A Graph-based Approach for Rectangle Detection Using Corners

Shengkai Wu, Lijun Gou, Hui Xiong, Xiaoping Li
School of Mechanical Science and Engineering
Huazhong University of Science and Technology
Wuhan, China
lixiaoping@hust.edu.cn

ABSTRACT

In this paper, a new rectangle detection approach is proposed. It is a bottom-up approach that contains four stages: line segment extraction, corner detection, corner-relation-graph generation and rectangle detection. Graph structure is used to construct the relations between corners and simplify the problem of rectangle detection. In addition, the approach can be extended to detect any polygons. Experiments on bin detection, traffic sign detection and license plate detection prove that the approach is robust.

CCS Concepts

Computing methodologies → Computer graphics → Image manipulation → Image processing

Keywords

Rectangle detection; Polygon detection; Shape recognition; Bottom-up approach.

1. INTRODUCTION

Rectangle detection is an important task in computer vision and has a wide range of application, such as bin detection, traffic sign detection, license plate detection.

Most of the rectangle detection approaches make use of line segment. Dmitry [1] extracts linear primitives first and then detects line segments by cluster analysis. Pairs of parallel line segments are then detected and combined for rectangle detection. Kim [2] defines four kinds of line segment relations and constructs the line-relation-graph. Then the problem of rectangle detection is transformed into the problem of finding any closed loops in the graph. However, the criteria of searching closed loops in the graph easily leads to the missing of rectangles when the sides of different rectangles form a long line segment. This phenomenon is common when rectangular objects are adjacent to each other. Tao [3] defines three primitive-structures using line segments and combines the primitive-structures to detect rectangle. But the definition of primitive-structures decides that the approach also can't deal with the problem mentioned above. Liu [4] extracts line segments first and constructs an MRF model on line segments to label lines belonging to rectangles. Then rectangles are detected by combining the labeled lines.

There are also rectangle detection approaches that don't use line segment feature. Li [5] proposes a different framework for rectangle detection. This approach detects edges first and treats all the connected components as candidate shapes. Then rectangular shapes are detected by removing outliers of candidate shapes, detecting rectangle's vertices and combining complementary open shapes. Obviously, the approach can't deal with the problem that one component is formed by more than one incomplete rectangles which restricts the application of the approach. Huertas [6] detects corner chains by tracking the edges first and then combines the corner chains to detect rectangles. Jung [7] proposes a windowed Hough Transform to detect rectangles. This approach defines a ring window first and then slides the predefined window to compute Hough Transform at every pixel location. Finally, a rectangle is detected if the peaks satisfy the rectangular geometric constraints. However, the size of the ring window has to be predefined approximately based on the size of rectangles in the image, which, however, is usually not known in the application.

In this paper, a bottom-up hierarchical approach for rectangle detection is proposed based on the hierarchical features (line segment, corner, rectangle) of rectangles. We extract line segments first and then detect the corners by computing the intersections between line segments. A corner-relation-graph is then constructed based on the predefined corner relations. Finally, rectangles are detected by searching the corner-relation-graph. Compared with the similar approach in [2], our approach takes advantage of corners which are the prominent feature of rectangle and only one corner relation is defined to construct the corner-relation-graph. As a result, the graph is simpler, which makes the rectangle detection simpler and robust. Compared with approach in [6], we both utilize the corner feature, but the methods of detecting corners and rectangles are different.

The rest of this paper is organized as follows: Section 2 introduces the method of line segment extraction. Section 3 explains the definition and detection of corners. Section 4 illustrates the definition of corner relation and the construction of the corner-relation-graph. Section 5 describes the method of rectangle detection. The experiments are presented in Section 6. Finally, conclusions are presented in Section 7.

ICVIP 2017, December 27–29, 2017, Singapore, Singapore
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5383-0/17/12…$15.00
DOI: https://doi.org/10.1145/3177404.3177417

2. LINE SEGMENT EXTRACTION

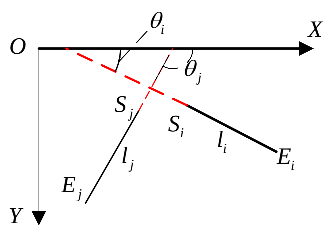


Figure 1 Line segment

Line segments are extracted using the state-of-the-art line segment detector[8]. Every line segment should be assigned a direction as Figure 1 shows. The "start" point is always on the top of the "end" point. When the line segment is horizontal, the "start" point is on the left of the "end" point. The extracted line segments are described by the "start" point, "end" point, length and direction angle. For example, line l_i can be represented as $l_i\{S_i(x_{is}, y_{is}), \mathrm{E}_i(x_{ie}, y_{ie}), length_i, \theta_i\}$.

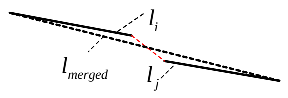


Figure 2. Merged line segment

After extracting lines, some complete line segments may be fractured into short lines because of noises and low contrast in the image. So it's necessary to merge lines that have narrow gaps into longer lines as Figure 2 shows. Line l_i and line l_j are merged to form a longer line l_{merged}.

3. CORNER DETECTION

3.1 Corner Definition

A corner is defined as a L-junction as Figure 3 shows. P_m, P_l and P_r are the vertex point, left point and right point respectively. The dashed line with an arrow is the bisector of the corner angle. θ_c, θ_l and θ_r represent the direction angle of the bisector, left side $\overrightarrow{P_mP_l}$ and right side $\overrightarrow{P_mP_r}$ respectively($\theta_c, \theta_l, \theta_r \in [0°, 360°)$). $length_l$ and $length_r$ represent the length of the left side $\overrightarrow{P_mP_l}$ and right side $\overrightarrow{P_mP_r}$ respectively. A corner can be described as follows:

- $\left\| \theta_l - \theta_r \right| - 90 \right| < angleThreshod$ or $\left\| \theta_l - \theta_r \right| - 270 \right| < angleThreshold$;
- $cornerSize = length_l + length_r$;

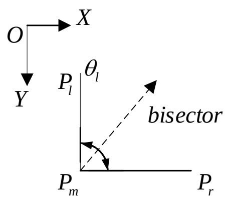


Figure 3 Corner

In general, the rectangular objects won't appear as standard rectangles in the image due to perspective distortion. As a result, the corners of a distorted rectangle won't be an ideal right angle. The angle threshold $angleThreshold$ is set to deal with this problem and can be set according to the distortion degree of the rectangle. The $cornerSize$ is used to define the size of the corner.

3.2 Corner Detection

Corners are detected based on line segments. Assuming there are a pair of line segments: line l_i and line l_j, the intersection point is computed first and then four distances(dis_{i1}, dis_{i2}, dis_{j1} and dis_{j2})between intersection point and four endpoints are computed. If the four distances satisfy certain conditions, corners are detected. A pair of line segments can produce one corner (single corner), two corners (double corners) or four corners. That a pair of line segments produce four corners is uncommon in application and is not considered. Single corner and double corners are shown in the Figure 4.

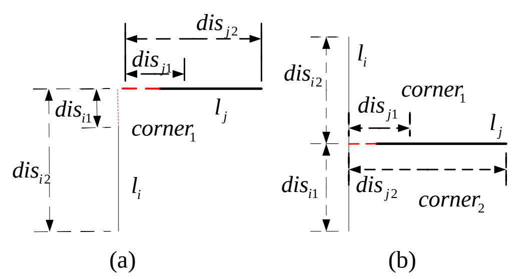


Figure 4 (a) Single corner; (b)Double corners

The conditions detecting the single corner and double corners are presented as follows:

Single corner conditions:

- $\left\| \theta_i - \theta_j \right| - 90 \right| < angleThreshold$;
- $dis_{i1} < T_{\min} * length_i$ or $dis_{i1} < T_{\min 0}$;
- $dis_{j1} < T_{\min} * length_j$ or $dis_{j1} < T_{\min 0}$;
- $CornerSize_1 = dis_{i2} + dis_{j2} > sizeThreshold$;

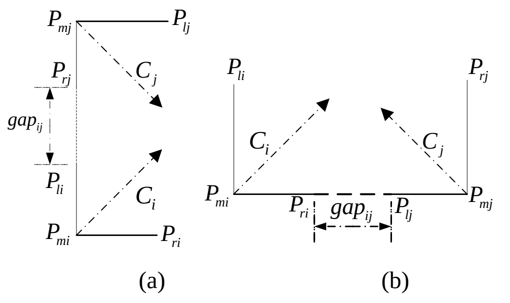
Double corners conditions:

- $\left\| \theta_i - \theta_j \right| - 90 \right| < angleThreshold$;
- $dis_{j1} < T_{\min} * length_j$;
- $dis_{i1} > T_{\max} * length_i$ or $dis_{i1} > T_{\max 0}$;
- $dis_{i1} + dis_{i2} = length_i$;
- $CornerSize_1 = dis_{i1} + dis_{j2} > sizeThreshold$ and $CornerSize_2 = dis_{i2} + dis_{j2} > sizeThreshold$;

dis_{i1} and dis_{i2} represent the minimum and the maximum distances from intersection point to the endpoints of line l_i respectively. dis_{j1} and dis_{j2} represent the minimum and the maximum distances from intersection point to the endpoints of line l_j respectively. $T_{\min}$ and $T_{\max}$ is recommended to be within $[0.1, 0.3]$ and $T_{\min}$ is less than $T_{\max}$. $T_{\min 0}$ is set to ensure the short lines can also form the sides of corners. $T_{\max 0}$ is set to avoid that the corner sides are too short. $sizeThreshold$ is set to remove small corners.

4. CORNER-RELATION-GRAPH CONSTRUCTION

4.1 Corner Relatioin Definition



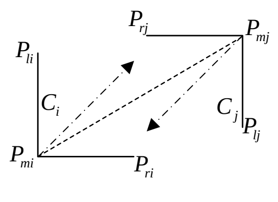


Figure 5 (a) Left corner relation; (b) Right corner relation; (c) Opposite corner relation

Based on the characteristics of rectangle's corners, three corner relations are defined. Figure 5 shows the three corner relations: left corner relation, right corner relation and opposite corner relation. There are two interesting properties about the three corner relations as follows:

- If corner C_j is the left corner of corner C_i, corner C_i is sure to be the right corner of corner C_j;
- If corner C_j is the left corner of corner C_i and corner C_k is the left corner of corner C_j, then corner C_k is sure to be the opposite corner of corner C_i.

The properties mentioned above can be called the transitivity between corner relations. Due to the transitivity, only one corner relation is enough to know the other two corner relations. Left corner relation is chosen and the detailed descriptions are presented as follows:

- $T_1 < dis(C_i, C_j) < T_2$;
- $\left||\theta_{ci} - \theta_{cj}| - 90\right| < T_3$ or $\left||\theta_{ci} - \theta_{cj}| - 270\right| < T_3$;
- $|\theta_{ij} - \theta_{li}| < T_4$ and $\left||\theta_{rj} - \theta_{ij}| - 180\right| < T_4$;
- $gapValue_{ij} = dis(gap_{ij}) / dis(C_i, C_j) < \text{T}_5$;

As Figure 5(a) shows, $dis(C_i, C_j)$ is the distance between the vertex points of the two corners. T_1 and T_2 are set to specify the search region when detecting left corner relation. θ_{ci} and θ_{cj} are the direction angle of bisectors of the two corners respectively and T_3 is set to specify the degree of parallelism between side $\overrightarrow{P_{mj}P_{lj}}$ and side $\overrightarrow{P_{mi}P_{ri}}$. θ_{ij}, θ_{li} and θ_{rj} are the direction angles of line $\overrightarrow{P_{mi}P_{mj}}$, left side $\overrightarrow{P_{mi}P_{li}}$ and right side $\overrightarrow{P_{mj}P_{rj}}$ respectively. T_4 is set to ensure the left side $\overrightarrow{P_{mi}P_{li}}$ of corner C_i and the right side $\overrightarrow{P_{mj}P_{rj}}$ of corner C_j are approximately on the line $P_{mi}P_{mj}$. $dis(gap_{ij})$ is the gap length of line $P_{mi}P_{mj}$ and can be detected by searching zero pixels along line $P_{mi}P_{mj}$ on the binary image. In order to dealing with the problem that detected line segments are not ideally on the line $P_{mi}P_{mj}$, a strip search region is adopted as Figure 6 shows. $gapValue_{ij}$ is computed to measure the degree of fracture of line $P_{mi}P_{mj}$. If corner C_i and corner C_j satisfy these conditions, corner C_j is called the left corner of corner C_i.

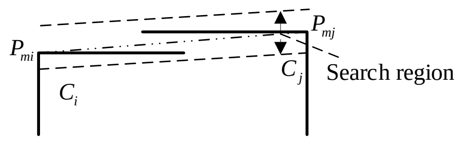


Figure 6 Gap detection

4.2 Corner-relation-graph Construction

Left corner relations are detected by searching all the corners and are stored in a graph. Nodes of the graph represent the corresponding corners and arcs between the nodes represent the corner relations. As the left corner relation is directional, the graph is a directional graph. The weights of arcs store the gap values between the corners. Figure 8 shows the corner-relation-graph generated from Figure 7.

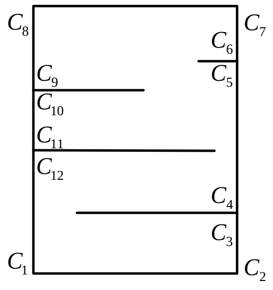


Figure 7 Corners and rectangles

$$\begin{tikzcd}
 & C_9 & C_6 & \\
C_{11} & C_8 \arrow[l, "w_8^{11}"'] \arrow[u, "w_8^9"'] \arrow[d, "w_8^1"] & C_7 \arrow[l, "w_7^8"'] & C_4 \arrow[l, "w_4^7"'] \arrow[d, "w_4^5"'] \\
C_{10} \arrow[u, "w_{10}^{11}"] \arrow[r, "w_{10}^1"'] & C_1 \arrow[r, "w_1^2"'] & C_2 \arrow[u, "w_2^7"'] \arrow[r, "w_2^5"'] \arrow[d, "w_2^3"] & C_5 \\
 & C_{12} \arrow[u, "w_{12}^1"'] & C_3 &
\end{tikzcd}$$

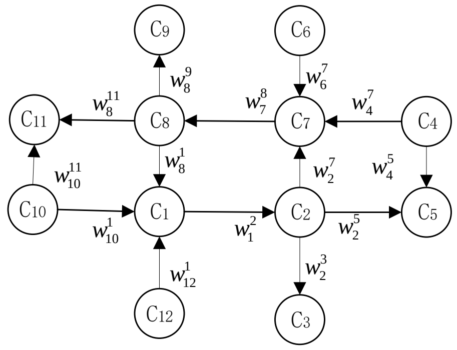


Figure 8 Corner-relation-graph

5. RECTANGLE DETECTION

As complete rectangles are formed by four corners, it is straightforward to detect rectangles by finding closed loops consisting of four nodes in the corner-relation-graph, such as the closed loop $C_1C_2C_7C_8$ in Figure 8. However, it's difficult to ensure the four corners can be detected completely in application due to noise and low contrast in the image. As a result, it is necessary to use three corners to detect the complete rectangles, which can be achieved by detecting the continuous three nodes in the graph, such as $C_1C_2C_5$ and $C_1C_2C_3$ in Figure 8. A depth-first search algorithm is used to detect the continuous three nodes and the closed loops consisting of four nodes. The rectangles detected by searching the graph is called candidate rectangles and should be verified based on the gap value of the rectangles. $rectGap$ is defined to measure the fracture of rectangles which is the sum of the $gapValue$ of the four sides.

$$rectGap = \sum_{i=1}^{4} gapValue_i$$

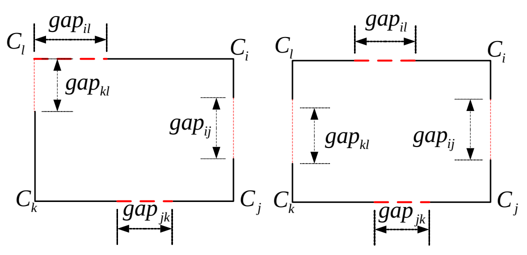


Figure 9 Rectangle detected by three nodes (a) and four nodes (b)

For the candidate rectangles detected by three continuous nodes, the missed corner needs to be computed. As Figure 9 (a) shows, the coordinate of the missed corner C_l can be computed by computing the intersection point of the right side of C_i and the left side of C_k. Then gap_{il} and gap_{kl} are detected and the $gapValue_{il}$ and $gapValue_{kl}$ are computed. If the following conditions are satisfied, then the candidate rectangle is considered to be a true rectangle.

- $gapValue_{il} < T_5$ and $gapValue_{kl} < T_5$;
- $recgap < T_6$;

The candidate rectangle detected by the closed loops consisting of four nodes is considered to be the true rectangle if the value $recGap$ is smaller than threshold T_6.

The rectangle detection approach can be extended to detect any polygons. For example, if a closed loop consisting of three or five nodes is found in the corner-relation-graph and the sum of the weight is smaller than threshold, then the corresponding corners form a triangle or a pentagon. The extensibility can expand the application field of the proposed approach greatly.

6. EXPERIMENTS

Rectangular objects are very common in everyday life, such as bins, license plates, traffic signs. Theoretically, the proposed rectangle detection approach can be applied to detect any rectangular objects. In addition, rectangular object verification is necessary in the rectangular object detection application. However, as different rectangular objects have different appearance, the verification algorithms are different. As a result, the verification algorithms aren't discussed and applied in this paper.

Figure 10 shows the rectangle detection result for synthetic image. As Figure 10 shows, one corner of the rectangle is covered by a circle and the sides are also fractured by the other two circles. But the rectangle can still be detected successfully, which proves that the approach can recover rectangle shape from three corners and overcome the problem of fracture of rectangle shape.

Figure 11, Figure 12 and Figure 13 show the rectangle detection results for bins, traffic sign and license plate. Obviously, all the rectangular shapes of bins, traffic sign and license plate are detected successfully. Besides, as Figure 11 shows, other true rectangles belonging to no rectangular objects are also detected. These rectangles are formed by interferential line segments and can be removed easily by applying object verification algorithm. The detection of rectangles not belonging to any rectangular objects should not be seen as the defect of rectangle detection approach and the basic reason is that the information about the line segment affiliation that indicates which image region the line segment belongs to is lost during line segment detection, which results in the interferential line segments. This problem can be solved by image segmentation and will be done in the future work.

Figure 14 shows the triangle detection and pentagon detection, which shows the extensibility of the approach. Theoretically, the approach can be applied to detect any polygons, which shows the broad application prospects of the approach.

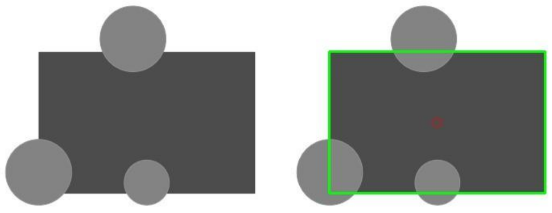

(a) (b)

Figure 10 (a) Synthetic image; (b) Rectangle

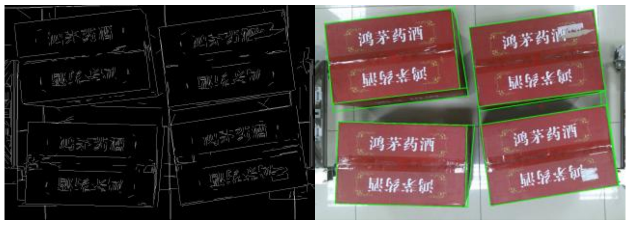

(a) (b)

Figure 11 (a) Line segments; (b) Bin rectangular shapes

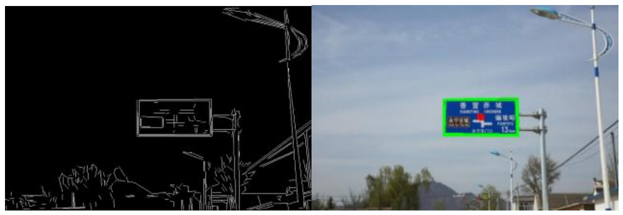
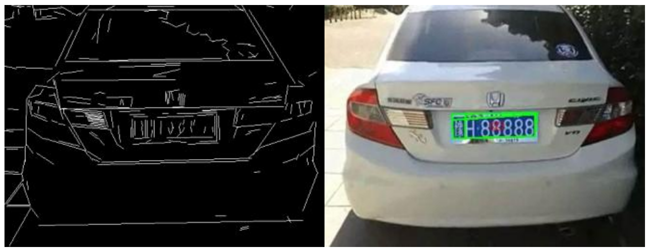
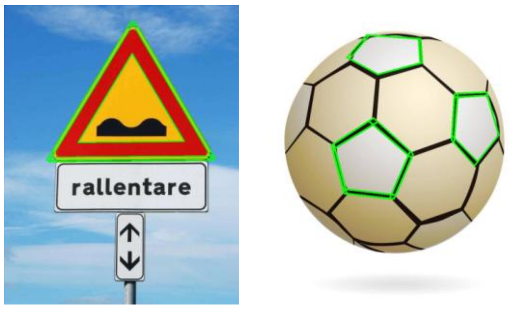

(a) (b)

Figure 12 (a) Line segments; (b) Traffic sign rectangular shape



(a) (b)

Figure 13 (a) Line segments; (b) License plate rectangular shape



(a) (b)

Figure 14 (a) Triangle; (b) Pentagon

7. CONCLUSIONS

In this paper, a bottom-up approach for rectangle detection is proposed, which constructs the hierarchical feature structure: line feature, corner feature and rectangle. Graph theory is applied to simplify the problem of rectangle detection. In addition, the approach can be extended to detect any polygons. Experiments shows that the proposed approach have excellent performance to overcome the fracture of rectangles. The future work will be concentrated on the image segmentation to deal with the problem of interferential line segments, which will certainly improve the performance of the proposed rectangle detection approach.

8. REFERENCES

[1]. Lagunovsky, D. and S. Ablameyko, Straight-line-based primitive extraction in grey-scale object recognition. Pattern Recognition Letters, 1999. 20(10): p. 1005 - 1014.

[2]. Kim, T. and J. Muller, Development of a graph-based approach for building detection. Image and Vision Computing, 1999. 17(1): p. 3 - 14.

[3]. Tao, W., J. Tian and J. Liu, A new approach to extract rectangular building from aerial urban images, in 6th International Conference on Signal Processing, 2002. 2002. p. 143-146 vol.1.

[4]. Liu, Y., T. Ikenaga and S. Goto, An MRF model-based approach to the detection of rectangular shape objects in color images. Signal Processing, 2007. 87(11): p. 2649 - 2658.

[5]. Li, Q., A Geometric Framework for Rectangular Shape Detection. IEEE Transactions on Image Processing, 2014. 23(9): p. 4139-4149.

[6]. Huertas, A. and R. Nevatia, Detecting buildings in aerial images. Computer Vision, Graphics, and Image Processing, 1988. 41(2): p. 131 - 152.

[7]. Jung, C.R. and R. Schramm, Rectangle detection based on a windowed Hough transform, in 17th Brazilian Symposium on Computer Graphics and Image Processing. 2004. p. 113-120.

[8]. R., G.V.G., et al., LSD: A Fast Line Segment Detector with a False Detection Control. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010. 32(4): p. 722-732.