been multiplied with the radius.

To ensure all edge pixels are drawn the resolution of θ can be selected very high, but this will also increase the computational cost. One trick lower the computational cost is to precalculated the values for $\sin, \cos$ and use a lookup table.

Although the above solution works another and better approach exist. Instead of using the equations in (4) Bresenham's Algorithm can be used for drawing raster circles [1]. Bresenham's algorithm was designed for drawing lines/circles for digital monitors without any overdraw and is thus ideal in this context as well. One nice property is that, it is possible to know the exact number of pixels used to draw the perimeter, and this information can be useful when the center of the circles are to be found from the accumulator data.

An implementation of Bresenham Algorithm can be found here: `http://en.wikipedia.org/wiki/Bresenham's_line_algorithm`

5.3 How to Find Circle From the Hough Transform Data

Although this is not a part of the Hough Transform it desirable to be able to find circles from the accumulator data. If no a priori knowledge is known about the number of circle and their radii then this process can be quite challenging. One approach is to find the highest peaks for each a,b plane corresponding to a particular radius, in the accumulator data, see Figure 4. If the height of the peak(s) is equal compared to the number of edge pixels for a circle with the particular radius, the coordinates of the peak(s) does probably correspond to the center of such a circle. But the center of a circle can also be represented by a peak with a height less than the number of edge pixels, if for instance the circle is not complete or is ellipse shaped.

If it is difficult to locate exact peeks, the accumulator data can be smoothed [5].

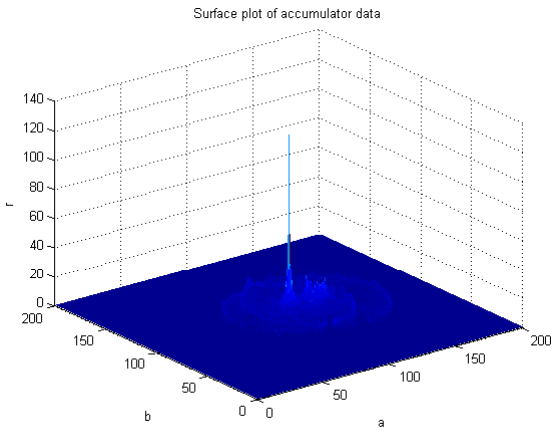


Figure 4: Surface plot of the a,b-plane with radius = 20 for the example circles shown in Figure 3

References

[1] Jack Bresenham. A linear algorithm for incremental digital display of circular arcs. *Communications of the ACM February, Volume 20,Number 2*, February 1977.

[2] Chester F. Carlson. Lecture 10: Hough circle transform. Rochester Institute of Technology: Lecture Notes, October 11, 2005.

[3] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2007. ISBN 0-13-168728-x.

[4] Carolyn Kimme, Dana Ballard, and Jack Sklansky. Finding circles by an array of accumulators. *Communication of the ACM, Volume 18, Number 2*, February 1975.

[5] Bryan S. Morse. Lecture 15: Segmentation (edge based, hough transform). Brigham Young University: Lecture Notes, 2000.

[6] Mohamed Rizon, Haniza Yazid, Puteh Saad, Ali Yeon Md Shakaff, Abdul Rahman Saad, Masanori Sugisaka, Sazali Yaacob, M.Rozailan Mamat, and 1M.Karthigayan. Object detection using circular hough transform. *American Journal of Applied Sciences 2 (12)*, 2005.