CIRCLE DETECTION USING VARIOUS HOUGH TRANSFORM ALGORITHMS AND THEIR COMPARISONS

Abhishek Kimmoji, 50246582 Kshitij Goel, 50246430 Roshni Murali, 50247191

Introduction

In the modern world, humans highly depend on the computing devices to perform analysis on large amounts of data, advanced body imaging, aesthetics and automation. For many of these practices, we use computer vision for detection of different features in images for different kinds of applications. We detect edges for aesthetic stitching, corners for motion detection and 3D modelling, blobs for object tracking and stereo stitching, ridges for detecting roads in aerial images and blood vessels in retinal images. Multiple of these detection types use hough transform for their basic segregation of data, thus making it a necessary tool for an analyst's toolbox. When using the hough transform, we come across multiple types of hough transform which can make it confusing for the user to choose. Thus, a comparison of these applications is a necessity as it makes an index for the user to make appropriate use when in need.

Finding Circles

The general equation of a circle is:

$$(x-a)^2 - (y-b)^2 = r^2$$

where,

a,b = coordinates of circle center,

r = radius of circle.

Any point on the circumference of the circle will be transformed into a cone of parameter space **(a,b,r)**. If the image intersects the circle at all points of the circle's circumference, the cones will intersect at a single point. Early uses of the circle detection made use of the gradient direction at each point of the edge. This has a pre-condition that the center of the circle should be at a normal to the point on the edge. This resulted in a faster way as only the segments of the cone had to be incremented and the accuracy just depended on the accuracy at which the edge detection was performed. A very important part of hough transform is the detection of peak which can be solved in a very interesting manner by considering a second daa pass which finds the maximum value in accumulator array of the edge points as compared to other parameter values of the point. The edge point a this location is then labelled thus making this method useful in detecting final peaks as well.

Standard Hough Transform

In the standard application of hough transform, extensive use of the 3D accumulator array is made for voting limit for a cone section. Generally, the circle center lies on the normal of the edge direction thus making the process of finding possible center locations an easier task. A real world usage of this process, although correct, does not provide with accurate results which makes local maxima detection harder to find.

Gerig And Klein Hough Transform

SHT suffers from storage problem, it happens when there's a big range for radii of the circles. SHT's uses a 3d Accumulator array while GKHT on the other hand works on 3 different 2d accumulator arrays for hough space, this method saves space. 2 of them is used to store the centers and the other is used to store the radius of the circles. BKHT doesn't use edge detection, this saves a lot of storage space as well but because it can only save one radius for each center, it won't be able to detect all the concentric circles.

GHKT with Edge Detection

Since GKHT doesn't use edge detection, it has to increment at every value of radius of the circle for each edge point. as this is an computationally expensive operation GKHT was modified to include Edge detection in the algorithm, the modified hough transform is named Gerig Hough Transform with Gradient. It is efficient because it only works on a sector of a circle dependent on the edge detection, to do this efficiently it uses approximation of the sector by its square. This algorithm is quite similar to GKHT, only difference is that it uses a smaller region of the circle for post processing and accumulation matrix.

Fast Hough Transform

FHT uses a multi dimensional quadtree structure for storing the hough space, it requires less space and is computationally efficient compared to SHT. FHT uses Hyperplane formulation, since Hyperplane formulation on its own generates non-linear parameters for finding circles. a modified version was introduced which incorporates edge detection into it. this improves the efficiency of the algorithm, this new method is called Modified Fast Hough Transform (MFHT). it tries to determine if the 2 orthogonal straight lines formed using the locus of the parameters (a,b,r) at an edge point with vector angle of phi from edge point (x,y) on a 3d hough space have been intersected by the Hypercube, perpendicular distance is calculated between the center of the of hypercube using the diagonal and the the straight lines. If the lines are shorter, then the hypercube gets a vote. MFHT does not involve incrementing value for checking if the hypercube and lines are intersecting, instead it chooses an threshold value adaptively based on the range of radius and also it reduces this range. all these modifications results in an improved and more efficient Hough transform algorithm

2-1 Hough Transform

Another way to reduce the storage requirement when edge direction is available is to decompose the circle finding problem into two stages. Since the center of a circle must lie on the normal of each point on the circle, the common intersection point of these normals is actually the center of the circle. A two dimensional array is required to accumulate votes along the normal of each edge point. The detection of false peaks in the center finding stage can lead to significant computational cost for the second stage, especially if a low threshold is used to detect small circles. The storage space required for the method is quite small, since only a single 2-D accumulator and a 1-D histogram are necessary.

Algorithm and Experimental Analysis

Trying to segregate the hough transform approaches distinctively is a very hard task and there exists no clear winner to any

segment but still there exists a need to coarsely define the approaches as better or worse. We, thus try to compare the algorithms on many different factors to lay out a scale based on the points of accuracy, robustness and computational complexity while still keeping in mind the storage space needed by each algorithm. This helps in the analyses of the algorithms by comparing absolute error of the absolute calculated radii and and the center coordinates in custom synthetic images. A test image of 10 randomly generated circles was used as input to all the algorithm and the speed of calculation and number of false and undetected circles were counted.

Mojarly, all algorithms take considerable amount of time while performing the transform accumulation and the Gerig-Klein post processing. The storage space used by these algorithms is dominated by the space used by individual accumulator arrays. The edge direction of the input image is noised by a filter with range from -5 to 5. This is to help the analyze the performance. A surprising result to this approach is that the Gerig and Klien method does not have an effect on the accuracy of detection. Considering other methods, a gradient of error detection was introduced which classified the methods as to number of false detections. The first bound was set at 0, the second at 5 and the third at 10. While GKHT achieves zero errors with the radius parameters without using gradient, 21HT due to its high sensitivity to the value of error bound produces very small errors for an error bound of 5. MFHT produces great results at bound 5 but at bound 10 uses way more storage. GHTG while producing very accurate results also stays stable at other error bounds. based on these comparisons 21HT and GHTG produces more accurate results. While testing for their ability to identify all the circles, we found that GHTG performs the best with lowest missing circles and false positives for circles.

21HT has the problem of error propagation where if errors occur during the first stage then they are amplified during the second stage histogram. if the center of the circle has an error greater than the histogram cell width then that circle will produce 2 radii for the same circle in the histogram. there radii peaks in the histogram will be located symmetrically located around the real radius and the distance between will be equal to the center error. this makes it difficult to identify concentric circles since they'll result in the same 2 peaks as the error propagation problem.

MFHT while being very accurate and efficient at lower bounds fails to perform at higher bounds to its high storage requirements. this problem arises due to it needing to scan through a large number of fake peaks through the histogram before identifying the legit ones.

In terms of computational speed, SHT, 21HT and GHTG are very close to each other. And as expected GKHT performs considerably poorly due it not using gradient. But in terms of identifying all the circles, GHTG outperforms them all by missing the least number of circles. When it comes to storage efficiency 21HT is the best due to it using lower dimensional arrays and 2 stage algorithm. GKHT and GHTG would be second best with them using lower dimensional arrays for accumulator and histogram with other efficient post processing techniques. SHT would be the worst as it uses 3d accumulator arrays. MFHT is rather unpredictable as it depends on the parameters such as the error bound,

complexity of the image and threshold values.

Conclusion

GKHT and MFHT have serious difficulties when dealing with large complex images. The main reason for that in GKHT is it not incorporating edge detection in its algorithm. MFHT suffers from storage problem and all in all its an very complex algorithm to implement in program. But they are very accurate when working on simple images with low noise.

As said earlier SHT, 21HT and GHTG are all very similar in terms of performance. SHT suffers from huge storage requirement. GHTG fails at identifying concentric circles. But is still comparatively better than 21HT in terms of robustness, this is due to 21HT using a 2 stage method where when error propagation occurs its accuracy gets way worse.

References

1.http://www.bmva.org/bmvc/1989/avc-89-029.pdf

2.Li H., Lavin M.A. and LeMaster R.J. "Fast Hough Transform: A hierarchical approach," CVGIP, 36, pp 139-161, 1986

3.Canny J. "A computational approach to edge detection," IEEE Trans. Pattern Analysis & Machine Intelligence, vol 8, no. 6, pp 679-698, 1986.

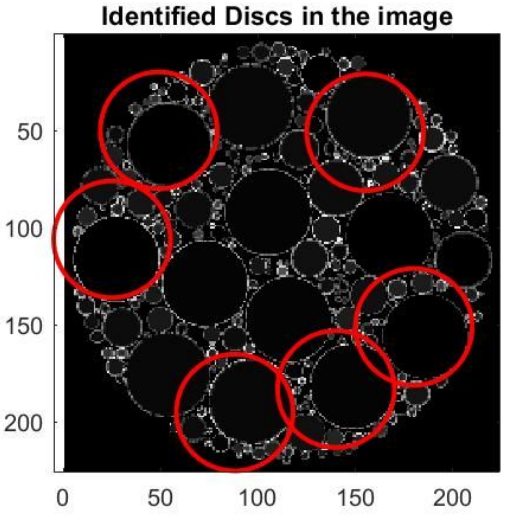
4.Princen J., Yuen H.K., Illingworth J. and Kittier J. "A comparative study of Hough Transform Algorithms: Part I - Line detection methods," Department of Electronic and Electrical Engineering, University of Surrey, U.K.

5.Princen J., Yuen H.K., Illingworth J. and Kittler J. "A comparative study of Hough Transform Algorithms: Part II- Circle detection methods," Department of Electronic and Electrical Engineering, University of Surrey, U.K.

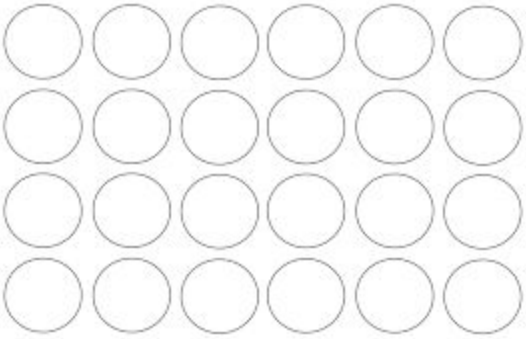
RESULTS

Input Image 1 :

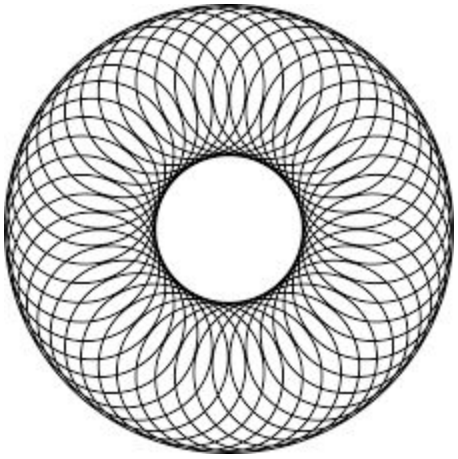
Output image 1 :



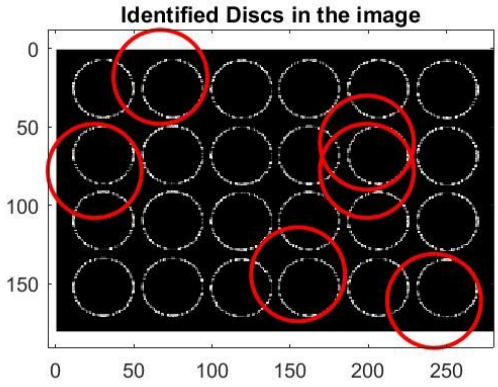
Input image 2 :



Input image 3 :



Output image 2 :



Output image 3 :