IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 21, Issue 1, Ver. I (Jan - Feb 2019), PP 74-85
www.iosrjournals.org

Polygon Detection Method by Direction Clustering Based on the Peeling Technique

Mengxi Tan
(SpaceSoftwares)
Corresponding Author: Mengxi Tan

Abstract:*In this paper, we propose a fast real time clustering method and an algorithm for detecting polygon lines and their orientations. The content is presented in two parts. In part I, a new clustering method based on density peeling technique is proposed, which sets up the basis of objects features clustering; in part II, a new polygon detection method is proposed based on the peeling technique and a statistical histogram of directions of pixels of the polygon lines; the effective pixels, which contributes to the orientations and positions of polygon lines, are extracted as soon as the polygon shape is obtained. The methods proposed in these two parts are based on the principle that iterative searching and comparisons should be maximally avoided and thus are fast algorithms suitable for real time application, especially for embedded systems.*

Keywords:*clustering; contour tracing; peeling technique; polygon detection; direction histogram*

Date of Submission: 12-01-2019 Date of acceptance: 27-01-2019

I. Introduction

Polygons, such as rectangles, are widely used as standard shapes of industry products and daily goods. Besides general image processing, polygon recognition and positioning are the basic techniques required and massively applied in modern auto-industry such as electronic chip mounting, goods organization, and so on, where the high efficiency is still challenging.

To satisfy the demanding efficiency requirements in industry application, a new polygon detection method is proposed in this paper. The relevant content is divided into two main parts.

The first part presents a fast clustering method based on the so called "peeling technique" in this paper, which identifies the densely located parts of clusters by removing the sparsely located area.The second part gives a new algorithm for fast polygon detection, which is based on the peeling technique and the histogram of directions from the polygon. The two parts can work independently, that is to say, we use and recommend to use the density peeling technique as proposed in part I to do the direction clustering as proposed in part II, while it is feasible to use some other general clustering methods to do the direction clustering, as long as efficiency becomes less important or someone has any special reason.

The algorithms presented here mainly manipulates object positioning (including orientation) after the object border contour has been detected, so we will not go in detail about object border detection since it is not part of our main discussion. You may use Sobel[1] edge detection method in a real time case or Canny[1] edge detection method to extract the object borders depending on your requirements.

II. Part I, Clustering Method Based On Density Peeling Technique

Clustering is a basic data mining technique, it is a task of classifying data objects into different group sets, generally referred to as clusters, in which objects in the same group (or cluster) shares more similarities (or features) to each other than to those in other groups (or clusters). Depending on the data model and cluster constitution, the algorithm efficiencies differ significantly. In typical unsupervised situation, most traditional algorithms progressively examines the data features and make advancement until all features are fully checked, thus the time complexities remain high. Typical examples may include the K-means[2,3], where the centroid as the mean of the objects features is updated repeatedly when new samples are kept rolling in, and similar algorithm EM[4], whose parameters are kept iteratively being updated and optimized in order to fit the data objects better in stepwise advancement. For high speed clustering, some density based algorithms such DBSCAN[5] and some grid-based algorithms, including STING[6,7], CLIQUE[6,8], and Wave Cluster[6,9], exhibits good performance. But their problems are similar, certain amount of repeated checks of all density features are still requirements, and they have similar difficulties in finding complex clusters such as nested ones. Besides, methods for specific scenarios give further limits to their application, e.g., the CLIQUE method proposes to find high dimensional data by checking high density regions in sub dimensional spaces before checking the higher dimension, while in application we are more likely to face cases with low dimensions, or to transfer high dimension problems to lower dimension to facilitate the process of problem solving, it will not save much time except to increase computation complexities since under these circumstances the sub dimensional spaces are generally full of data distribution.

We noticed that, for general problems the key part of clustering is to find the dense cluster centers, which are referred to as “core” in this paper, and on most occasions data objects can be divided into “boundary” (or “surface”) parts and core parts; that is to say, finding the boundary parts and finding the core parts are equivalent in expression, indicating a final solution to the task of clustering. Following this idea, we found that the core parts can be obtained by “peeling off” the boundary parts with some simple minus-plus operation. Compared to those traditional grid and density based algorithms, the complexity is further reduced with this peeling technique since the core parts remain intact during the process of clustering. The computational complexity reduction is of special significance in cases where the grid mesh resolutions have to be extremely fine due to highly irregular data distribution or in high dimensional data objects distribution.

2.1 Contour Tracing

One of the main tasks of the peeling technique is the contour tracing of density distribution, thus the efficiency of the contour tracing algorithm is demanding. In the following description in this section, we take the general image object's border tracing as an example, and paint the border pixels gray in purpose of demonstration. Please note that the tracing principles are the same for both boundary extraction from image object and contour extraction from density distribution, just in density contour extraction it is the grid cells instead of image pixels that are being traced.

In an 8AC (8 adjacent connections) case as shown in Fig.1(a), each pixel has 8 neighbors which are named as “quad”s in this paper, similar to the concept of “quadrant” in a plane coordinate system. Please note that quad is a property of a certain pixel, and it cannot exist independent of pixel in relevance.

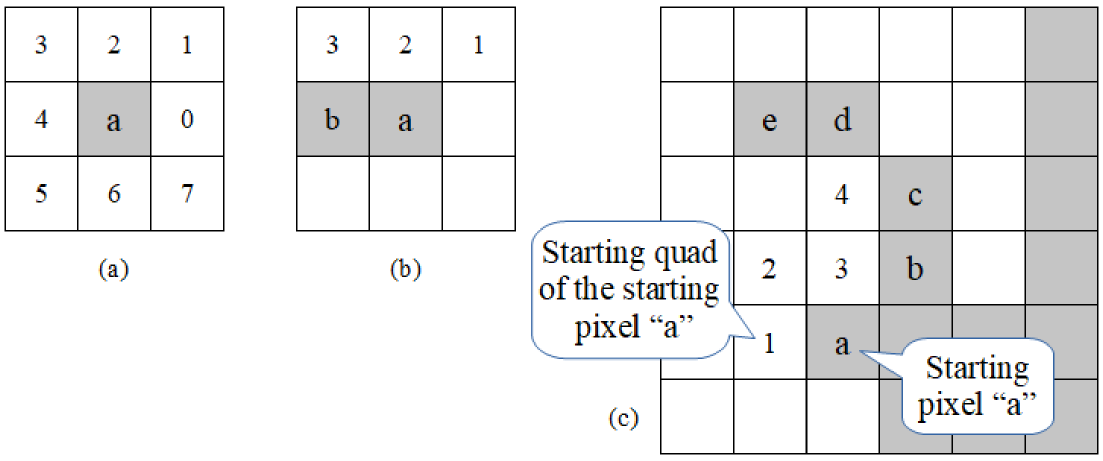


Fig.1 Demonstration of a contour. (a) pixel “a” and its 8 “quad”; (b) illustration of the searching quad pattern around pixel “a”, where “b” is in quad 4 of pixel “a”, if we search around “b” clockwise we will meet pixel “a”, then after moving to pixel “a” the next search should start from quad 1 of pixel “a”; and (c) the 2 important locations for stop criteria: the starting gray pixel “a” and its starting quad number 4 where the white pixel is marked “1”.

Without loss of generality, the tracing can be done clockwise or anticlockwise, similar to the traditional contour tracing algorithms[10-12]. Here we present our method in a clockwise way. The tracing can be started from any direction outside of the target object. As shown in Fig.1(c), we randomly select a line and move from left to right until the first gray pixel is hit, one important thing to remember is that we need to mark the 2 starting positions for stop of tracing: (1) the first gray pixel being hit (pixel “a” in Fig.1(c)), and (2) the quad of the gray pixel from which the search started, (the white pixel marked “1” in Fig.1(c), which is in quad 4 of pixel “a”).

We start from pixel marked “1” in Fig.1(c) and tour around pixel “a” clockwise to find the next gray pixel. In the process the white pixels “2” and “3” are examined before we reach the target pixel “b”. We move to pixel “b” once it is identified as a gray pixel, since pixel “1”, “2” and “3” are already examined when we are touring around pixel “a”, so this time we directly start the search from pixel “b”’s next quad where the pixel is marked “4”, touring around pixel “b” clockwise and we find the gray pixel “c”. The process continues until all gray boundary pixels are collected, and terminates until we hit the starting pixel “a” and it’s starting quad number 4 for the second time. The collected gray pixels are generally recorded in a list referred to as "contour list" in this paper.

Now the problem is, how we can avoid repeating checking of the white pixels. As illustrated in Fig.1(b), considering the current pixel is "b" and the next gray pixel "a" is in quad 0 of pixel "b" (in such a case pixel "b" is in quad 4 of pixel "a"), we know that after moving to pixel "a", the next quad to be checked for pixel "a" should be quad 1, so we mark the 2 important quad info: quad 0 of pixel "b" and quad 1 of pixel "a". For each pixel there are only 8 quads, i.e., 8 neighboring searching cases, which is a rather limited number so we can build a look-up table to for this relationship, as shown in Table.1.

Quad	0	1	2	3	4	5	6	7
Next search quad of the next pixel	1	3	3	5	5	7	7	1

Table 1 Current pixel's quad number vs. the next search quad of the next pixel

Thus, in contour tracing the repeating checking of the white pixels have been maximally avoided, the computational cost are kept very low, in a case of straight line tracing the cost is just half of the traditional method as described in documents [10,12]. Another feature worth mentioning is that, compared to the stop criteria of the traditional methods, which requires a repeat check of several gray pixels [12], our stop criteria is simpler both in understanding and coding.

It should be mentioned that, in contour tracing of polygons of general goods, the dangling lines are generally noise sections, such as the section consisting of pixels "c", "d", and "e" in Fig.1(c) where pixel "c" is the "pivot" pixel of the dangling section. A dangling section can be removed by identification of its pivot pixel, which bears two typical characteristics: (1) it has been traced more than once in the contour extraction; (2) its first predecessor is exactly its last successor.

2.2 Density Peeling Technique

Considering data distribution in a 2D plane as shown in Fig.2, the algorithm is presented as follows

```
Process of the Density Peeling

Step 1: divide the whole area into a grid with limited number of cells
Step 2: register data objects into relevant grid cells
Step 3: refine the grid resolution or distribution (optional)
Step 4: set the initial parameters
while end criteria has not been met
Step 5.1: remove a certain amount of density from each grid cell;
Step 5.2: trace the density contour, and remove the noise cell;
Step 5.3: check the meet-end-criteria condition
Step 5.4: update parameters and sub clusters (optional)
end of the while loop
Step 6: use the separated core's average and variations to finalize the clustering
Done
```

Step 1: create a grid with a certain resolution.

For general problems of normal distribution, a grid with homogeneous resolution would be OK. As shown in Fig.2, the data groups are normally distributed and a homogeneous mesh with 15×15 cells is to be used. Here the grid resolution is kept low for purpose of demonstration.

In a real time application for such a similar case the grid resolution can be much higher since the overall computational cost is very low.

Step 2: register data objects into relevant grid cells.

Data objects are collected in each grid cell according to the statistical attributes located in the relevant region. The initial density distribution of our demonstration is given in Fig.3 (a).

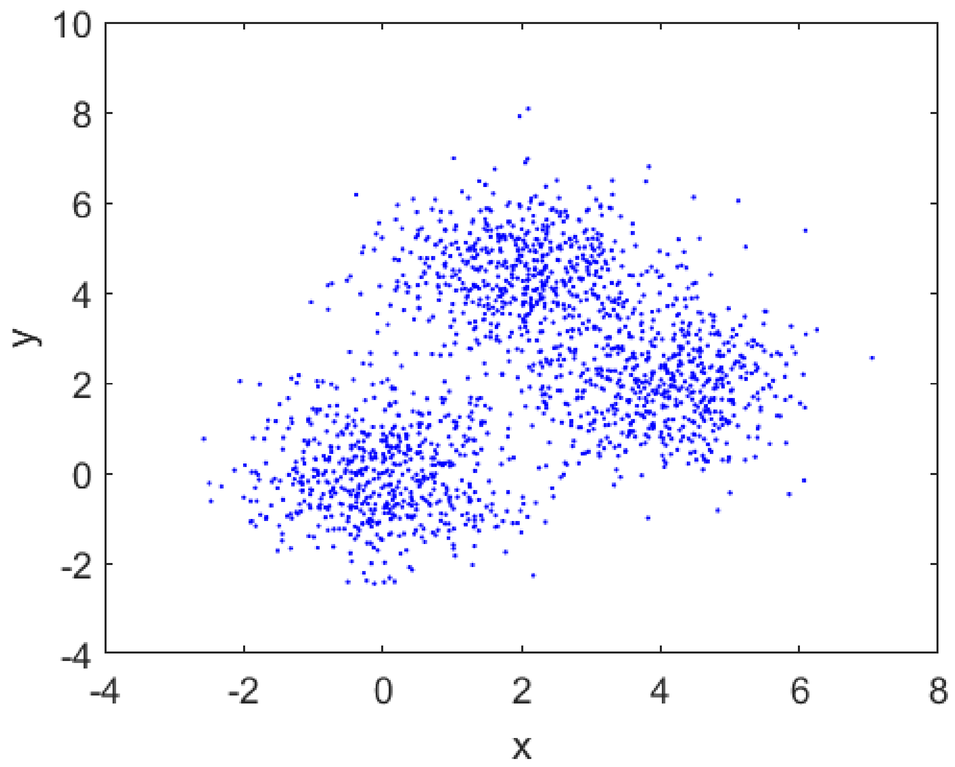


Fig.2 An example of data groups in normal distribution.

Step 3: refine the grid resolution or distribution (optional).

The grid resolution refinement is specific for high irregular data distribution, where a grid with homogeneous mesh resolution may not be able to fulfil the task of high quality clustering. Under these circumstance, we recommend low-resolution meshes for sparse distribution regions and high-resolution meshes for densely distributed regions; that is to say, grid cells with higher density of attributes distribution should be adaptively refined with higher granular resolution.

It should be noted that, for scenarios require refinement, there may be lots of localized variations since the data distribution is irregular, which may bring lots of localized noisy peaks in the process of clustering and seriously increase the time of noise removing; so in this stage it may be better to do some data smoothing, such as mean average filtering.

Step 4: set the initial parameters.

For each grid cell (*i*, *j*) (*i*, *j*=0, 1,..., 14) the initial peeling interval $P^0_{(i,j)}$ should be set. The value is homogeneous in this step, in our demonstration it is set to 4.

Step 5.1: subtract each grid cell (*i*, *j*) with a certain peeling interval$P^r_{(i,j)}$, where *r* is the number of iterations.

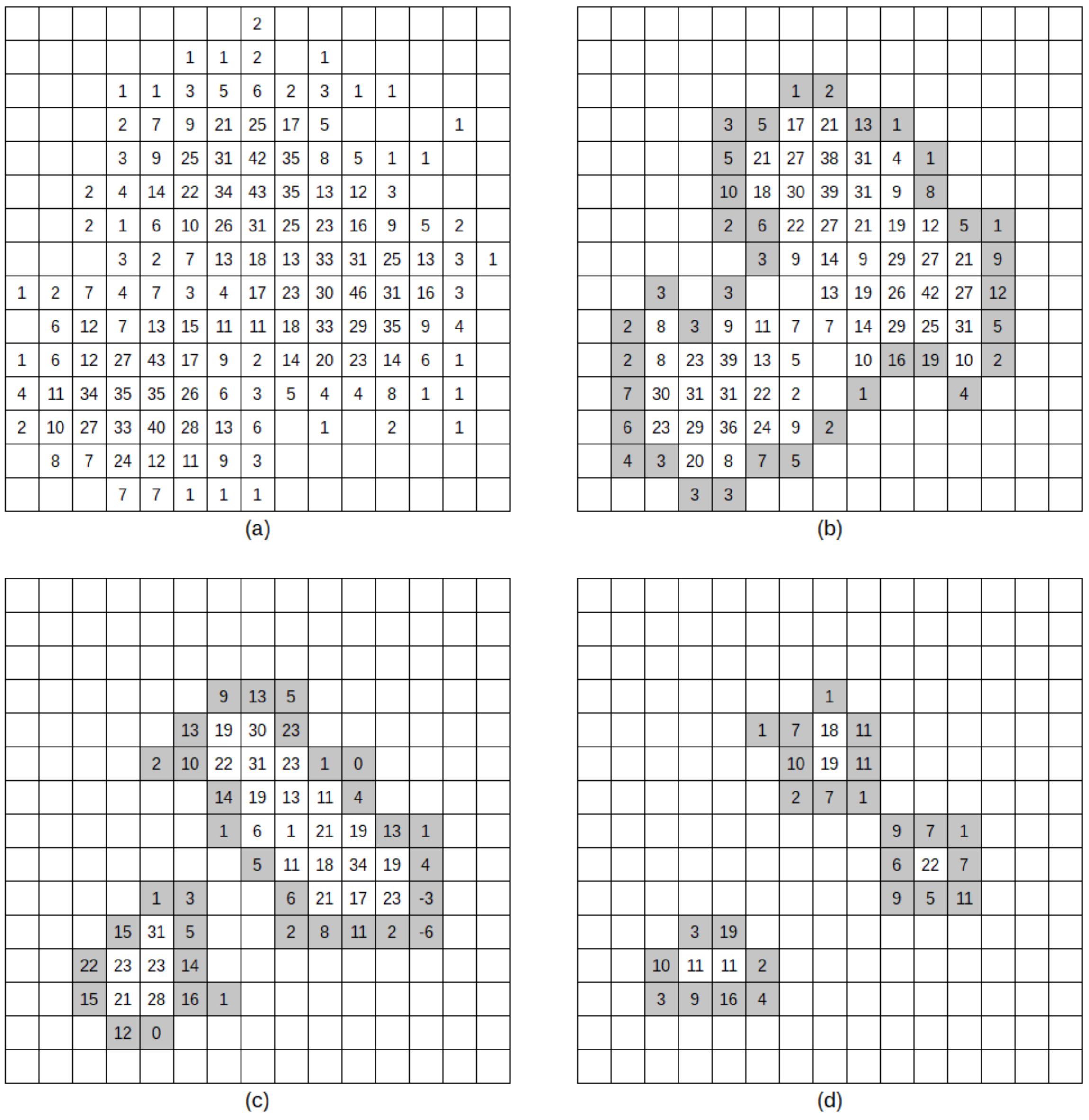
Generally the value $P^r_{(i,j)}$ is constant and homogeneous for all grid cells; but, in order to obtain a higher clustering quality it may also vary according to specific requirements such as irregular or exceptionally dense distribution. In addition, in searching potential sub-clusters in different parent clusters, some different peeling intervals may be used for data objects of different density distribution.

Fig.3 (b) gives the result of the first peeling.

Step 5.2: trace the density contour.

As show in Fig.3(b), (c) and (d), the boundary cells are extracted sequentially according to the contour tracing algorithm presented in the above section, and a list recording these boundary cells is made for each separated contour, which is considered as the border of a potential cluster "core".

In this stage, some extra check is required to avoid taking the noisy variation as cluster cores. Generally a contour can be regarded as noise instead of a cluster core if the number of grid cells in this contour is less than a certain threshold.



(a)

							2							
					1	1	2		1					
			1	1	3	5	6	2	3	1	1			
			2	7	9	21	25	17	5				1	
			3	9	25	31	42	35	8	5	1	1		
		2	4	14	22	34	43	35	13	12	3			
		2	1	6	10	26	31	25	23	16	9	5	2	
			3	2	7	13	18	13	33	31	25	13	3	1
1	2	7	4	7	3	4	17	23	30	46	31	16	3	
	6	12	7	13	15	11	11	18	33	29	35	9	4	
1	6	12	27	43	17	9	2	14	20	23	14	6	1	
4	11	34	35	35	26	6	3	5	4	4	8	1	1	
2	10	27	33	40	28	13	6		1		2		1	
	8	7	24	12	11	9	3							
			7	7	1	1	1							

(b)

						1	2							
				3	5	17	21	13	1					
				5	21	27	38	31	4	1				
				10	18	30	39	31	9	8				
				2	6	22	27	21	19	12	5	1		
					3	9	14	9	29	27	21	9		
		3		3			13	19	26	42	27	12		
	2	8	3	9	11	7	7	14	29	25	31	5		
	2	8	23	39	13	5		10	16	19	10	2		
	7	30	31	31	22	2		1			4			
	6	23	29	36	24	9	2							
	4	3	20	8	7	5								
			3	3										

(c)

						9	13	5						
					13	19	30	23						
				2	10	22	31	23	1	0				
						14	19	13	11	4				
						1	6	1	21	19	13	1		
							5	11	18	34	19	4		
				1	3			6	21	17	23	-3		
			15	31	5			2	8	11	2	-6		
		22	23	23	14									
		15	21	28	16	1								
			12	0										

(d)

							1							
					1	7	18	11						
						10	19	11						
						2	7	1						
									9	7	1			
									6	22	7			
									9	5	11			
			3	19										
		10	11	11	2									
		3	9	16	4									

Fig.3 The peeling process of data distribution as shown in Fig.2. (a) The initial density distribution; (b) the result of the first peeling ; (c) the result of the 3rd peeling; (d) final result after 6 peeling iterations.

Step 5.3: check that whether the termination criteria has been met.

The end criteria may vary according to specific application. In our demonstration, the peeling process stops when 3 groups of data objects are identified, the final result is given in Fig.3 (d), where 6 peeling iterations are performed.

If the termination criteria has not been met, a new cycle of density peeling process will continue, as exemplified in Fig.3(c) as a result of the 3rd peeling iteration.

Step 5.4: update parameters and clusters (optional).

In order to gain a high clustering quality, it is possible that nested clusters may require higher level of peeling resolution, in such a case parameters such as peeling intervals $P^{r}_{(i,j)}$ may be specifically adjusted.

Step 6: use the separated core's average and variations to finalize the clustering

Now the core parts have been successfully identified, generally data objects with normal distribution can be clustered according to Gaussian models with the obtained averages and relevant variations.

2.3 Application in One Dimensional Data Distribution

Similar to the 2D plane peeling, the basic idea remains the same in a one dimensional case of data distribution. A typical example is shown in Fig.4 where 2 mixed Gaussian cores are illustrated. The idea of the peeling technique is to remove, as minimally as possible, a certain amount of out-layer population (indicated with light gray) from the distribution so that the 2 dark gray Gaussian cores can be spatially identified.

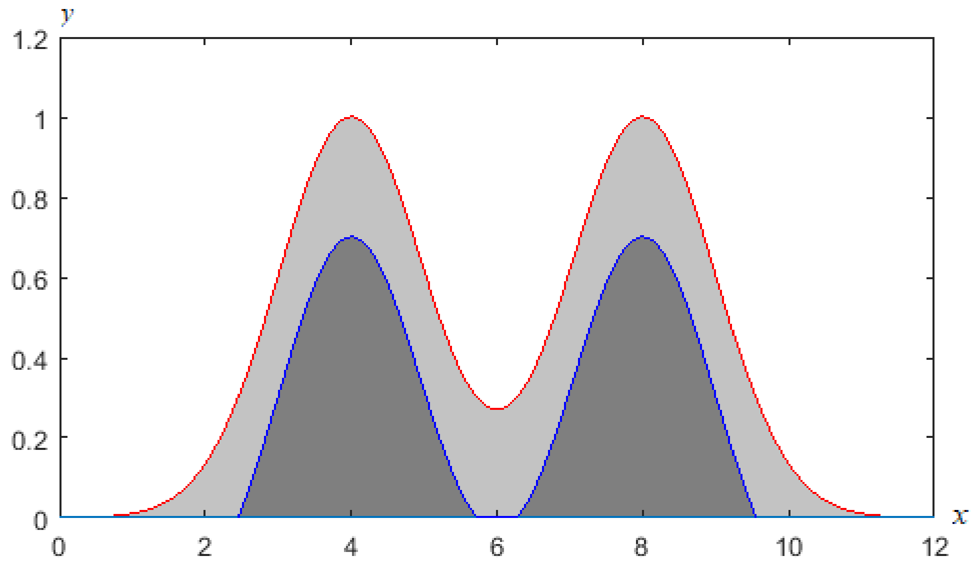


Fig.4 An example one dimensional data distribution with 2 mixed Gaussian cores

2.4 Comparison to Other Density Related Algorithms and Discussions

General density based algorithms keeps merging data object with a density value defined by the number of its neighbor objects within a given radius, in which DBSCAN is a typical representation with a general complexity of $O(n^2)$ [5]. The iterative density calculation for each data object makes this algorithm extremely expensive in high density regions. Another cost comes from the initial parameter setting, the clustering quality may be very low if the threshold is not within a suitable range, and the situation becomes worse when more than one threshold is required for finding nested clusters or sub-clusters, making it impractical in real time application.

The other grid-based clustering algorithms have similar problems for threshold setting; and, as mentioned earlier in this paper, they require repeated check of the individual cells during clustering. Furthermore, some grid-based algorithm such as STING[7] requires individual statistics of data objects in each individual grid cell, which is relatively expensive in computation when the grid resolution is high.

The peeling methods proposed overcomes the above problem.Its computation involves mainly simple minus-plus operation; and distinctively, the peeling method only checks the out-layer contour of data distribution.

Most important of all, as previously described in this paper, density peeling is a fast try-and-test method. Since for most general problems we have no idea beforehand about the number and status of potential clusters, sub-clusters, and their relations, so we have no knowledge of the optimal threshold, with traditional clustering algorithm it is difficult to achieve a successful cluster separation, especially for closely mixed ones. The key idea of the peeling method is that, since the threshold is unknown, we can accumulatively increase it to get the optimum, by setting a certain small interval as the amount to be removed at each step (where the total removal is equivalent to the threshold), and each time we check the results to see whether we got newly separated clusters or not.

It should be mentioned that, since the core gives the known part of a certain cluster, in order to reduce the unknown part we need to get the separated cores as big as possible; thus, as a general rule the removal (peeling interval) should be as small as possible when the computational cost allows.

A typical example is that in a 3D distribution with dimensionality of (k, m, n) in the worst cases the task of a peeling process requires 2×(k×m+m×n+k×n) times of searching and comparison in each iteration instead of (k×m×n) times as required by other grid-based algorithm. This means the complexity of the peeling algorithm is relatively one magnitude lower and can be much more time saving when the grid resolution is high, especially in high dimension.

In a grid with adaptive resolution, the peeling technique brings extra benediction. This is because in contour tracing we only check the sparsest grid cells which is in the boundary contour region of distribution, and check-ups of the dense grid cells in high density areas are avoided in each iteration.

It should be noted that the only exception where the inner core cells requiring specific check is that the clusters consist of data group (or groups) being fully encircled by some other data group (or groups). As a general rule, we don’t check and don’t recommend to check the data core parts unless there are special requirements, especially in high dimensions, though it is feasible to do any statistic check as anyone who might expect to.

III. Part II, Polygon Detection with Direction Histogram

There are various techniques to identify object shapes in computer vision, concerning line and circle detection the Hough transform may be the most widely used method [13, 14] in application, which constructs a parameter space and then uses a voting procedure to obtain the object candidates as local maxima in that space; similar voting mechanism was adopted by Shaw et.al [15] to detect circles and octagons of the high way speed signs.Since these voting methods need to build all possible combinations of featured values of different parameters, the computational complexity is high and difficult to apply in real-time industry applications especially of embedded systems.

The more efficient polygon extraction method may be the RDP (Ramer–Douglas–Peucker) algorithm [16-18] with a general computational complexity of $O(n\log(n))$ [18]. It makes an imaginary line between the first and last point in a set of points of a certain section, and finds which point in between is farthest away from this line. If the farthest point is closer than a given distance threshold 'epsilon', it removes all these in-between points and the line is saved as a reduced form of the two points: the first point and the last point; on the other hand, if this farthest point is farther away from this imaginary line than epsilon, the curve is split in two parts: (1) from the first point up to and including the outlier, and (2) the outlier and the remaining points. The function is recursively called on both of the resulting curves, until all reduced forms of the curves are collected.

But the RDP method is sensitive to noise, as shown in Fig.5 (a), if there is a noise curve outside the polygon, the wrong (dashed) lines will be detected as border lines instead of the correct (solid) ones; similar problems can happen to rounded corners which are the common shapes of industry products, as illustrated in Fig.5 (b) where the dotted lines will be mistakenly accepted as border lines of the polygon.

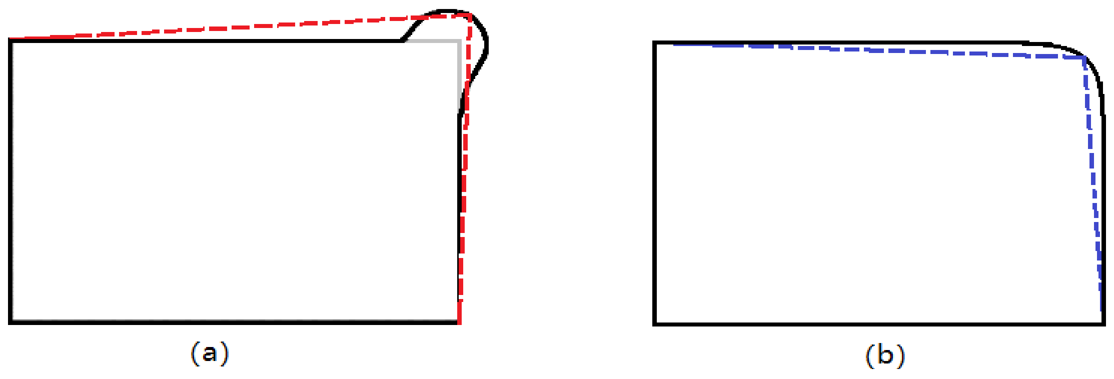


Fig.5 (a) noise curve on the corner leading to wrong border line detection; (b) rounded corner of a rectangle leading to wrong border line detection

3.1 Pixel Directions in Curves

Considering pixel “k” of a border line as illustrated Fig.6, which are consecutively collected pixels in the contour tracing, two definitions are given as follows,

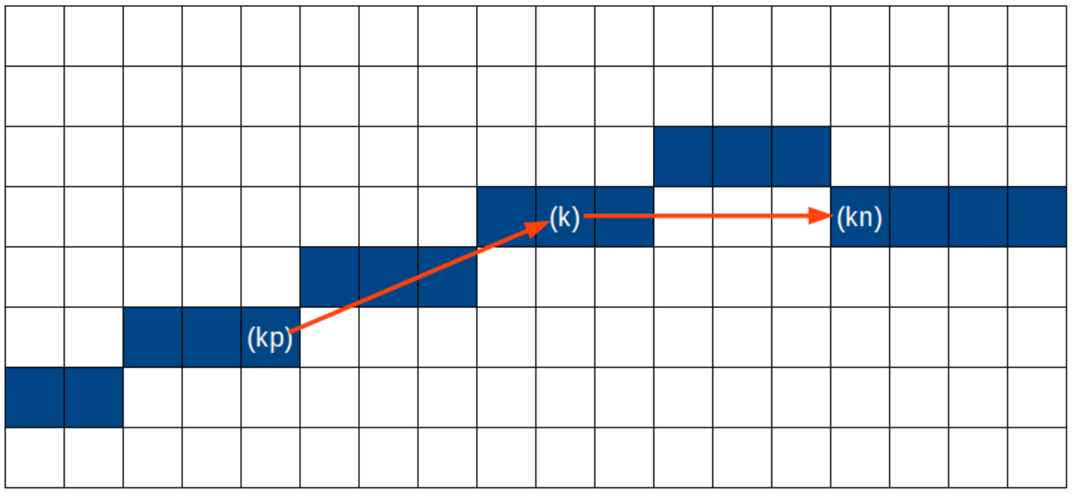


Fig.6 Illustration of pixel directions of pixel “k”, where pixel “kp” is its *n*-th predecessor and pixel “kn” is its *n*-th successor.

(1) Direction Stride (*DS*). Suppose the list is obtained with the previous contour tracing technique as described in section 2.1, if pixel “kp” is pixel “k”’s*n*-th predecessor, then this stride *n* is called direction stride when a directed (arrow) line is drawn from pixel “kp” to pixel “k”. In Fig.6 *DS* is 5 for both the line pointing from pixel “kp” to pixel “k” and the line pointing from pixel “k” to pixel “kn”.

(2) Pixel Directions. Assuming there are *N* pixels in the contour list, given a certain *DS*, the directions of a pixel consists of 2 parts:the front direction and the back direction. As shown in Fig.6, the arrowed line pointing from pixel “kp” to pixel “k” is the back direction and the arrowed line pointing from pixel “k” to pixel “kn” is the front direction of pixel “k”. The values of the 2 pixel directions are given as follows,

$$\theta_{kp} = arctan\left(\frac{y_k - y_{kp}}{x_k - x_{kp}}\right)$$
$$\theta_{kn} = arctan\left(\frac{y_{kn} - y_k}{x_{kn} - x_k}\right)$$

where for $k = 0, 1, \ldots, N\text{-}1$
$kp = k - DS,$ when $k \geq DS$ and $kp = k - DS + N$ when $k < DS$;
$kn = k + DS,$ when $k < N - DS$ and $kn = k + DS - N$ when $k \geq N - DS$.
It should be noted that, the above angle range is in (-90°, 90°), and we need to extend the range to (0°, 360°] to satisfy the coming analysis, as given below,
$\theta = \theta,$ when$dx > 0,\ dy > 0;$
$\theta = \theta + 360°,$ when$dx > 0,\ dy < 0;$
$\theta = \theta + 180°,$ when$dx < 0;$
$\theta = 90°,$ when$dx = 0,\ dy > 0;$
$\theta = 270°,$ when$dx = 0,\ dy < 0;$
where$dx = x_k - x_{kp}$, $dy = y_k - y_{kp}$ for $\theta = \theta_{kp}$
and$x = x_{kn} - x_k$, $dy = y_{kn} - y_k$ for $\theta = \theta_{kn}$.

3.2 Peeling on Direction Histogram

The histogram is a statistic graph of angles of all pixels in the contour list, including the front directions and the back directions for a given *DS*. The initial parameters *DS* and number of buckets of the histogram varies depending on specific application scenario and accuracy requirements.

For purpose of demonstration, an example of randomly scribbled hexagon with 554 pixels are given in Fig.7, and its direction histogram is given in Fig.8, which uses 36 buckets and the *n*-th bucket holds the number of angles in the range of (10 *n*, 10 *n*+10] degrees for *n* = 0, 1, …, 35.

It should be noted that for a given *DS* the angular values are discrete since the coordinate values are discrete. Polygons with seriously twisted or jagged border lines may bring lots of jagged variations in the direction histogram. In such a case, some filtering techniques such as MAS (Moving Average Smoothing) should be used to smooth the histogram and to reduce the effect from these random variations. Please note that the example demonstrated in Fig.8 is a histogram without filtering, and our tests showed that, for various simple polygons such as triangles, rectangles, hexagons, and so on, generally this filtering is not necessary if *DS* is properly selected, which will be further explained in the next subsection.

Since this direction histogram is a one dimensional data distribution, the peeling process can be performed as described in part I, subsection 2.3. The initial peeling interval is set to 5 which is 1/20 of the max value round to the floor integer, the process terminates when all the 6 border lines are identified. The horizontal line in Fig.8 gives the final result after 2 consecutive peelings, above which 6 separated peaks are detected and relevant pixels are extracted for further analysis.

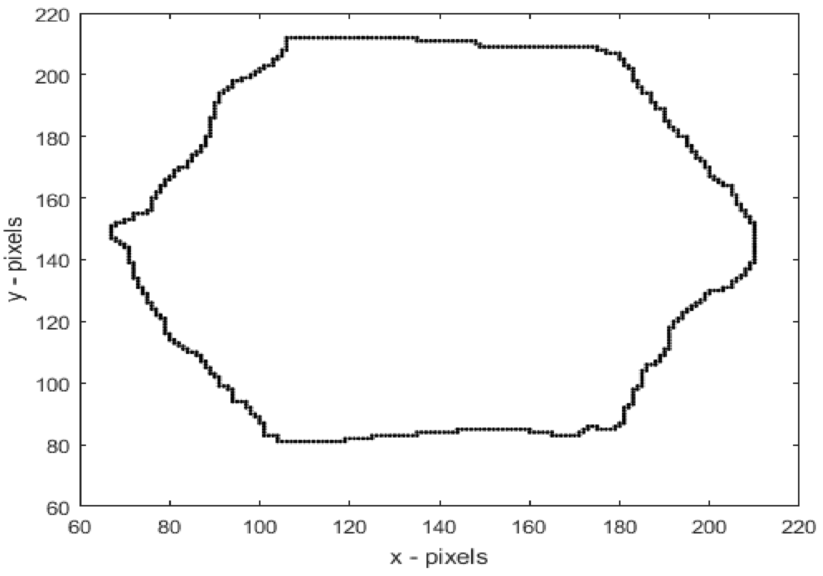


Fig.7 A randomly scribbled hexagon.

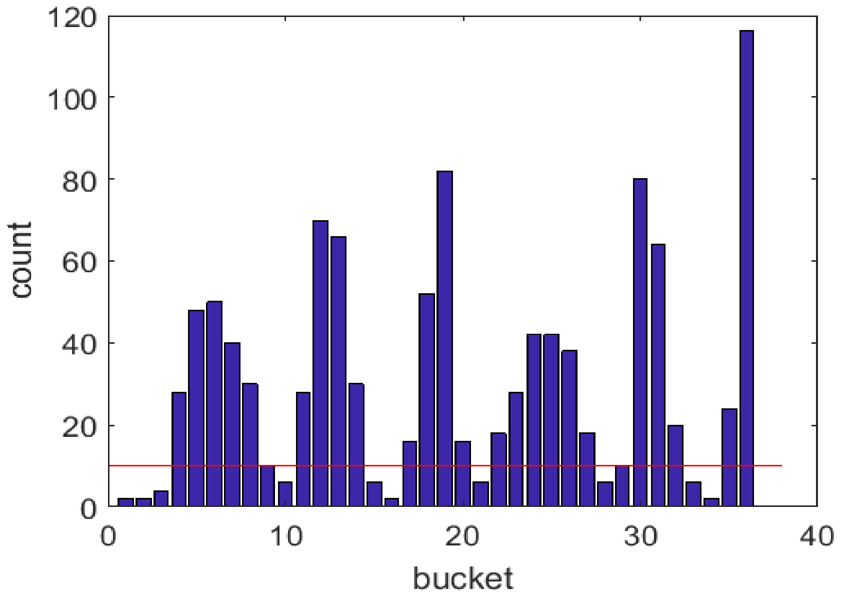


Fig.8 Direction histogram of all pixels in the contour list as shown in Fig.7.

3.3 Clean the Angle Clusters

After the peeling of the direction histogram, a check of all the clustered pixels is required to remove the noise pixels who have fallen into the wrong angular cluster.

An illustrative example of the cluster cleaning is given in Fig.9, where the direction from pixel “k” to pixel “a” obviously deviates from the main direction of the illustrated border line. This direction may fall into some other angular cluster in the direction histogram. A general cleaning method is to set a number threshold ‘epsilon’, if the number of consecutive pixels of a certain line section is less than epsilon, and their angular values fallen into a histogram cluster which is different from angular clusters of this section’s nearest preceding section and succeeding section, then pixels of this section are identified as noises and should be removed. As shown in Fig.9, if the angles of the section, which consists of pixels “a”, “b”, and “c”, are deviating from angle clusters of this section’s predecessor and successor sections, then these pixels are identified as noises.

It should be mentioned that *DS* has a direct influence on deviation and should be selected appropriately, for a regular polygon a *DS* value less than half of its shortest border line is recommended. Please bear in mind that we are expecting a high confidence with small tolerance. Assuming the histogram clusters are in normal distribution, generally larger *DS* will result in smaller deviation and an angle cluster will have a smaller tolerance interval[19] for the same confidence. The noise level of a cluster depends on the tolerance interval, as shown in Fig.9, assuming that for *DS*=5 the average angular value is 18°, if the tolerance interval of this cluster is set to the range of (18°-45°, 18°+45°), then pixels “a” and “b” will be detected as noise and pixel “c” will be a general effective pixel in the line.

Convex polygons, which are the common shapes of industry products, provide another convenience for noise detection. The sequence of the polygon lines are consecutively in correspondence to the sequence of the clusters in the direction histogram; thus, noise sections can be easily identified and removed subsequently once they are out of order of this consecutive sequence.

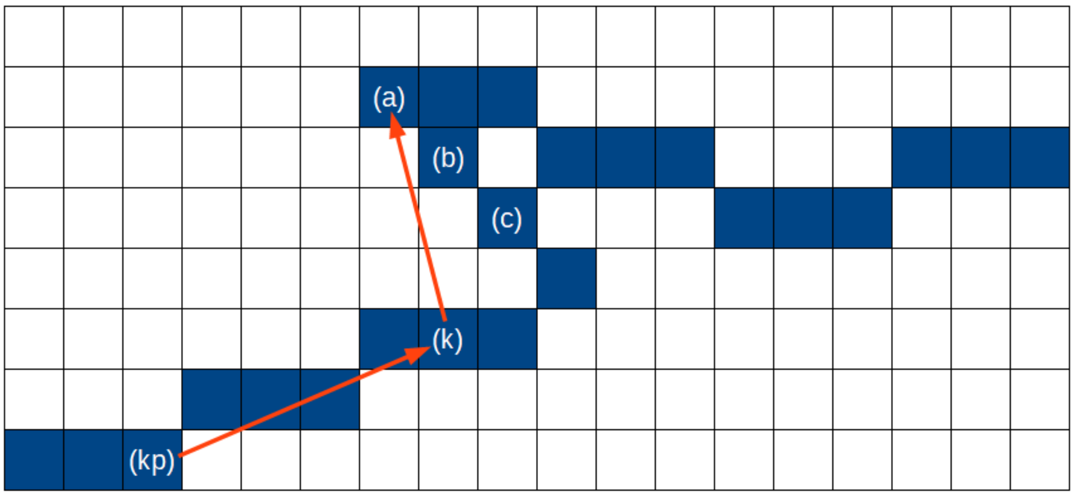


Fig.9 Illustration of noise pixels that may fall into the wrong cluster of the direction histogram

3.4 Polygon Detection

The polygon border lines are obtained once the pixels are correctly clustered and cleaned, the lines directions are the mean values of the clustered angular values. In general application for regular shapes of objects these average values should be accurate enough. For polygon lines with zigzag noises a higher accuracy can be achieved by fitting the clustered angular values to Gaussian models.

For systems with positioning requirements, the line position of the clustered pixels can be extracted with some simple linear regression [20].

The overall polygon detection process is summarized as follows,

```
Process of Polygon Detection with Direction Histogram

Step 1: set the initial parameters such as direction stride (DS), number of histogram buckets, etc.
Step 2: build the direction histogram;
Step 3: smooth the histogram with moving average filtering (optional);
Step 4: peeling the histogram to get the clustered directions;
Step 5: clean the clusters;
Step 6: extract direction of polygon border lines;
Step 7: extract position of polygon border lines with linear regression  (optional);
Done.
```

3.5 Result and Discussion

The high efficiency of this polygon detection algorithm comes from the idea that different border lines of certain polygons have different directions, which can be clustered at a very low computation cost. The main task of the algorithm is the establishment of direction histogram, which is a typical $O(n)$ process. In certain application scenarios, the efficiency can be further improved by building look-up tables for the angular calculation; thus, this method can be well adapted to real time applications in the embedded systems.

This polygon detection technique is tested on various polygons, further evaluation showed that this method is insensitive to initial parameters setting, e.g., in our demonstrative example, the peeling intervals are set to some rather different values ranging from 1 to 20, *DS* ranging from 10 to 60, and similar final results were obtained.

Another merit of this detection method is that it circumvents the problem of jagged noise and corner rounding, which is of special significance in positioning systems in automatic control application.

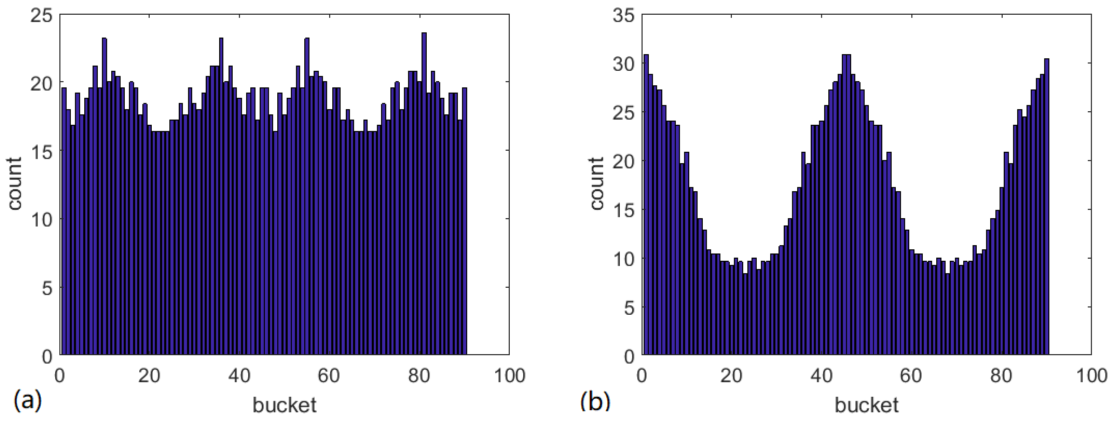


Fig.10 (a) direction histogram of a circle with radius of 103 pixels; (b) direction histogram of an oval with 2 radii of 120 and 76 pixels. The histogram (horizontal axis) has 90 buckets standing for angle range of (0°, 360°].

In addition, we found that this direction histogram method may also be applicable to detection of circles and ovals, as shown in the Fig.10. Fig.10(a) gives the histogram of a circle with radius of 103 pixels, where no obvious peak can be detected until the total peeling amount reaches 70% of its maximum value, Fig.10(b) gives the histogram of an oval with 2 axial radii at 120 and 76 pixels respectively, where 2 obvious peak are detected when the peeling amount reaches 30% of its maximum (please note that the left half peak at 0 degree is in connection with the right half peak at 360 degrees since the angles have a period of 360 degrees).

3.6 Limitations of Direction Histogram Method

When the polygon sizes become small, the corresponding direction strides (*DS*) has to decrease and the relative angular variations of the histogram will increase; that may make the algorithm inapplicable. In our example, the parameters will become difficult to tune when a hexagon border line has pixel numbers less than approximately 10. In such a case a contour re-scaling may be performed before the polygon detection.

A second limit of this method is that, it has almost no capability to extract lines from multiple interweaved polygons or from long zigzag curves since certain angular ranges may be fully occupied by lines directions with various deviations, that may make separation of correct peaks from each other difficult in the histogram of angular distribution.

References

[1]. Wikipedia - Edge Detection (https://en.wikipedia.org/wiki/Edge_detection)

[2]. Hamerly Greg, Elkan Charles, (2002),Alternatives to the k-means algorithm that find better clusterings, Proceedings of the eleventh international conference on Information and Knowledge Management (CIKM).

[3]. David Arthur,Sergei Vassilvitskii, (2007),K-means++: the advantages of careful seeding, Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics Philadelphia, PA, USA. P.1027-1035.

[4]. Dempster Arthur P., Laird Nan M., Rubin Donald B., (1977), Maximum Likelihood from Incomplete Data via the EM Algorithm,Journal of the Royal Statistical Society, Series B. 39 (1): 1-38.

[5]. MartinEster,Hans-Peter Kriegel, Jörg Sander, XiaoweiXu, (1996), A density-based algorithm for discovering clusters in large spatial databases with noise,Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. P.226-231.

[6]. Amandeep Kaur Mann, Navneet Kaur,(2013),Review Paper on Clustering Techniques, Global Journal of Computer Science and Technology, Volume 13 Issue 5 Version 1.0, P.43-47

[7]. Wei Wang, Jiong Yang, Richard Muntz,STING: A Statistical Information Grid Approach to Spatial Data Mining,Proceedings of 23rd International Conference on Very Large Data Bases (VLDB), August 25-29, 1997, Athens, Greece

[8]. Rakesh Agrawal, Johannes Gehrke, DimitriosGunopulos, PrabhakarRaghavan, Automatic subspace clustering of high dimensional data for data mining applications, Proceedings of the 1998 ACM SIGMOD international conference on Management of data, P.94-105.

[9]. GholamhoseinSheikholeslami,Surojit Chatterjee, Aidong Zhang, (1998),WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases, Proceedings of the 24th VLDB Conference New York, USA, P.428-439.
[10]. Image Analysis, Processing, and Machine Vision, 4thedition, by Milan Sonka, Vaclav Hlavac, Roger Boyle, Chapter 6.2.3 Border Tracing
[11]. Satoshi Suzuki, Keiichi Abe, Topological structural analysis of digitized binary images by border following, Computer Vision, Graphics, and Image Processing, Volume 30, Issue 1, April 1985, P.32-46
[12]. P.Rajashekar Reddy, V.Amarnadh, MekalaBhaskar, (2012),Evaluation of Stopping Criterion in Contour Tracing Algorithms, International Journal of Computer Science and Information Technologies, Vol. 3 (3) , P.3888-3894
[13]. Wikipedia - Hough Transform (https://en.wikipedia.org/wiki/Hough_transform)
[14]. Wikipedia - Circle Hough Transform (https://en.wikipedia.org/wiki/Circle_Hough_Transform)
[15]. David Shaw, Nick Barnes,(2004), Regular Polygon Detection as an Interest Point Operator for SLAM, Conference paper in Proceedings of the 2004 Australasian Conference on Robotics and Automation
[16]. Urs Ramer, (1972), An iterative procedure for the polygonal approximation of plane curves, Computer Graphics and Image Processing, 1(3), P.244–256
[17]. David Douglas, Thomas Peucker,(1973), Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, The Canadian Cartographer 10(2), P.112–122
[18]. Wikipedia – Ramer–Douglas–Peucker algorithm (https://en.wikipedia.org/wiki/Ramer–Douglas–Peucker_algorithm)
[19]. Applied Statistics and Probability for Engineers, 3rdEdition, Douglas C. Montgomery, George C. Runger, Section 8.7, P.270
[20]. Wikipedia – Simple Linear Regression (https://en.wikipedia.org/wiki/Simple_linear_regression)

Mengxi Tan."Polygon Detection Method by Direction Clustering Based on the Peeling Technique" IOSR Journal of Computer Engineering (IOSR-JCE) 21.1 (2019): 74-85.