Circles Finding with Clustering Method

Computational and Biological Vision

Final Project Report – Shimon Machluf 034883876

1. Project Goals

The main goal of my project is to find circles in a picture.

A secondary goal (or maybe a secondary result) is to find symmetric shapes, like circles or squares.

2. Course of Action

Unlike Hough transform for line and circle detection, which basically needs an edge detector that return the edge gradient, my algorithm is based on different idea.

My idea is that since circle are symmetric shapes, their "center of mass", or if we'll take all the pixels that create the circle, treat them as 2 dimension vectors and find their mean value (this is exactly the way to find the center of mass), the result is the center of the circle.

Since I'm using a clustering method, the picture is divided into clusters; each shape is a different cluster, and all the pixels of each shape, which are all the points of the analogous cluster, are getting the same unique label. If I'll calculate the mean value of all the points with the same label, I'll find the center of that shape.

Now, if I'll go to all directions and will find the shape's end, or edge, I can check that I went to each direction the same distance till I reached the edge, and if so, this is a circle.

This description is very simplified, and there are some problems with it, some can be solved, and were solved, and some can't. The algorithm will be explained later, and the problems also.

2.1. Brief Description of the Algorithm

(full Matlab code is linked in the web page)

2.1.1. First, I'm using a clustering algorithm (I choose a mean shift clustering, by Dorin Comaniciu: http://www.caip.rutgers.edu/~comanici/), which return the clusters in the picture, means a map from the pixel index to its label value (this is a matrix the size of the picture and each element in it has an index as a pixel in the picture and the value of that matrix' element is the label of that pixel).

An explanation on the algorithm can be found in http://www.caip.rutgers.edu/~comanici/Papers/Cluster.pdf (section 5).

There is no necessity to use the mean shift clustering; any kind of clustering is good enough, as long as every pixel in the picture has a cluster label.

2.1.2. When I have the clusters in the picture (each cluster has its unique label) I know that the edges in the picture are the pixels that have one cluster on one side and a different cluster on the other side.

Some Advantages of this method will be presented in next part.

2.1.3. Independently from the edge detection, I find the cluster centers. As was explained earlier, a cluster center is the mean of all the pixels in that cluster (or the mean of all the pixels with the same label).

Of course it is not as simple as just sum all the pixels with the same label, because a shape can hide another shape (looks in the picture like it on top of the other one), and in this case, the center of the larger shape, that is partially hidden, is the mean of the pixels with these two labels. And the same if there are more shapes on top of each other.

I do that by going to all directions, and save all the different labels I go through till I reach the final edge of this cluster. To find the final edge I need to check each time I reach en edge that the last pixel has this cluster label, and save it as the last one, for now. I continue till the end of the picture and use the information I save on the last edge.

This is done in "findNewCenters.m".

2.1.4. After I have the clusters' centers and the edge map I can find all the symmetric shapes in the picture.

I do that by defining an angle (start with 0, increases with a given parameter, till π), find the cluster edge, and then check if on the same distance but the opposite direction (angle + π), there is an edge.

Finding the cluster edge is not just to go in the same direction till I reach an edge. Again, because of there can be a shape on top of another shape and in that case the first edge is not the edge I'm looking for. I can solve this problem by checking that the last pixel has a label like the cluster I'm currently checking.

This is half of the solution, because I can go through several shapes till I reach the end of the shape, so I need to find the last edge that the pixel before it has the same label as the cluster label, and now I can be sure this is really the end of the cluster, the real edge of the shape.

This part returns for each symmetric shape the coordinate of its center and a list of all the edge points of this shape.

2.1.5. The last part of the algorithm is to take all the symmetric shape, and find out which ones are circles.

This is done by finding the maximum and minimum distance of any edge pixel (of this specific shape) to the cluster center, and if the subtraction of the minimum distance from the maximum is smaller then a given threshold, then it is a circle. The threshold separates circles from ellipses or other semi-circle shapes.

3. ***Discussion on This Way of Action***

3.1. ***Disadvantages***:

3.1.1. The main goal of this project is to find circles in a picture, but it work only if all the edge of the circle is visible. If not, the cluster center in not in the circle center.

3.1.2. This algorithm is depending greatly on the clustering algorithm, in computation time and the results themselves.

For example, all parts of my algorithm are fast enough to run on personal computer except the 6'Th part of the mean shift (the k-nearest-neighbors, as I implemented it) which take too long to run, and that limits the size of the pictures I could test.

3.2. Advantages:

3.2.1. The secondary goal, and result, is to find symmetric shapes.

3.2.2. After I find all the symmetric shapes, I'm looking for circles, but I don't have to stop there. I can, for example, look for squares (or ellipses, etc…) by only writing the findSquare (or findEllipse, etc…) function that will use the existing findSymmetric function (just like findCircles method does), so almost all the work is already done and the new function will be short and simple one.

3.2.3. The circles that the algorithm finds are circles of one object, if there are several objects that look like a circle, two half circles put together for example, it will not find it to be a circle because the algorithm check each object for itself.

3.3. Dealing with Noise:

As I mentioned before, this algorithm is greatly dependent on the clustering output. If the clustering is good and can handle noise well, then the rest of the algorithm will work fine. Because that after the clustering the noise disappears, every pixel doesn't have it value anymore, but just a cluster label, and if the clustering is working well, then every pixel have the right label.

3.4. Some Advantages for Using Edge Detector Based on Clustering Method:

3.4.1. Algorithms based on gradient edge detectors trying to calculate the edge angle and this method has some problems, like:

3.4.1.1. Because we're dealing with pictures, the angles can have only discrete values, and as the radius is getting larger the different between the real center and the line calculated from the edge angle is getting significant.

3.4.1.2. Any method based on gradient is sensitive to noise, because the derivatives just increase the noise.

3.4.2. On the other hand, edge detector based on clustering has a few advantages:

3.4.2.1. Clustering can work pretty well with noise; it is a smoothing method in addition to the clustering ability, just because of the way it works (the mean shift clustering does that).

3.4.2.2. Even if the noise still damage the picture and the edge of the clusters (and the edge itself) is not smooth and have some noise in it, since the algorithm is looking for the center of the shape, the vibrations in the edge will have small or no effect at all on the shape center.

3.4.2.3. The edges that the edge detector finds are only two pixels wide. That because I consider a pixel as an edge only if it touches the line between two different clusters, and the line has only two pixels on its two sides.

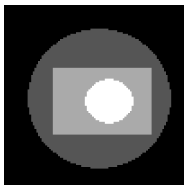
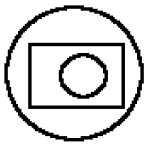
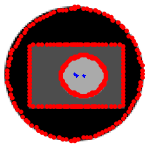
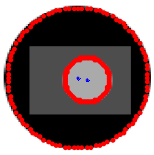
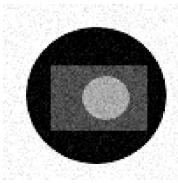
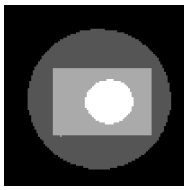
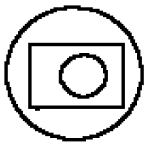
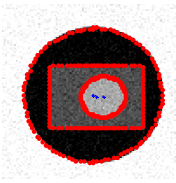
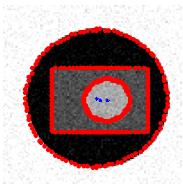
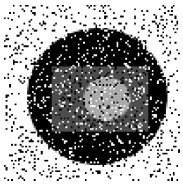
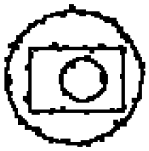
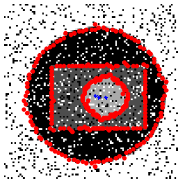
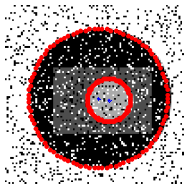
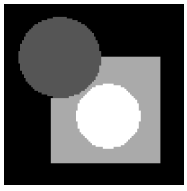
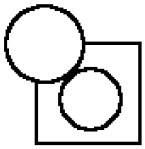
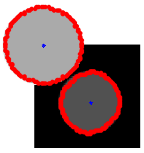
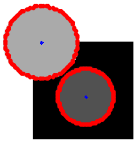
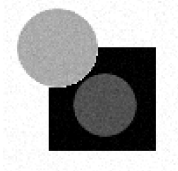
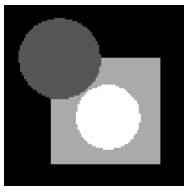
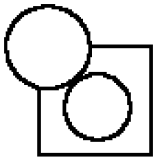
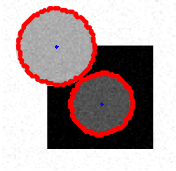
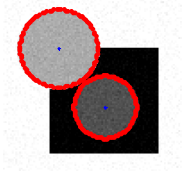
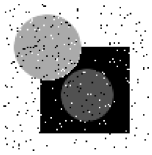
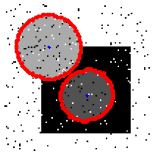
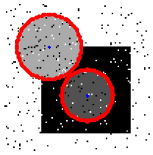
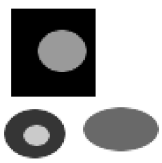
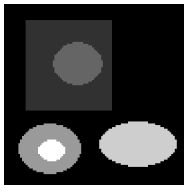
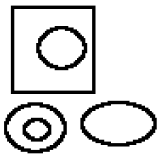
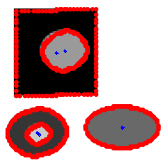
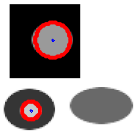
3.4.2.4. Unlike other edge detectors, this method doesn't have any problem with junctions of edges. There are no blank parts in the edge map near the junction of edges, and that also because of the algorithm is based on clustering and not gradient of the picture.

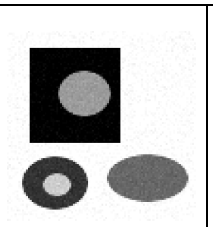
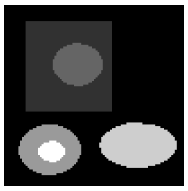
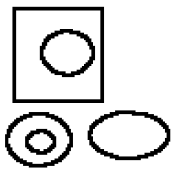
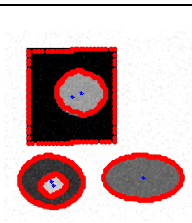
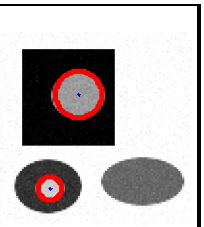
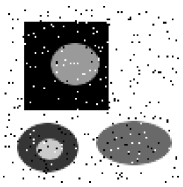
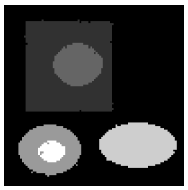
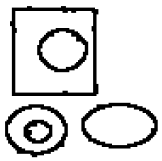
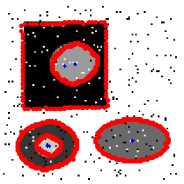
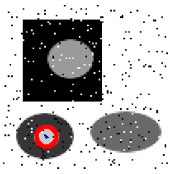
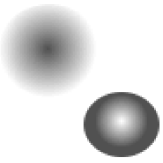
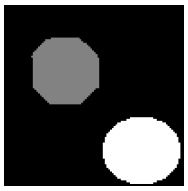
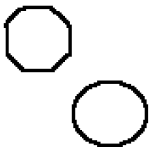
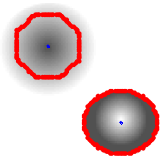
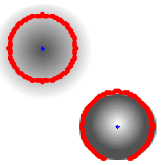
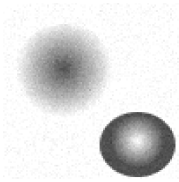
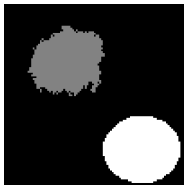
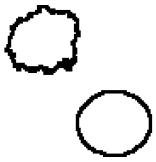
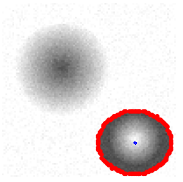
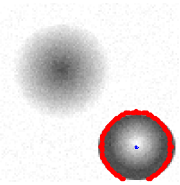
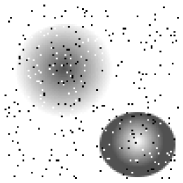
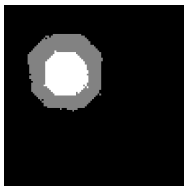
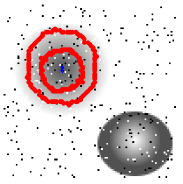
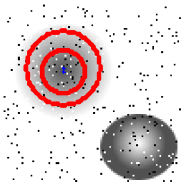
4. Results

Next there is a table with some examples. It has the pictures, with and without noise, and the clusters the mean shift algorithm has found, the edge map, the symmetric shapes and the circles.

This part should be seen in an electronic format (doc or pdf file) because the printing doesn't do well to the pictures.

All the test were done with the same parameters (except the mean shift parameters): Tround = 3, and 50 tests for each shape.

	Original picture	Clusters	Edge map	Symmetric shapes	circles
Test 1					
Test 1 with Additive noise					
Test 1 with Salt & Pepper noise					
Test 2					
Test 2 with Additive noise					
Test 2 with Salt & Pepper noise					
Test 3					

Test 3 with Additive noise					
Test 3 with Salt & Pepper noise					
Test 4					
Test 4 with Additive noise					
Test 4 with Salt & Pepper noise					

5. ***Conclusions***

5.1. As we can see, the algorithm is working well. Especially with pictures without noise or with additive noise, and even with some of the examples with salt & pepper noise.

5.1.1. The major problem with salt & pepper noise is that a black pixel on white background (or white pixel on black background) can be considered as a cluster of its own.

5.1.2. The algorithm handles additive noise much better because the way the mean shift algorithm works is to smooth the picture.

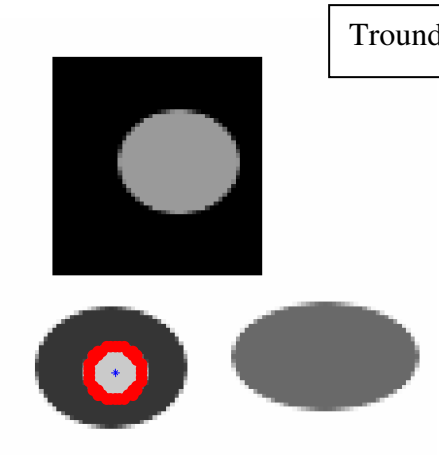
In the 5'Th test it didn't. That's because there is a shape that is fading into the background, and in the areas closer to the background the noise change the color enough to be in the color of the background, and its close enough to the end of the shape, therefore it is considered as part of the background.

5.2. In the 3'rd test there are circles and ellipses that are very close to circles.

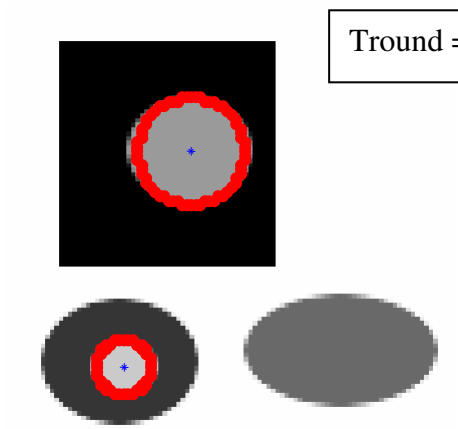
If I change the parameter Tround (which determined how round the circle needs to be, or the different between the largest distance from the edge to the center and the shortest one) I can add or remove shape from the result of "findCircles.m". But the shape will still be considered as a circle.

For example:

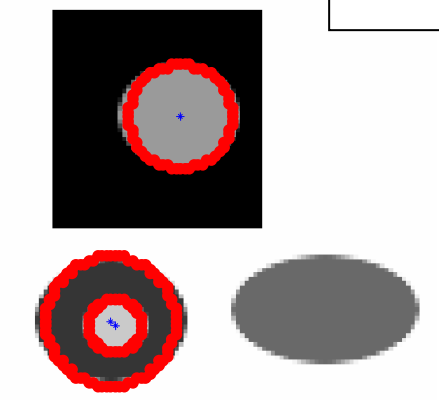
Tround = 2



Tround = 3



Tround = 4



5.3. The algorithm can't find circles if the entire edge of the shape isn't visible.

That's because the calculation of the center of the shape is based on summing positions of pixels of full clusters, and if the cluster edge is in the middle of another cluster the algorithm can't know where the edge is, and then the center it will find won't be the correct center.

5.4. The other side of this problem is that if there are several shapes that their edge map look like a circle, for example, two half circles put together will look like a circle, and other algorithms based on edge's gradient may find this case as a circle, but cluster based algorithm is checking each object if it is a circle without knowing anything on other objects or the picture, and in this example the object is half a circle.