Accurate junction detection and characterization in line-drawing images

The Anh Pham[a,b], Mathieu Delalandre[a], Sabine Barrat[a], Jean-Yves Ramel[a]

phamtheanh@hdu.edu.vn, {mathieu.delalandre, sabine.barrat, jean-yves.ramel}@univ-tours.fr

[a]*Laboratory of Computer Science (LI), Francois Rabelais University, Tours city, France.*
[b]*Hong Duc University, Thanh Hoa city, Vietnam.*

Abstract

In this paper, we present a new approach for junction detection and characterization in line-drawing images. We formulate this problem as searching for optimal meeting points of median lines. In this context, the main contribution of the proposed approach is three-fold. First, a new algorithm for the determination of the support region is presented using the linear least squares technique, making it robust to digitization effects. Second, an efficient algorithm is proposed to detect and conceptually remove all distorted zones, retaining reliable line segments only. These line segments are then locally characterized to form a local structure representation of each crossing zone. Finally, a novel optimization algorithm is presented to reconstruct the junctions. Junction characterization is then simply derived. The proposed approach is very highly robust to common geometry transformations and can resist a satisfactory level of noise/degradation. Furthermore, it works very efficiently in terms of time complexity and requires no prior knowledge of the document content. Extensive evaluations have been performed to validate the proposed approach using other baseline methods. An application of symbol spotting is also provided, demonstrating quite good results.

Keywords:
Junction Detection, Junction Characterization, Dominant Point Detection, Graphical Documents, Line-Drawings

1. Introduction

A *junction* point, by definition in the computer vision (CV) field [1], is formed by the intersection of at least two homogeneous regions. As a result, the junctions are often detected by finding the prominent points in the image at which the boundaries of the adjacent regions meet. The edges meeting at a junction point are regarded as the arms of the junction and are used to characterize junctions into different types such as L-, T-, or X-junctions. Even though much of work for junction detection has been proposed in CV [2, 3, 4, 5, 6], it is difficult to directly apply these methods for the same problem in document image analysis (DIA) for several reasons. First, most of these methods are scale-dependent, and restricted to limited experiments without comparative evaluation with other methods. Few [2, 6] discuss junction characterization, which is very important for junction features. Second, all of these methods are limited to single junction detection. Finally, the conception of a junction in DIA is different from that in CV, making CV methods unsuitable.

In DIA, the junction points are treated as the intersections of at least two line segments and the problem of junction detection is usually formalized as finding the intersections of median lines in images. This point, in fact, constitutes the main challenge of junction detection in DIA as the task of extracting the median lines is not trivial. For these reasons, dedicated methods [7, 8, 9, 10, 11] have been proposed in DIA, in which the junction detection problem has been mainly considered as a post-processing of a vectorization process. Although most of these well-known techniques for junction detection are vectorization-based systems, such methods rely on vectorization, which is known to be sensitive to setting parameters, and present difficulties when heterogeneous primitives (e.g., straight lines, arcs, curves, and circles) appear within a single document. Knowledge about the document content must be included, making the systems less adaptable to heterogeneous corpus.

In this work, we directly address the problem of junction detection by searching for optimal meeting points of median lines. At first glance, it seems that our approach would directly encounter the well-known problem of junction distortion. However, it is important to note that, apart from distorted zones (i.e., the areas where several line segments meet), the median lines are known to be very representative for the rest of line segments. This point suggests that if we can successfully remove the distorted zones, the remaining disjointed strokes would be not subjected to the problem of junction distortion. We therefore present a new algorithm to precisely detect and conceptually remove the distorted zones. The remaining line segments are then locally characterized to form structural representations of the crossing zones. Finally, an optimization algorithm is proposed to reconstruct the junctions.

We review the related work for junction detection in Section 2. The details of the proposed approach are presented in Section 3. Junction characterization and matching are presented in Section 4. A complexity evaluation of the proposed system is

given in Section 5. Experimental results are investigated in Section 6. An application of symbol spotting is provided in Section 7. Key remarks and future works are given in Section 8.

2. Related work

2.1. Junction detection techniques in CV

The methods for junction detection in CV are often classified into two categories: edge grouping and template matching [1]. For the former approaches of *edge grouping*, Bergevin et al. [2] first detect edge points by applying several local oriented-based filters. The detected edge points are then used in conjunction with two criteria (i.e., spatial dispersion and occupancy rate) to generate hypotheses about junction branches from which junction points will be constructed. Deschênes et al. [3] estimate the local curvature at each point and then update the estimated curvatures by propagating the orientation vectors staring from the low curvature end-points. Next, the junctions are identified by extracting the local maxima from the updated curvatures. Köthe [4] introduces an integrated system for edge and junction detection with boundary tensors computed based on the responses of polar separable filters. To improve the performance, higher order filters should be incorporated; however, in this case, it is difficult to design robust filters. Maire et al. [5] first detect contours by combining local and global features to form a *globalized probability of boundary* (gPB). The resulting contours are then used to localize the junctions based on an optimization process. Recently, Xia [6] introduces the concept of meaningful junctions based on the *a contrario* detection theory. Compared to other junction detection methods, this approach requires fewer parameters while providing accurate junctions with respect to a globally visual perception grouping.

Regarding later approaches to *template matching*, Baker et al. [12] argue that most of the features in the real world are parametric, such as edges, corners, junctions, and circles. In their work, a feature model is first represented by a densely sampled parametric manifold in a subspace of Hilbert space. Next, given a local image window, an intensity vector constructed from this window is projected to the subspace yielding a candidate feature point. The candidate points are finally verified using a distance threshold. Parida et al. [1] formalize the junction model as an energy function. A matching process is then performed by minimizing the energy function to yield the best fitted junction parameters. The results applied for several synthesized and real images show the performance of the proposed method to some extent. Sluzek [13] combines local operator and template matching to detect junctions. The local operator is applied to every edge point to compute its 1D profile within a local circular window. These profiles are matched with the junction templates to obtain final junctions. Tabbone et al. [14] carry out an in-depth study of the behavior in the scale space of linear junction models (e.g., L, Y, and X), nonlinear junction models, and linear junction multi-models. Based on these behaviors, they conclude that the extrema of the Laplacian of a Gaussian function can be good starting points for detecting junctions. However, no experiment has been performed to validate their work.

2.2. Junction detection in DIA

In the domain of DIA, most of the proposed methods for junction detection are concerned with the post-processing of vectorization results. These methods address the problem of the correction of the vectorization results to ensure a correct detection of junction points and vectors using contextual information about line drawings. Therefore, the junction detection methodology becomes pre-processing dependent, as the employed data structures and techniques are directly derived from the vectorization step. Junction detection has been addressed in the context of skeleton-based [10, 15, 8], contour-based [16, 17, 18], and direct vectorization [9, 7, 11].

For the skeleton-based approaches, Liu et al. [10] introduce a new set of candidate junction points, computed based on crossing-points obtained from skeletons and dominant points obtained from a polygonal approximation step. A heuristic criterion, called *Criterion A*, is then proposed to correct spurious junctions. The basic idea of *Criterion A* is that two junctions P_1 and P_2 are merged if we can find a sufficiently long straight line segment P^*P_b fully included in the black region of the image. The authors suggest choosing P^* from those points on the branch P_1P_2 and choosing P_b from those points on every branch of P_1 or P_2 (except P_1P_2). Although the authors proved the efficiency and accuracy of this criterion on a Chinese character dataset, our experiments carried out on Kanungo noise symbols have shown that it performs poorly and often produces false merging of the junctions.

Hilaire et al. [15] detect junctions based on topological correction of vectorization results. In their work, every skeleton segment is classified into either short or long primitive by simply comparing its length with the local line-thickness. Next, the long primitives are clustered into different groups by calculating the intersection zones from the uncertainty domains. Finally, junction points are reconstructed from the primitives in each group. This work, as discussed by the authors themselves, has several weaknesses, including being time-consuming, being sensitive to interrupted patterns, and featuring the ambiguous step of merging junction points. A further extension of this work has been made in [8], where the main improvement relies on the process of skeleton optimization. The improved algorithm traverses all possible paths in a connectivity graph, starting from a long segment and leading to either another long segment or the last segment of a sequence of the short ones. A heuristic rule, which is quite similar in spirit to *Criterion A*, is designed to check whether the first segment could join the final one of the considering path. Using this rule, it is agreed that multiple solutions could be outputted by the proposed system.

To avoid the distortion problem resulting from the skeletonization step, some works [17, 18] detect junctions from contour-based vectorization. In [17], the lines are tracked using two contour followers on opposite sides of each line. When arriving at a junction location, the tracking algorithm detects the relevant branches by constructing a series of circles centralized at the current interrupted point with increasing radii. The intersections between these circles and the contours of the line structures are used to identify the branches involved at the considering junction zone. Next, the user is asked to determine the

branches for the continuation of the line tracking process. This approach requires the intervention of the user in both seed initializing and junction handling. In addition, it is sensitive to contour noises and the tracking length is difficult to maintain properly. The work in [18] formulates the junction detection problem as an identification of relations (i.e., intersection, succession, parallelism) between contour primitives (i.e., quadrilaterals). Such an approach is therefore quite sensitive to contour noises and presents some difficulty in terms of correctly interpreting the pair of matched contours.

To avoid the problems resulting from skeletonization- and contour-based vectorization, direct vectorization methods have been proposed in [7, 9, 11]. Dori et al. [7] propose a direct vectorization system based on pixel-tracking called sparse pixel vectorization (SPV). During the tracking, the widths of the tracked lines are estimated and compared to a threshold. If the width run conflicts the width preservation, a junction recovery process is triggered. The junction recovery process works as an iterative search along the junction area, composed of three steps: reverting back to the last medial point, updating the tracking step length with half the length of the previous step, and exploring the new tracking position. The authors in [9] propose a region-based vectorization system called *maximal inscribing circle* (MIC). A MIC is defined as a maximal-radius circle fitting inside a line segment in the sense that it has at least two contact points with the border of this line segment. The line segments are then tracked using the MIC(s), followed by a labeling algorithm to analyze different detected lines, identify junction zones, and construct spatial relations among them. A final approach is related to an object-oriented vectorization system [11]. In this method, a raster image is progressively simplified by erasing recognized bars (e.g., lines, circles) to eliminate their interference with subsequent recognition. However, the deletion operator at junction zones is not trivial. At such a junction zone, the authors suggest detecting the contours of the branches involved in this zone and then analyzing their trends to calculate the part of the junction zone to be preserved. No additional suggestions for improving this process are presented.

In summary, the common major problems with direct vectorization are the tracking initialization and reading order. Direct vectorization is a recursive process, wherein next tracking steps are initiated from the previous ones. Depending on how the system drives the tracking, detection results can change significantly. In addition, as we have mentioned before, these methods are sensitive to the error-prone caused by the vectorization process. The achievements above convince us that a robust approach for junction detection in DIA is possible. In this paper, we attempt to create such an approach with the following major features: junction distortion avoidance, accurate junction detection, efficiency, and robustness. We will justify these features in the following sections.

3. The proposed approach

We directly formulate the problem of junction detection as searching for optimal meeting points of line-like primitives from input images. However, as it is impossible to obtain ideal line primitives (i.e., 1-pixel-thick lines) from a digitalization process, the intersection areas of the line primitives can not converge or contract to one pixel as expected. Therefore, to achieve exact junction localization, the line primitives must be represented in suitable forms that facilitate the step of finding their intersections. Apart from the major drawback of junction distortion, median axis lines have been known to be very good representations of such line primitives. Naturally, if we can identify and remove all distorted zones, the remaining line segments could be well represented as the mean of the median lines with little (or even without) disturbance concerning with the issue of junction distortion. We therefore develop our approach based on this idea.

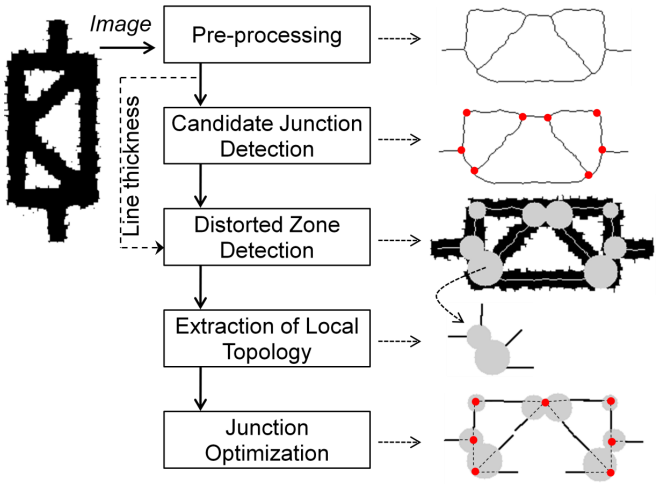

Figure 1: Overview of the proposed approach.

It is noted that similar ideas have been explored in the literature, e.g., [19, 15, 8]. However, the approaches taken in those works are quite different from ours. In particularly, the works in [15, 8] partition skeletons into reliable and unreliable line segments relying solely on a single line-thickness criterion: if the length of one segment is less than the line thickness, it is considered to be a short segment and *vice versa*. The short segments are treated as unreliable segments and thus removed. Using such a threshold is too vulnerable and would lead to many mis-classifications among unreliable/reliable segments. Fan et al. [19] partition images into irregular regions and regular bars; however, their approach is based on run-length tracking, which is sensitive to slanted bars and complicated crossing areas.

In our approach, a distorted zone is identified by addressing two probing questions: where such a zone is likely to appear and how large this distortion zone would be. Naturally, the distorted zones occur at crossing locations and these zones would be restricted to small areas fitting inside the crossing structures (see Figure 1). The former fact indicates that we could make use of *candidate* junctions to determine the locations of distorted zones, whereas the latter observation suggests that a maximal inscribing circle [9] would be useful to determine the areas of the distorted zones. Next, the reliable segments are extracted and then locally characterized to form a local structure repre-

sentation of every distorted zone. Accurate junction localization and characterization are then achieved based on a novel junction optimization algorithm. The overall approach is outlined in Figure 1.

3.1. Pre-processing

Our method is applied to binary images. These images could be obtained following some enhancement processes, such as noise filtering and binarization, depending on the specific application. Next, the median lines are pre-extracted using the technique presented in [20] because it is probably argued as one of the most robust techniques for skeleton extraction in the literature. In addition, this method is time-efficient, supporting all our processes at a low time cost.

3.2. Detection of Candidate Junctions

In the second stage, candidate junction points are detected from the median lines extracted previously. These candidate junctions, in combination with the line thickness information, are used to detect distorted zones and drive our junction optimization process in the next stages of our system (see Figure 1). In our case, the candidate junctions are classified into 2-junctions and n-junctions (i.e., the junctions formed by n arms with $n \geq 3$). The n-junction candidates are easily extracted by detecting the crossing-points obtained from the skeletonization step.

The 2-junction candidates are detected as the dominant points of the median lines. A large number of works in the literature have addressed the problem of dominant point detection from digital curves. Recent surveys of this work are presented in [21, 22]. In general, the approaches for dominant point detection are categorized into two classes: multi-scale and single-scale. Although the former approaches [23, 24] have been known to be robust to noise, they often produce high false positive rates. This matter is realized based on the fact that curve smoothing and curvature estimation are two of the most critical stages of a dominant point detector [21, 25]. However, the choice of an appropriate smoothing scale is not trivial and the use of a multi-scale framework to smooth the curve does not solve the problem of scale selection. For the single-scale-based approaches [26, 27, 25], the major challenge is the determination of the *region of support* (ROS) or *local scale*. In our work, the candidate 2-junctions are detected by exploiting Teh-Chin's method [25] with a major change in the step of determination of ROS. The key idea in Teh-Chin's work relies on the observation that ROS could be determined by measuring the sudden change of the chord length: given a point p_i of a digital curve, the chord length l_k is defined as the Euclidean distance from p_{i-k} to p_{i+k}, given a parameter $k > 0$. By applying the Teh-Chin's criterion, the ROS at p_i is determined as: $ROS(p_i) = k$ if $l_k \geq l_{k+1}$ and $l_{j-1} < l_j$ for any j: $1 \leq j \leq k$. This criterion could be useful for continuous curves but is fragile in the case of digital curves. Figure 2 shows several examples in which that Teh-Chin's method fails to correctly determine the ROS.

Our solution to the problem of ROS determination relies on the observation that for every point p_i of a curve, there exists a

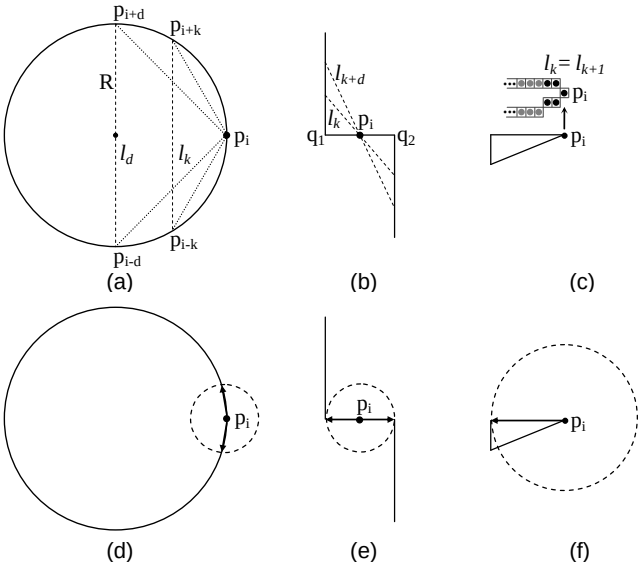

Figure 2: Some examples in which Teh-Chin's technique fails to correctly determine the ROS. Top row: (a) $ROS(p_i) = d$ for any point p_i on a circle with radius R, where $l_d = 2R$; (b) $ROS(p_i) = +\infty$ (i.e., p_i is the middle point of segment q_1q_2 and thus $l_{k+d} > l_k$ for any $k, d > 0$); (c) $ROS(p_i) = 1$ (i.e., $l_{k+1} = l_k$). Bottom row: expected ROS for each case in the top row.

trailing line segment (i.e., the segment composing of the points $\{p_i, p_{i-1}, \ldots, p_{i-k_t}\}$) and a leading line segment (i.e., the segment composing of the points $\{p_i, p_{i+1}, \ldots, p_{i+k_l}\}$), where $k_l > 0$ and $k_t > 0$, such that both line segments together constitute a meaningful view of that point regardless of how smooth the curve is. This observation is especially true at dominant points on the curve, where a dominant point is usually treated as the point at which two edges meet and form a vertex. This fact suggests that the ROS of a point could be determined by finding the straight lines fitted to the leading and trailing segments of that point. It turns out that this task could be efficiently accomplished using linear least squares (*LLS*) line fitting technique. In the proposed approach, given a curve consisting of N ordered points $p_1, p_2, \ldots, p_N$, the ROS at a point p_i is determined as follows:

- Step 1: Start with $k_l = 1$ and gradually increase k_l in increments of one to estimate the straight line d_f of the form $y = \alpha + \beta x$, which provides the best fit for the points $\{p_i, p_{i+1}, \ldots, p_{i+k_l}\}$. The parameters α and β are derived by minimizing the following objective function:

$$Q(\alpha, \beta) = \sum_{j=i}^{i+k_l} (y_j - \alpha - \beta x_j)^2 \tag{1}$$

  Next, we define the distance error $h(p_j, d_f)$ computed as the Euclidean distance from a point $p_j(x_j, y_j)$ to the straight line d_f as follows:

$$h(p_j, d_f) = \frac{|\beta x_j - y_j + \alpha|}{\sqrt{\beta^2 + 1}} \tag{2}$$

The step of searching the local scale on the leading segment of p_i will be terminated at some point p_{i+k_l} if either of the two following conditions is satisfied:

$$\frac{1}{k_l}\sum_{j=i}^{i+k_l} h(p_j, d_f) \geq E_{min} \quad (3)$$

$$h(p_{i+k_l}, d_f) \geq E_{max} \quad (4)$$

The first condition (i.e., Condition (3)) requires that the average distance error associated to the fitting line is less than E_{min} pixels. The second condition (i.e., Condition (4)) is designed to limit the maximum distance error from a point p_j to d_f: no point is E_{max} pixels away from d_f ($E_{max} > E_{min}$). The value $s_l = k_l - 1$ is then treated as the local scale on the leading segment of p_i.

- Step 2: Repeat Step 1 to find the optimal scale $s_t = k_t - 1$ with respect to the trailing segment $\{p_i, p_{i-1}, \ldots, p_{i-k_t}\}$.
- Step 3: The ROS of p_i is finally computed as: $ROS(p_i) = Min(s_t, s_l)$.

Our empirical investigation showed that the values of E_{min} and E_{max} produce a negligible impact on the detection rate provided that $E_{min} \in [1.2, 2.0]$ and $E_{max} \in [1.5, 3.0]$. In our implementation, we fixed the following setting for all the experiments: $E_{min} = 1.3$ and $E_{max} = 1.8$. Once the ROS is determined, we then apply Teh-Chin's algorithm to detect the dominant points from skeleton branches. Figure 3 shows the dominant points and the corresponding ROS(s) detected from an image. The detected points, in combination with crossing-points (i.e., skeleton points with at least three 8-connected neighbors), are treated as the candidate junctions and will be used to detect distorted zones in the next stage.

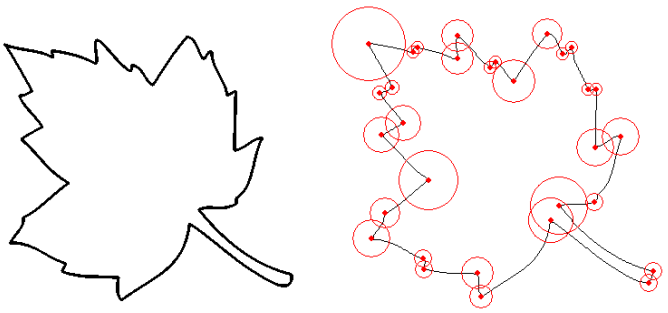

Figure 3: Left: an original image; Right: the detected dominant points (small dots) and the corresponding local scales (small circles).

3.3. Distorted Zone Detection

The candidate junction points we have detected previously are used in conjunction with the line thickness information to first detect distorted zones and then conceptually remove these distorted zones to eliminate their interference in terms of the distortion of median lines. In our approach, a distorted zone is identified by tackling two probing questions: where is such a zone likely to occur and how large this distortion zone would be. Naturally, the distorted zones occur at junction locations, and these zones would be restricted to small areas fitting inside the crossing structures. Furthermore, line thickness is also one of main causes of skeleton/junction distortion (i.e., *thin* objects are not or weakly subjected to skeleton distortion). Relying on these observations, the distorted zones could be easily identified by using the information of the line thickness at the candidate junction points detected in the previous steps. More precisely, we define a distorted zone Z_J for a given candidate junction point J as the area constructed by a circle centered at J whose diameter equal to the local line thickness computed at J. This definition is actually a variation of the maximal inscribing circle, as presented in [9]. By making use of line thickness information, these maximal inscribing circles are easily determined with a high degree of accuracy. We call several distorted zones that intersect together a connected component distorted zone (*CCDZ*). Once the CCDZ(s) have been detected, the skeleton segments lying inside these zones are treated as distorted segments and thus removed. From this point, our subsequent stage of junction reconstruction proceeds based on the reliable line segments only. Figure 4 (left) shows the reliable segments remaining after removing all distorted zones (marked as gray connected components).

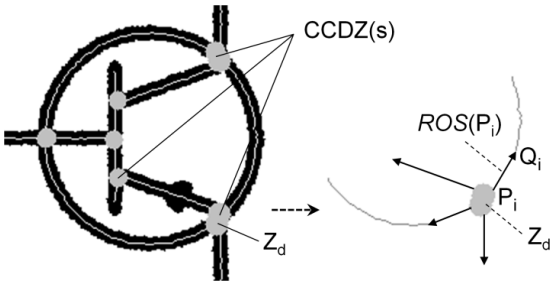

Figure 4: Left: an input image with the detected *CCDZs* (gray connected components) and reliable line segments (thin white lines). Right: the local topology defined for a *CCDZ*.

3.4. Junction Reconstruction

The junction reconstruction exploits candidate junction points to remove possible false alarms, merge candidate junction points, and correct final junction locations. This reconstruction is initiated in a first step by extracting local topologies, corresponding to sets of segments belonging to the same distorted zone or set of intersecting distorted zones. These local topologies will drive a second step in our junction optimization process. We will present these two steps in the following subsections.

3.4.1. Extraction of local topology

This step defines and constructs the local topology at each *CCDZ*. In particularly, given a *CCDZ*, its local topology is defined as the set of local lines segments, $\{P_iQ_i\}_{i=1,\ldots,n}$, stemming from this *CCDZ*. That is, for each reliable skeleton segment

stemming from a *CCDZ*, we characterize the first part of this segment by a local line segment starting from the extremity linked to the *CCDZ*. By defining and analyzing these local geometry topologies, we have significantly simplified the complexity of the objects of input images; thus, the proposed approach is able to work on any type of shape rather than straight lines and/or arc primitives exclusively. Moreover, this step can be performed efficiently by reusing the results of the ROS determination stage applied at each extremity of each reliable skeleton segment stemming from the *CCDZ*. As a result, for each *CCDZ*, we obtain a list of local line segments describing its local geometry topology. In addition to these local lines, the foreground pixels lying inside the *CCDZ* are also recorded for use as a local search neighborhood for the subsequent step of junction optimization. In summary, the local topology associated with a *CCDZ* is now represented by a list of n local lines, $\{P_iQ_i\}_{i=1,\dots,n}$, and a set, Z_d, containing the foreground pixels located inside the *CCDZ*. Figure 4 (right) illustrates a local topology extracted for a *CCDZ*.

3.4.2. Junction optimization

The goal of this step is to reconstruct the junction points for a specific *CCDZ* represented by n line segments $\{P_iQ_i\}_{i=1,\dots,n}$ and a set, Z_d, of foreground pixels lying inside the *CCDZ*. We accomplish this goal by clustering the line segments into different groups such that the clustered lines in each group will be used to form a junction point. Concerning this problem of clustering segments, the authors in [15], as discussed above, calculated the intersection zones from the uncertainty domains of the long primitives. This approach is subjected to the constraint that each primitive is allowed to be clustered in one group only, increasing the difficulty of the subsequent junction linking step. Another approach to segment clustering was presented in [5] based on the idea that if we know the position of the junction, the associated line segments passing through this junction could be easily identified and *vice versa*. However, this work assumed that each neighborhood (see Figure 5) contains one junction only, and this approach is subjected to high computation load because the optimization step must include sufficiently large neighborhoods likely containing junctions to reduce the error introduced by the previous step of contour detection. In addition, the reweighting step does not consider the weights accumulated during the previous iterations. This omission may lead to incorrect convergence, as shown in Figure 5.

We therefore develop an integrated solution for both clustering and optimizing tasks to address the aforementioned weaknesses, described below. In particularly, the proposed algorithm is able to handle the following issues simultaneously: (1) each neighborhood can contain multiple junctions, (2) each line segment can be clustered into more than one group, and (3) junction linking and characterization are automatically derived.

The key spirit behind our algorithm is as follows. Starting from a *CCDZ* (e.g., Figure 7(c)), a new junction point is constructed by iteratively searching for a foreground pixel of the *CCDZ* such that the distance error, computed as the sum of weighted Euclidean distances from this pixel to the line segments of the *CCDZ*, is minimized (e.g., Figure 7(d)). To achieve

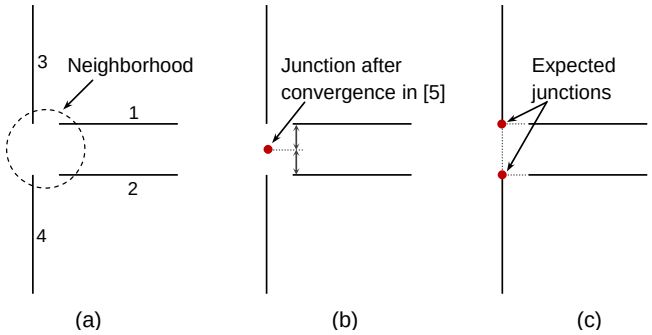

Figure 5: Incorrect convergence of the junction optimization step in [5]: (a) four line segments with the same length; (b) the detected junction, which is the same distance from the lines 1 and 2; (c) expected junctions.

this goal, each line segment has to be assigned with a proper weight in the sense that the lines, which are close to the junction, would have higher weights than the ones away from the junction. This implies that the weights are set mainly relying on the distances from the junction to the lines. In practice, a smooth function (e.g., Gaussian function) should be used to update the weights. As the algorithm evolves, the junction tends to converge towards the lines with higher weights and move away from the lines with lower weights. Hence, the junction would converge to a fixed point after several iterations (e.g., Figure 7(d)). Once a newly optimized junction is derived, the weights are re-assigned in such a way that higher weights are given to the lines which have not been involved in constructing the junctions previously. The optimization process is then repeated to find a new junction (e.g., Figure 7(e, f)). This continues until every line segment has been participated in constructing at least one junction. A final post-processing step is then applied to make the topology of the obtained junctions be consistent (e.g., Figure 7(g)). The proposed algorithm works as follows.

Let w_i be a weight assigned to the line segment P_iQ_i with $1 \le i \le n$, and Ω_J be a set of optimal junctions found during the optimization process. At the beginning, $\Omega_J \leftarrow \emptyset$ and $w_i \leftarrow |P_iQ_i|$, and all line segments $\{P_iQ_i\}$ are marked as *unvisited*. The followings are the main steps of the proposed algorithm with respect to those in Figure 6.

- Step 1: Search for an optimal junction J:

$$J = \arg\min_{J \in Z_d} \{ \sum_{i=1}^{n} w_i \cdot d(J, P_iQ_i) \} \tag{5}$$

  where $d(J, P_iQ_i)$ is the Euclidean distance from J to P_iQ_i.

- Step 2: Update the weights $\{w_i\}$:

$$w_i = w_i \cdot \exp\left(\frac{-\pi \cdot d(J, P_iQ_i)^2}{S_{CCDZ}}\right) \tag{6}$$

  where S_{CCDZ} is the area of the *CCDZ*.

- Step 3: Enact a penalty (i.e., smaller weight) for the line segment farthest from J: $w_{imax} = \frac{w_{imax}}{\tau}$, where $imax$ =

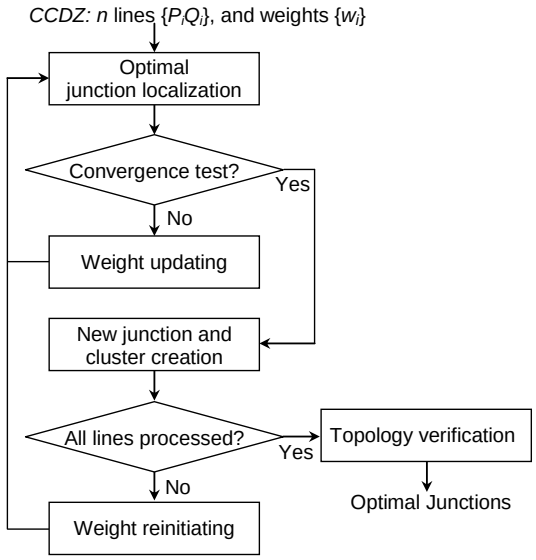

Figure 6: Outline of our junction optimization algorithm.

$\arg\max_i\{d(J, P_iQ_i)\}$ and $\tau > 1$. If there are several line segments having the same greatest distance from J, one is randomly selected to assign a penalty.

- Step 4: Repeat steps $\{1, 2, 3\}$ until J converges to a fixed point or a given number of iterations has been reached. Then, insert the newly obtained junction to Ω_J: $\Omega_J \leftarrow \Omega_J \cup \{J\}$, and go to Step 5.

- Step 5: Determine the line segments that pass through the junction J and mark them as *visited*. A new cluster is constructed corresponding to these line segments. If all line segments have been marked as visited, go to Step 7. Otherwise, go to Step 6 to look for other junctions.

- Step 6: Reinitiate the weights: $w_i = 1$ if the label of P_iQ_i is *visited*; otherwise:

$$w_i = \prod_{k=1}^{L} \exp\left(\frac{\pi \cdot d(J_k, P_iQ_i)^2}{S_{CCDZ}}\right) \tag{7}$$

  where L is the number of times that steps $\{1, 2, 3, 4, 5\}$ have been fulfilled, and J_k is the optimal junction found during the corresponding cycle. Return to Step 1.

- Step 7: Topology consistency verification by resetting the weights: $w_i = 1$ if the line P_iQ_i is involved in only one cluster; otherwise $w_i = w_\infty = K \cdot \sqrt{H^2 + W^2}$, where W and H are the width and height of input image respectively, and $K = |\Omega_J|$. Then, apply Step 1 to the line segments in each cluster to obtain the final junctions. In this way, the lines assigned with the weight w_∞ are fixed in one place.

The distance error minimization in Step 1 has been used in several works [15, 5]. Hilaire et al. [15] employed least squares error minimization to find the optimal position of the junction, but this process is performed separately from segment clustering. Maire et al. [5] developed this idea by incorporating a reweighting step like that in Step 2 but differing in that it does not incorporate the weights accumulated during the previous iterations and requires a training step to empirically derive a parameter controlling the decay of distance tolerance.

Our investigation has shown that Step 1 will quickly converge to the optimal junction if the weights are updated taking into account the weights derived during the previous iterations. In addition, we avoid the training step by normalizing the distances, $d(J, P_iQ_i)$, based on the area of the *CCDZ* (i.e., the factor $\pi \cdot d(J, P_iQ_i)^2/S_{CCDZ}$ is equal to the ratio of the area constructed by a circle with radius of $d(J, P_iQ_i)$ centralized at J and the area of *CCDZ*). Step 3 enacts a penalty (the parameter $\tau = 2$ in our implementation) for the line segment farthest from J. If there are several line segments with the same greatest distance from J, one is randomly selected to assign a penalty. This step is used to allow the optimization process to quickly converge to a correct junction location. More importantly, it acts as a *trigger* to break the balance state or incorrect convergence, if any, as discussed in Figure 5. Note that if one line segment is penalized, it does not imply that this line segment will not pass through the *latest* optimal junction.

Step 4 is used to repeat the three steps above until the optimal junction is found. The obtained junction is then added to the set Ω_J (e.g., the junction J_1 in Figure 7(d)). Next, the line segments that actually form this junction are determined by looking for the lines whose distances from the detected junction tend to form a monotonically decreasing order (e.g., the lines $\{1, 3, 4\}$ in Figure 7(d)). This is accomplished because at each iteration, the optimal junction would converge towards the lines with higher weights and move away from the lines with lower weights. Therefore, the distances from the optimal junction in each iteration to the line segments are recorded and then used to determine the real lines passing the most recently found junction. The obtained lines are then associated to a new cluster, and marked as visited (i.e., already involved in at least one junction). If all lines have participated in constructing at least one junction, the algorithm terminates after checking topology consistency in Step 7. Otherwise, Step 6 is invoked to initiate a new cycle to find other junctions (i.e., a cycle is composed of the first five steps $\{1, 2, 3, 4, 5\}$ to completely find a new optimal junction).

In Step 6, the weights are reinitiated such that more priority or higher weights are given to the lines that have not yet been involved in junction construction (e.g., the lines $\{2, 5\}$ in Figure 7(d)). To this end, the recorded distances that violate the monotonic decrease are used to accumulate the weights for the lines. In this way, these lines increasingly gain weight, and at the end, when the weights are large enough, the optimization process (i.e., steps $\{1, 2, 3, 4\}$) will be driven by these weights, leading to a new junction convergence at the corresponding lines (e.g., J_2 in Figure 7(e) and J_3 in Figure 7(f)).

Step 7 is aimed at verifying the topology consistency of all line segments in the obtained clusters. At this time, we obtain K clusters, each containing one optimal junction. As one line segment, say P_iQ_i, can be clustered in several groups (e.g., the

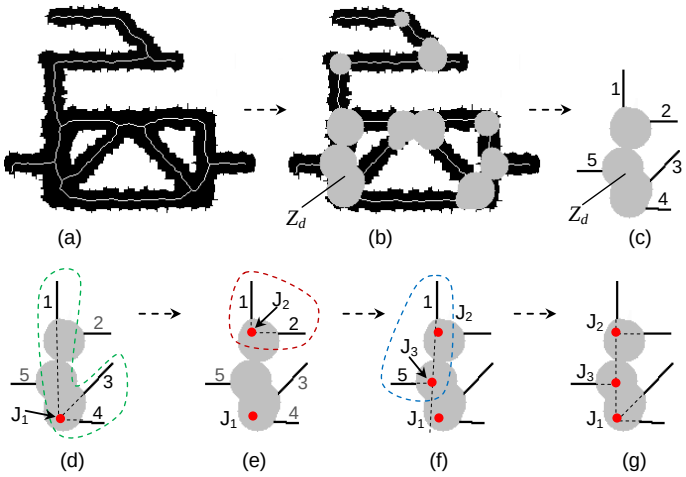

Figure 7: (a) An image with its skeleton; (b) the *CCDZ*(s) and the reliable line segments; (c) the local topology configuration extracted for one *CCDZ* (marked by Z_d); (d) the first cycle of steps $\{1, 2, 3, 4, 5\}$: the junction J_1 is found corresponding to the cluster containing line segments $\{1, 3, 4\}$; (e) the second cycle: the junction J_2 is found corresponding to the second cluster comprised of lines $\{1, 2\}$; (f) the third cycle: the junction J_3 is found corresponding to the third cluster of $\{1, 5\}$; (g) topology correction for three junctions.

line 1 in Figure 7(f)) and there is no warranty that all the optimal junctions in these groups will form a straight line that fully contains P_iQ_i, such situations must therefore be identified and corrected. This step could be easily processed by setting a large enough weight for the line P_iQ_i and then performing Step 1 once for each cluster. In this way, a small change in the distance error computed from each point $J \in Z_d$ to the line P_iQ_i will cause a large change in the objective function in Step 1. The line P_iQ_i is thus fixed in one place, and the new optimal junctions found in the clusters, in which P_iQ_i is involved, become consistent (Figure 7(g)). Figure 7 demonstrates the steps of our junction optimization algorithm, and Figure 8 shows all the detected junctions and the corresponding local scales.

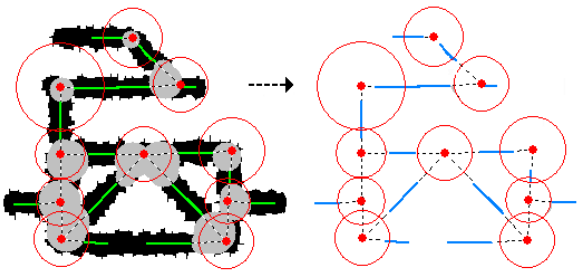

Figure 8: Detected junctions (red dots) and local scales (circles).

4. Junction characterization

One of the main advantages of our junction reconstruction process is that the detected junctions could be automatically characterized and classified into different types, such as T-, L-, and X-junctions. More generally, we wish to characterize any complicated junctions in the same manner based on the arms

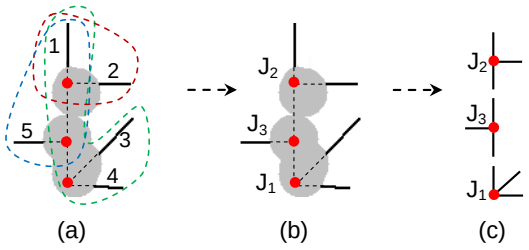

Figure 9: (a) A *CCDZ* cropped from Figure 7 with the superposition of three clusters; (b) detected junctions $\{J_1, J_2, J_3\}$; (c) the detected junctions are classified as a 3-junction even though J_2 and J_3 are constructed from two clusters, $\{1, 2\}$ and $\{1, 5\}$, respectively, each of which only contains two local line segments.

forming the junction. In our case, as each junction point is constructed from the local line segments of one group, we can consider these line segments as the arms of the junction point. However, as each *CCDZ* can contain multiple junctions and each local line segment of the *CCDZ* can participate in several groups, the exact arms of a junction could therefore be greater than the line segments forming this junction (Figure 9). Given a local topology represented by n straight line segments $\{P_iQ_i\}$ with $1 \le i \le n$, the process of determination of the exact arms of each junction is as follows:

- Let O_J be a set of arms of junction J where $O_J \leftarrow \emptyset$ at the beginning for every junction.
- If the line segment P_iQ_i is clustered in a group whose an optimal junction J is then constructed, the line P_iQ_i is considered as one of the arms of the junction J: $O_J \leftarrow O_J \cup \{P_iQ_i\}$.
- For each line segment P_iQ_i that is clustered in several groups, the corresponding junctions involved in P_iQ_i are sorted in the order of increasing distance to P_i. Then, for each junction J *except the last one* in the list, the corresponding set O_J is updated as: $O_J \leftarrow O_J \cup \{JG\}$, where JG is a straight line segment constructed at J with the same length as P_iQ_i but the point G lies in the opposite direction of vector $\vec{P_iQ_i}$.

Once we have correctly determined the arms of each junction, the junction characterization could easily be accomplished as follows. Given a junction J associated with a set of m arms $\{U_iV_i\}_{i=0,\ldots,m-1}$, the characterization of this junction is described as $\{p, s_p, \{\theta_i^p\}_{i=0}^{m-1}\}$, where:

- p is the location of J, and s_p is the local scale computed as the mean length of the arms of J;
- θ_i^p is the difference in degrees between two consecutive arms U_iV_i and $U_{i+1}V_{i+1}$. These parameters $\{\theta_i^p\}_{i=0}^{m-1}$ are tracked in the counterclockwise direction and the θ_{m-1}^p is the difference in degrees between the arms $U_{m-1}V_{m-1}$ and U_0V_0.

The description of each junction point derived in this way is rather compact, distinctive, and general. The dimension of

this descriptor is limited to the number of arms of each junction point, and in practice, this value is quite small (e.g., 3 for a T-junction, 4 for an X-junction). This point constitutes a great advantage of the detected junctions that provides a very efficient approach to the subsequent task of junction matching. In addition, the junction descriptor is distinctive and general, such that we can describe any junction points appearing in a variety of complex and heterogeneous documents. After this step, junction matching can be performed by simply comparing the descriptors of two junctions. Figure 10 shows the corresponding matches of the junctions detected in a query symbol (left) and those of an image cropped from a large document (right). For simplicity, the matches are shown after performing geometry checking.

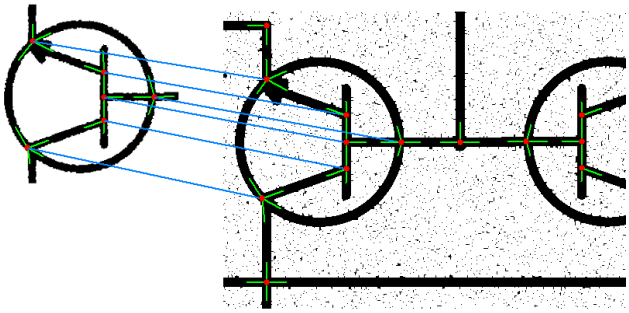

Figure 10: Corresponding junction matches between a query symbol (left) and a cropped document (right).

5. Complexity Evaluation

In this section, we provide a detailed analysis of the complexity of the proposed method given an image I of the size $M \times N$. In the pre-processing stage, before applying the (3,4)-distance transform skeletonization algorithm, several basic pre-processing steps, such as hole filling, small contour removing, and image dilation, are performed, as discussed in the original work of Di Baja [20]. Such steps can be processed in parallel using one scan over the image. The skeletonization step is then applied, which requires two scans of the image to calculate the (3,4)-chamfer distance. In summary, the computation complexity for the pre-processing stage is basically linear (i.e., $O(MN)$).

In the next stage of scale selection and dominant point detection, the ROS determination step is applied for each skeleton point, thus using a single loop of length S, where S is the number of skeleton points. The technique of least squares line fitting is a second-order linear computation of the length that it traverses. In practice, it is not necessary to traverse a full skeleton branch; rather, a short path of the branch with a length $k_\rho = 50$ (pixels) may be sufficient, for example. As the technique of least squares line fitting is performed in both directions at each point, it is equal to a complexity of $O(2Sk_\rho^2)$ in total for this step. The 2-junctions are then detected as dominant points by applying Teh-Chin's algorithm, which is a sequential 4-pass process, where the first pass is performed on the full length of the median lines to detect a list of H candidate dominant points and the other passes are conducted on one of these candidate points, where H is much smaller than S. Furthermore, the crossing-points could be detected in parallel with a cost of first-order linear polynomial time $O(S)$. The overall computation complexity for these processes is essentially linear to the length of the median points (i.e., $O(Sk_\rho^2)$).

For the last stage of junction reconstruction, let K be the number of candidate junctions comprised of 2-junction points and crossing-points. The distorted zone Z_i defined at each candidate junction p_i ($1 \le i \le K$) has an area of $\pi r_i^2/4$ where r_i is the line thickness at p_i. Given a distorted zone Z_i, the maximum complexity of the computation to find an optimal junction in Z_i is $O(T\pi r_i^2/4)$ where the first factor, T, is the number of times that steps $\{1, 2, 3\}$ are repeated and the second factor, $\pi r_i^2/4$, is the number of foreground pixels in Z_i (i.e., the local searching neighborhood). Our investigation has shown that the number of iterations, T, is very small and is typically less than 10. As Z_i can contain multiple junctions, say L junctions, it implies that the junction optimization process applied to Z_i will be terminated after running L iterations of the steps $\{1, 2, 3, 4, 5\}$. The value of L is also very small in practice, often 2; thus, for a wide range of situations, we have set $L = 5$ in our implementation. Overall, the maximum complexity of computation for this stage, applied to K distorted zones, is $O(KLTr^2)$ where r is the average line thickness of the image I. In other words, this stage is linear time complexity for the areas of the distorted zones. Note that the distorted zones, in practice, could intersect, resulting in connected component distorted zones and making the searching areas much smaller.

6. Experimental results

6.1. Evaluation Metric and Protocol

We use *repeatability* criterion to evaluate the performance of our junction detector because this criterion is standard for the performance characterization of local keypoint detectors in CV [28]. This criterion works as follows. Given a reference image I_{ref} and a test image I_{test} taken under different transformations (e.g., noise, rotation, scaling) from I_{ref}, the repeatability criterion signifies that the local features detected in I_{ref} should be repeated in I_{test} with some small error ϵ in location. We denote $D(I_{ref}, I_{test}, \epsilon)$ as the set of points in I_{ref} that are successfully detected in I_{test} in the sense that for each point $p \in D(I_{ref}, I_{test}, \epsilon)$, there exists at least one corresponding point $q \in I_{test}$ such that $distance(p, q) \le \epsilon$. Let n_r and n_t be the number of keypoints detected by one detector from I_{ref} and I_{test}, respectively. The repeatability score of this detector applied to the pair (I_{ref}, I_{test}) is computed as follows:

$$r(I_{ref}, I_{test}, \epsilon) = \frac{|D(I_{ref}, I_{test}, \epsilon)|}{Mean(n_r, n_t)} \tag{8}$$

Our strategy to perform evaluation in the experiments is as follows. We first apply each detector to reference images and test images in each dataset to obtain reference junctions (S_r)

and detected junctions (S_t), respectively. Then, groundtruth information is used to compute repeatability scores of this detector from two sets of junctions: S_r and S_t. The average score on the entire dataset is finally outputted. We vary the value of parameter ϵ in the range of $[1, 8]$ to obtain a *ROC-like* curve of the repeatability score.

6.2. Baseline Methods

To compare the proposed system with other methods, we have selected two baseline systems [8, 10] dedicated to junction and fork-point detection. The work of Liu et al. [10] is dedicated to fork-point detection in handwritten Chinese characters, whereas the work of Hilaire et al. [8] is a vectorization-based system for line-drawings. We wish to highlight that although the work in [8] is designed for vectorization, the major contribution in this work is the process of skeleton optimization to correct skeletons and reconstruct junctions. As the implementations of these works are not publicly available, we have developed our own implementation for these two systems[1]. For each system, several running trials have been performed to select the best parameter settings, and only junctions or fork-points are compared with our detected junctions. It is worth noting that we have applied the same pre-processing steps for all three systems and used the same parameter settings in all the experiments.

6.3. Datasets

The datasets for experiments have been selected from the Symbol Recognition Contest in GREC2011 (SymRec-GREC11)[2], the low resolution diagram dataset SESYD[3], and the UMD Logo Database of University of Maryland, Laboratory for Language and Media Processing (LAMP)[4]. The SymRecGREC11 dataset is composed of 4 folders: *setA*, *setB*, *setC*, and *setD*. *SetA* and *setB* are excluded from our experiments because *setC* is the largest set, with 150 model symbols and 7500 test images covering all degradations included in *setA* and *setB*. *SetD* is very different from *setC* in terms of noise disturbance. Instead of using traditional Kanungo noise, the *setD* is disturbed by context noises (i.e., symbols cropped from full line-drawing images). The diagram SESYD dataset contains 100 reference images and 936 test images by applying four levels of low resolution, corresponding to the scaling factors $\{1/2, 1/4, 1/8, 1/16\}$, and using a compression scheme (JPG) from gray-level images. In addition, for the evaluation of single parameter changes (i.e., rotation and scaling), we have used 150 model symbols from GREC2011 to generate 1339 test images taken under different levels of rotation (e.g., from 10^0 to 90^0) and 1200 test images taken under different scaling factors (i.e., $\{1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0\}$). The UMD Logo Database consists of 104 model logos, which have been used to generate 1272 test images by applying a combination of Kanungo noise and geometric transformations.

[1]The source codes for the three systems and the demonstration of our junction detector and symbol spotting are publicly available at https://sites.google.com/site/ourjunctiondemo/

[2]http://iapr-tc10.univ-lr.fr/index.php/final-test-description

[3]http://mathieu.delalandre.free.fr/projects/sesyd/

[4]http://lampsrv02.umiacs.umd.edu/projdb/project.php?id=47

6.4. Comparative Results

6.4.1. Evaluation of rotation and scaling change

In this experiment, the repeatability scores are computed over single parameter changes while the location error is fixed at 4 (pixels). Table 1 presents the effect of rotation change for three systems. It can be noticed that the proposed approach far outperforms (almost 25%) the two other systems and that our repeatability scores tend to be quite stable and almost over 88% when varying the rotation parameter. These results confirm that the proposed system is very robust to rotation change. The systems of Liu et al. and Hilaire et al. are theoretically rotation invariant; however, the results reported here show that these systems are less adaptive to rotation change under real-world conditions. In the context of scaling change, as shown in Table 2, the same situation is repeated for the baseline methods, whereas the proposed system still strongly outperforms the others. It is noticed, in particularly, that the results obtained by the system of Hilaire et al. are significantly degraded when increasing the scaling factor. This degradation could be due to the two baseline systems being quite sensitive to the digitization effect caused by rotating and scaling the input images. In fact, the results reported in the work of Hilaire et al. are applied to several line-drawing images that are typically used in the context of vectorization contests, whereas the results reported by Liu et al. are applied to handwritten Chinese characters, which are not taken under extreme rotation/scaling changes.

Table 1: Repeatability scores on rotation change (%).

System	Rotation change in degrees								
	10	20	30	40	50	60	70	80	90
Our system	**97**	**93**	**88**	**92**	**90**	**89**	**93**	**97**	**99**
Hilaire et al.	76	69	65	67	65	66	66	69	76
Liu et al.	69	65	62	66	68	62	66	67	79

Table 2: Repeatability scores on scaling change (%).

System	Scaling factor							
	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
Our system	**95**	**92**	**85**	**78**	**74**	**70**	**64**	**61**
Hilaire et al.	61	49	27	15	11	10	6	5
Liu et al.	70	63	52	46	37	31	25	22

Table 3: Repeatability scores on Kanungo noise and rotation/scaling change for *setC* of GREC11 (%).

System	Location error (in pixels)							
	1	2	3	4	5	6	7	8
Our system	**45**	**63**	**75**	**80**	**83**	**85**	**85**	**86**
Hilaire et al.	30	44	52	61	70	74	75	76
Liu et al.	22	38	53	60	68	71	73	76

6.4.2. Evaluation of a mixture of Kanungo noise and rotation/scaling change

The purpose of this experiment is to determine how well each system can work under different combinations of binary noise and geometric transformations. The results are presented in Table 3. On average, the repeatability scores obtained by the proposed system are 15% higher than those of the other systems, being especially higher for the first small range of location errors. Taking the location error at 4 (pixels) for example, the proposed system gives a score of 80%, which is quite interesting considering the severe degradation in *setC*, as illustrated in Figure 11 (a, b). Under the same conditions, we can see that the scores of two baseline systems are approximately 60%, which is much less than that of the proposed system. Two main factors explain these results. First, the polygonization process in the system of Liu et al. and the skeleton segmentation step in the system of Hilaire et al. are rather sensitive to the distortion of contours. Second, the post-process of junction merging using *Criterion A* is quite sensitive to the variation and distortion of the line thickness of foreground objects. The results of our system suggest that the proposed system can satisfactorily resist degradation including common binary noise and geometric transformations.

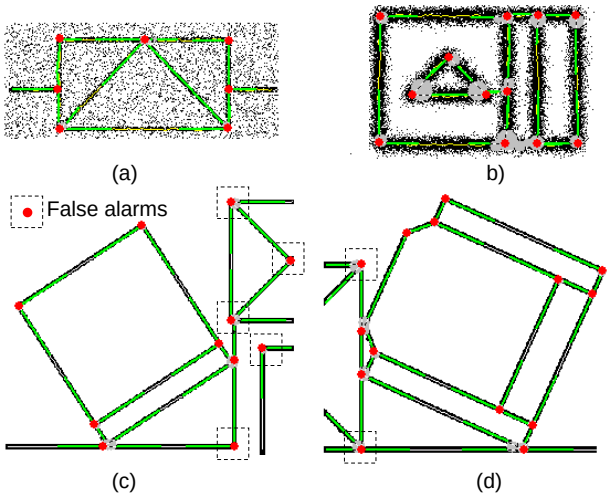

Figure 11: Junctions detected (red dots) by the proposed system for *setC* ((a) and (b)) and *setD* ((c) and (d)) of GREC11. False positives caused by context noise in *setD* are marked by dashed-line boxes.

6.4.3. Evaluation of context noise

Although Kanungo noise and geometric transformations are very common degradation models in DIA, a more realistic type of degradation is known as *context* noise. By definition, context noise is concerned with a type of disturbance caused by background or context information. For this purpose, we have selected *setD* from the final recognition dataset in GREC2011. The test images in this set have been cropped from full line-drawing documents where each reference image could be touched with other context information. The repeatability scores of three systems are reported in Table 4. Although the proposed system still outperforms the others, the repeatability scores of all systems are quite low. This finding is attributed to the fact that the images in *setD* are embedded into other context information, resulting in many false positives being detected in this set, as shown in Figure 11 (c, d). However, without any prior knowledge about groundtruth information, these false alarms correspond to the mismatches of correct detected junctions missing in the reference images.

Table 4: Repeatability scores for *setD* of GREC11 (%).

System	Location error (in pixels)							
	1	2	3	4	5	6	7	8
Our system	**4**	**8**	**12**	**16**	**21**	**25**	**28**	**32**
Hilaire et al.	2	5	8	11	15	18	21	25
Liu et al.	3	5	9	13	18	21	25	29

6.4.4. Evaluation of the low resolution dataset

In this experiment, we wish to assess the performance of the three systems for very severely low resolution images. We have selected the low resolution diagram SESYD dataset, in which the test images have been generated from the reference images by applying four exponential levels of low resolution corresponding to the scaling factors $\{1/2, 1/4, 1/8, 1/16\}$ and incorporating a compression scheme (JPG) from gray-scale images. The results of the three systems are shown in Table 5, where the proposed system again performs better than the baseline systems. On average, the proposed system provides 10% and 5% better results than those obtained from the systems of Hilaire et al. and Liu et al., respectively. All three systems perform quite well on the first levels of low resolution, but their performance rapidly degrades for the later levels of low resolution. This behavior is mainly due to the loss of much of the original information, especially finer features, when reducing the resolution.

Table 5: Repeatability scores on SESYD dataset (%).

System	Scale factor			
	1/2	1/4	1/8	1/16
Our system	**88**	**79**	**54**	**41**
Hilaire et al.	73	64	47	32
Liu et al.	85	76	50	34

6.4.5. Evaluation of the filled-shape and non-uniform stroke dataset

In this experiment, we want to justify how different kind of images, such as filled-shape and non-uniform stroke images, can impact the performance of the three systems. For this purpose, we have selected the UMD logo dataset, which typically composes of filled-shape and non-uniform stroke objects at a hard level. The repeatability scores are presented in Table 6. Even though the skeleton-based representation for such kind of images is not perfect, the obtained results are encouraging. The system of Hilaire et al. achieves the lowest scores because of a

lower number of outputted junctions. As the line thickness of the filled shapes is greatly varied compared to the typical line-drawings, many short skeleton segments are produced. Consequently, few long skeleton segments are retained, and thus the number of detected junctions is rather limited in the system of Hilaire et al. Our system also produces a limited number of junctions, but it still noticeably outperforms the system of Hilaire et al. and almost gives the same results as those of the system of Liu et al. These results confirm the accuracy of the detected junctions of the proposed system. Some visual results of the proposed system, applied to the logo images and Chinese characters, are shown in Figure 12.

Table 6: Repeatability scores for the UMD logo dataset (%).

System	Location error (in pixels)							
	1	2	3	4	5	6	7	8
Our system	**10**	19	**31**	**39**	45	49	53	55
Hilaire et al.	5	12	23	31	39	44	48	51
Liu et al.	**10**	**20**	**31**	**39**	**47**	**52**	**56**	**59**

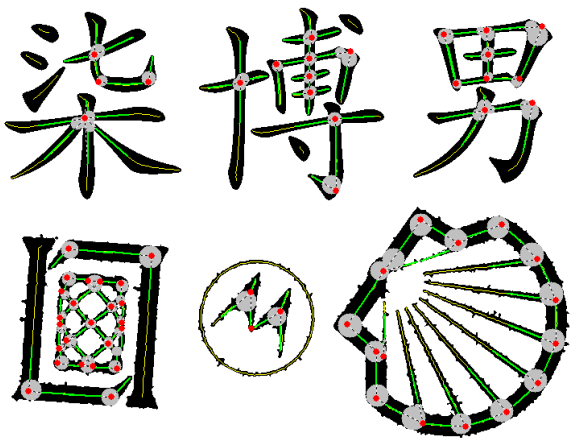

Figure 12: Junctions detected (red dots) by the proposed system for few Chinese characters and logo images.

6.4.6. *Evaluation of the built-in aspects*

In addition to the evaluations discussed above, we have investigated several additional trials to understand the behavior of the proposed approach at the system level. In particularly, we wish to present a detailed analysis of the impact of the stage of determination of ROS and the computation time of our junction detector. Regarding the first aspect, we have computed the repeatability scores for the *setC* up to the stage of dominant point detection over three different scenerios: the use of ROS based on local line thickness (i.e., the ROS at given a point p is set as the local line thickness at p), the use of ROS proposed by Teh-Chin [25], and the use of ROS proposed by our approach. The results are presented in Table 7. As is clear, our proposed use of ROS achieves much better results than the others (by almost 23%). The results linked to the ROS proposed by Teh-Chin are quite low because, as we have discussed above, Teh-Chin's ROS determination step is sensitive to digitization effects, whereas in this dataset, the noise applied to these images is quite severe, distorting their shapes. The results shown in Table 7 also reveal that line thickness could be a good feature to estimate local scales.

Table 7: Comparison of the dominant point detection rates for three scenarios.

Dominant point detection mode	Repeatability Score
With ROS of the proposed method	**67.5** %
With ROS based on line thickness	44.3 %
With ROS proposed by Teh-Chin	21.3 %

We also performed an additional experiment to study the impact of the two parameters E_{min} and E_{max} in the stage of ROS determination. For this purpose, we vary the values of E_{min} and E_{max}, and compute the repeatability score of the proposed system for the *setC* up to the stage of dominant point detection. The obtained results are presented in Figure 13. As expected, the score curves are quite stable (e.g., varying in the range of $[63, 68]$) given various settings of E_{min} and E_{max}. It is noted that the proposed system achieves the repeatability score of 67.5% in Table 7 with respect to the following setting: $E_{min} = 1.3$ and $E_{max} = 1.8$.

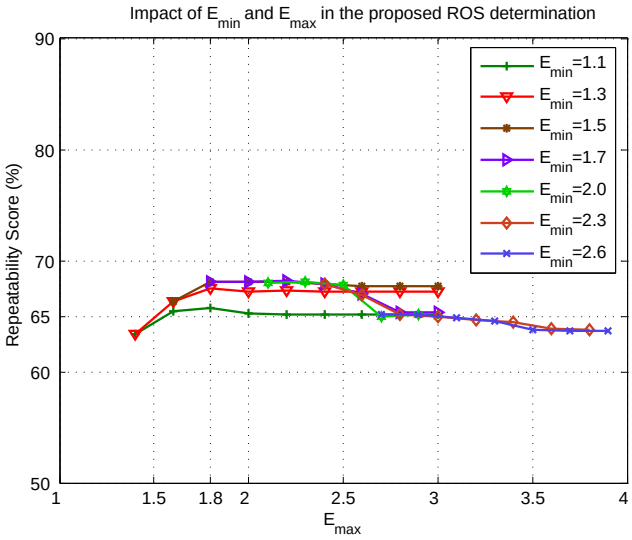

Figure 13: Impact of the parameters E_{min} and E_{max}: the repeatability score is computed on the *setC*.

For time complexity evaluation, we provide in Table 8 some information about the processing time (excluding the pre-process step) of three systems applied on several images with different sizes. The processing time has been recorded on our specific computer configuration: Intel(R) Core(TM) i5 CPU 2.4 GHz, RAM 2.4 GB, Windows 8.

In general, the system of Liu et al. is subjected to a high computation load because distorted skeleton correction using *Criterion A* is very time-consuming. The system of Hilaire et al. seems to provide a reasonable level of processing time because the criterion to merge two discrete primitives is somewhat similar in spirit to *Criterion A* but with the elimination of much of the redundant computation. Our system works most efficiently, not only for the cases of the several images reported in Table

Table 8: Report of the processing time (ms) and the number of detected junctions (in brackets).

System	Image size (Width × Height)		
	900 × 984	3600 × 3938	2100 × 4433
Our system	**16.0 (67)**	**187.0 (79)**	**140.0 (105)**
Hilaire et al.	110.0 (89)	297.0 (147)	265.0 (146)
Liu et al.	563.0 (82)	14953.0 (157)	4078.0 (174)

8 but also throughout the extensive experiments we have performed. It is also noted that the number of detected junctions (in brackets) provided by our system is much smaller than those outputted by the other systems.

7. Application to symbol spotting

We present here additional experiments at the application level to prove that our junction detector is robust and discriminant enough to be used in a retrieval, spotting, or recognition systems. We have used our approach in a baseline symbol spotting system. A symbol spotting system is often composed of the following modules: the decomposition of document images into primitives; the characterization of the primitives; primitive matching, grouping and localization; and verification. In our case, the end-points are first added to the detected junctions to form a set of interest points. Without loss of generality, an end-point is also characterized as a special 2-junction whose arms form an included angle equal to 180^0. Given a query symbol, its interest points are detected and characterized as described in the proposed approach. Next, the matching step is performed to find the correspondences among the interest points of the query symbol and those of database documents. The obtained matches are finally verified by the use of Generalized Hough Transform [29]. This step will remove false matches and cluster the remaining matches into different groups such that each group indicates an instance of the query symbol.

For performance evaluation of the proposed approach, we have selected the latest dataset for symbol spotting in GREC2011[5]. The detail of this dataset is described in Table 9. The evaluation metric in [30] has been selected, including precision (P), recall (R), and F-score (F) as follows.

$$P = \frac{S_{Int}}{S_{Ret}}, \quad R = \frac{S_{Int}}{S_{GT}}, \quad F = 2 \cdot \frac{P \cdot R}{P + R}$$

Where S_{Int} is the sum of intersection areas between the bounding boxes retrieved by the spotting system and ground-truth, S_{Ret} is the sum of areas of the bounding boxes retrieved by the spotting system, and S_{GT} is the sum of areas of the bounding boxes in ground-truth. It is worth mentioning that each bounding box (B_{gt}) in ground-truth is counted at most once. Typically, B_{gt} will be marked as already considered if there exists a bounding box (B_{ret}) retrieved by the spotting system such that the ratio of the intersection of their areas to the union of their areas is greater than a given threshold (e.g., 75% in our experiment). This is needed to avoid biased scores caused by multiple detections of a same symbol at a same location.

Table 9: The spotting results of the proposed system

Test	Queries	Symbols	Noise	P	R	F
Elec1.	118	246	Ideal	0.74	0.82	0.78
Elec2.	127	274	Level 1	0.79	0.78	0.78
Elec3.	114	237	Level 2	0.85	0.71	0.77
Elec4.	156	322	Level 3	0.71	0.76	0.73
Archi1.	247	633	Ideal	0.88	0.96	0.92
Archi2.	245	597	Level 1	0.86	0.95	0.91
Archi3.	245	561	Level 2	0.88	0.89	0.88
Archi4.	249	593	Level 3	0.88	0.94	0.91

The obtained results are reported in Table 9. On average, the F-score(s) of the proposed system are 0.77 and 0.91 for electrical and architectural datasets, respectively. It is noted that the results achieved for the architectural dataset are much better than those on the electrical dataset. This difference is attributed to the fact that in the electrical dataset, more query symbols are used and many query symbols look very similar, making them difficult to correctly distinguish.

8. Conclusions and future works

In this paper, a new approach for junction detection and characterization in line-drawings has been presented. The main contribution of this work is three-fold. First, a new algorithm for the determination of the region of support is presented using the linear least squares technique. The crossing-points, in combination with the dominant points detected from median lines, are treated as candidate junctions. Next, using these candidate junctions, an efficient algorithm is proposed to detect and conceptually remove all distorted zones, retaining reliable median line segments only. These line segments are then locally characterized to construct the topological representations of the crossing zones. Finally, a novel junction optimization algorithm is presented, yielding accurate junction localization and characterization. The proposed approach is extremely robust to common geometry transformations and can resist a satisfactory level of noise/degradation. Furthermore, it works very efficiently in terms of time complexity and requires no prior knowledge of the document content. All of these prominent features of the proposed approach have been validated relative to other baseline methods by our extensive experiments. An application of symbol spotting has also been provided to demonstrate the usefulness of the detected junctions.

In addition to these advantages, we are also aware of the following shortcomings of the proposed approach. First, as this approach is dedicated to working with line-like primitives, its performance would be degraded if applied to filled-shape objects, such as logo images. In addition, the junction optimization process could lead to some difficulties in correctly interpreting the junction position as originally produced by craftsmen. However, although this point is valid for some specific

[5] http://iapr-tc10.univ-lr.fr/index.php/final-test-description

domains of exact line-drawing representation, such as vectorization, we are interested in detecting local features that would be useful to addressing the problem of large-scale document indexing and retrieval. In this sense, a low rate of false positives in the final results is not problematic. A last noticeable point is that the detected junctions could be used in combination with other types of features (e.g., end-points, *isolated* straight lines, arcs, and circles) to obtain a complete representation of document images. Further work on the use of the detected junctions will be investigated in the future, such as document indexing and retrieval.

References

1. Parida, L., Geiger, D., Hummel, R.. Junctions: detection, classification, and reconstruction. *IEEE Trans Pattern Anal Mach Intell* 1998; **20**(7):687–698.
2. Bergevin, R., Bubel, A.. Detection and characterization of junctions in a 2d image. *Comput Vis Image Underst* 2004;**93**(3):288–309.
3. Deschênes, F., Ziou, D.. Detection of line junctions and line terminations using curvilinear features. *Pattern Recogn Lett* 2000;**21**(6–7):637–649.
4. Köthe, U.. Integrated edge and junction detection with the boundary tensor. In: *Proceedings of the 9th IEEE International Conference on Computer Vision*; vol. 2. 2003, p. 424–431.
5. Maire, M., Arbelaez, P., Fowlkes, C., Malik, J.. Using contours to detect and localize junctions in natural images. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. 2008, p. 1–8.
6. Xia, G.S.. *Some Geometric Methods for the Analysis of Images and Textures*. Ph.D. thesis; Télécom ParisTech (ENST); 2011.
7. Dori, D., Liu, W.. Sparse pixel vectorization: An algorithm and its performance evaluation. *IEEE Trans Pattern Anal Mach Intell* 1999; **21**(3):202–215.
8. Hilaire, X., Tombre, K.. Robust and accurate vectorization of line drawings. *IEEE Trans Pattern Anal Mach Intell* 2006;**28**(6):890–904.
9. Chiang, J.Y., Tue, S., Leu, Y.C.. A new algorithm for line image vectorization. *Pattern Recognition* 1998;**31**(10):1541–1549.
10. Liu, K., Huang, Y., Suen, C.Y.. Identification of fork points on the skeletons of handwritten chinese characters. *IEEE Trans Pattern Anal Mach Intell* 1999;**21**(10):1095–1100.
11. Song, J., Su, F., Tai, C.L., Cai, S.. An object-oriented progressive-simplification-based vectorization system for engineering drawings: Model, algorithm, and performance. *IEEE Trans Pattern Anal Mach Intell* 2002;**24**(8):1048–1060.
12. Baker, S., Nayar, S.K., Murase, H.. Parametric feature detection. *Int J Comput Vision* 1998;**27**(1):27–50.
13. Sluzek, A.. A local algorithm for real-time junction detection in contour images. In: *Proceedings of the 9th International Conference CAIP'01*. 2001, p. 465–472.
14. Tabbone, S.A., Alonso, L., Ziou, D.. Behavior of the laplacian of gaussian extrema. *J Math Imaging Vis* 2005;**23**(1):107–128.
15. Hilaire, X., Tombre, K.. Improving the accuracy of skeleton-based vectorization. In: *Proceedings of the 4th International Workshop GREC'01, LNCS 2390*. 2001, p. 273–288.
16. Lee, C., Wu, B.. A chinese-character-stroke-extraction algorithm based on contour information. *Pattern Recognition* 1998;**31**(6):651–663.
17. van Nieuwenhuizen Peter R. Kiewiet, O., Bronsvoort, W.F.. An integrated line tracking and vectorization algorithm. *Computer Graphics Forum* 1994;**13**(3):349–359.
18. Ramel, J.Y., Vincent, N., Emptoz, H.. A structural representation for understanding line-drawing images. *Int J Doc Anal Recognit (IJDAR)* 2000;**3**(2):58–66.
19. Fan, K.C., Wu, W.H.. A run-length-coding-based approach to stroke extraction of chinese characters. *Pattern Recognition* 2000;**33**(11):1881–1895.
20. di Baja, G.S.. Well-shaped, stable, and reversible skeletons from the (3,4)-distance transform. *J Vis Commun Image R* 1994;**5**(1):107–115.
21. Mohammad Awrangjeb, G.L., Fraser, C.S.. Performance comparisons of contour-based corner detectors. *IEEE Trans Pattern Anal Mach Intell* 2012;**135**(9):4167–4179.
22. Mokhtarian, F., Mohanna, F.. Performance evaluation of corner detectors using consistency and accuracy measures. *Comput Vis Image Underst* 2006;**102**(1):81–94.
23. Awrangjeb, M., Lu, G.. An improved curvature scale-space corner detector and a robust corner matching approach for transformed image identification. *IEEE Trans Image Process* 2008;**17**(12):2425–2441.
24. Mokhtarian, F., Suomela, R.. Robust image corner detection through curvature scale space. *IEEE Trans Pattern Anal Mach Intell* 1998; **20**(12):1376–1381.
25. Teh, C.H., Chin, R.T.. On the detection of dominant points on digital curves. *IEEE Trans Pattern Anal Mach Intell* 1989;**11**(8):859–872.
26. Carmona-Poyato, A., Fernández-García, N.L., Medina-Carnicer, R., Madrid-Cuevas, F.J.. Dominant point detection: A new proposal. *Image Vision Comput* 2005;**23**(13):1226–1236.
27. Reisfeld, D., Wolfson, H., Yeshurun, Y.. Context free attentional operators: the generalized symmetry transform. *Int J Comput Vision* 1995; **14**(2):119–130.
28. Tuytelaars, T., Mikolajczyk, K.. Local invariant feature detectors: a survey. *Found Trends Comput Graph Vis* 2008;**3**(3):177–280.
29. Ballard, D.H.. Generalizing the hough transform to detect arbitrary patterns. *Communications of the ACM* 1981;**13**(2):111–122.
30. Rusinol, M., Llados, J.. A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices. *Int J Doc Anal Recognit (IJDAR)* 2009;**12**(2):83–96.