Circle Recognition using the “Folding” method

Guy Ben-Yosef, CS and Physics, Ben-Gurion University

Abstract

The problem of circle identification on an image is a known problem on the field of computational vision.

A common solution to this problem uses Hough transform, a Search And Score method, taking computational complexity of $O(N^3)$, where $N$ the image dimension, although its results can be distrusted by noise and large information edge detection result.

The algorithm presented, can be regard as a different version of the Hough transform algorithm, using the same search and score basic and the “folding” idea, aimed to overcome the weak points of the Hough transform algorithm.

A Folding of an image

Given a certain axis, on a certain 2D image, we can part the image into two sub image identical on its size, each one represent a different side of the axis. (The size of the two sub images will be the size of smaller one).

We will define an AND operation between these two sub image as a folding of the image around the given axis.

Such AND operation results an image includes only the symmetric points around the given axis, where we regard, of course, a non-zero number as 1.

The main purpose of folding is to reduce the amount of information leaving only the information relevant for identify the symmetry.

As a circle being an all around symmetric shape, we can use folding around its diameter as much as we want without losing valuable information for detecting it.

The Hough Transform base algorithm for circle detection

One way to cluster points that could lie on the same structure is to record all the structures on which each point lies and look for the structure that get many votes. This (quite general) technique is known as the Hough transform. We take each image token and determine all structures that could pass through that token [1].

Suppose this structure is a circle. Given an edge picture of a certain image, we would like to identify whether there is a circle on that image, where locates its center and what is its radius. The algorithm goes over each edge point and vote for its fitting to be part of any circle on the image. (Under the constraints). A circle would be mark as the triple <$X$ center, $Y$ center, Radius>.

Given the three-dim voting matrix we would choose the highest ranked cells (upon a threshold) to be the circles on our image.

Using the gradient direction of an edge point, we can vote for all the circles which its center lays on the perpendicular to the edge tangent. That would be all the structures that could pass through that token (edge point). Such voting process takes $O(N^3)$ time complexity, result total time complexity of the Hough algorithm to be $O(N^3)$ as well.
The algorithm advantages include robustness to shape distortions and to occlusions/missing parts of an object [3].

The normal for gradient approach works well for high signal to noise ratios and/or simple environments. As the signal to noise ratio decreases, the accuracy of the gradient estimation decreases. The fact that gradient-based methods are heavily dependent upon the accuracy of the gradient estimation explains why they are not robust to noise[3]. Another disadvantage is that, sometimes, the edge detector of choice does not provide gradient information [3].

In addition, the attraction of the Hough transform is that it connects widely separated tokens that lie close to some form of parametric curve. That is also a weakness; it is usually possible to find quite good phantom circles in a large set of reasonably uniformly distributed edge points [1].

**The use of folding for circle detection**

Taking advantage of the circle being an all around symmetric shape, and symmetric around the X axis and the Y axis in particular, we can go over each point on the X axis and fold the image around the Y parallel perpendicular to that X axis point. Furthermore, for each such fold image, we can go over the Y axis and fold the image again around the X parallel perpendicular.

Suppose this process is performed on an edge map, the result is significant information reduce image which represent the symmetry around the parallel and the perpendicular axis of each original edge image points. For an image point being a center of a circle, such double fold image will be unique: it will contain a quarter of a circle. It is important to emphasis, that the folding process can result as well all X and Y symmetric shapes, and noise, of course. The classification of which would be the algorithm next step.

**A Voting process on the double fold image**

Given a double fold image, the algorithm goes over each edge point on that image, and calculates the distance of the point from the up right corner (regard as the center of a circle). For a distance R it would vote one more voice for radius R. As a result of this process, we would get an array of all possible distances from the up right corner, for each distance - the amount of points being on it. At this level, the algorithm chose whether the image origin is a center of a circle, and if yes, what its radius is. We should be careful not to choose the highest ranked distance, and pay attention that the amount of votes should be related to the distance size. A smaller radius circle would result less votes then larger radius and therefore, may be confused by noise. That is why the algorithm picks the distance R which its amount of votes is satisfying:

\[
\text{Votes}(R) > R^* K, \quad K = 1.2
\]
Where $K$ is empirical constant around $\sqrt{2} \approx 1.4$ for estimating the size of the quarter circle arc and the amount of votes each distance should be look for in order to be a circle radius.

Dealing with noise, the smaller distances are more sensitive to it then larger distances. That is because, a small distance (1-5) would not be needed much votes to satisfy this inequality, and those votes can be noise. That is why the algorithm receives a threshold parameter $T$, for the minimum votes needed for a distance to be declared as a circle radius:

$$\text{(2)} \quad \text{Votes}(R) > T \text{ & Votes}(R) > R \times K, K = 1.2$$

Determining the parameter $T$ would effect much the radius sizes that can be detected. For example, choosing $T=10$ would make it impossible for the detector to detect a 1-8 radius distance circles. On the other hand, the bigger $T$ gets the better the algorithm avoid noise. Hence we should be careful when determining $T$, and if possible, use some pre-knowledge whether the circles are big or small, and the size of the image. However, satisfying (2) is not enough for a distance $R$ to be declared as a circle radius. It also should be the highest ranked among those who satisfy this term. This feather designated for dealing with fat and noisy edge contours. Again, on the other hand, it prevents the algorithm to detect two different circles on the same origin.

**Complexity**

The algorithm time complexity is $O(N^4)$ computational steps, where $N$ is the image dimension. The fold operation is rather expensive and take $O(N^2)$ computational steps. However, it is not the bottle neck of the algorithm:
The idea of going over all the edge points and regard each one of them as an origin, therefore vote for radius on each one of them would take as well $O(N^4)$ computational steps.

To conclude, time complexity analysis for this algorithm result:

$$(1,...X) \times (\text{fold around } X + (1...Y) \times (\text{fold around } Y + \text{vote process}) = N(N^2 + N(N^2 + N^2) = N^3 + N^4 + N^4 = O(N^4))$$

**Performance**

Given a various test image, the algorithm result was as follow:

A simple synthetic circle picture: (edge detector threshold=200, $T=20$)

Original: Edge picture: Fold-and-vote result: (detector threshold=200, $T=20$)
Parts image: (edge

Parts: Additive noise (edge detector threshold=200, T=20)

Parts: with Salt and paper noise: (edge detector threshold=200, T=20)
A real basketball (The Amsterdam university database): (edge detector threshold=200, T=20)

My magnum opus: A real basketball game image:
(Edge detector threshold=300, T=20)
The Olympics: (edge detector threshold=200, T=20)

The Edge picture:

The algorithm’s result:
The main advantages

By no means, the algorithm big advantages related to its performance. Theoretically, the algorithm is able to detect any circle, on any size and any center, as long as its edge picture, keeps the circle structure as a whole.

- However, on a real world there is noise, and the algorithm performance upon noise is an advantage, over other colleague algorithms. The Error measured on the algorithm’s test image, is:
  \[
  \Delta X_{\text{center}} \leq 2 \quad \Delta Y_{\text{center}} \leq 2 \quad \Delta R \leq 3 \quad \text{on average.}
  \]
  So that the algorithm shows good results in locating the position and calculating the radius.
- Another advantage is the ability to deal and recognize fat edges contour. The process of choosing the highest ranked distance among those who satisfy the conditions (as mention before) ensure the most accurate distance to be chosen for that origin. That is why the algorithm draws the circle as fat as it shows on the edge picture.
- In addition, the chance the algorithm will result circle that are not exist on the image is low. That is because it has three conditions to be satisfy along the way: the symmetry over X, the symmetry over Y and the threshold exam. That is in different then other algorithms which required the threshold exam alone.
- The algorithm reliability is not differing while dealing with noisy pictures, such as additive and salt and paper noise. Large amount of noise can not pass through the symmetric conditions (statistically 0.75\% of the noise on the circle area), and the noise that can pass fails on the vote process.

To conclude, the algorithm stay stable on its good performance when it deal with noisy and much detailed images.

The main disadvantages

First, the algorithm has major flaw giving result on an \( O(N^4) \) computational step.
- Dealing around 300x300 image size, that can be substantial. I must add, that I have made much effort trying to reduce the complexity using those methods and other, with no success.
- Another disadvantage, is the inability of the algorithm to detect an hidden circle behind another object (when its major area is shown, of course) . that is because small miss became big miss after folding.
- As mention before, the algorithm has problems to detect small circle as well, while lowering the T threshold may add noisy data to results.
- As mention before, the algorithm has problems to detect two different circles of the same of the same origin (center). That is due to the voting process which takes only the highest vote.
The Fold and Vote Vs. Hough

**Complexity:** HT (Hough transform) takes $O(N^3)$ while Fold&Vote algorithm takes $(N^4)$. **Accurate center location and accurate radius:** On a clear sharp edge image, both algorithms get good results, since they both use a voting method. Their both accuracy can be determined by a given threshold. **Dealing with noise image:** As shown on examples and mention before, the Fold&Vote algorithm performs well on an noisy image and therefore a noisy edge detection (both salt and paper and additive noise). The folding method gives for the Fold&Vote algorithm a small benefit upon HT. **Dealing with noisy edge-detection:** As mention before, the HT is heavily dependant on a good clear edge detection, both in edges and gradients, while the Fold&Vote algorithm is capable of handing noisy edge detection, as long as the circle edge stays as a whole. **Robustness to shape distortions and to occlusions/missing parts of an object:** The HT is capable of finding such circles while the Fold&Vote algorithm is not. **False detection:** As mentioned before, HT algorithm can fall into phantom circles, while the strict and simple voting process on The Fold&Vote algorithm minimize false detections.

Conclusions and Further work

The problem of circle detection on an image is a tough problem. The algorithm presented has its disadvantages, and the major of it, the way I see it is its time complexity. For example, it takes about 3-4 hours for it to detect a 260x355 image.

At first, I have spent great deal of time trying to find smaller time complexity algorithm, though I got the feeling that simple ideas would not lead me anywhere. It seems that smaller time complexity algorithm would take its price on the performances.

My way if thinking was how to use the circle being the only all around symmetric shape for its detection.

However, the main goal was to find a good time complexity algorithm, using all kinds of image manipulations, and on that I did not succeed.

I kept compare my algorithm to the Hough transform base algorithm, that we studied in class, and to the naïve circle detection algorithm, where you go over all center points and all possible radiuses, and search for a circle. I guess my algorithm is somewhere between. The more I kept thinking about this problem the more I realized how difficult it is. I do think now, that progress on solving the problem need the use of other tools then image manipulation, such as inference, for example.

The algorithm, of course can use for detection of all kinds of symmetric shapes (such as ellipse), after making few changes on the voting process.

I hope that the idea of folding may be in use for future solving other problems.
References

3. Circle recognition through a 2D Hough Transform and radius histogramming
   Dimitrios Ioannoua, Walter Hudab, Andrew F. Lainece,*, Columbia University. 1999