https://doi.org/10.26682/sjuod.2020.23.2.57

Journal of University of Duhok,Vol.23, No.2 (Pure and Eng. Sciences),Pp 731-744, 2020 (Special Issue)

A THINNING-BASED JUNCTION DETECTION AND RESOLUTION ALGORITHM FOR DOCUMENT IMAGES

HASAN S.M. AL-KHAFFAF[1,*] and ABDULLAH Z. TALIB[**]

[*]Dept. of Computer Science, College of Science, University of Duhok, Kurdistan Region-Iraq

[**]School of Computer Sciences, Universiti Sains Malaysia, 11800 USM Penang, Malaysia

(Accepted for Publication: December 8, 2020)

ABSTRACT

Junction detection plays a significant role in document image recognition. High recognition rate of graphical primitives is correlated with the proper detection of junctions. In this paper, a junction detection algorithm is presented in thinning-based raster to vector conversion process. The method has three stages that leverage junction representation from pixels to features (i.e. junctions). The input image is thinned to its skeleton. Edges were found next and pixels with many neighbours are designated as a low level junction. Polygonal approximation on edges is used to detect L-junctions while connected component analysis is used to find intermediate junctions. Intermediate junctions with a distance less than a threshold are combined to form a high level X- and Y-junctions. Performance evaluation on mechanical engineering drawings shows precision rate of 82.38% and a recall rate of 97.29%.

KEYWORDS: junction detection, document image, binary image, document analysis and recognition, performance evaluation

1. INTRODUCTION

In document images, junctions are formed by the intersection of at least two regions (Pham et al., 2014). Junctions can be found by locating the prominent points at regions intersection. Lines intersect forming many types of junction configurations like L-, X-, and Y-Junctions. Raster to vector conversion methods includes some mechanism to deal with junctions. Different approaches are available to tackle this issue. Zhang et al. (2019) presented a method based on discrete curvature by modelling the corner points into three types: stair, end, and L-shaped models. They also presented a corner measure for corner resolution. Ma et al. (2019) presented a junction detection method in real images through a set of heuristics to preserve only meaningful junctions. Experimental results on real images of it3f dataset show efficient and robust results. Wang et al. (2019) presented a junction detection method to detect cracks in pavement images. They used a correlation structure analysis. Then they calculated the correlation structure index to differentiate between crack, stone mixture, and pavement. Tensor voting is followed next to indicate the shape of the tensor to either ball tensor or stick tensor. To extract crack junction, the local maximum of ball tensor saliency is applied. Document images by nature differ from other realistic images in many aspects. Document images contain text, line like shapes and drawings. Also, it is common that document images need fewer colours to represent image content compared with photorealistic images. Actually, two colours (black and white) are quite enough to present content in many types of document images. Hence, dominant point detection methods and techniques for photorealistic images might not be suitable or adequate to detect dominant point (i.e. junctions)

hasan.salim@uod.ac; azht@usm.my

[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

in document images hence many methods are developed by researchers specifically for document images.

The rest of this paper is organized as follows. In section 2, related work is presented. Section 3 explains junction detection in thinning-based methods. Section 4 presents the proposed junction detection method within a suggested framework. The background and details of the proposed three-stage junction detection are presented in Section 5 and 6. The performance evaluation of the proposed method is presented in Section 7. The paper ends with a conclusions section.

2. RELATED WORKS

Pham et al. (2012, 2013) presented a method for junction detection consisting of many stages. The first three stages are skeletonisation, curvature point detection, and distorted zone detection. A junction is then detected and optimized by merging information of line characteristics and pixel-selection in distorted-zones. Seo et al. (2014) proposed a junction detection method in document images with the focus on table detection. They used probabilistic progressive Hough transform for line detection. The lines are then refined using random sampling consensus (RANSAC) method. They imposed a graph over the intersected lines and create 9 cases for intersections depending on how the lines are intersected. A cost function is then used to refine the junctions by removing falsely detected junctions. They used an in-house dataset of 85 images of tables. Yuan et al. (2015) presented two-step junction detection in tables of Golf scorecards. In the first step, table boundaries are detected based on the assumptions that tables will have the largest connected component in the image while in the second step, junctions where detected at the intersection between horizontal and vertical lines. Pham et al. (2014) presented a reliable junction detection method. First they remove noise and then perform thinning to get the skeleton. Then they detected candidate junctions as 2-junction or n-junctions based on the number of arms forming the junction. Candidate junctions and line width information is used to detect and remove distorted zones. Finally, the candidate junctions are merged and false alarm junctions are removed and the final junction position is located.

Seo et al. (2014) method detect tables in camera captured images. Because lines in tables are intersected in predefined patterns, the authors predefine 9 intersection patterns. While this method worked well with tables but it might not perform well on other types of drawings with lines intersected at arbitrary orientations. As with Seo et al. (2014) method, Yuan et al. (2015) method detects tables in mobile captured images of tables of scoreboards by means of a novel junction pattern matching. The method used 9 predefined intersection patterns hence has the drawback of recognizing junctions of lines intersected at predefined orientations rather than the lines intersect at arbitrary orientation. The proposed method and Pham et al. (2014) methods are both thinning-based methods however we have different approach to the junction detection solution. In our method, we found many points inside real junctions area then merge them to form a detected junction while in Pham et al. (2014) they found connected component distorted zone (i.e. several zones intersected together) and they removed all intersections lying inside that zone then they reconstruct junctions by relying on reliable line segments.

In this paper a junction detection algorithm is proposed to detect the real junctions in document images. As opposed to Seo et al. (2014) and Yuan et al. (2015) methods, the proposed junction detection method can detect junctions created by intersections of any number of lines in any orientation. A thinning-based framework

hasan.salim@uod.ac; azht@usm.my
[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

for raster-to-vector conversion process is also proposed where the proposed junction detection algorithm is suitably placed in the framework. The proposed methodology was first introduced in Al-Khaffaf and Talib (2020). In this paper, however, full aspect of the methodology is presented showing the detail of the junction detection algorithm as well as the mathematical formulation of the proposed solution.

3. JUNCTION DETECTION IN THINNING-BASED RASTER-TO-VECTOR CONVERSION

Raster-to-vector conversion is a well-known topic in the area of graphics recognition. Thinning-based methods are one of the most popular families of methods to be used for the conversion purpose (Hilaire and Tombre, 2006). Although thinning-based methods produce high quality lines, its junction detection ability is poor. This family of methods is also sensitive to noise causing the creation of artificial junctions. Hence there is a need to propose better ways to detect junctions in order to detect a graphical element in large extent rather than detecting it in many small fragments (i.e. avoiding/minimising manual editing). A good noise removal algorithm should clean the noise and produce clean edges for the graphical elements to avoid creating false junctions which will participate in breaking graphical elements into smaller fragments.

4. THE PROPOSED THINNING-BASED RASTER TO VECTOR FRAMEWORK

Thinning-based raster-to-vector conversion methods rely on thinning (skeletonisation) algorithms to simplify the input image before proceeding to the next analysis stages. As with other raster to vector methods, the proposed framework includes the general steps of such methods as shown in Figure 1. The input raster image is processed to find medial axis pixels by removing redundant layers of pixels. Some sort of line tracking is then used to find and accumulate data points into a list of points. Polygonal approximation removes redundant points leaving only the dominant one which is used to create raw vectors (i.e. vectors that need further processing).

Figure 2 shows a flowchart of the proposed raster-to-vector conversion framework. The vertical arrow on the left of the figure represents the type of data processed at each step of the proposed raster-to-vector conversion. The earlier stages process raster data while the steps close to the end of the framework process vector data. The steps in the middle process data stored in special data structures. The first step is thinning which is the process of obtaining the skeleton of the graphical elements. The subsequent stages constituting the proposed junction detection method are low-level junction (LLJ) detection, mid-level junction (MLJ) detection, and high-level junction (HLJ) detection. The proposed LLJ detection is used primarily to detect junctions, but it also helps in

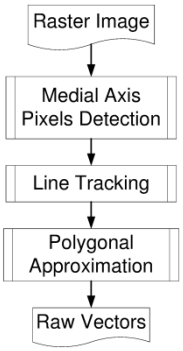


Fig. (1): General steps of vectorisation algorithms.

detecting/classifying other types of pixels. In the next step, MLJ detection merges two or more LLJs into one MLJ. It also includes labelling of each newly created MLJ. Finding edges is a process that transforms graphical elements in a raster image into a form suitable for conversion into vector form. At this stage, the graphical elements may be fragmented due to the junctions. Polygonal approximation finds the breakpoints

hasan.salim@uod.ac; azht@usm.my
[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

(if any) in the detected edges. If a breakpoint is found, the edge will be broken into smaller pieces. In the proposed HLJ detection, MLJs will be inspected for possible merging into HLJs. If no such merging is possible for any MLJ, it will be promoted to an HLJ. After the creation of each junction, junction resolution is necessary in order to update edge information with the newly created junctions.

The final stages of the framework start with the process of fitting curves into straight lines and circular arcs. The Arc Fit Error (AFE) and Line Fit Error (LFE) are used to decide whether the curve (also called edge) is a straight line or a circular arc (Lim et al., 1995). In this stage, all pixels belonging to a curve are used to calculate its vector representation. Two end points are calculated for straight lines. A centre, a radius, a starting angle, and an

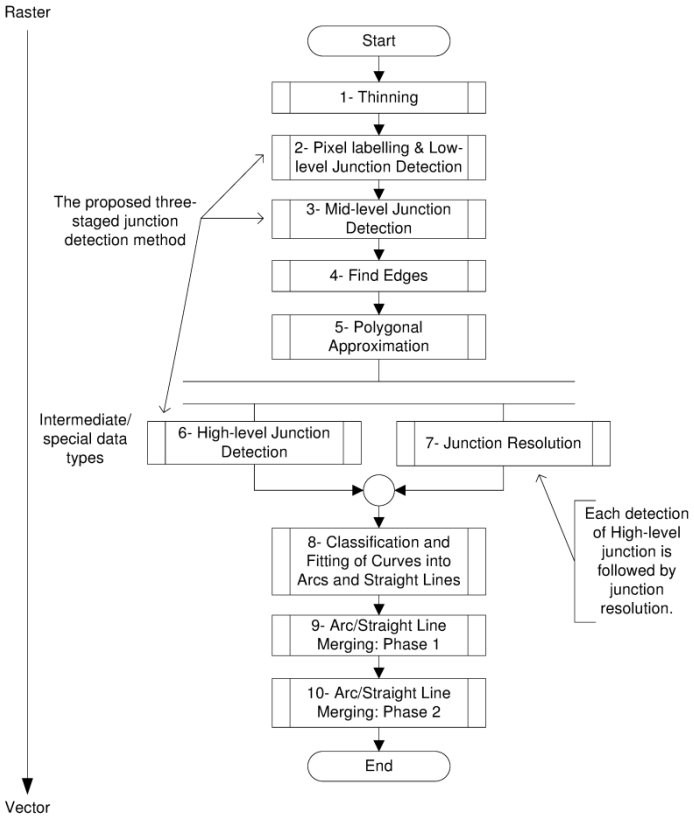


Fig. (2): Proposed Raster-to-Vector Framework.

ending angle are calculated for each circular arc. Since curves may be fragmented due to intersections, curve merging is used to merge possible curves. In the first phase of merging, curves around breakpoints are investigated for possible merging. In the second phase of merging all the curves are investigated for possible merging. HLJs are used at this stage to decide possible merging candidates. After curve merging, the vector data is prepared to be saved into a proper vector file format such as a document exchange file (DXF).

The steps of the proposed framework are grouped into three stages according to the type of data processed in each step: raster stage, intermediate representation stage, and vector stage.

4.1 Raster Stage

Raster stage consists of four steps: noise removal, thinning, LLJ detection, and MLJ

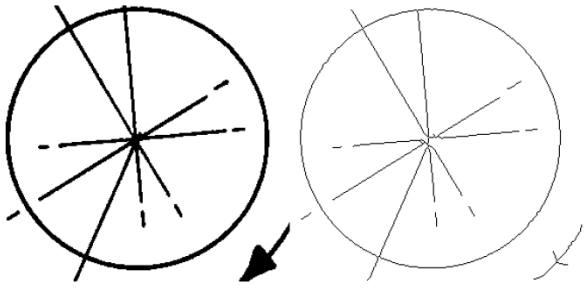
hasan.salim@uod.ac; azht@usm.my
[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

detection.

1. Thinning reduces most of the image data to thin skeletons. After thinning, all line-like graphical elements are being reduced to one-pixel-wide. This will simplify the analysis of primitives (e.g. straight lines and arcs). However, line intersections are still complex and their representations range from an intersection by one pixel to an intersection by many pixels that are connected with each other. Note the complex intersection junction in Figure 3. Hence, many levels of processing may be needed to get real junctions. In the implementation, we use the parallel thinning algorithm with two subiterations (PTA2T), a multi-pass thinning algorithm with two subiterations (Zhang and Wang, 1996). The advantage of using this method is that it produces perfectly 1-pixel-wide skeleton. A moving window will slide horizontally (from top to bottom and from left to right) and the type of pixel p is inspected using a lookup table. Pixel p will be marked for removal if it is found to be stable. All marked pixels will be removed after the end of each subiteration. The algorithm terminates when no further removal of any pixel is possible.

2. Pixel labelling and LLJ detection stage start with classifying image pixels into different categories in order to prepare data for further processing. The advantage of the classification is the ability to detect pixels (i.e. LLJs) with a high probability of belonging to a real junction. The other advantage is the ability to determine the end points of the thinned curves.

3. MLJ detection is applied to merge adjacent LLJs into larger junction areas called MLJs. The merge operation is performed by means of Connected Components (CC) computation. An algorithm developed by Di Stefano and Bulgarelli (1999) is used at this stage to compute image CC's. Each MLJ will be given a label to distinguish it from others.



(a) (b)

Fig. (3): Complex junction. (a) before and (b) after thinning by PTA2T.

4.2 Intermediate Representation Stage

Intermediate stage consists of four steps: finding edges, polygonal approximation, HLJ detection, and junction resolution.

1. Finding edges of the graphical elements involves tracking and storing the coordinates of all pixels belonging to one raster curve into a special data structure that facilitates further analysis by the next stages. This stage makes use of the information created by the previous stages such as pixel labelling, LLJ detection, and MLJ detection. The proposed labelling scheme makes it easier to find all the edges in the image since each edge (in general) starts and ends with an end pixel. The tracking operation starts by searching for an end point. After getting an end point the sequence of pixels between the two end points are tracked by chaining from one pixel to the other till all pixels belonging to an edge is tracked. It is important to note that at this stage of processing each edge found is not necessarily representing one unique curve as different curves may be connected with each other in a way that is not distinguishable to the algorithm of the

hasan.salim@uod.ac; azht@usm.my
[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

current stage. For example, a detected edge may start with a straight line and at some point, it starts to turn into other direction creating a circular arc. This fact is acceptable at this stage and such edge will be broken into its basic curve structures (i.e. straight lines and circular arcs) by the next stages. The data at this stage starts to move away from the raster representation used in previous stages to a new representation (a special data structure) that will simplify the analysis and conversion of data into a vector representation.

2. Polygonal Approximation (PA) is used next to analyse the pixels of each edge and to find points where two different graphical elements connect. These points are called breakpoints. Nakagawa and Rosenfeld (1979) method is designed to work with mechanical engineering drawings hence it is used in this work. The first step in this method is to calculate the curvature of all pixels of the edge under investigation. This curvature is calculated relative to two reference points before and after the pixel under investigation. A local maximum of the calculated curvature within a short segment of the edge will signal the existence of a sharp change in slope, and hence the detection of a breakpoint.

3. HLJ detection is the final step in the proposed junction detection. The HLJ detector will search in the neighbourhood for all MLJs and merge them (if their distance is within a predefined threshold) to form a HLJ.

4. Junction resolution is applied after each merge of MLJs to form an HLJ. Junction resolution will take place in order to search for all affected edges and updating their data properly. This is important since the merging of MLJs into HLJs changes the label of the latter. Hence, the edge data need to be updated accordingly.

4.3 Vector Stage

Vector stage consists of three steps: classification and fitting of curves, arc/line merging (phase 1), and arc/line merging (phase 2).

1. Classification and fitting of curves into arcs and straight lines starts with the fitting of each edge by a straight line and a circular arc. Curve fit quality is then calculated. The two criteria used are AFE and LFE (Lim et al., 1995). AFE shows the error when fitting the curve by a circular arc while LFE shows the error when fitting the curve by a straight line. If AFE is less than LFE this means the curve is better fitted as an arc than a straight line; otherwise the curve segment is better fitted as a straight line than as an arc.

2. Arc/straight line merging: Phase 1 is a process that analyses each edge that was broken by PA into two or more pieces for a possible merge. If such a merge is possible the merged curve is refitted again and converted to vector form.

3. Arc/straight line merging: Phase 2 analyses all edges for possible merge with adjacent edges. Information on HLJ is used to decide potential candidates for merging.

5 BACKGROUND TO JUNCTION DETECTION

In the following sections a description of the proposed junction detection in images of mechanical engineering drawing is explained. Many definitions will be set to simplify the formation of the algorithm.

Let I be a binary image with B as the set of background pixels and F as the set of foreground pixels such that $B \cup F = \phi$. The polarity of the binary image is 1 for foreground and 0 for background pixels. The words pixel and point are used interchangeably in this paper to denote the smallest accessible image element. Every pixel p in F may have up to eight direct neighbours (or 8-neighbours) belonging to F as shown in Figure 4. The distance between two pixels is denoted by $d(.,.)$. For a set of pixels $p_0, p_1, \ldots, p_n$ the distance $d(p_0, p_n) = n$ if p_i is 8-neighbour of p_{i+1} for all $0 \leqslant i < n$. In other words, $d(p_0, p_n)$ is the

hasan.salim@uod.ac; azht@usm.my
[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

number of steps we need to perform in order to move from pixel p_0 passing through the adjacent neighbours till we reach pixel p_n, and each pixel is visited only once.

p[7]	p[0]	p[1]
p[6]	p	p[2]
p[5]	p[4]	p[3]

Fig. (4): Configuration of pixel p.

Neighbour Number: Denoted as NN(p) is the number of 8-neighbours for pixel p and it is defined by:

$$NN(p) = \sum_{k=0}^{7} p[k] \quad (1)$$

Neighbour Set: Denoted as NS(p) is the set of 8-neighbours for pixel p and it is given by:

$$NS(p) = \{q | d(p,q) = 1 \wedge q \in F\} \quad (2)$$

Many types of junctions may appear in the image are shown in Figure 5. However, lines are intersected at more complex configurations such as that shown in Figure 3-a. After thinning, the intersection is no more a single point, but a series of pixels connected by short line segments and possibly contain loops (Figure 3-b).

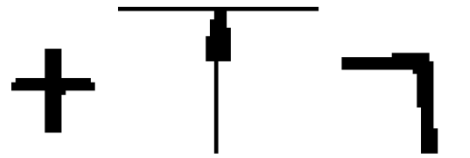


Fig. (5): Different junction types in document images.

6 PROPOSED JUNCTION DETECTION METHOD

The proposed junction detection method works from low level representation of junctions (i.e. pixel representation) to high level representation called HLJs. The proposed junction detection method consists of three stages:

1. Low-level (isolated junction pixels, also called pseudo junctions)
2. Mid-level (connected component junctions)
3. High-level (real junctions)

6.1 Low-level Junction Detection

In the proposed method, image pixels are classified into four different groups according to the number of neighbours for each pixel as well as pixel value. The first group includes all background pixels, which will not be modified. The second group (junction points) includes all foreground pixels with three or more 8-connected neighbours. A junction pixel in the image is $p \in F$ such that $NN(p) \geqslant 3$.

The set of all LLJ points (pseudo junctions) in the image could be defined as:

$$J = \{p | p \in F \wedge NN(p) \geq 3\} \quad (3)$$

The third group (edge points) includes all foreground pixels with only two 8-connected neighbours. An edge pixel in image is $p \in F$ such that $NN(p) = 2$.

The set of all edge points in the image is given by:

$$G = \{p | p \in F \wedge NN(p) = 2\} \quad (4)$$

The last group (end points) includes all foreground pixels having either only one 8-connected edge pixel as a neighbour or a pixel that has only two 8-connected neighbours where one or both of these pixels should be a junction pixel. An end pixel in the image is $p \in F$ such that $NN(p) = 1$ or $NN(p) = 2$ where either or both of the neighbouring pixel q and r belongs to J.

The set of all end points in the image is given by:

$$D = \{p | p \in F, q \in NS(p) \vee r \in NS(p), NN(p) = 1 \ \vee [NN(p) = 2 \wedge (r \in J \vee q \in J)]\} \quad (5)$$

Figure 6 shows four samples of X-junctions. All of the four junctions have different pixel configuration after performing thinning. Original image, thinned image, and labelled image were shown for each sample. Edge points, junction points, end points are shown in black, red, and

hasan.salim@uod.ac; azht@usm.my
[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

blue, respectively. The sample thinned images shows that the pixel configuration of different X-junctions are different from junction to junction even when such junctions look similar to the human eye.

In practice the pixels of the image are labelled as follows:

• Background points carry their original value of 0 (or BACKGROUND).

• Junction points are labelled as 3 (or JUNCTION).

• End points are labelled as 4 (or END).

• Edge points are labelled as 255 (or EDGE).

Pixel labelling is applied using the same sequence demonstrated above.

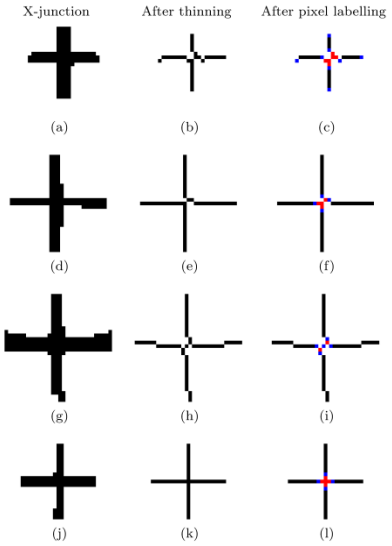


Fig. (6): Different samples of X-junction. In the 1st row, the X-junction is in its normal size. In the 2nd, 3rd, and 4th rows, only the X-junction created by the intersection of a long lines are shown- hence end points for those X-junctions are not shown.

6.2 Mid-level Junction Detection

The LLJ detection might produces many junction pixels for each real junction. These pixels may be adjacent to each other (i.e. they are part of the same connected component). While other junction pixels related to the same real junction are separated by other edge or end pixels. The next step is to group all LLJs that are connected in order to form an MLJ. The grouping is performed by calculating image connected components. A connected component C is the maximum set of pixels in F that are adjacent to each other (Di Stefano and Bulgarelli, 1999). Two pixels P and Q are connected if for all pixels $p_0, p_1, \ldots, p_n$ we have p_i and p_{i+1} as direct neighbours $0 \leqslant \square i \leqslant n-1$, $p_0 = P$ and $p_n = Q$. MLJ is formed by calculating the CC for junction pixels only. All other kinds of pixels are ignored (i.e. don't care) at this stage of processing. The MLJ can be formulated as:

$$M_l = \{p | p \in J \wedge p \in C_l\} \quad (6)$$

where C_l is a connected component, the subscript l is a class label that belongs to the set of labels L. No two CC's share the same pixel (i.e. $\forall k \neq z, k \in L, z \in L, M_k \cap M_z = \emptyset$).

At this level, each junction (M_l) is distinguished from other junctions. The number of MLJs could drop compared with the number of LLJs since many LLJs are now merged into one MLJ. The MLJ count is less than or equal to LLJ count.

6.3 High-level Junction Detection

The MLJs could be merged and promoted to an HLJ if the former junctions are close to each other and satisfy pre-determined conditions. Two MLJ junctions M_i and M_j will be merged iff there is a path of foreground pixels from the pixels of M_i to those of M_j that is less than T pixels in length.

A path (denoted as $\widehat{path}$) is a set of connected pixels that can be defined as follows:

$$\widehat{path}(p_0, p_n) = \{p_0, p_1, \ldots, p_n | p_0 \in M_i, p_n \in M_j, 0 < \forall k < n, p_k \in G \cup D, \forall k < n \quad d(p_k, p_{k+1}) = 1\} \quad (7)$$

We can also extend Equation (7) to define the path between two MLJs as follows:

$$\widehat{path}(M_i, M_j) = \{p_0, p_1, \ldots, p_n | p_0 \in M_i, p_n \in M_j, \exists \widehat{path}(p_0, p_n)\} \quad (8)$$

If such a path exists between any two of the MLJs, they will be merged. The proposed algorithm will search in all possible directions for such a path. Any existing path has a length

hasan.salim@uod.ac; azht@usm.my
[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

which is equal to the number of pixels in the path excluding the first and last pixels.

An HLJ consisting of two MLJs could be represented by:

$$H_h = \{p | p \in (M_i \cup M_j) \wedge (|\widehat{path}(M_i, M_j)| < T)\} \quad (9)$$

where |.| denotes the cardinality of a set. Note that $h \in L$ and h arbitrary takes the value of either i or j in case of Equation (9), for example.

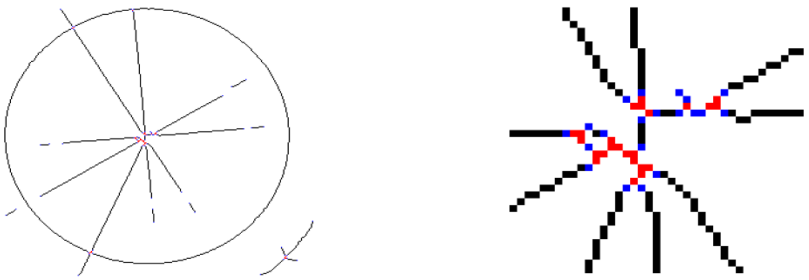
An HLJ can be defined with correspondence to MLJs (considering more than two MLJs) as follows:

$$H_h = \{M_i^0 \cup M_j^1 \cup \ldots \cup M_k^n | 0 \leq \forall p \leq n, 0 \leq \exists q \leq n, \exists \widehat{path}(M^p, M^q) \wedge (|\widehat{path}(M^p, M^q)| < T)\} \quad (10)$$

If an MLJ cannot be merged with any other MLJ to form an HLJ, it is directly promoted to an HLJ. The number of HLJs is equal or less than the number of MLJs due to the merge of some MLJs to produce HLJ. Figure 7 shows an example of a complex junction that will be handled by the proposed algorithm.

The next paragraphs demonstrate an example that shows the internal working of the proposed junction detection method. Since showing the details of the proposed junction detection algorithm on images with their original sizes is impractical due to the large amount of data generated at junction detection stage, a small part of the image (shown in Figure 3) will be used and the focus will be on the complex junction formed by the intersection of many straight lines. At each stage of junction detection, junction's information as well as edges information will be shown within the corresponding figures.

The original image (Figure 3-a) is first thinned by PTA2T algorithm. The thinned image is shown in Figure 3-b. The labelled image with LLJ detected is shown in Figure 7-a. Figure 8 zooms into the complex junction shown in Figure 7-b. The LLJ pixels are shown in red and the end points are shown in blue. At this stage, each LLJ is represented by one pixel. However, the LLJs are scattered inside the junction and some LLJs are adjacent to each other. A total of 24 LLJs are detected in this junction created by the intersection of many straight lines.



(a) Labelled Image (b) Magnified junction of part (a)

Fig. (7): A sample image with a complex junction that will be handled by the proposed algorithm. Note the loop in (b) created by noisy pixels.

The next stage is to perform MLJ detection. At this stage, all adjacent LLJs will be merged to form an MLJ. The creation of MLJs is based on calculating CCs. Only LLJ points are used to calculate CCs. Merging adjacent LLJs to form CCs will reduce the number of detected junctions and further simplify the analysis process of the image. Each MLJ is assigned a unique identity number (ID). Figure 9-a shows MLJs with their IDs.

Finding the edges is the next stage of the analysis. At this stage, all edges will be tracked and their information (which include their IDs, end points, and all pixels belonging to the edge)

hasan.salim@uod.ac; azht@usm.my
[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

will be recorded and stored in a data structure. One important piece of information that needs to be recorded is the IDs of the MLJs adjacent to each edge. This information is necessary for the Arc/Straight Line Merging stage in order to merge pieces of the primitive that is broken due to intersection. Each edge may be adjacent to at most two junctions. If one (or either) end point of an edge is not adjacent to a junction then the junction field in the data structure of the edge will be set to 0 to indicate a loose end. Figure 9-b shows the information for all visible edges of the test image. The information includes edge ID, the ID of the first MLJ (*j1*), and the ID of the second MLJ (*j2*).

Polygonal approximation is applied next. The PA algorithm will break an edge into smaller primitives at specific

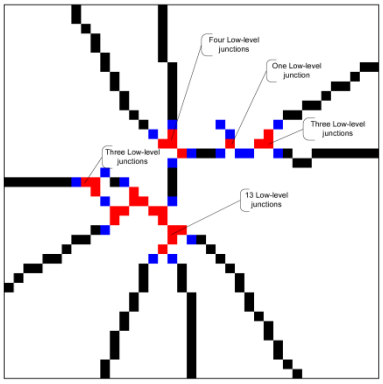
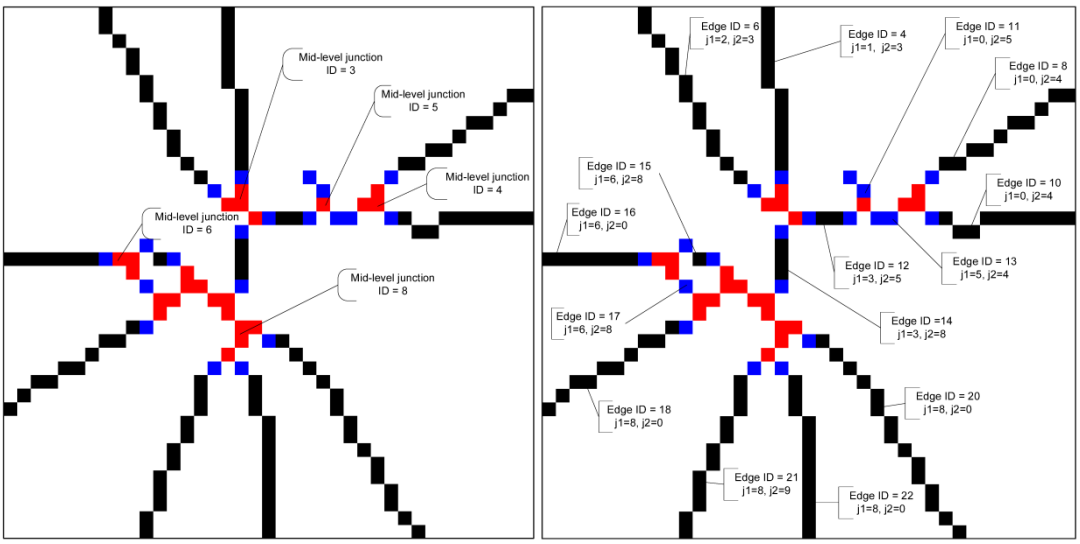


Fig. (8): A sample image showing LLJs. At this step, each LLJ is represented by one pixel (red).

points basing on the local curvature of the point. The next stage is HLJ detection. At this stage, MLJs were already detected and the set of edges were already created. The HLJ detection algorithm will find and merge MLJs that are connected by short edges. Let's assume that HLJ detection starts at MLJ with ID = 3 (we will call it junction 3), any pixel p belonging to that MLJ will be arbitrarily selected to start the merging process. This pixel p will be used as a seed to start the recursive MLJ merging process. In this process, the algorithm will search in all possible directions for MLJs to be merged with junction 3. However, the search will be bounded by existing paths of foreground pixels and with a predefined distance. When this recursive process ends, the information for the set of MLJs to be merged with junction 3 is collected. In the example, a set of five MLJs with IDs 3, 4, 5, 6, and 8 are detected by the proposed algorithm. The information of the four detected MLJs will be merged together to create an HLJ. The newly created HLJ will be assigned a unique ID (taken as 8 in this example). Figure 10-a shows the complex real junction after HLJ creation. The process of HLJ resolution will directly follow the process of HLJ creation. In HLJ resolution, the information of all edges will be updated to reflect the change in junctions information considering that some MLJs are no longer exist due to merging with other MLJs. Figure 10-b shows the updated edges information after HLJ resolution. In the figure, all edges are now referencing one real big junction which is junction 3.

hasan.salim@uod.ac; azht@usm.my
[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

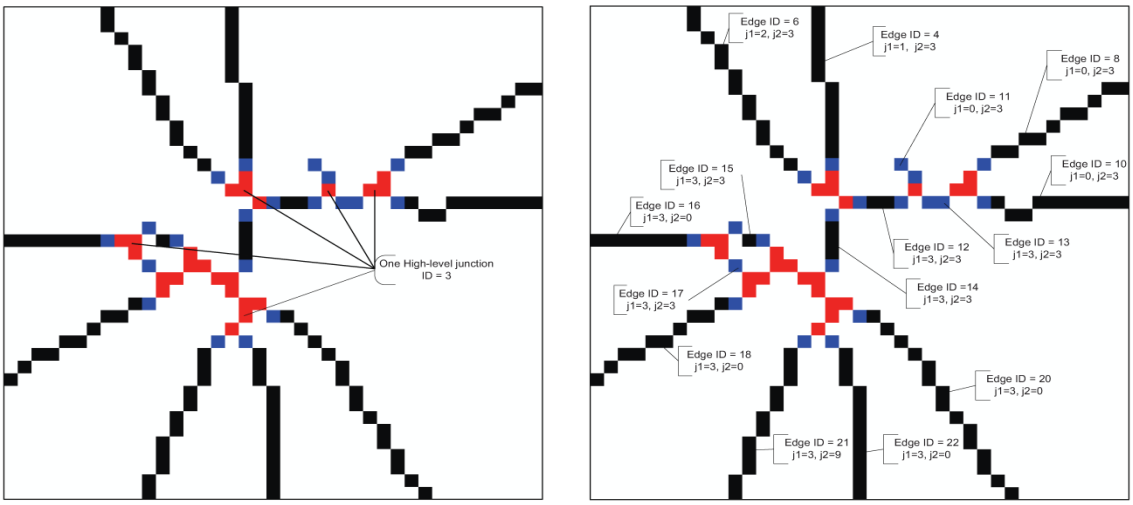


(a) MLJs after merging of LLJs of Figure 8 into an MLJ.

(b) Edge information after MLJs creation.

Fig. (9): A sample image showing MLJs. At this step, a connected component of adjacent LLJs are grouped together to form MLJs.

After the creation and resolution of HLJs, it is now possible to perform line merging by relying on edges information which is already updated to hold HLJs information. For example, it is possible now to merge edge 10 (of Figure 10-b) with edge 16 since they have approximately the same slope and they share the same junction (junction ID = 3). This means that lines before and after this complex junctions could be merged to get fine lines with attributes close to the ones drawn by the draftsman.



(a) HLJs after merging of MLJs of Figure 9-a

(b) Edge information after junction resolution

Fig. (10): A sample image showing HLJ. At this step, MLJs that are connected by short path are merged to form HLJ.

hasan.salim@uod.ac; azht@usm.my
[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

7 PERFORMANCE EVALUATION OF THE PROPOSED JUNCTION DETECTION METHOD

In this section, we evaluate the proposed junction detection method by counting the number of junctions detected and the number of real junctions. The number of real junctions for each raster image is found manually by counting the junctions in each image. The junctions counted are T-, X- and Y-junctions. The L-junctions are omitted since it is less problematic to thinning based methods and will be detected later on during the polygonal approximation process.

Manual counting involves assigning a different colour (yellow, for example) to one pixel (arbitrarily selected) that belongs to a real junction using any raster image editor. The step is repeated for all the junctions in the raster image. The advantage of this scheme is to simplify the counting of all junctions. Missing of junctions will be prevented since visual inspection is to be used. The actual junction counting will be reduced to finding the number of yellow pixels in each image. Image editors can also be used in performing this step.

7.1 Experimental Results and Discussion

The proposed junction detection algorithm is implemented in C++ within our EasyVec vectorisation software. The performance of the proposed junction detection method is studied based on original images. The images are taken from the Arc Segmentation Contest (Wenyin, 2006) attached to the International Workshop on Graphics Recognition (GREC'05).

In junction detection process, real junctions are either detected (true-positive) or missed (false-negative). Also some false junctions (false-positive) might be detected as real junctions. Hence, precision (P), recall (R), and F-measure (F) as harmonic mean of precision and recall are used for objective performance evaluation as shown below:

$$P = \frac{TP}{N_D}, R = \frac{TP}{N_R}, F = \frac{2*P*R}{P+R} \qquad (11)$$

where TP means the number of true positive detection rate of junction, N_D is number of detected junctions, and N_R is the number of real junctions in the image. The results of the proposed method based on the GREC'05 dataset is shown in Table 1 and the results of the proposed method compared with other three methods are shown in Table 2. Different articles used different datasets. However, both works of Seo et al. (2014) and Yuan et al. (2015) tested their junction detection methods on tables' datasets. The mobile/camera images complicate table recognition, however, tables are easier to recognize compared with engineering drawings due to tables' intuitive features of mostly having horizontal/vertical lines. Pham's and our method are tested on engineering drawings, by nature more complex than tables, and have many lines and curves that intersect with other elements at varying angles hence forming complex junction configurations. The Pham experiment is based on ideal electrical and ideal architectural images. Hence, Pham et al. (2014) work is the focus of our comparison since we are using images of similar complexity.

Table (1): Junction detection performance of the proposed method

Image (.BMP)	P	R	F
1	0.90	0.88	0.89
2	0.86	0.96	0.91
5	0.90	1.00	0.95
6	0.64	1.00	0.78
10	0.81	1.00	0.90
36	0.82	1.00	0.90
Mean	**0.82**	**0.97**	**0.89**

It is shown from Table 2 that R for our algorithm is better than Pham's et al. work. The F value of our algorithm is better than Pham et al. considering electrical images dataset. The R

hasan.salim@uod.ac; azht@usm.my
[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

value of 0.97 shows that our algorithm is capable of detecting most junctions in the test images. However, our algorithm detects many false junctions due to noisy edges of test images.

Table (2): Comparison of the proposed method with other methods

Method	Image type	P	R	F
Seo et al. (2014)	Camera captured tables	0.99	0.97	0.98
Yuan et al. (2015)	Mobile captured golf scorecard	0.94	0.95	0.94
Pham et al. (2014)	Electrical	0.74	0.82	0.78
Pham et al. (2014)	Architectural	0.88	0.96	0.92
The proposed method	Mechanical	0.82	0.97	0.89

8 CONCLUSIONS

In this paper, a thinning-based junction detection and resolution method is proposed. Unlike other methods that work well with only predefined line intersections, the proposed method can detect junctions formed by intersections of lines at any line orientation making it suitable to detect junctions in many image types. Experimental results on scanned mechanical engineering drawings show that the method detects most junctions of the tested images. However, the proposed method produced more false positives, on average, than the other studied methods. This is a weakness in the proposed method that is to be tackled in the future. Line width information will be studied and utilized for calculating the T threshold of Equations (9) and (10) to reduce the detection of artificial junctions. Using other types of images like architectural and electrical are also suggested.

Acknowledgements

The authors would like to thank the two anonymous reviewers for their valuable comments that helped in improving the quality of this manuscript.

REFERENCES

Al-Khaffaf, H. S. M. and Talib, A. Z. (2020), Three-stage Junction Detection in Document Images, 2020 International Conference on Computer Science and Software Engineering (CSASE), Duhok, Iraq, pp. 142-145, doi: 10.1109/CSASE48920.2020.9142083.

Di Stefano, L. and Bulgarelli, A. (1999). A simple and efficient connected components labeling algorithm. In Proceedings of the International Conference on Image Analysis and Processing, pages 322–327, Venice, Italy. Doi: https://doi.org/10.1109/ICIAP.1999.797615.

Hilaire, X. and Tombre, K. (2006). Robust and accurate vectorization of line drawings. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(6):890–904.

Lim, K. B., Xin, K., and Hong, G. S. (1995). Detection and estimation of circular-arc segments. Pattern Recognition Letters, 16(6):627–636.

Ma, J., Wang, X., He, Y., Mei, X., and Zhao, J. (2019). Line-based stereo slam by junction matching and vanishing point alignment. IEEE Access, 7:181800–181811. doi: https://doi.org/10.1109/ACCESS.2019.2960282

Nakagawa, Y. and Rosenfeld, A. (1979), A note on polygonal and eliptical approximation of mechanical parts, Pattern Recognition, 11, 133-142.

Pham, T.-A., Delalandre, M., Barrat, S., and Ramel, J. (2012). Accurate junction detection and reconstruction in line-drawing images. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), pages 693–696, Tsukuba, Japan.

Pham, T.-A., Delalandre, M., Barrat, S., and Ramel, J.

hasan.salim@uod.ac; azht@usm.my

[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.

(2013). Robust symbol localization based on junction features and efficient geometry consistency checking. In 2013 12th International Conference on Document Analysis and Recognition, pages 1083–1087, Washington, DC, USA.

Pham, T.-A., Delalandre, M., Barrat, S., and Ramel, J.-Y. (2014). Accurate junction detection and characterization in line-drawing images. Pattern Recognition, 47(1):282–295.

Seo, W., Koo, H., and Cho, N. (2014). Junction-based table detection in camera-captured document images. International Journal on Document Analysis and Recognition (IJDAR), 18. pages 47–57.

Wang, Y., Huang, Y., and Huang, W. (2019). Crack junction detection in pavement image using correlation structure analysis and iterative tensor voting. IEEE Access, 7:138094–138109.

Wenyin, L. (2006). The third report of the arc segmentation contest. In Lecture Notes in Computer Science, volume 3926 NCS of Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 358–361, Hong Kong, China. Springer Verlag, Heidelberg.

Yuan, J., Chen, H., and Cao, H. (2015). An efficient junction detection approach for mobile-captured golf scorecard images. Procedia Computer Science. 3rd International Conference on Information Technology and Quantitative Management, ITQM 2015, 55:792–801, Rio De Janeiro, Brazil.

Zhang, W., Sun, C., Breckon, T., and Alshammari, N. (2019). Discrete curvature representations for noise robust image corner detection. IEEE Transactions on Image Processing, 28(9):4444–4459.

Zhang, Y. Y. and Wang, P. S. P. (1996). A parallel thinning algorithm with two-subiteration that generates one-pixel-wide skeletons. In Proceedings of the International Conference on Pattern Recognition, volume 4, pages 457–461, Vienna, Austria.

hasan.salim@uod.ac; azht@usm.my

[1]College of Science, University of Duhok, Kurdistan Region of Iraq, Iraq.