Junction Detection based on Line Segments

Zhefu Tu
Dept. of Electrical Engineering and Information Science,
University of Science and Technology of China,
Hefei 230027, P.R.China
Email: tuzhefu@mail.ustc.edu.cn

Xuejin Chen
Dept. of Electrical Engineering and Information Science,
University of Science and Technology of China,
Hefei 230027, P.R.China
Email: xjchen99@ustc.edu.cn

***Abstract*—We present a novel method for junction detection. A junction is defined as the point where several lines intersect. Given the line segments in the image, our junction detector consists of three steps. First, potential junctions are located from a small neighborhood around the intersection of each pair of lines. Second, our detector searches the branches connecting to each potential junction in a circular neighborhood according to its scale. Finally, the actual junctions are selected using a two-step method according to their connecting branches and their distance. We test our algorithm on a variety of images. The experiment results demonstrate that the proposed algorithm can robustly detect junctions in different scenes.**

I. Introduction

Local geometrical structure is an important part for many problems in computer vision. While carrying the topology of lines, junctions play a critical role in many visual tasks, such as figure/ground separation [1], [2], [3], image segmentation [4], [5], object recognition [6], [7], 3D modeling, and so on. In the past decades, corner detection has been an active research area in image analysis. However, most of the existing methods look for features at high-curvature location in the gradient domain. They are very sensitive to varieties of intensity patterns. Junctions, defined as the image points where two or more edges meet, present more geometrical information for further processing.

Many works have been done for the corner or junction detection. They can be generally divided into two types, one is region-based and the other is edge-based. Region-based methods [8], [9], [10], [11], [12] use the information of the surrounding area of a pixel, while edge-based methods [12], [13], [14] use the edge detection result of an image as reference. The famous region-based corner detector, Harris corner detector [8] is widely used in many computer vision algorithms. It computes the Hessian matrix of a square window to check if there are two dominant edge directions in the region. [9] sets a circular mask over pixels and computes corner response function. These edge-based methods do not distinguish different junction types, such as Y-junction, L-junction and so on. Therefore, they are only suitable for key point extraction with little geometrical information. The junction detector proposed in [10] constructs a junction template consisting of sector partitions and uses a template deformation framework to detect the radial partitions of the template. Its application is very limited because of its high computation complexity. The scale of a corner is taken into account by making use of linear scale-space in [11]. However, the precision of location and scale is not precise enough for applications such as segmentation and 3D modeling.

Two Bayesian methods are proposed for both region-based and edge-based corner detection [12]. The region-based method identifies the corner by several regions of homogeneous intensity around it. It computes the average intensity along each direction diffused from each potential corner and maps them to a 1-D intensity profile. Smooth regions are found from the profile by a growing algorithm. The edge-based method identifies the junction by several converging edges. A different profile is computed and the edges forming a junction are found at the contrast peaks in the profile. The junction detector presented in [13] generates inspiring results. However, it requires the user to tune a set of parameters every time. Moreover, the edge continuity requirement in the algorithm is so strict that some junctions whose branches are fractured are ignored. Junctions are considered as local visual events under a contrario methodology in [14]. However, its heavy computation makes it inapplicable in real-time scenarios.

This paper proposes a new and fast junction detection algorithm to accurately locate the junctions and classify the junctions into different types according to their connecting branches. By combining the region-based and edge-based methods, our method firstly compute the locations of potential junctions from the intersections of every two line segments. Then a circle region around each potential junction is traversed to find the supporting branches that converge at the point. Using straight line segments in the image to find the junctions, the proposed algorithm can robustly detect junctions in the scenes of architectures and man-made objects. The obtained junctions can be used as features for further applications, such as scene reconstruction, structure analysis, etc.

The rest of the paper is organized as follows. Section II gives an overview of the algorithm. The algorithm details are described in Section III. Section IV shows a series of experimental results on different types of images to demonstrate the effectiveness of our junction detector. Finally, we make a conclusion in Section V.

II. Overview

Our junction detection algorithm consists of three parts: *potential junction localization*, *branch searching*, and *junction verification*, as Fig. 1 shows. In the first step, we compute the intersections of each pair of line segments and choose the intersections near the line segments as junction candidates. In the second step, we search the connecting branches to each junction candidate in a circular region with adaptive radius. Finally, the junctions are verified according to the number of branches and the distance to the branches.

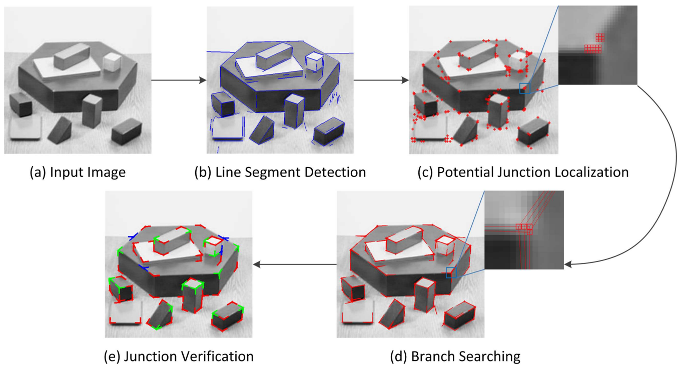

(a) Input Image
(b) Line Segment Detection
(c) Potential Junction Localization
(d) Branch Searching
(e) Junction Verification

Fig. 1. System overview.

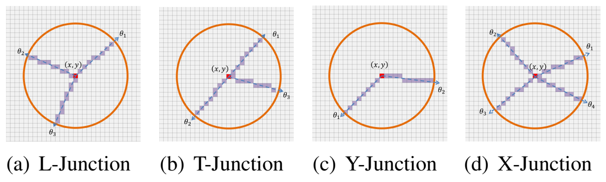


Fig. 2. Example junction templates.

III. Junction Detection

A junction is defined as the intersection point of two or more line segments. We model a junction by a parametric template $J = \{x, y, r, M, \{\theta_i\}_{i=1}^{M}\}$, where (x, y) is the location coordinate, r is the scale, M is the number of branches, θ_i in the orientation of the i^{th} branch.

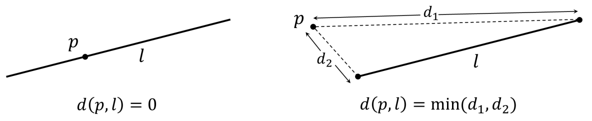
The junction is classified based on the number of interacting edges and the intersection angles. Four types are used in our work, which are adequate for describing most man-made scenes. The L-junction is a corner. The Y-junction is a common vertex of three line segments. The T-junction is a special case of the Y-junction, while the two line segments of the three are collinear. The X-junction is a point where four or more lines meet. Example junction templates are shown in Fig. 2.

Line segment detection. In order to compute the junction candidates, long straight edges are detected from the input image. Line Segment Detector (LSD) [8] is able to detect most of the straight line segments with acceptable computation complexity. Fig. 1(b) shows the line segments detected from a grayscale image. This step can be replaced by any other line detectors to obtain a series of line segments from the image.

A. Potential Junction Localization

By applying the line segment detector on the input image, we get a set of line segments $L = \{l_i | l_i = [a_i, b_i, c_i, len_i, ang_i, p_i^{start}, p_i^{end}]\}$, where $a_i x + b_i y = c_i$ is the parameter expression of l_i, len_i is the length, ang_i is the line direction, the last two terms represent the line's two endpoints.

To find the true junctions, we first compute the intersections of each pair of the line segments l_i and l_j unless the angle between them is smaller than τ, which means they are nearly parallel. Because the length of line segments is not infinite, the true junctions usually locate at the endpoints of the line segments. We simply filter the intersection points using their belonging distance to their connecting line segments.



(a) p is on the line segment l. (b) p is not on the line segment l.

Fig. 3. Computing the belonging distance.

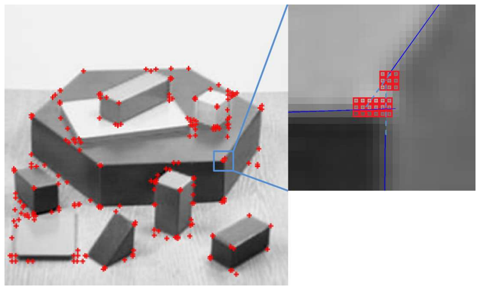


Fig. 4. Potential junctions in an image. The zoom-in figure shows all the potential junctions in a 30x30 regions.

We define the belonging distance $d(p, l)$ between a point p and a line segment l as Eq. 1. Fig. 3 shows the two cases for computing the belonging distance. In order to get reliable junction candidates, we choose the intersections whose belonging distances to their connected two line segments are both smaller than T_d as potential junctions. On the other hand, the intersection point might not be the exact junction because of the error of line detection and fitting. Therefore, we set all the nine pixels in the 3×3 small window around the intersection of line segments l_i and l_j as potential junctions. Fig. 4 shows the potential junctions in an image. From the zoom-in, we can see that very dense candidates are generated from the image.

$$d(p, l) = \begin{cases} 0, & \text{if } p \text{ is on the segment } l, \\ min(\|p, p_l^{start}\|, \|p, p_l^{end}\|), & \text{otherwise.} \end{cases} \quad (1)$$

B. Branch Searching

In order to identify an actual junction, we look for its supporting branches. For each junction candidate p obtained from two segments l_i and l_j, we search its supporting branches in its surrounding circular region. We present an adaptive searching range to identify the junctions of different scales, as the junctions A and B in Fig. 5. Generally, the junction formed by the short line segments should have a small search region while the junctions formed by long line segments should have a larger search region to get strong support.

We determine the radius of the searching region as follows: First, we compute the possible searching range r_l for each line segment as Eq. 2 to make sure it can be reached in this range. Second, we set the searching range r as the minimum of (r_{l_i}, r_{l_j}, T_r). We use T_r to avoid too large search region so that the computation can be limited in a reasonable range. The performance is not reduced since the results tend to be stabilized when the search range is big enough.

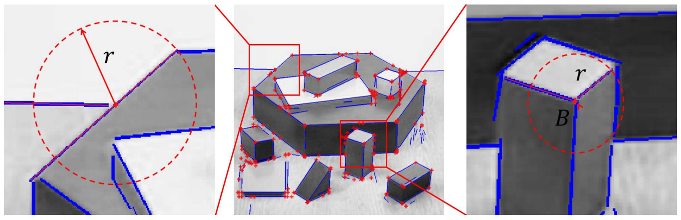


Fig. 5. Junctions at different scales are formed by the line segments with different lengths.

$$r_{l_i} = \begin{cases} max(T_d, min(\|p, p_{l_i}^{start}\|, \|p, p_{l_i}^{end}\|)), & \text{if } p \text{ is on } l_i. \\ len_i, & otherwise. \end{cases} \quad (2)$$

Given the search range r of a potential junction, we search the supporting branches as follows.

1) Generate a point set $\mathbf{CP}$ of all the points on the circle with radius r centered at $p(x, y)$.
2) For each circle point $q_i \in \mathbf{CP}$, we select the three-pixel width band region $\mathbf{B_i}$ along the line segment $l(p, q_i)$ using Bresenham's algorithm [16], as shown in Fig. 6.
    a) We create a chain from p to q_i. For each row in the band, if there exists an edge pixel, set the corresponding element in the chain as value 1; otherwise, the element is 0.
    b) If the longest connected path on the chain is longer than $0.8r$, we check if all the pixels on the link belong to a single line segment. If so, $l(p, q_i)$ is accepted as a branch candidate. Otherwise it is ignored.
3) The searching operation is repeated with the next circle point.
4) The distance $D(p, l(p, q_i))$ from the junction candidate p to each branch candidate $l(p, q_i)$ is computed. Ideally, the distance is 0.
5) A non-minimum suppression is used to avoid the ambiguous lines' appearance near the edge line. For each branch candidate, if there exists an adjacent branch candidate whose distance to p is smaller, the branch is removed.

If there is only one support branch found for p, we reject p as a junction. For the junctions with two or more support branches, we finally verify each candidate according to its structure.

C. Junction Verification

Because we compute check nine pixels in a small neighborhood of a real junction and a junction formed by more than two lines is checked twice or more, there are many redundant responses. We verify the junction candidates having two or more support branches based on its number of branches and the distance to the branches. If a potential junction has only two branches, we compute their intersection angle. If the intersection angle is smaller than τ, it is removed in order to avoid the case that the two branches are actually on the same line. For the remaining junction candidates, we compute the number of branches and the direction of each branch.

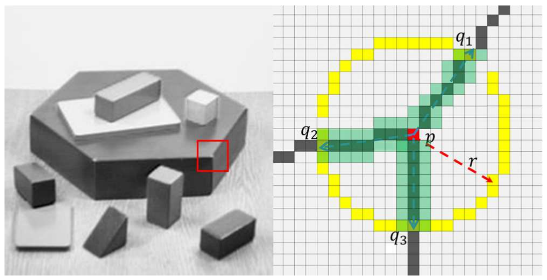


Fig. 6. A Y-junction in image. Three branches are found in the green band region along the direction (p, q_i) $i = 1, 2, 3$.

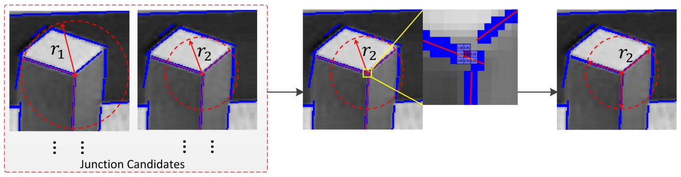


Fig. 7. Different kinds of junctions can be detected in various scales. The more complex junctions are obtained, and their distance to the furthest branches are computed. Finally, the central red point is selected with its scale equal to r_2.

We verify the junctions using a two-step selection method. As Fig. 7 shows, a Y-junction candidate is detected three times from the three pairs of line segments. In the smaller circular region, three branches are found while only two branches are found in the other region. For this case, the real junction can be easily picked out with the largest number of branches. After this, if there are more than one junction in a 3×3 window, we select the candidate having the smallest distance to all its supporting branches. With this two-step verification process, our junction detector finally obtains high accurate detection rate and low false detection rate.

IV. Experimental Results

We test our algorithm on a computer with a 3.30GHz CPU and 2.0GB RAM. The three thresholds used in our system are set as belonging distance threshold $T_d = 10$, intersection angle threshold $\tau = 15°$, and scale upper limit $T_r = 30$ after numerous experiments.

We first test our junction detector on the synthetic images. Our system is able to extract all the junctions in the synthetic square image containing a set of rectangles with different rotation angles and grey values, as Fig. 8 shows. We also compare our method with the popular Harris detector [8] and the recent JUDOCA method [13] on the noised image and a natural image. From Fig. 9, we can see that our method outperforms the other two detectors as the Harris detector produces many false corners in the noise regions and the JUDOCA misses many real junctions. In comparison, our method produces more robust results without false detection.

There are more types of junctions in natural images. We classify the junctions according to the number of their connected branches and the intersection angle. Our method is able to clearly detect the junction locations and classify

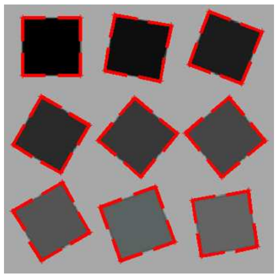


Fig. 8. The algorithm is applied in synthetic square image.

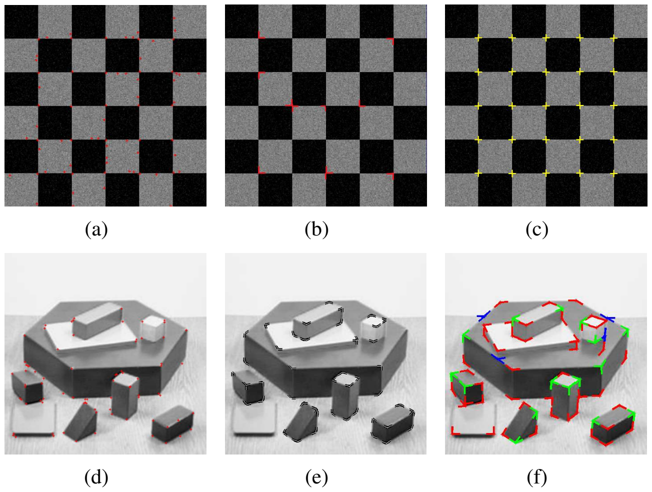


Fig. 9. Comparison of Harris corner detector (a, d), JUDOCA (b, e) and our detector (c, f), Gaussian noise with mean zero and variance 0.002 is added to the square image. L-junctions are marked in red, T-junctions in blue, Y-junctions in green, and X-junction in yellow.

the junctions in natural images. Fig. 10 shows a series of junction detection results on architectural images of different resolutions and different structure details.

V. Conclusion

This paper proposes an edge-based junction detection algorithm. From the intersections of each pair of line segments, we look for the potential junctions in a small neighborhood. Then branches are searched in a circular region for each junction candidates. By selecting the junction with the largest number of branches and the smallest distance to its connecting edges, we finally obtained real junctions in the image. The junctions obtained by our detector have strong ties with the geometrical and topological structure of the scene. Therefor, it can be used as a prominent feature in many further applications such as 3D modeling and image segmentation.

References

[1] D. Geiger, K. Kumaran, and L. Parida, "Visual organization for figure/ground separation," in *Proc. of Computer Vision and Pattern Recognition*, vol. 0, pp. 155-161, 1996.

[2] X. Ren, C. C. Fowlkes, and J. Malik, "Figure/ground assignment in natural images," in *Proc. of European Conference on Computer Vision*, vol. 2, pp. 614-627, 2006.

[3] I. Leichter and M. Lindenbaum, "Boundary ownership by lifting to 2.1d," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 9-16, 2009.

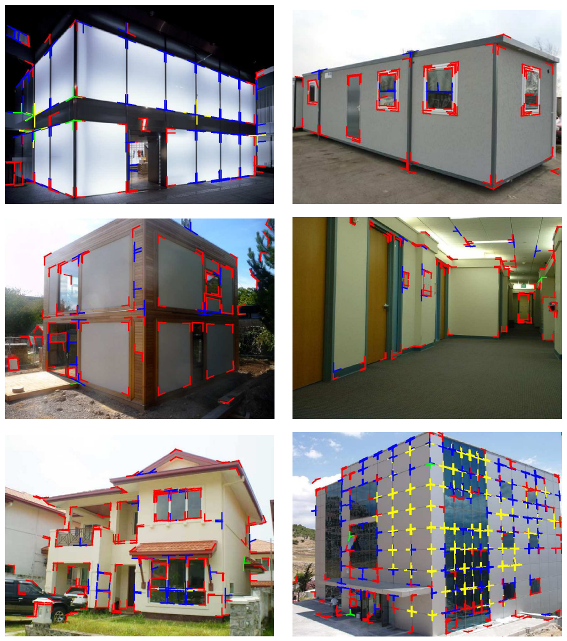


Fig. 10. Junctions obtained by our algorithm for architectural images.

[4] C. Fuchs and W. Forstner, "Polymorphic grouping for image segmentation," in *Proc. of Inter-national Conference on Computer Vision*, pp. 175-182, 1995.

[5] T. Lindeberg and M.-X. Li, "Segmentation and classification of edges using minimum description length approximation and complementary junction cues," *Comput. Vis. Image Underst.*, vol. 67, no. 1, pp. 88-98, 1997.

[6] A. Guzmán, "Decomposition of a visual scene into three-dimensional bodies," in *Proc. of Fall Joint Computer Conference, part I*, pp. 291-304, 1968.

[7] R. C. Bolles and R. A. Cain, "Recognizing and locating partially visible objects: The local-feature-focus method," *Int. J. Robotics Research*, vol. 1, no. 3, pp. 57-82, 1982.

[8] C. Harris, M. Stephens, "A combined corner and edge detection," in *Proc. of The Fourth Alvey Vision Conference*, pp. 147-151, 1988.

[9] S.M. Smith, J.M. Brady, "SUSAN: A new approach to low level image processing," *Proc. of International Journal of Computer Vision*, vol. 23, pp. 45-78, 1995.

[10] T. Lindeberg, "Junction detection with automatic selection of detection scales and localization scales," in *Proc. of International Conference on Image Processing*, pp. 924-928, 1994.

[11] R. Hummel, L. Parida, D. Geiger, "Junctions: Detection, classification, and reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 687-698, 1998.

[12] M.A. Cazorla, F. Escolano, "Two bayesian methods for junction classification," *IEEE trans. on Image Processing*, vol. 12, no. 3, pp. 317-327, 2003.

[13] R. Elias, R. Laganière, "JUDOCA: junction detection operator based on circumferential anchors," *IEEE Trans. on Image Processing*, vol. 21, no. 4, pp.2109-2118, 2012.

[14] G.-S. Xia, J. Delon, Y. Gousseau, "Accurate junction detection and characterization in natural images," *TechReport-HAL-00631609*, pp. 1-33, 2011.

[15] R.G. von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, "LSD: A fast line segment detector with a false detection control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 722-732, 2010.

[16] J.E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, 1965.