2011 Sixth International Conference on Image and Graphics

A Rectangle Detection Method for Real-Time Extraction of Large Panel Edge

Zhe Wu, Qingjie Kong ,Jiapeng Liu , Yuncai Liu
Department of Automation and Key Laboratory of China MOE for System Control and Information
Shanghai Jiaotong University
Shanghai, China
zjuwuzz@gmail.com

***Abstract*—The problem of real-time rectangle detection on high-resolution image arises in actual panel production. This paper proposes a robust real-time method for panel rectangle detection. In line extraction part, a new technique called "Crossing Dilation"is proposed to improve the Progressive Probabilistic Hough Transform performance, to extract long line segments that are not perfectly straight in large sparse binary image. The rectangle is detected using the proposed rectangle fitting method. Experiments on a variety of real cases of large panels are executed, some of which are badly disturbed by the fake edges caused by the dust in the background region. The results show great real-time performance on all the cases, and the proposed method is robust to the disturbances and different panel sizes.**

***Keywords*-rectangle; detection; Hough; panel;**

I. INTRODUCTION

The problem of real-time rectangle detection on high-resolution image arises in actual panel production. Thousands of edge points are given in form of 2D coordinates, whose magnitude is extremely huge: from hundreds to tens of thousands. These edge points are provided by the panel manufacturers through basic image processing techniques. The use of the high-resolution image is due to the type of the camera adopted by the manufacturers. The purpose of the rectangle detection in this situation is to extract the right rectangle from the large amount of edge points and should not be influenced by the interferences. The size of panel is not a priori knowledge, but the minimum width and height is known. The main disturbance comes from the fake edge introduced by dust in the background region, which is illustrated in Fig.1. Besides, to enable the algorithm work in real time, good efficiency and accuracy are demanded. In this paper, we will present a rectangle detection method that is very effective in such situations.

Rectangle detection algorithm is needed in many fields, such as buildings or vehicles detection in aerial images, and recognizing license plates for cars in video sequences[1][2]. However, there are many differences between the situations mentioned above and the problem of rectangle detection for large panel in this paper. These situations are probably the multi-rectangle situations with the normal image size. While in the problem of panel rectangle detection, there is usually no more than one complete rectangle in each case. The data size is huge in point number and magnitude of point coordinates. Moreover, the rectangle detection algorithm should be robust to the panel size differences and dust influences.

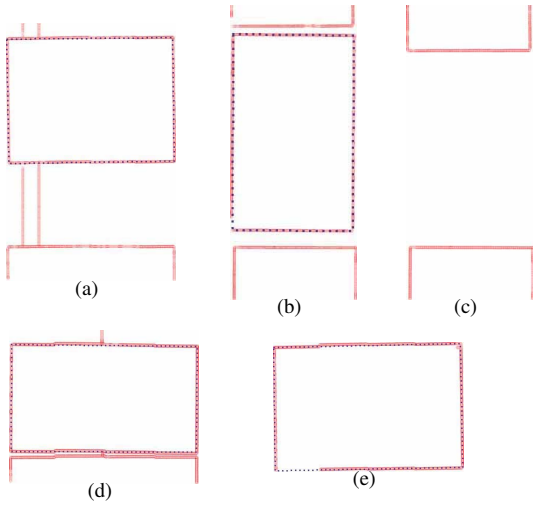


Figure 1. Some panel thumbnail images, formed by thousands of edge points each. The right rectangle is denoted with dark dotted line. In(a),(b),(d), fake edge disturbances are shown; in(c), there is no completed rectangle.

Jung and Schramm[5] proposed a rectangle detection method based on edge and line primitives. Given the edge binary image, they first extracted line segments, and then combined these straight lines to form rectangles. In the part of line segments extraction, Hough Transform[8] is a very efficient tool, even in the presence of noise and missing data. However, highly computational cost is the drawback of this method.Some other recent works on rectangle detection lies in various regions with specific usage. D. Shaw[3] provided a perspective rectangle detection mainly uses for Simultaneous Localisation and Mapping (SLAM). J. Bazin[4] brought out a rectangle extraction method in catadioptric images that is adopt in robotic systems equipped with catadioptric cameras. All of these works don't have s significant relationship with our work on the rectangle detection in large panel manufacture, due to the differences in the usage.

In this paper, we propose a fast rectangle detection algorithm effective for the high-resolution image. The proposed

978-0-7695-4541-7/11 $26.00 © 2011 IEEE
DOI 10.1109/ICIG.2011.83

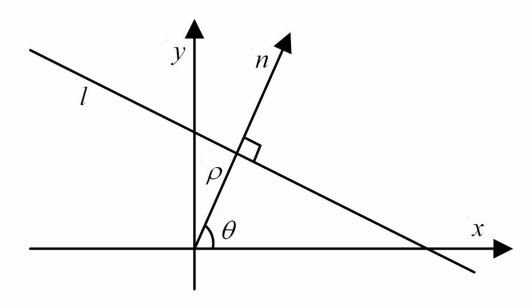
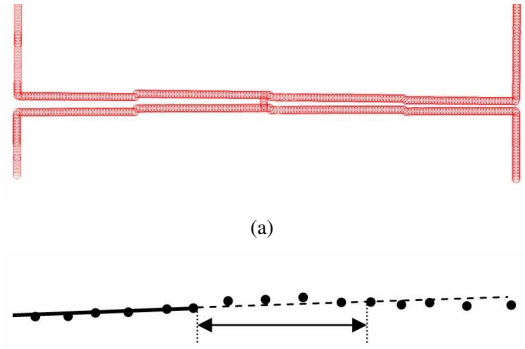


Figure 2. The line l is represented by ρ and θ.



(a)

(b)

Figure 3. (a) The line is usually not perfectly straight because of the distortion; (b) the reason why failed to extract the whole line.

method contains two parts: line segments extraction and the rectangle fitting using the extracted lines. The line extraction algorithm is an extension of the Progressive Probabilistic Hough Transform (PPHT)[6]. It uses a "crossing point dilation"technique to extract the whole line, which could save lots of computational cost and makes the Hough Transform work more efficiently. The line extraction algorithm is applied directly to the binary edge map. After line extraction, a fast rectangle fitting method is proposed to form a rectangle using the extracted line segments. The rectangle fitting method is designed to work on all the cases, which is also a fast detection method.

The rest of this paper is organized as follows. Section 2 describes the Progressive Probabilistic Hough Transform (PPHT) algorithm and the results when applying it to our problem. Then our line extraction algorithm IPPHT (Improved Progressive Probabilistic Hough Transform) and fast rectangle fitting method are elaborated in Section 3 and Section 4 respectively. Experiments and results are given in Section 5.

II. PPHT ALGORITHM

The Hough Transform (HT) is a popular method for extracting lines from an image. This algorithm is essentially a voting process, where each point belonging to the patterns votes for all the possible patterns passing through that point. The votes are accumulated in a Hough space array, the pattern receiving the maximum votes is detected line. Duda and Hart[7] found that any 2 Dimension line can be described as (1).

$$\rho = x \cdot cos\theta + y \cdot sin\theta \tag{1}$$

where ρ is the normal distance to the coordinate origin, θ is the normal angle of the line, (x,y) is the coordinates of the pixel, as Fig.2 shows.

Galambos et al.[6] proposed a Progressive Probabilistic Hough Transform (PPHT). PPHT exploits the difference in the fraction of votes, which are needed to detect reliable lines with different numbers of supporting points, and minimizes the amount of computation needed to detect lines in this way. The following are the PPHT algorithm.

1. Collect non-zero points
2. Choose random point out of the remaining ones.
3. Remove the point from the sequence.
4. Accumulate the parameter in the accumulator space until the max value is larger than the threshold. Then we found the most probable line. If not, go to 2.
5. From the current point, walk in each direction along the line, stop at the image border in case of too big gap. Then get line side points in both directions.
6. Remove the pixels in the segment from the sequence.
7. Unvote from the accumulator all the pixels from the line that have previously voted.
8. If the line segment is longer than the minimum length, add it into the output list.
9. Go to 2.

When applied to the panel rectangle detection problem, PPHT failed to extract the whole lines even after the data was condensed to the resolution of about 2000*2000. The results are very bad, many obvious lines that are easy to be recognized by human could not be detected, as shown in Fig.8.

The main reason why PPHT could not detect lots of long lines is that the binary image is a large sparse matrix. Besides, we could find out that many lines are not perfectly straight due to the camera distortionas Fig.3(a) shows. Therefore, in step.5 of the PPHT, when walking along the line to find the side points, the algorithm is usually failed to find the next point along the line, as Fig.3(b) shows. The results would be better if we continue to condense the data but the extracted rectangle would be less accurate.

III. PROPOSED ALGORITHM OF LINE EXTRACTION

The basic idea of the proposed algorithm is to dilate each pixel (x,y) to a cross pattern using the technique that

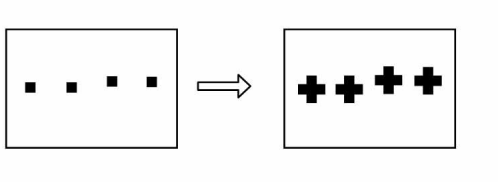


Figure 4. The "Cross Dilation" technique with n_{around}

.

we called "Crossing Dilation". Use all pixels including the dilation ones to vote. Then in the unvoting part, unvote from the accumulator all cross patterns from the line that have previously voted. These steps are described in details next.

To solve the problem described in Section 2, we need to get the whole line side points. Noticed that if the sphere of the influence of each pixel is represented with a region centered at the pixel, rather than the pixel itself, then we could reach the region influenced by next pixel, then the whole line could be extract. To denote the sphere of the influence, we dilate each pixel *(x,y)* to a cross pattern by setting n_{around} neighbor pixel in the four direction of the pixel 1, as shown in Fig.4. Bigger influence means a larger n_{around}. 12

After using this C.D technique, all the effective line segments could be extracted. However, the result line segments include lots of similar segments, most of which belonged to the same completed line. Such situation happens because after we extract an effective line, we only remove the pixels in the segment, this will leave lots of pixels in the same cross. And these pixels would vote for other same lines, as shown in Fig.5. They should be removed also since their function is only to help get the line side points. To remove all the pixels of the cross, we build a two-way relationship between the cross and its pixels. Then, given a pixel $p_i(x, y)$, all pixels $P = \{(x_1, y_1) \dots (x_n, y_n)\}$ that share the same cross could be calculated with this mapping.

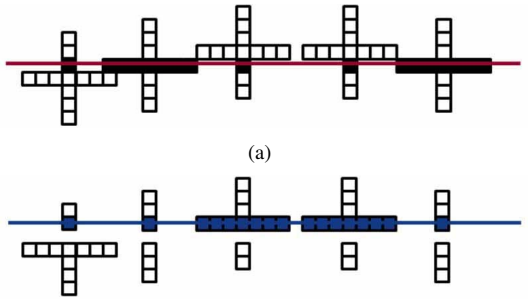
$$P = \{(x_1, y_1)...(x_n, y_n)\} = F(p_i) \quad , n \leq 4 \times n_{around} + 1 \tag{2}$$

where *F()* is the a two-way relationship between the cross to its pixels. Thus, if we know a pixel in the segment, we could know all the pixels share the same cross with it, and when extracted the line, we could remove the whole cross. The following are the IPPHT algorithm.

1. Collect non-zero points

2. Using the "Cross Dilation"technique to dilate the pixels and build the two-way relationship between the cross to its pixels

3. Choose random point out of the remaining ones. If the number of remaining 1 pixels is less than a set ratio of the total number, algorithm over.



(a)

(b)

Figure 5. (a) Remove the pixels not the "+" masks; (b) After erasing part, similar lines are still being extracted.

4. Remove the point from the sequence.

5. Accumulate the parameter in the accumulator space until the max value is larger than . If not, go to 3.

6. From the midpoint of the cross, walk in each direction along the line, stop at the image border in case of too big gap, then get line side points in both directions.

7. Compare the length of the line with the lines having adjacent angle to find the longest line, as the best line.

8. Remove the crosses which include pixels in the segment from the sequence .

9. Unvote from the accumulator all the pixels from the line that have previously voted.

10. If the line segment is longer than , add it into the output list.

11. Go to 3.

IV. Rectangle fitting algorithm

By using the proposed IPPHT algorithm, we could detect all the line segments that are longer than a minimum length threshold. The proposed rectangle fitting algorithm is to use the extract lines to fit the right rectangle.

In [5], Jung et al. proposed a rectangle detection method in Hough space, which can't adapt to our problem because of the fake edge disturbances, as shown in Fig.1. By analyzing the cases, a contradictory situation was found, which is shown in Fig.6. We could see that in Fig.6 (a), two lines are very close; while in Fig.6(b), one line has been divided into two segments. This means in this case, we can't extract the ideal line segments no matter how the parameter $MaxGap$ was chosen.

To make our rectangle detection algorithm robust, we must find the characteristics of the panel rectangle pattern. Noticed that in every case given, if we can detect one precise parallel pair P with a perfectly matched orthogonal line L_1, and a less perfectly matched orthogonal line L_2, we can fit the right rectangle, which is shown in Fig.7. Here a precise parallel pair means two parallel lines with same length, and their connecting sides are perpendicular to each other; a perfectly matched orthogonal line means a

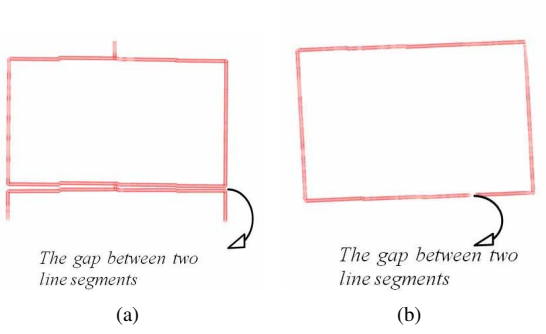


Figure 6. Two contradictory cases. In (a), two panels are very close, the gap between two edges lines is very small; in (b), one line is divided into two separate.

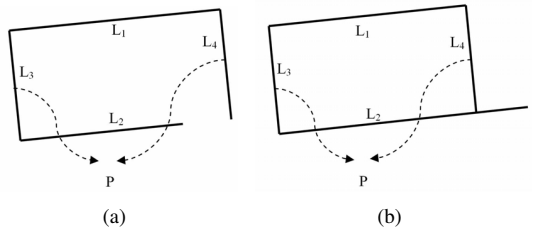


Figure 7. The key pattern to fit the rectangle, P is the perfectly matched parallel pair, formed by L_3 and L_4; L_1 is a precise orthogonal line and L_2 is a less perfectly matched orthogonal line.

line that is perpendicular to the parallel pair, and the side points are very closely to the side points of the parallel lines; a less perfectly matched orthogonal line means a line with one side point closely to the side point of the parallel pair, no limit on the other side point. The following are the rectangle fitting algorithm.

1. Calculate the line parameters, including the length and the angle.

The angle should be processed to be in the range of 0 180. The length is calculated using Euclidean distance.

2. Find a precise parallel pair from the lines.

Firstly we should find two lines with same length and same angle. Then ensure the line formed by two lines mid points is perpendicular with the parallel lines. There is a need to exclude the pair between which the distance is not big enough. If no effective parallel pair is found, go back to find the pair, otherwise, continue to next step. If all lines have been detected, end with "No Rectangle. "result.

3 and 4. Try to find a orthogonal pair.

Firstly find a perpendicular line segment, secondly check if it is a perfectly matched line or a less perfectly matched line. Then continue this step until we find two perfectly matched lines or one perfectly matched line and one less perfectly matched line. The rectangle fit successfully. Otherwise go back to 2.

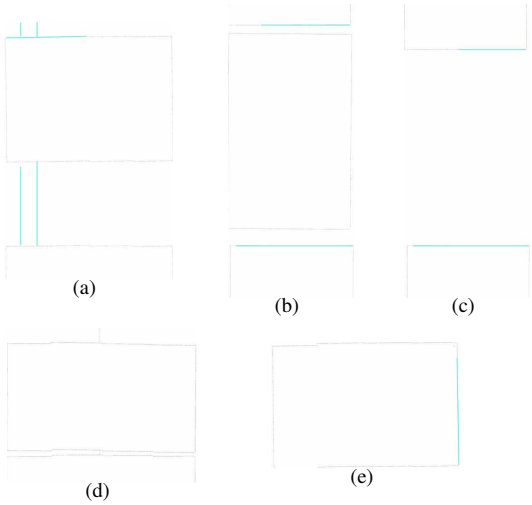


Figure 8. The line extraction results by using PPHT line extraction algorithm. We could see many good lines are failed to be detected, no lines are detected in case (d).

V. Experiments and Results

In this section, we first give the results of the PPHT algorithm to see its performance. Then results of the proposed line extraction algorithm IPPHT will be shown. Finally, we will show the results of rectangle fitting algorithm.

We evaluate our approach on the cases provided by a panel manufacturer. There are several sizes of panels in all cases. Each case contains about 2000 edge points, which are not continuous. The coordinates of the pixel are from hundreds to tens of thousands. The data size is so large that the later digits of the pixels are not significant figures. Therefore, we first condense the data, which will not affect the accuracy of the results. Some of the cases are badly disturbed by fake edges which are caused by dust in the background region, as shown in Fig.1. There are more than one hundred cases in total, the disturbed cases only account for a small percentage (about 10%) of them.

A. Results of PPHT line extraction algorithm

By using the PPHT method, the best results we could get is shown in Fig.8. As analyzed previously in Section.2, no matter how the parameters are chosen, the problem could not be solved, due to the high resolution images and the distortion problem of the data.

B. Results of IPPHT line extraction algorithm

The experiments use the proposed line extraction algorithm IPPHT to extract line sections on the cases shown in Fig.1. $MinLength$ is set to 280 pixel, $MaxGap$ is set to 25 pixel,$n_{around} = 7$,the Hough peak threshold $K_{Threshold} = 80$.

In this experiment, our goal is to extract all the line segments whose length is larger than $MinLength$. Besides,

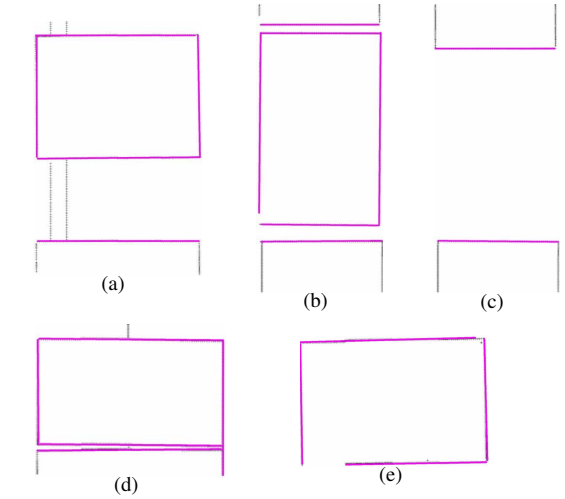


Figure 9. The line extraction results using IPPHT algorithm. We could see all the line segments whose length is larger than the threshold $MinLength$ are extracted successfully.

the line segments should be the extracted completely. When we extract a line segment, remove the crosses which include pixels in the line segment from the sequence.

The results in Fig.9 show that our proposed IPPHT line extraction algorithm achieves great performance. All the lines whose length is larger than the set $MinLength$ are extracted completely. In Fig.9 (d), the right edges belonged to two different rectangles are extracted as one line, which is because the gap is too small, as discussed in Section 4. But this won't affect the rectangle fitting results, which will be discussed later.

The process of IPPHT is shown in Fig.10, we can see after each line is extracted, most pixels that share the same cross have been removed, but some are still left, this is because mid-point of the cross coincides with points of other cross therefore has been removed before. The proposed IPPHT has improved the efficiency of PPHT greatly.

C. Results of rectangle fitting algorithm

Fig.11 shows the results of the panel rectangle fitting algorithm, all the cases (180 cases) with different sizes of panel have been tested, and the correct detection rate is 100%. The results show that our method is robust to fake edges and different panel rectangle sizes. The computational cost of five cases is shown in Table.1 (CPU 2.33GHz). The result is very representative because the common cases without so many disturbances are much faster (about 10ms).

In Table.1, the second column shows the number of edge pixels. The third and fourth column show the time (in milliseconds) involved in the execution of IPPHT and overall rectangle fitting algorithm, respectively. These times were computed by averaging the results of 50 executions of the five cases.

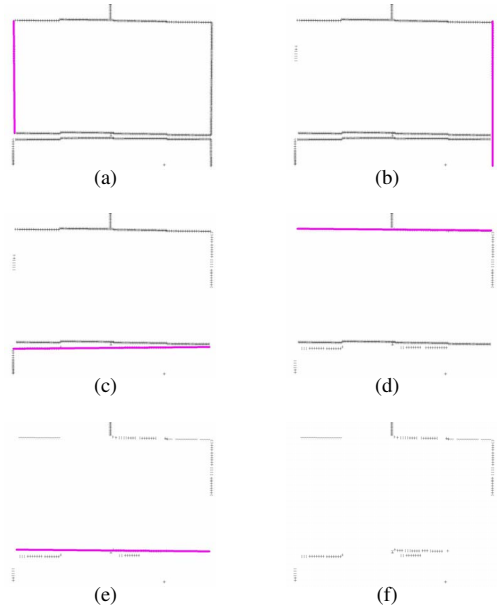


Figure 10. The process of line extraction results using IPPHT algorithm.

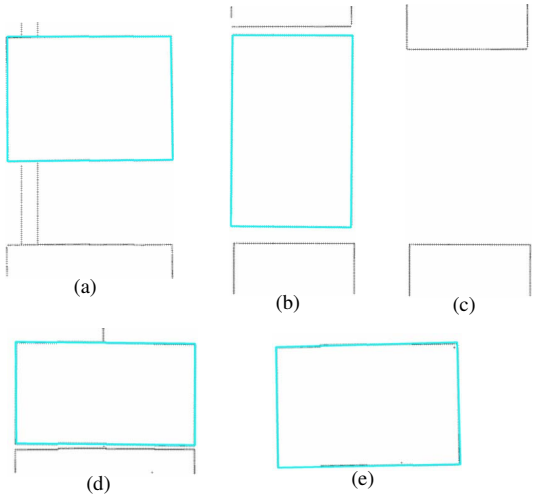


Figure 11. The rectangle fitting results using our proposed panel rectangle detection algorithm. We could see all the line segments whose length is larger than the threshold $MinLength$ are extracted successfully.

As shown in Table.1, the time for IPPHT occupies most of the total time and correlates with the pixels number. Fig.11 (c) is an exception, which could be explained that image size is large. The overall time shows that our proposed method is capable of real-time panel rectangle extraction.

VI. Conclusion

In this paper, we presented a robust real-time method for panel rectangle detection. We proposed an IPPHT line extraction method using the "Crossing Dilation"technique to extract long line segments that are not perfectly straight

Table I
COMPUTATIONAL COST

Image	Edge pixels count	IPPHT (ms)	Overall algorithm (ms)
a	1840	20.00	20.94
b	2123	24.50	24.95
c	916	17.85	18.36
d	1576	12.78	13.30
e	1142	10.56	11.07

in large sparse binary image. The efficiency of Hough transform was improved. Using the extracted lines, a fast rectangle fitting method was presented to detect the panel rectangle. The proposed method was extensively tested on a variety of real cases of large panels, some of which are badly disturbed by the fake edges caused by the dust in the background region. The results show great real-time performance on all the cases, and the proposed algorithm is robust to the disturbances and different panel sizes. In the future, we plan to improve the IPPHT algorithm and increase the accuracy in rectangle detection. We also intend to extend this work to other applications.

ACKNOWLEDGMENT

This work was supported in part by the National Science Key Foundation of China under Grant 60833009 and in part by the National Science Foundation of China under Grant 60975012.

REFERENCES

[1] S. Noronha and R. Nevatia, “Detection and Modeling of Buildings from Multiple Aerial Images,”IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, May. 2001, pp. 501-518, doi: 10.1109/34.922708.

[2] D. S.Gao and J. Zhou, “Car license plates detection from complex scene,”International Conference on Signal Processing Proceedings, vol. 2, Aug. 2000, pp. 1409-1414, doi: 10.1109/ICOSP.2000.891808.

[3] D. Shaw and N. Barnes, “Perspective Rectangle Detection,”Proc. Workshop of the Application of Computer Vision, in conjunction with ECCV 2006, Apr. 2008, pp. 119-127, doi: 10.1.1.94.7972.

[4] J. C. Bazin, Inso Kweon and C. Demonceaux, “Rectangle Extraction in Catadioptric Images,”IEEE 11th International Conference on Computer Vision(ICCV2007), Oct. 2007, pp. 1-7, doi: 10.1109/ICCV.2007.4409208.

[5] C. R. Jung and R. Schramm, “Rectangle detection based on a windowed Hough transform,”Proc. Computer Graphics and Image Processing, Oct. 2004, pp. 113-120, doi: 10.1109/SIBGRA.2004.1352951.

[6] C. Galambos, J. Matas and J. Kittler, “Progressive Probabilistic Hough Transform for line detection,”IEEE Conf. Computer Vision and Pattern Recognition(CVPR1999), Jun. 1999, pp. 303-316, doi: 10.1109/CVPR.1999.786993.

[7] R. E. Duda and P. E. Hart, “Use of the Hough transformation to detect lines and curves in pictures,”Communications of the ACM, vol. 15, Jan. 1972, pp. 11-15, doi: 10.1145/361237.361242.

[8] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Beijing: Publishing House of Electronics Industry, 2007.