Ellipse Detection Using a Randomized Hough Transform based on Edge Segment Merging Scheme

Kwangsoo Hahn, Youngjoon Han, Hernsoo Hahn
Dept. of Electrical Engineering, Soong Sil University
1 Sangdo-Dong, Dongjak-Ku, Seoul, 156-743, Korea
http://visionlab.sssu.ac.kr

Abstract: - This paper proposes a new ellipse detection scheme using a Randomized Hough Transform (RHT) modified to use line segments. It detects line segments in an edge image and selects every pair of them to test whether they are pertained to the same ellipse of not using the RHT. If they pass the test, they are merged. Since the proposed algorithm uses line segments, it reduces the computation time of the RHT significantly, and detects all ellipses included in an image without missing. The experimental results have shown that its performance is more prominent in detection of ellipses when they are overlapped and partially occluded.

Key-Words: - Ellipse detection, Segments Merging, Randomized Hough Transform, Occluded Ellipse.

1 Introduction

Elliptical shapes appear in most images dealt with in computer vision. Not only in most industrial parts such as volts, rings and tires etc., included they are but also in natural object such as a human body. To detect elliptical shapes included in a 2D image, Hough transformation as well as its variations has been used most popularly and they have shown reasonable performances.

Standard Hough Transform (SHT) [1] used the concept of Hough Transform that projects the edge pixels on a X-Y plane onto the parameter (a,b) plane to find straight line segments. To find an ellipse, it projects every edge pixel on a X-Y plane onto the 5D parameter space consisted with 5 parameters of a conic equation. Since each pixel requires 5 memory spaces, the total memory size becomes pretty large and computational complexity also increases significantly.

Probabilistic Hough Transform (PHT) [3] was provided as a solution for a large computational complexity problem. Instead of using all edge pixels, PHT selects a specific set of edge pixels using probabilistic models to project onto a parameter space. Since the number of pixels to be used in the determination of the ellipse parameters is reduced, the time complexity is reduced, but the memory size is large since each pixel is still requires 5 memory spaces.

Generalized Hough Transform (GHT) [4] is another variation for fast detection of object shapes in an image. If a model shape to detect in an image is determined, GHT constructs its R-Table first where individual pixels are represented as a set of distance and angle with reference to a selected reference point. Then the R-Table is used as the model to search the image. It reduced the space and time complexity by reducing the number of parameters required for each pixel, and included arbitrary shapes in its application domain. However, it has a disadvantage that it should provide separately all models of ellipse shapes to detect.

As a solution of these space and time problems involved in ellipse detection, Randomized Hough Transform (RHT) [5, 6] was proposed. It represents an ellipse using a second order polynomial equation which has three parameters so that they can be derived if the coordinates of three edge pixels are given. That is, it reduces the computational complexity by dividing a 5D space, required by the traditional Hough Transformation, by a 2D space for the epicenter of ellipse and a 3D space for the rest of the ellipse parameters. It also selects edge pixels in random, not by probabilistic model, to reduce the computational time. However, a random selection of three edge pixels causes a detection of false ellipses when objects are overlapped.

This paper proposes a new method of grouping edge pixels by line segments and merging them if they are in the same ellipse. Whether two line segments are in the same ellipse or not is tested by comparing their parameters of RHT. Because it can estimate the exact number of ellipses in the image through the number of the merged segment set, it reduces significantly the probability of false ellipse detection. Also, this method can reduce the total execution time because it

estimates the ellipse parameters in the line segment level not in the individual edge pixel level.
The rest of this paper is organized as follows: Section 2 describes the overview of the system and the preprocessing of an input image. Section 3 illustrates the line segment detection scheme and Section 4 explains how to use the line segments to detect ellipses using RHT. Section 5 shows the experimental results and Section 6 gives the conclusion.

2 System Overview and Preprocessing

2.1 System overview

The schematic diagram of the proposed algorithm is summarized in Fig. 1. The algorithm begins with the edge detection and thinning process in the first stage to find the boundaries of the objects in the image. The detected edge pixels are grouped by line segments using a corner pattern detector and individual line segments are labeled so that each of them may belong to only one ellipse later. In the second stage, every pair of edge segments is tested to see whether they satisfy the ellipse condition to merge. If so, they are merged. This test and merge process is repeated until all line segments are tested.

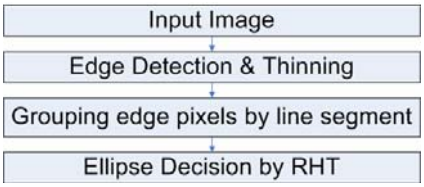
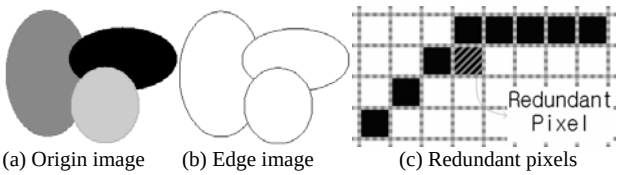


Fig. 1 Schematic diagram of the proposed algorithm

2.2 Preprocessing

The preprocessing includes the edge detection and thinning processes. For detecting edges in the input image, a modified Canny operator is used which removes noise using a Gaussian filter and estimates the local maximum edge point using the magnitude and orientation of the first differential of each pixel. Fig. 2 (b) is shows the result of applying the Canny edge detector for an input image given in Fig. 2 (a).



(a) Origin image (b) Edge image (c) Redundant pixels

Fig. 2 Canny edge detector

For grouping the edge pixels by line segments to use for RHT, a thinning operation is required. As shown in Fig. 2(c) which is the zoom-in image of a junction included in Fig. 2(b), Canny edge detector leaves redundant pixels around corners which hinders from correctly segmenting the boundary edge pixels. Those pixels shown in the corners making L shape as shown in Fig. 3 are considered as the redundant ones and they are eliminated. By removing these redundant pixels, the time required for finding line segments can be reduced.

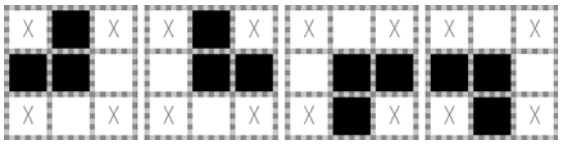


Fig. 3 Redundant pixels to be removed

3 Grouping Edge Pixels by Line Segments

To find ellipses using RHT, three edge pixels should be selected in the object image. As shown in Fig. 2, if there are ellipses more than one, the RHT has a high chance of selecting pixels from different ellipses to construct an ellipse, resulting in false ellipses with spending a large amount of processing time. To solve this problem, edge pixels are grouped by line segments first in this paper using the process given in Fig. 4 and the RHT uses these line segments instead of edge pixels to detect ellipses.

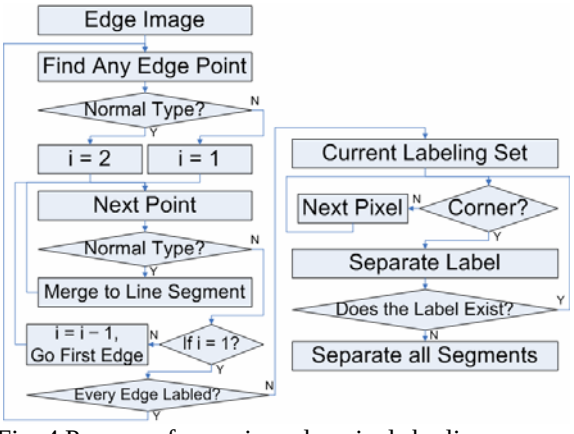
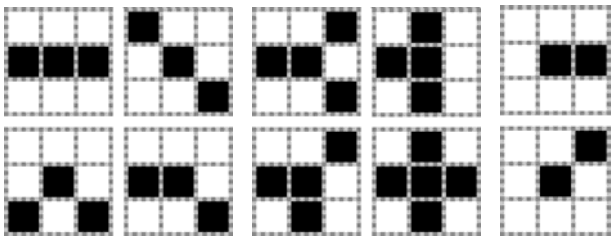


Fig. 4 Process of grouping edge pixels by line segments

To find line segments from the edge image, the process illustrated in Fig. 4 finds the crossing pixels in the first step. For this purpose, edge pixels are classified by four patterns, using the 8-connectivity window. The first one is the normal type having two

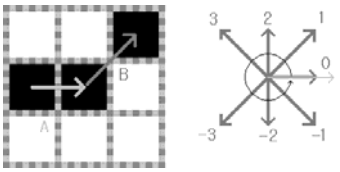
neighboring edge pixels as shown in Fig. 5(a), the second one is the crossing type having more than three neighboring edges as shown in Fig. 5(b), and the third one is the end type having one neighboring edge as shown in Fig. 5(c).



(a) Normal (b) Crossing (c) End

Fig. 5 Types of edge pixels

The fourth one is a corner type to be defined by a difference chain code on the way of grouping process, since it cannot be determined simply by a 3x3 window. The difference chain code expresses an angle difference between two vectors, A and B, each of which can take one of 8 directional vectors shown in Fig. 6(a). The angle difference between these two vectors can be represented by one of 7 difference codes ranging from -3 to 3 including 0 as shown in Fig. 6(b).



(a) Vector definition (b) Code

Fig. 6 Definition of a difference code

Fig. 7 shows an example of describing a sequence of edge pixels using the difference chain code. A part of the edge image is described by the difference chain code.

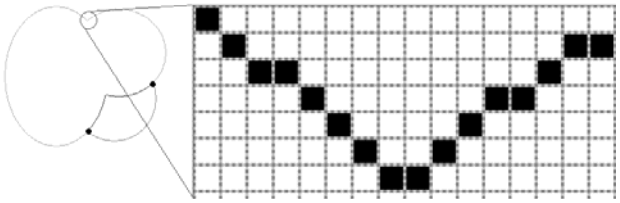
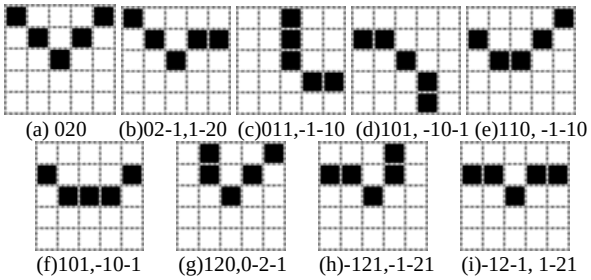


Fig. 7 Example of representing a curve using a difference chain code: 1-10001100-110-1

By empirically testing the images, the difference chain code patterns of corners appearing in a 5x5 window are summarized in Fig. 8.

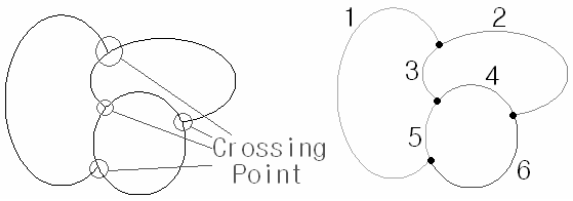


(a) 020 (b)02-1,1-20 (c)011,-1-10 (d)101, -10-1 (e)110, -1-10

(f)101,-10-1 (g)120,0-2-1 (h)-121,-1-21 (i)-12-1, 1-21

Fig. 8 Corner patterns in a 5x5 window

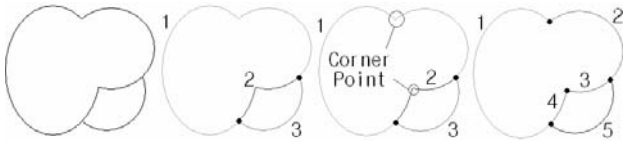
Based on the types of edge pixels, the grouping and labeling process begins with testing the leftmost top edge pixel in the input image to see which type it is. If it is a normal type, it is considered as a starting pixel of a new line segment and the edge pixel located first in the counterclockwise rotation with reference to X axis is selected to be tested until an edge pixel of crossing or end type is found, while merging the edge pixel as the same line segment if it is a normal type. Once a crossing or end type is found, the process returns to the first edge pixel of the line segment and begins the same procedure with the edge pixel located in the other direction. One line segment is completed when the tests along both neighboring edge pixels are finished. If the first edge pixel is a crossing type, it is the first pixel of three different line segments and its neighboring pixels are tested one by one repeatedly. Fig. 9(b) shows the line segment image obtained by applying the proposed algorithm to the edge image in Fig. 9(a) where there is no edge pixel of corner type.



(a) Crossing edges (b) Labeled line segments

Fig. 9 Edge grouping based on crossing edges

On the grouping process, if an edge pixel of corner type is found, then it stops the testing and merging operation. The edge pixel of corner type becomes the starting pixel of a new line segment from which the grouping process begins again. For example, let's consider an edge image given in Fig. 10(a) where edge pixels of corner type are included. In this case, if the pixels of corner type are not detected correctly, then the wrong line segments which include sudden curvature changes are resulted. In Fig. 10(b), line segments 1 and 2 include one corner pixel respectively.



(a) Edge image (b) Wrong result (c) Corner finding (d) Correct result

Fig. 12 Grouping process considering corner pixels

If representing the line segment 1 in Fig. 10(b) with the difference chain code, 1-10001100-110-1 is resulted where 110, one of corner patterns, is included. At the edge pixel where this pattern is found, the line segment is divided and the correct resulting line segments are obtained as given in Fig. 10(d).

4 Ellipse Decision

4.1 Randomized Hough Transform

Ellipses in a X-Y plane can be completely represented by the following quadratic equation with five parameters (*a*,*b*,*c*,*d*,*e*) as other 2nd order curves can be.

$$ax^2 + by^2 + cxy + dx + ey + 1 = 0 \tag{1}$$

Eq. (1) can be restructured into Eq. (2) to explicitly include the coordinates of the ellipse center (*p*,*q*). Eq. (3) is given as the ellipse condition of Eq. (2). Still five parameters (*A*,*B*,*H*,*p*,*q*) are required to represent an ellipse.

$$A(x-p)^2 + 2H(x-p)(y-q) + B(y-q)^2 = 1 \tag{2}$$

$$AB - H^2 > 0 \tag{3}$$

Since as the number of parameters increases the computational complexity in Hough Transformation also does rapidly, Yuen et al [2] proposed a method of representing an ellipse with 3 parameters to use for Randomized Hough Transformation. It takes two points (x_1,x_2) on an ellipse and finds their midpoint (m_1)and intersection(t_1) of their tangents, as shown in Fig. 11.

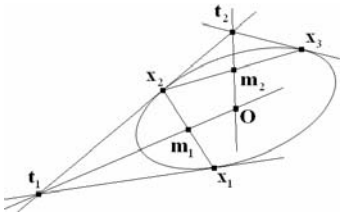


Fig. 11 Estimate center of ellipse

The line connecting m_1 and t_1 passes the center (O) of the ellipse. Therefore if three points (x_1,x_2,x_3) on an ellipse are given, its center can be determined as the intersection of two lines, $\overline{\mathbf{t_1m_1}}$ and $\overline{\mathbf{t_2m_2}}$ as shown in Fig. 11. Thus, the parameters (*p*,*q*) in Eq. (2) can be removed and Eq. (4) can be used to represent an ellipse where three parameters (*A*,*H*,*B*) are sufficient.

$$Ax^2 + 2Hxy + By^2 = 1 \tag{4}$$

These three parameters in Eq. (4) can be obtained in the parameter space if any three points on an ellipse are given in X-Y plane, using Eq. (5).

$$\begin{pmatrix} x_1^2 & 2x_1y_1 & y_1^2 \\ x_2^2 & 2x_2y_2 & y_2^2 \\ x_3^2 & 2x_3y_3 & y_3^2 \end{pmatrix} \begin{pmatrix} A \\ H \\ B \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{5}$$

4.2 RHT with line segments

To test if a pair of line segments consists of the same ellipse of not, the process given in Fig. 12 is used where RHT takes the main role in ellipse decision.

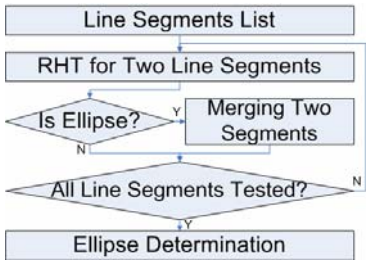


Fig. 12 Block diagram for ellipse decision

In the first step, the line segments are sorted by length in a decreasing order and their ellipse parameters of RHT are calculated using Eq. (5). If a line segment satisfies the ellipse condition given in Eq. (3), it stays in the list. Otherwise, it is considered not a part of an ellipse and omitted from the list. The merging operation considers the first, longest one in the list as the reference line segment. One of the rests in the list is selected and its ellipse matching ratio (EMR) defined in Eq. (6) is calculated.

$$EMR = \frac{N_c}{N_w} \tag{6}$$

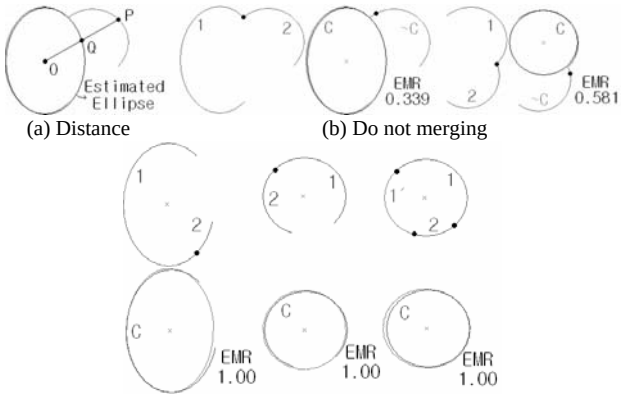
In Eq. (6), N_w is the total number of pixels included in two line segments and N_c is the number of those pixels of segments pair that can be included in the reference ellipse E_R which is derived by RHT from the reference line segment. Thus, N_c can be represented by Eq. (7).

$$N_c = \sum_{i=1}^{N_w} Near(i) \tag{7}$$

$$Near(i) = \begin{cases} 1 & \text{if } \overline{PQ} < th \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

In Eq. (8), if Q is the pixel to be tested and P is the pixel of the reference ellipse at the intersection with the line from the center of the ellipse and P, $\overline{PQ}$ is the distance between the reference ellipse and the pixel included in the selected line segment, as shown in Fig. 13(a). If the selected line segment has an EER with the reference ellipse within the threshold to be determined empirically, then it is merged to the reference line segment.

Fig. 13 shows the example images acquired as the results of the line segment merging two line segments. The first image in Fig. 13(b) includes two line segments and line segment 1 (longest one) is selected as the reference one to derive the reference ellipse C given in the second image. Then line segment 2 is selected and its EMR is calculated which is smaller than the threshold 0.95. Thus it cannot be merged with Line segment 1. Fig. 13(c) shows the cases where the line segments can be merged.



(a) Distance (b) Do not merging

(c) Merging

Fig. 13 Merging decision

5 Experiments

The performance of the proposed algorithm has been tested with the ellipse image sets given in Fig. 14 where ellipses with various parameters are overlapped arbitrarily. It is also compared with those of SHT [1] and RHT [6], in terms of accuracy of detecting ellipses and counting their number in the image. The execution time is also calculated to compare the computational time.

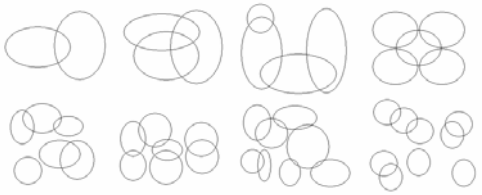
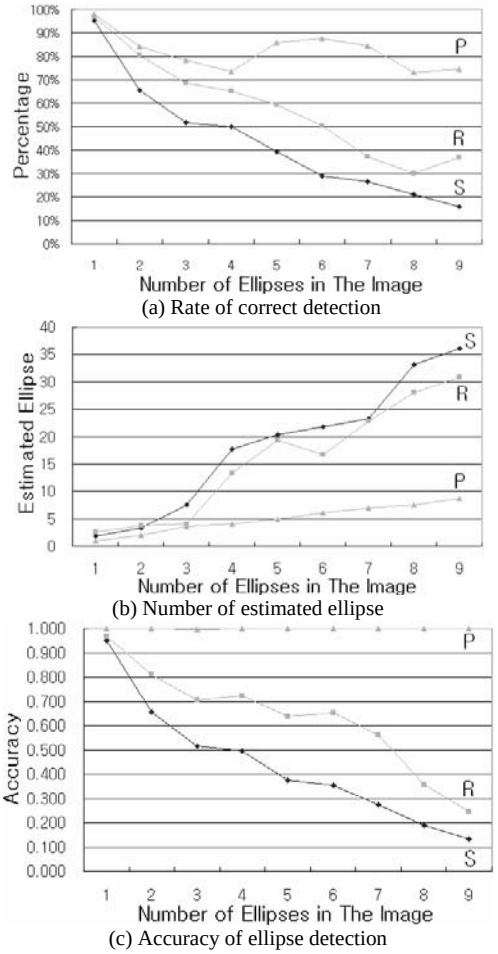


Fig. 14 Test images used in the experiments

Fig. 15(a) shows the rates of correct detection of ellipses, plotted with changing the number of ellipses in the image. The figure shows that the performance of the proposed algorithm gets better outstandingly compared to the two conventional ones, as the number of ellipses in image increases.



(a) Rate of correct detection

(b) Number of estimated ellipse

(c) Accuracy of ellipse detection

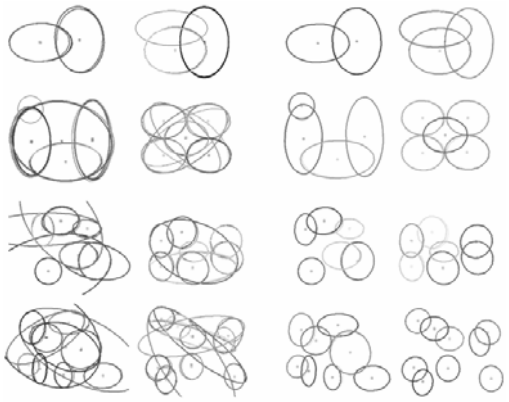
Fig. 15 Performance comparison (S: SHT, R: RHT, P: Proposed)

Fig. 15(b) displays the number of ellipses that is estimated in each image. From this figure, the accuracy (S) of the ellipse detection can be calculated as given in Eq. (10) and is illustrated in Fig. 15(c).

$$S=\frac{N_d}{N_e} \tag{9}$$

In Eq. (9), N_d and N_e denote the number of correctly detected ellipses and the number of total ellipses in an input image, respectively.

To show visually the performances given in Fig. 16, the ellipses actually detected by the RHT algorithm and the proposed algorithm as depicted in Fig. 16. It shows that RHT algorithm detects ellipses with the pixels located in different ellipses, resulting in detecting more ellipses than actual ones.



(a) RHT algorithm (b) Proposed algorithm

Fig. 16 Actually detected ellipses

In the experiments, the computation times of three algorithms are also measured. Since the order of the spent time for SHT is out of the range, those of other two methods are depicted in Fig, 17.

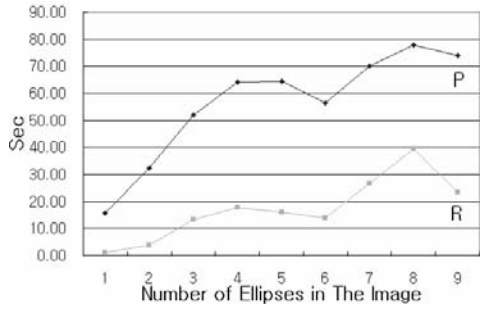


Fig. 17 Processing time

Fig. 17 shows that the proposed algorithm is faster than the RHT of using edge pixels, as expected, since the number of elements to be tested is reduced significantly by grouping the edge pixels by line segments. The execution time is measured while executing the algorithm implemented with Visual C^{++} in Pentium PC.

The experiments have shown outstanding results in terms of both detection accuracy and execution time. But it is also conformed in the experiments as the problems that should be solved that its performance depends on the accuracy of edge detection and it needs to reserve all the corner patterns.

6 Conclusion

This paper has proposed a new ellipse detection algorithm using the RHT based on line segments. The algorithm intended to solve the problems of RHT which has a tendency of detecting false ellipses with edge pixels pertained to different ellipses and whose computational complexity depends on the number of edge pixels. These problems have been solved in this paper by reducing the number of inputs to RHT by grouping the edge pixels by line segments. The experimental results have shown that the proposed algorithm is superior to two conventional algorithms, SHT and RHT, by at least two times on average in terms of accuracy and processing time, even when ellipses are overlapped in an input image.

References:

[1] P.V.C. Hough, "Method and Means for Recognizing Complex Patterns," U.S. Patent 3069654, Dec. 18 1962.

[2] H. K. Yuen, J. Illingworth, and Kittler, “Detecting partially occluded ellipses using the Hough Transform,” Image and Vision Computing, vol. 7, no. 1, pp. 31-37, Feb. 1989.

[3] N. Kiryati, Y Eldar, and A. M. Bruckstein, “A probabilistic Hough Transform,” Pattern Recognition, vol. 24, no. 4, pp. 303-316, 1991.

[4] Sheng-Ching Jeng, Wen-Hsuang Tsai, "Scale and orientation-invariant generalized Hough Transform-A new approach," Pattern Recognition, vol. 24, no. 11, pp. 1034-1051, 1991.

[5] Lei Xu, Erkki Oja, and Pekka kultanena, “A new curve detection method: Randomized Hough Transform (RHT),” Pattern Recognition Letters, vol. 11, no. 5, pp. 331-338, May. 1990.

[6] Robert A. McLaughlin, “Randomized Hough Transform: better ellipse detection,” Digital Signal Processing Applications, vol. 1, pp. 409-414, Nov. 1996.