Effective Multiresolution Arc Segmentation: Algorithms and Performance Evaluation

Jiqiang Song, *Member*, *IEEE*, Michael R. Lyu, *Fellow*, *IEEE*, and Shijie Cai

Abstract—Arc segmentation plays an important role in the process of graphics recognition from scanned images. The GREC arc segmentation contest shows there is a lot of room for improvement in this area. This paper proposes a multiresolution arc segmentation method based on our previous seeded circular tracking algorithm which largely depends on the OOPSV model. The newly-introduced multiresolution paradigm can handle arcs/circles with large radii well. We describe new approaches for arc seed detection, arc localization, and arc verification, making the proposed method self-contained and more efficient. Moreover, this paper also brings major improvement to the dynamic adjustment algorithm of circular tracking to make it more robust. A systematic performance evaluation of the proposed method has been conducted using the third-party evaluation tool and test images obtained from the GREC arc segmentation contests. The overall performance over various arc angles, arc lengths, line thickness, noises, arc-arc intersections, and arc-line intersections has been measured. The experimental results and time complexity analyses on real scanned images are also reported and compared with other approaches. The evaluation result demonstrates the stable performance and the significant improvement on processing large arcs/circles of the MAS method.

Index Terms—Graphics recognition, arc segmentation, multiresolution, circular tracking, vectorization, performance evaluation.

1 Introduction

Arc segmentation is the process of recognizing arcs and circles from images, especially the scanned images of engineering drawings. Since arcs/circles are essential elements of many types of drawings, e.g., construction drawings, mechanical drawings, and circuit drawings, arc segmentation has received great attention in the last two decades [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22]. From 1995, the IAPR International Workshop of Graphics Recognition (GREC) regularly held graphics recognition contest to examine the state-of-the-art of graphics recognition methods. The first arc segmentation contest was held in 2001 [23]. According to the contest results [24], the current arc segmentation solutions are still far from satisfactory, especially for real scanned images, due to the known difficulties, including the degradation of arc images and the presence of other graphical elements, texts, and noises. This demonstrates that there is a lot of room for improvement in arc segmentation.

The existing arc segmentation methods can be classified into two classes: vectorization-based segmentation methods and direct segmentation methods. The former class first performs a raster-to-vector conversion on an image, and then segments arcs/circles from the vectors. The latter class works directly on the image and segments arcs/circles at the pixel level. A detailed review of these methods is provided in Section 2. In our previous research [22], we proposed a seeded pixel tracking method for arcs/circles; however, it has three major limitations. In this paper, we propose new techniques to remove these limitations for significant performance improvement. The new method has been tested systematically and exhaustively using the test images of GREC'01 and GREC'03 arc segmentation contests and other real images. The comparison result with other approaches shows that the performance on real images has been greatly enhanced by our method.

The rest of this paper is organized as follows: Section 2 reviews the existing arc segmentation methods. Section 3 describes the proposed multiresolution arc segmentation method in detail. Section 4 reports the experimental results on various test images and the comparison with other methods. Finally, Section 5 draws our conclusions.

2 Related Work

Vectorization-based segmentation methods all employ some vectorization approaches as preprocessing to convert a raster image into raw vectors (usually short segments); therefore, arc segmentation actually works at the vector level. Vectorization can be implemented by various methods [25], [26], [27], [28], [29], but it is not the focus of this paper. We focus on how to segment arcs/circles from raw vectors. According to the underlying techniques, these methods are further divided into arc-fitting methods [1], [3], [4], [7], [10], [12], [18], [20], [21], circular Hough Transform (HT) methods [2], [5], [8], [9], and stepwise arc extension methods [13], [14], [15].

- Arc-fitting methods first collect a group of vectors which are believed to form an arc or a sequence of arcs, and then find the fittest arc parameters by some iterative processes, such as least square computation [7] and iterative splitting [4]. The major limitation of these methods is the sensitivity to vector distortions, leading to localization errors.

- *J. Song and M.R. Lyu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong SAR, P.R. China. E-mail: songjq@ieee.org, lyu@cse.cuhk.edu.hk.*
- *S. Cai is with the State Key Lab of Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing, 210093, P.R. China. E-mail: sjcai@netra.nju.edu.cn.*

Manuscript received 26 July 2003; revised 19 Mar. 2004; accepted 2 Apr. 2004.
Recommended for acceptance by Y. Amit.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0192-0703.

- Circular HT methods globally map the vertices of all vectors to the circle parameter space, and then detect the parameters with high accumulative values as potential arcs and circles. These methods can resist partial vector distortions; however, the circular HT requires more computation and memory space (3D transform space instead of 2D space in the linear HT), which is not efficient for large arcs/circles. Moreover, they are in a dilemma of choosing a proper threshold for both accepting short arcs and eliminating noises.
- Stepwise arc extension methods are developed from the Perpendicular Bisector Tracing algorithm [13] to the Incremental Arc Segmentation algorithm (IAS) [15]. IAS algorithm first detects a candidate arc, and then uses the shape and style of the current arc as constraints to select a new fragment appended to the current arc. This algorithm has a certain capability in handling distortions; however, it cannot handle well the cases when there are big intersections on an arc or when an arc shares vectors with other graphic objects.

We note that vectorization-based segmentation methods share a common disadvantage: Their performance is affected by the information loss and distortions introduced by vectorization; that is, the resulting arc may fit the component vectors well, but it does not necessarily fit its raster image. This phenomenon is also called "uncontrolled location errors" [18].

Since such information loss and distortions are the inherent weaknesses of vectorization, direct segmentation methods aim at avoiding the disadvantages of vectorization by segmenting arcs/circles from an image directly. These methods include statistical methods [11], [17] and pixel tracking methods [6], [16], [19], [22].

- Statistical methods utilize the symmetric attribute of circles to detect them. They first scan an image horizontally (or vertically) to accumulate the times of each position being the midpoint of two edge points. Then, they perform linear HT on those positions with high accumulative values to generate a group of horizontal (or vertical) symmetrical axes. The intersections of horizontal and vertical symmetrical axes become candidate circle centers. Finally, they collect the contributing edge points to each circle center to form the subimage of this circle. These methods are robust for circles, even ellipses; however, they cannot handle arcs. Moreover, they are very time-consuming for large images.
- Pixel tracking algorithms are based on the geometric constraints of arcs/circles. Kovalevsky's algorithm [6] begins with two tracked straight-line segments, which give a hint of the arc curvature and construct a center polygon. If the next tracked straight-line segment conforms to the center polygon, the arc can be extended; otherwise, the tracking stops. This method can track through intersections and is less noise-sensitive on the condition that the center polygon is accurate. However, there is no guarantee to that. Wei's algorithm [16] proposes to compare the arc radius with the distances between candidate pixels and the arc center to select the next tracking pixel. But, the intensive distance computations slow down the tracking. Since neither of the above two algorithms contains adjustment to the initial arc parameters (center and radius), they cannot track out an entire arc/circle if the initial arc parameters are inaccurate. In our previous research [19], [22], we proposed a Seeded Circular Tracking (SCT) method with a dynamic adjustment algorithm; therefore, the fragmentation error rate is significantly decreased. SCT is also faster than the above two algorithms because the circular tracking path is generated by the efficient Bresenham algorithm for circle [30]. However, the SCT method has three major limitations: 1) The seed segments for arcs are detected from the potential arc positions produced during the straight-line recognition; therefore, it is not independent. 2) It fails to detect too-small or too-large arcs/circles. Moreover, the dynamic adjustment is not efficient for too-large arcs/circles. 3) The false detection removal largely relies on the progressive simplification of the OOPSV model [22].

The Multiresolution Arc Segmentation (MAS) method proposed in this paper makes major improvements over the SCT method by developing new techniques and improving the dynamic adjustment algorithm, and eliminates the above-mentioned three limitations. Therefore, the proposed algorithm is self-contained and is more efficient.

3 MAS Algorithms

At first, we need to analyze the limitations of the SCT method in more detail. Limitations 1) and 3) of the SCT method are mainly due to its close coupling with other algorithms in the OOPSV model for the purpose of utilizing the intermediate results of other algorithms conveniently. However, when we want to segment only arcs/circles without considering straight lines, noncircular curves, and texts, the SCT method cannot accomplish the task independently. The weakness of detecting too-small arcs is due to the length limit of seed segments for straight lines, which form potential arc positions. An arc is detectable when its length is longer than three seed segments. This length limit misses small arcs. On the other hand, if the radius of an arc/circle is too large, its curvature cannot be detected in a limited length, which leads to the weakness of detecting too-large arcs/circles. Moreover, with the increasing radius, the rapid growth of circumference length makes the pixel tracking less efficient. To overcome these limitations, the proposed MAS method devises a novel multiresolution arc segmentation paradigm to handle large arcs/circles, a sensitive pixel-level arc seed detection algorithm to detect arcs/circles from images directly, and a confidence-weighted arc verification algorithm to remove false detections.

3.1 Multiresolution Arc Segmentation Paradigm

The MAS method works not only at the original resolution, but also at lower resolutions, i.e., smaller images downsized from the original image. For a big arc which is not detectable at the original resolution, it may become detectable in some lower resolution because its radius is scaled down. The processing time for the arc is reduced as well. However, multiresolution segmentation also brings up two critical issues: 1) the repetitive detection of the same arc/circle at more than one resolution and 2) the localization of arcs/circles detected from lower resolution levels in the original resolution level. Fig. 1 shows the proposed MAS paradigm which handles these two issues well.

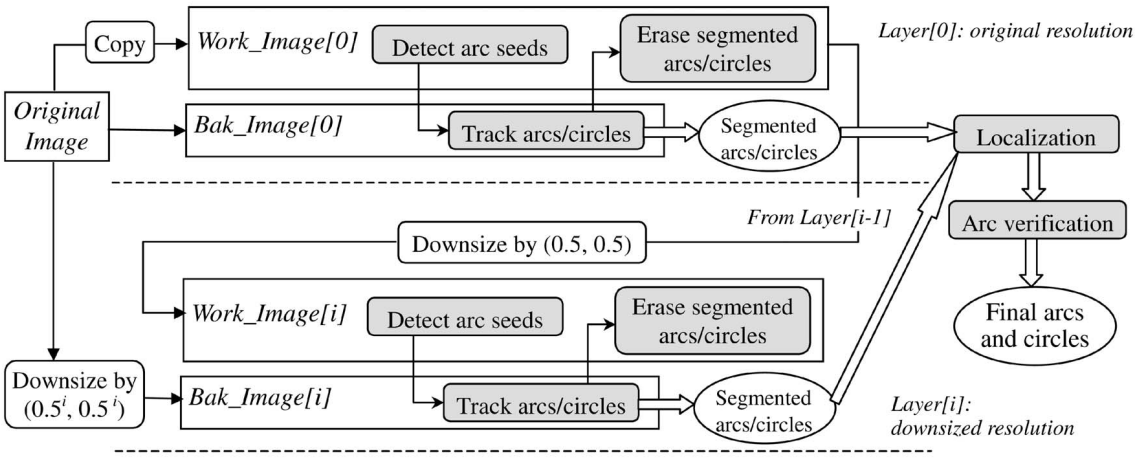


Fig. 1. Multiresolution arc segmentation paradigm.

The MAS paradigm divides the whole processing into N layers $(N \geq 1)$, where each layer corresponds to one resolution level. In each $Layer[i]$, there are two images (called twin images) with the same size, $Work\_Image[i]$ and $Bak\_Image[i]$, which are the key to eliminate repetitive detections. The whole processing starts from $Layer[0]$, i.e., the original resolution. The original image serves as $Bak\_Image[0]$ and, initially, $Work\_Image[0]$ is an identical copy of $Bak\_Image[0]$. First, the pixel-level arc seed detection is performed in $Work\_Image[0]$. A pixel-level arc seed is a segment of raster shape showing the circular curvature. Then, for each detected arc seed, the dynamic circular tracking is invoked in $Bak\_Image[0]$. If the tracking succeeds, a new arc/circle will be segmented. Finally, the pixels corresponding to the segmented arc/circle are erased from $Work\_Image[0]$ and the arc segmentation in $Layer[0]$ is completed. The twin images for the next layer are then generated as follows: $Work\_Image[i]$ is downsized by half in both width and height from the current $Work\_Image[i-1]$ (where the segmented arcs/circles are erased), and the $Bak\_Image[i]$ is always downsized by 0.5^i in both width and height from the original image (i.e., $Bak\_Image[0]$). After the twin images of $Layer[i]$ are ready, the execution of seed detection, tracking, and erasing processes is the same as that in $Layer[0]$. Therefore, the arcs/circles segmented in a high resolution level will not be detected in the lower resolution levels. Moreover, since the tracking works in Bak_Image, the possible overerasing at intersections in $Work\_Image$ will not affect the tracking. Before the segmented arcs/circles from each layer are added to the resulting arcs/circles, they need to first go through a localization process to obtain their accurate parameters in the original resolution, and then pass an arc verification process to eliminate false detections.

The layer number of the multiresolution image pyramid, i.e., N, is determined as follows:

$$N = 1 + Log_2 MAX\left\{1, \left\lfloor \frac{MIN(width, height)}{T_{size}} \right\rfloor\right\}. \quad (1)$$

T_{size} is the threshold to stop the iterative downsizing. According to (1), when the minimum of width and height is less than $2 \times T_{size}$, the image will not be downsized any more. The smaller T_{size} is, the larger N is. Usually, T_{size} should be set to be slightly larger than the maximum detectable radius in the pixel level, denoted by R_{max}, which is derived as follows:

R_{max} indicates the top limit of the radius, with which the curvature of an arc/circle can be detected from an arc segment with a certain length. Fig. 2 illustrates the way to calculate R_{max}. For an arc with radius (r), its offset (d) from its tangent line over length (l) is computed as (2).

$$d(r, l) = \sqrt{r^2 + l^2} - r. \quad (2)$$

We define the term "detectable scope" as the radius of the maximum circular neighborhood in which the arc seed detection algorithm searches. Suppose that l acts as the detectable scope, R_{max} is the radius of the potential arc when d is very small but still detectable, as shown in (3), where l equals 30 in our algorithm, as justified in Section 3.2.

$$R_{\max} = \underset{r}{Arg}[d(r, 30) = \varepsilon]. \quad (3)$$

Note that ε is not set to be 1, but 0.75 to simulate the quantization error of digital images. R_{max} is 600 by computing (3). Thus, we set T_{size} to be 800. Since R_{max} is the limit of each layer, the MAS paradigm can handle arcs/circles with radii as large as $2^{N-1} \times 600$. For example, a $5,000 \times 5,000$ image requires three layers and the largest detectable radius is 2,400.

As the MAS paradigm requires more memory than single resolution segmentation, we need to study its memory consumption. Let S denote the size of the original image, then the total memory requirement for an N-layer segmentation is

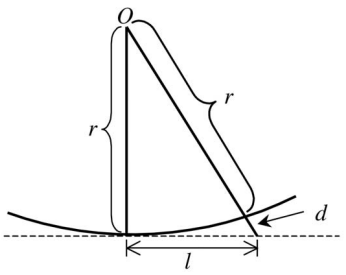


Fig. 2. R_{max} calculation.

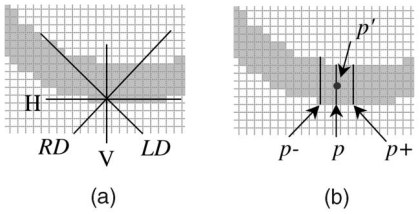


Fig. 3. Check linear shape.

$$2 \times \sum_{i=0..N-1} \left[\left(\frac{1}{2} \times \frac{1}{2} \right)^i \cdot S \right],$$

which is less than $3 \times S$. Considering an A0-sized engineering drawing scanned with 300dpi, which is the largest drawing size, in practice, with a fine enough scan resolution, the size of its binary scanned image is about 12MB. Therefore, the memory required by the MAS paradigm is less than 36MB, which is certainly affordable for modern computers.

Sections 3.2-3.5 depict the algorithms for arc seed detection, dynamic circular tracking, arc localization, and arc verification, respectively. The algorithm for erasing segmented arcs/circles is the same as that of [22], therefore, it is omitted in this paper.

3.2 Arc Seed Detection

Arc seed detection is aimed at locating potential arc positions in an image. Since it is the first screening process of arc segmentation and, therefore, is frequently invoked, it should be acute, accurate, and efficient. The requirement of both acuity and accuracy demands that the detection produces positive responses in arc areas and negative ones in nonarc areas. The requirement of efficiency demands that the detection should pass through nonarc areas very quickly. The proposed arc seed detection algorithm satisfies these requirements well. In the following description, we assume the foreground color of the image is *black*, while the background color is *white*.

The arc seed detection algorithm scans the image horizontally every k rows, where $k(\geq 1)$ depends on the image resolution (R_s), the minimum radii of arcs (R_{min}) and the maximum line thickness (W_{max}). R_s, measured by *dpi*, can be retrieved from the image header, while R_{min} and W_{max} are calculated by R_s multiplying real-life length limits, which are usually domain-specific. In this algorithm, the length limits for R_{min} and W_{max} are set to be 0.04 inches and 0.05 inches, respectively. Thus, when $R_s = 200$ *dpi*, $R_{min} = 8$, $W_{max} = 10$. k should be selected proportional to R_s and below the smaller one between R_{min} and W_{max}. A large k definitely speeds up the detection, but it also raises the risk of misses. We set k as 4. When the arc seed detection meets a black pixel, denoted by p, it first checks whether the neighborhood of p appears a linear shape by the following steps:

Step 1. Check the lengths of black runs passing p in four directions: horizontal, vertical, left diagonal, and right diagonal, respectively (Fig. 3a). If the length in a direction (denoted by dir) does not exceed W_{max}, go to **Step 2**. After all the four directions are processed, go to **Step 4**. In Fig. 3a, the vertical (V) direction and the right diagonal (RD) direction are eligible. Since the purpose of this step is to detect the potential directions of linear shapes, the tracking in four candidate directions will stop when the tracked length exceeds W_{max} or a white pixel is encountered. It does not follow the shapes for vectorization as the Orthogonal ZigZag tracking (OZZ) [13] does.

Step 2. Translate p for 0.01 inches, i.e., two pixels for $R_s = 200$ *dpi*, along the direction perpendicular to dir, both positive and negative, to $p+$ and $p-$, respectively. If both $p+$ and $p-$ are black, go to **Step 3**; otherwise, go to **Step 1** to check the next direction. In Fig. 3a, only the vertical direction is eligible.

Step 3. Check the lengths of black runs parallel to dir and passing $p+$ and $p-$, respectively (Fig. 3b). If both two lengths are smaller than W_{max} and are similar with the length of the black run passing p, dir is accepted as a candidate and the differences among these three lengths are summed as the linearity error for dir. Otherwise, go to **Step 1** to check the next direction. In Fig. 3b, the vertical direction becomes a candidate.

Step 4. If there is no candidate, return false. Otherwise, return true and take the midpoint of the black runs in the candidate direction with the minimum linearity error as the center of the linear shape, as p' in Fig. 3b.

This linear shape checking quickly eliminates intersections, corners, and complex overlapping parts, ensuring that arc seeds will always be detected at normal linear shapes. If p passes the linear shape checking, the returned p' is used as the new position to detect an arc seed. We use two concentric circle windows centered at p' to detect arc seeds. The algorithm of detecting an arc seed from p' is explained by the following C pseudocode, referring to Fig. 4.

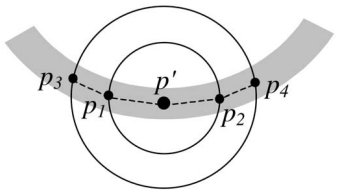


Fig. 4. Arc seed detection.

```
ArcSeedDetection(p') {
    Get the intersecting black segments on the inner
    window;
    Store those segments whose lengths ≤ W_max in IS_inner;
    if (the number of black segment in IS_inner < 2)
        return false;

    Get the intersecting black segments on the outer
    window;
    Store those segments whose lengths ≤ W_max in IS_outer;
    if (the number of black segment in IS_outer < 1)
        return false;

    Calculate the midpoints of the segments in IS_inner and
    IS_outer as candidate positions, i.e., {p_i};
    Find the best sequence of candidate positions that
    connect with p' in pixel level and form the best circular
    curvature, such as p_1, p_2, p_3, and p_4 in Fig. 4. Even if the
    best sequence contains only one position on the outer
    window (i.e., either p_3 or p_4 is absent), the sequence is
    still acceptable;
```

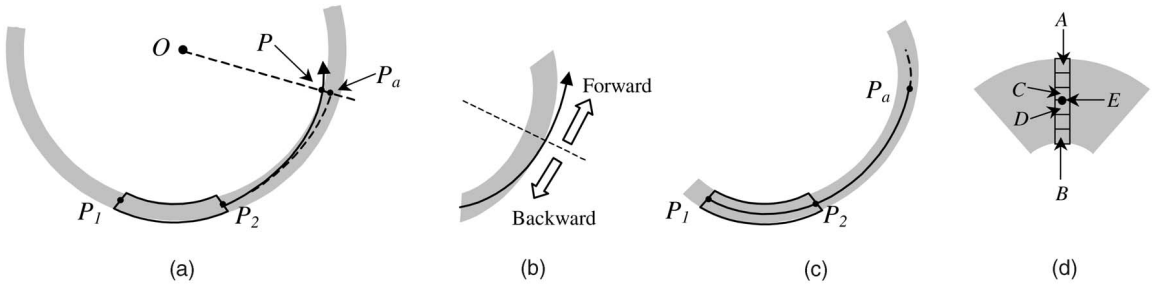


Fig. 5. Dynamic circular tracking.

```
    if (such sequence exists) {
        Compute the arc parameter from the best sequence,
        including arc center, radius, starting angle, ending
        angle, and line thickness;
        The detected arc seed is characterized by the arc
        parameter;
        return true;
    }
    else
        return false;
}
```

The reason for using circular windows is to ensure the isotropy of the detection, i.e., the length of the intersecting segment between a line and the circular window does not vary with the direction of the line. The advantages of utilizing two concentric windows are threefold: 1) to make the detection more efficient by fast eliminating the areas without long linear shapes by checking the number of intersecting segments on the inner window, 2) to make the detection more acute since it can handle the case when there is one intersecting segment on the outer window, and 3) to make the accepted arc seed more reliable since a sequence of candidate positions consists of four or five points, not three points. We know that any three points determine a circle, even if some points are distorted. With four or five points, we can divide them into two groups of three points and check the consistency of their circular parameters. The best sequence is the one with the best consistency between two groups.

The radii of the inner window and the outer window, denoted by R_{inner} and R_{outer}, respectively, are determined as follows: The upper bound of R_{inner} is R_{min}, while that of R_{outer} is $2 \times R_{min} - 1$. This is to ensure a small circle whose radius is R_{min} can still intersect the outer window. The lower bound of R_{inner} is $W_{max}/2 + 1$, while that of R_{outer} is $R_{inner} + 1$. Note that although R_{inner} and R_{outer} can be set lower, it is not recommended since this will diminish the first advantage mentioned above. This algorithm sets them at their upper bounds. According to the previous setting of R_{min}, R_{inner}, and R_{outer} are 8 pixels and 15 pixels, respectively. Therefore, the detectable scope is the diameter of the outer window, i.e., 30 pixels, and the arc seed detection algorithm can detect an arc as short as 23 (i.e., $15 + 8$) pixels long. In conclusion, the largest detectable radius is 600, as discussed in Section 3.1, while the smallest detectable radius is 8.

Note that the arc seed detection works in the $Work\_Image$, and the corresponding pixels of a segmented arc/circle in the $Work\_Image$ are erased immediately after it is segmented. Therefore, the segmented arcs/circles will not be detected again.

3.3 Dynamic Circular Tracking

After obtaining an arc seed, the dynamic circular tracking begins to segment the entire arc/circle. Our tracking algorithm is an improved version of the SCT (seeded circular tracking) algorithm proposed in [22]. For completeness, we briefly review the SCT algorithm here.

The SCT algorithm employs the Bresenham algorithm for circles to generate the circular tracking path points; therefore, it is much faster than computing distances. As depicted in Fig. 5, tracking begins from an arc seed, parameterized by O, P_1, and P_2 (Fig. 5a), and extends it from P_2 along the circular direction $P_1 \rightarrow P_2$. During the tracking, the dynamic adjustment is performed when tracking out of black areas to extend the tracking as long as possible. For example, P is adjusted to P_a in Fig. 5a. If successful, the tracking yields an extended arc with updated O and P_2. If it returns a circle, the tracking finishes; otherwise, the SCT algorithm interchanges P_1 and P_2 and tracks in the other direction again. After the tracking finishes, it either returns a failure flag indicating no arc/circle is segmented, or returns a success flag with the parameter of segmented arc/circle. The parameter of a circle includes center, radius, and line thickness. The parameter of an arc includes center, radius, line thickness, starting angle, and ending angle. Finally, we remove all the pixels covered by the newly segmented arc/circle from the image.

The improvement focuses on the dynamic adjustment when tracking out of black areas. It includes three aspects: the adjustment position selection strategy, the validity measure of an adjustable position, and the precision of adjustment. This improvement makes the dynamic circular tracking more robust, as follows:

1. The SCT algorithm selects the adjustment position using a first-found strategy, which first checks backward and then checks forward for adjustable positions (Fig. 5b), and uses the first-found valid position for adjustment. This strategy is simple, but the first-found position may not be the best one. This paper improves it to a best-of-all strategy, which first checks both backward and forward to form a collection of candidate positions, and then selects the best one for adjustment by the validity measure.
2. The SCT algorithm measures the validity of an adjustable position, denoted by P_a, only by checking the matching extent between the black area and the testing arc generated by P_1, P_2, and P_a (the solid arc in Fig. 5c). However, it does not check the extensibility of P_a. This paper improves it by including the extensibility in the validity measure.

The testing arc is extended forward for a few steps (the dashed arc in Fig. 5c), and the extensible length in black area is recorded. Therefore, the final validity measure is a weighted sum of the matching extent and the extensible length.

3. The adjustment precision of the SCT algorithm is 1 pixel. For example (cf. Fig. 5d), when the midpoint of AB is the adjustment position, the SCT algorithm will use either C or D because the line width is even. This may cause a few offsets of the adjusted arc. In this paper, the adjustment position can be of real coordinates; therefore, the adjustment precision is improved to half a pixel. For the case in Fig. 5d, C, D, and the real midpoint E are all accepted as candidates to compete for the best adjustment position.

3.4 Arc Localization

For the arcs/circles segmented in $Layer[i]$, where $i > 0$, we must localize their accurate parameters in $Layer[0]$, i.e., the original resolution. This process is called arc localization, which focuses on localizing center and radius, since other parameters (including line thickness, starting angle, and ending angle) can be easily measured after the center and radius have been determined. Considering an arc in $Layer[i]$ with parameter (x, y, r), where x and y are center coordinates and r is radius, its accurate parameter in $Layer[0]$ will be one point in the space

$$\begin{aligned} SP = \{(x', y', r') | & x \times 2^i \le x' < (x+1) \times 2^i; \\ & y \times 2^i \le y' < (y+1) \times 2^i; \\ & r \times 2^i \le r' < (r+1) \times 2^i\}. \end{aligned}$$

The dimension of SP is $2^i \times 2^i \times 2^i$, i.e., 8^i. Since the same process is performed for each possible parameter, the computation complexity of arc localization is $O(8^i)$, which is high when $i > 1$.

Apparently, localizing downsized arcs/circles directly to the original resolution is not efficient. Therefore, we propose a layer-by-layer localization algorithm. Since the MAS paradigm stores the Bak_Image array of all layers, which contains information as complete as the original image, we can perform a stepwise localization from the current layer to the original resolution, as shown in Fig. 6. Suppose the current layer is $i (i > 0)$ and the parameter is (x, y, r). First, the constructed SP contains eight possible parameters in $Layer[L-1]$. Then, each parameter is tested in $Bak\_Image[L-1]$ to obtain a score. Finally, (x, y, r) is updated with the parameter with the best score, and the current layer is set to the upper layer. The above process repeats until the current layer reaches 0. With this algorithm, the computation complexity of arc localization decreases to be $O(8i)$, which is significantly less than $O(8^i)$ when $i > 1$.

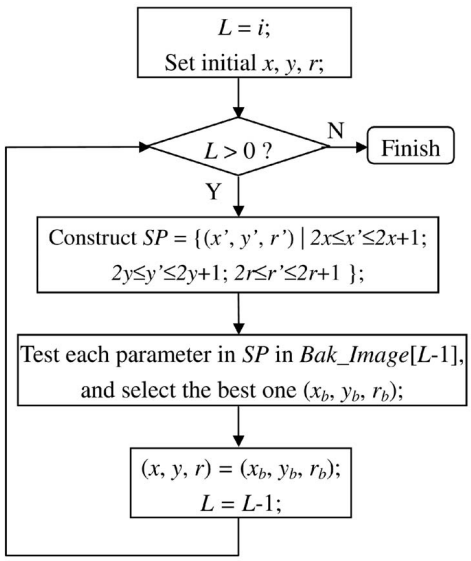


Fig. 6. Layer-by-layer arc localization.

The score of a parameter is measured as the percentage of the pixels covered by the arc/circle with the parameter being black in the Bak_Image. After the parameter in the upper layer has been selected and before proceeding to the next iteration, the line thickness is recalculated by voting along the circular path, and the starting and ending angles (for arcs only) are also refined by extending two extremities of the arc.

3.5 Arc Verification

The last step of the MAS paradigm is arc verification, which evaluates the confidence of segmented arcs to minimize false detections. Actually, not all arcs need verification since the confidence of a big and long arc is already high; therefore, only small or short arcs should be verified, where "small" means the radius is small and "short" means the length of arc is short. The known difficulty of this task is how to distinguish misdetected arcs (cf. Figs. 7a and 7b) from true arcs in complex environment, e.g., arcs touching or intersecting other objects (Figs. 7c and 7d). Failures in distinguishing them will lead to suppressing false detections at the cost of raising miss rate. To handle this problem properly, this paper proposes an image-based confidence-weighted arc verification protocol.

This verification protocol divides an arc into a sequence of continuous black or white segments according to the pixel color on the medial line of the arc. Let $S_i (i = 1..k)$

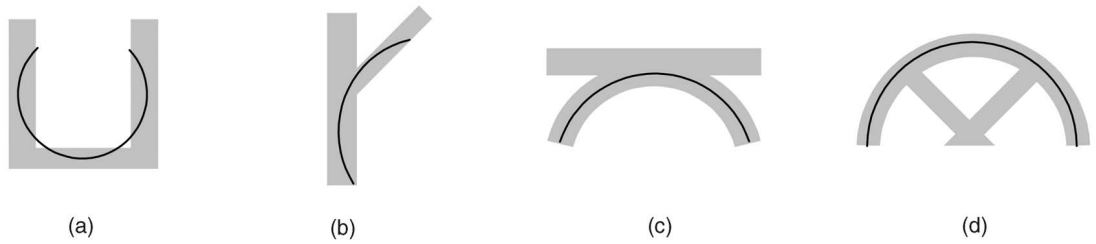


Fig. 7. Misdetected arcs and true arcs in a complex environment. (a) and (b) A false detection and (c) and (d) a true arc.

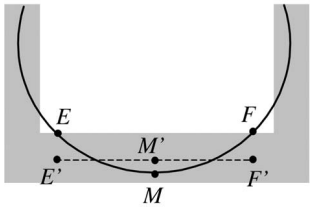


Fig. 8. Curvature check.

denote the sequence of segments, L denote the whole length of the arc, and l_i denote the length of S_i. The confidence of the arc (C) is calculated by (4), (5), (6), (7), and (8):

$$C = \frac{1}{L} \times \sum_{i=1}^{k} W_i^S, \quad 0 \le C \le 1.0, \tag{4}$$

$$W_i^S = \begin{cases} W_i^C - \sum_{j=1}^{l_i} W_j^T, & S_i \; is \; black \\ \sum_{j=1}^{l_i} W_j^D, & S_i \; is \; white, \end{cases} \tag{5}$$

$$W_i^C = \begin{cases} l_i \times P_N, & S_i \; is \; not \; linear \\ l_i \times P_L, & S_i \; is \; linear, \end{cases} \tag{6}$$

$$W_j^T = \begin{cases} 0, & |T_j - T| \le \\ & MAX(1,\ T/4) \\ MIN(P_T,\ |T_j - T| \times P_{\Delta T}), & otherwise, \end{cases} \tag{7}$$

$$W_j^D = MAX[0,\ P_D - (D_j - 1) \times P_{\Delta D}]. \tag{8}$$

According to (4), C is the weighted sum of the confidence of all segments, where the confidence of a segment S_i, denoted by W_i^S, is computed by (5). If S_i is black, W_i^S equals the curvature confidence minus the sum of abnormal-thickness confidence of all black points; otherwise (i.e., S_i is white), W_i^S is the sum of distance-to-black confidence of all white points. The reasons for the definition of W_i^S are described as follows: For a black segment, its curvature certainly increases the confidence, while the abnormal line thickness at some points lowers the confidence. For a white segment, no matter a real gap or just a degraded part, it should carry low confidence. However, the confidence of a white point ought to be related to its perpendicular distance to the black areas.

The curvature confidence of S_i, denoted by W_i^C, is calculated by (6). The curvature check is performed as shown in Fig. 8. For the endpoints E, F, and the midpoint M of the black segment, we locate their axis points on the image, i.e., E', F', and M' in Fig. 8. If E', F', and M' are not linear, it means the curvature of this segment is apparent. W_i^C is the product of l_i and a probabilistic constant (P_N or P_L), which indicates the probability of a nonlinear or linear segment being on a real arc.

The abnormal-thickness confidence of the jth point in a black segment, denoted by W_j^T, is calculated by (7), where T_j is the line thickness at the jth point and T is the line thickness of the current arc. If T_j and T are similar, W_j^T is zero, which means no deduction from W_i^S; otherwise, the more the difference between T_j and T is, the bigger W_j^T is, which consequently causes more deduction from W_i^S. The statistical result confirms that W_j^T is positively related, but not linearly, to the difference between T_j and T. To accelerate computation, (7) uses an approximated linear function, where P_T is the upper bound of deduction and $P_{\Delta T}$ is the increment of deduction.

The distance-to-black confidence of the jth point in a white segment, denoted by W_j^D, is calculated by (8), where $D_j (D_j \ge 1)$ is the minimal distance between the jth point and its neighboring black pixels. Zero-valued W_j^D indicates a complete gap. The statistical result shows that W_j^D is negatively related, but not linearly, to D_j. To accelerate computation, (8) uses an approximated linear function, where P_D is the upper bound of confidence and $P_{\Delta D}$ is the decrement of confidence.

The values of above-mentioned probabilities P_N, P_L, P_T, $P_{\Delta T}$, P_D, and $P_{\Delta D}$ are determined by the statistical study of 80 test images, including both synthetic images and scanned images of mechanical drawings and construction drawings. We first perform the MAS method without verification on each image, then manually delete false positives and, finally, use the measuring approaches of verification to obtain the following quantities of small or short arcs for each image:

1. NBS: the number of black segments,
2. NBSL: the number of linear black segments,
3. NBP: the number of black pixels on median lines of arcs,
4. NWP: the number of white pixels on median lines of arcs,
5. $H_T[1 \sim 10]$: the histogram of $|T_j - T|$ (7) for black pixels on median lines, and
6. $H_D[1 \sim 10]$: the histogram of D_j (8) for white pixels on median lines.

Note that all values larger than 10 contribute to $H_T[10]$ or $H_D[10]$, respectively. After summing up these six quantities of all images, we can calculate the probabilities as follows:

1. $P_L = N_{BSL}/N_{BS}$,
2. $P_N = 1 - P_L$,
3. $P_T = H_T[1]/N_{BP}$,
4. $P_{\Delta T}$ is obtained by linear fitting of $H_T[1 \sim 10]$,
5. $P_D = H_D[1]/N_{WP}$, and
6. $P_{\Delta D}$ is obtained by linear fitting of $H_D[1 \sim 10]$.

The values of P_N, P_L, P_T, $P_{\Delta T}$, P_D, and $P_{\Delta D}$ determined by the statistical study are 0.85, 0.19, 0.62, 0.09, 0.23, and 0.06, respectively. According to (4) and (5), the upper bound of C is P_N; therefore, these six weights are scaled by the same factor in our algorithm to be 1.0, 0.22, 0.53, 0.11, 0.27, and 0.07, respectively. Note that P_T is inversely scaled. A confidence threshold, denoted by C_{min}, is set to be 0.75 accordingly. With (4), (5), (6), (7), and (8), we can obtain the overall confidence of an arc. An arc with $C \ge C_{min}$ is added to the final result; otherwise, it is discarded. The proposed arc verification protocol is quite robust, as shown in the experimental results in Section 4, since it relies on the original image and evaluates the curvature, gaps, and line thickness simultaneously.

4 Performance Evaluation

The proposed MAS method has been implemented in a graphics recognition system—VHVector, which takes scanned images as input and produces graphics files. We utilize the test images with ground-truth graphics files of the GREC '01 and GREC '03 arc segmentation contests [23],

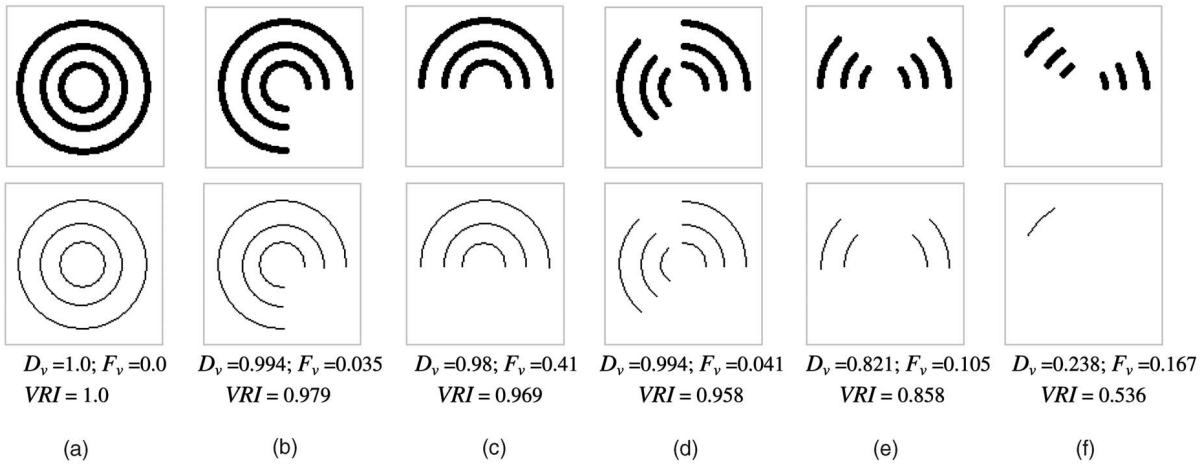


Fig. 9. Arc segmentation results over various arc angles and lengths.

[31] to perform a systematic evaluation of the MAS method. The performance evaluation tool [32] is the same as the one used in these contests. It generates a Vector Recovery Index (VRI) taking into account the localization accuracy, the endpoint precision, and the line thickness accuracy. VRI is computed by $0.5 \times D_v + 0.5 \times (1 - F_v)$, where D_v (correct detection rate) is the length-weighted sum of the fraction of every ground truth arc being correctly detected, and F_v (false detection rate) is the length-weighted sum of the fraction of every detected arc being false. VRI, D_v, and F_v all range between 0 and 1. A good arc segmentation method should achieve high D_v, low F_v and, therefore, high VRI.

We first evaluate the detection capability over various arc angles. The testing images are a group of arcs/circles with various subtending angles and radii, as shown in the first row of Fig. 9. The radii of the inner, middle, and outer arcs/circles are 18, 32, and 50 pixels, respectively. Since the radii are relatively small, these images also examine the detection capability over short arc lengths. The segmentation results are shown in the second row of Fig. 9, where most arcs/circles are correctly detected. The quantitative evaluation result is listed under each result. The D_vs of Figs. 9a, 9b, 9c, and 9d are very high and the F_vs are low. The D_v of Fig. 9e drops a little due to missing two short inner arcs, while the longer arcs with the same subtending angles are correctly detected, which demonstrates that the cause of missing is not the small subtending angle, but the short arc length. The D_v of Fig. 9f drops to 0.238 because only one arc is correctly detected. It should be mentioned that the reason for missing the outmost arc at the right side and that for missing the other four arcs are not the same. The outmost arc has firstly been detected and, finally, been discarded by the arc verification due to lacking curvature, while the other four arcs have never been detected since they are too short. This evaluation result shows that the proposed method works well for the arcs with detectable curvature, determined by the arc seed detection.

Second, we evaluate the arc segmentation performance over various line thicknesses. Fig. 10 displays five test images containing the same group of arcs/circles with various line thicknesses. The experimental result of each image is displayed on the right side, demonstrating that the arc segmentation performance is very robust over a wide range of line thickness. The D_vs of thin arcs (thickness ≤ 3) are a little lower than those of thick arcs since the proposed method, being a pixel tracking method, prefers thick arcs. However, we also find the F_vs of thick arcs (thickness ≥ 7) are a little higher than others, since the localization of thick arcs is less accurate. σ is the average difference between the ground-truth line thickness and the detected one. We can see that is σ very small, far below 1 pixel.

The above two experiments are performed on clean synthetic images. Next, we evaluate the performance under noisy environment. Four types of noises are added to the test images: Gauss noise, hard pencil noise, high frequency noise, and geometry noise. The noise models and the algorithms to generate them are described in [32]. As shown in Fig. 11, the Gauss noise simulates the random noise introduced during scanning paper drawings, the hard pencil noise simulates the noise caused by using hard pencils to paint drawings, the high frequency noise simulates the abrasion effects during the use of paper drawings, and the geometry noise simulates the effects of drawing arcs/circles by hand or the distortion caused by scanning. Each type of noise is added to the test image in several levels, where higher levels result in heavier noises. Fig. 11 shows the four types of noisy images with relatively high levels.

The numbers of levels of the four types of noisy images are different. The Gauss noise images contain 10 levels. The hard pencil noise images contain seven levels. The high

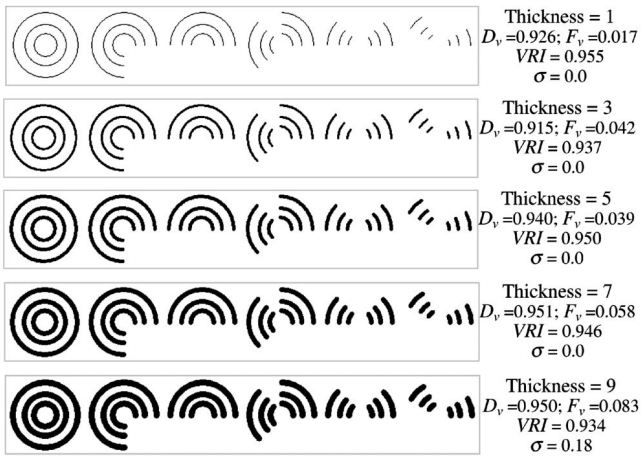


Fig. 10. Test images of various line thickness.

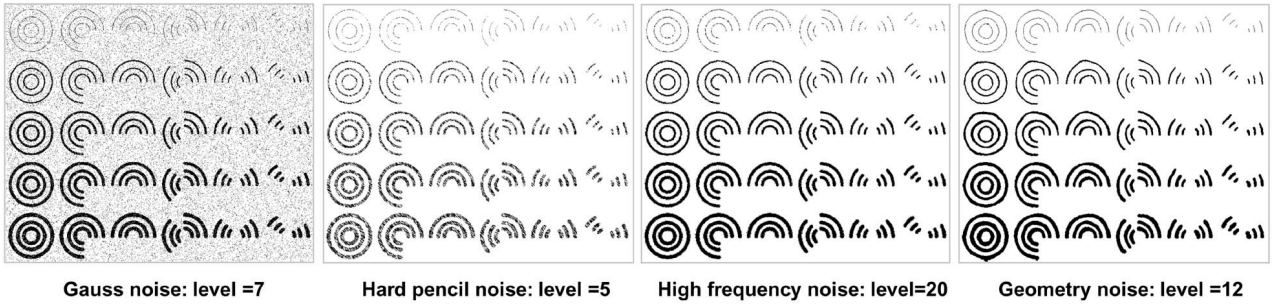


Fig. 11. Four types of noisy images.

frequency noise images contain 25 levels. The geometry noise images contain 15 levels. For all the four types, the zero-leveled image contains no noise, i.e., it is the same as the original image. The performance-to-noise-level curves of the four types of noisy images are plotted in Fig. 12. Note that some noise levels do not have corresponding data points since the noisy images of those levels are not available.

A simple connected-component-based despeckle filter is placed before the MAS method to simulate the noise removal effect of the scanning process. This filter alleviates noises of Gauss, hard pencil, and high frequency types at lower levels, but it has no effect on geometry noises, which are not pixel-level noises. With the effect of the filter, the D_v curve of Gauss noise keeps high when noise level is below 5. After that, it drops gradually. The D_v curve of hard pencil noise and that of high frequency noise also drop faster after the level 3 and the level 13, respectively. We observe that even at high noise levels, the F_v curves of Gauss, hard pencil, and high frequency noise types always stay very low, which means that although the arc image is degraded too much to be detected correctly, the MAS method is still robust enough to avoid producing many false detections. The performance curves in Fig. 12d illustrate that the geometry noise affects the MAS algorithm most. For the test images containing geometry noise, most arcs/circles are still segmented; however, since this type of noise distorts the circular shape of arcs/circles, the parameter of a segmented arc/circle often offsets from its ground-truth parameter, which lowers the performance.

The above evaluations focus on isolated arcs/circles. Next, we evaluate the performance of the MAS method when an arc is intersecting other arcs or lines. This experiment includes two parts: arc-arc intersection and arc-line intersection. The test images of arc-arc intersection contain five images, where the small circle moves from the outside of the big circle to the inside gradually, as shown in Fig. 13. The segmentation results and performance indices are also shown in Fig. 13. Both circles are successfully detected in all the five images, demonstrating the MAS method handles arc-arc intersections well. The centers of segmented circles, except those in a04.tif, have one-pixel offsets. Since there are only two circles in an image and the circles are thin, even one-pixel offset affects the performance indexes obviously. The segmented circles in a04.tif have no offset; therefore, the performance is perfect.

The test images of arc-line intersection contain 15 images, which emphasize the case that a line is tangent to an arc, as shown in Fig. 14. This is considered as one of the hardest problem for vectorization-based arc segmentation methods since the overlapped part becomes a shared vector of both the arc and the line after vectorization. Thus, it is difficult to keep the entirety of both the arc and the line. Moreover, the distortion of the shared vector may cause offset to the arc's parameter. The proposed MAS method is a direct segmentation method without vectorization; therefore, it does not suffer from this limitation. The arcs in all the fifteen images are successfully segmented. The performance curves for the sequence of images are plotted in Fig. 15. We observe that most VRIs are higher than 0.9. The performance drops a little at L111, L112, and L113 because the detected line thickness has one-pixel offset from the ground-truth line thickness. Overall, the experiments on arc-arc intersection and arc-line intersection demonstrate that the MAS method has a good capability in handling intersections.

Next, we show the experimental results on three real images used in the GREC '01 contest, as shown in the first row of Fig. 16. P1.tif (707×590 pixel^2) is scanned from a small drawing with a moderate resolution. P2.tif (503×500 pixel^2) and P3.tif ($2,125 \times 2,113$ pixel^2) are scanned from the same

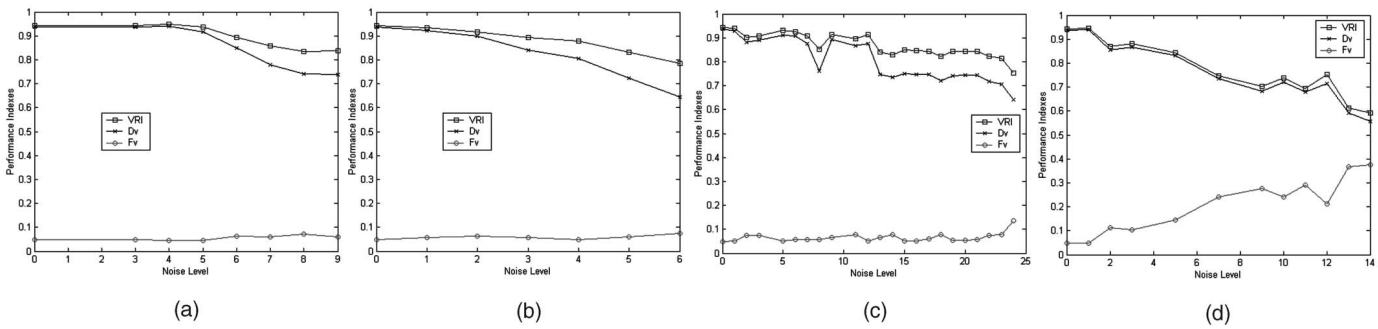


Fig. 12. Performance-to-noise-level curves. (a) Gauss noise, (b) hard pencil noise, (c) high frequency noise, and (d) geometry noise.

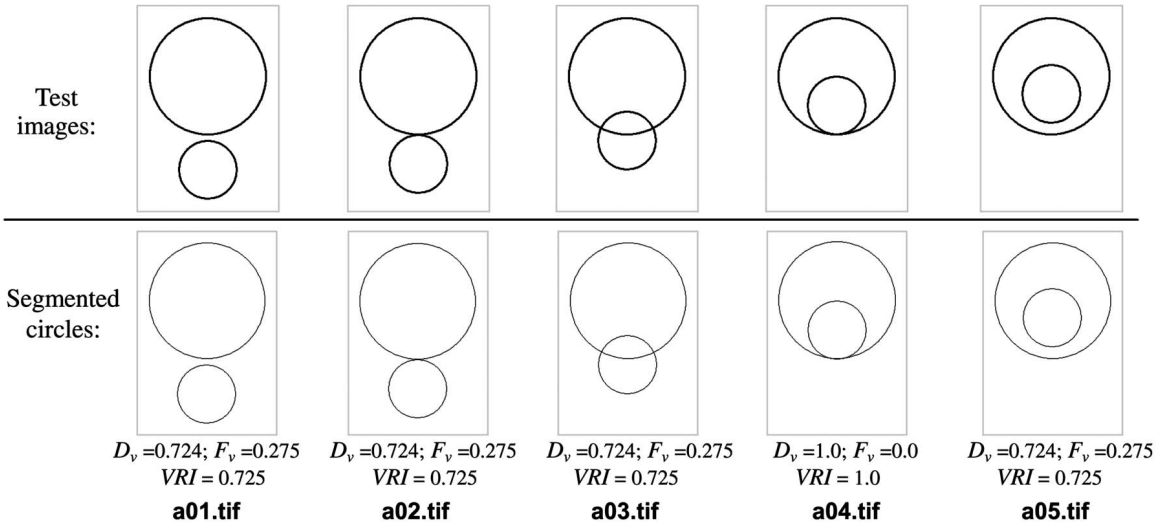


Fig. 13. Experimental results of arc-arc intersection.

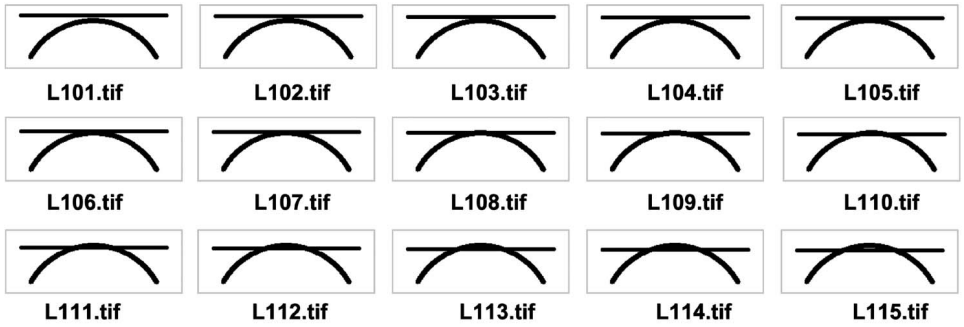


Fig. 14. Test images for arc-line intersection.

drawing with a low resolution and a high resolution, respectively. The quality of P1.tif is satisfactory, but most circles are not circular enough, resembling the geometry noise. The quality of the other two images is not good, especially P2.tif, where many arcs are thin and disconnected. The results of segmented arcs/circles and the performance evaluation indices are displayed below the images, respectively. The experimental results demonstrate that the MAS method is robust over different resolutions; however, being a pixel-tracking method, it prefers a high resolution, which implies a high signal-to-noise ratio. The correct detection rate of P2.tif drops to 0.722 because there are two concentric arcs that are too close to be distinguished; therefore, only the longer one is successfully detected.

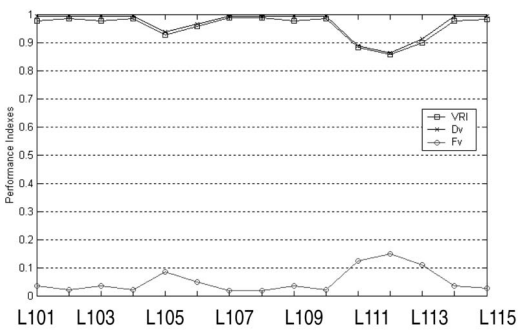
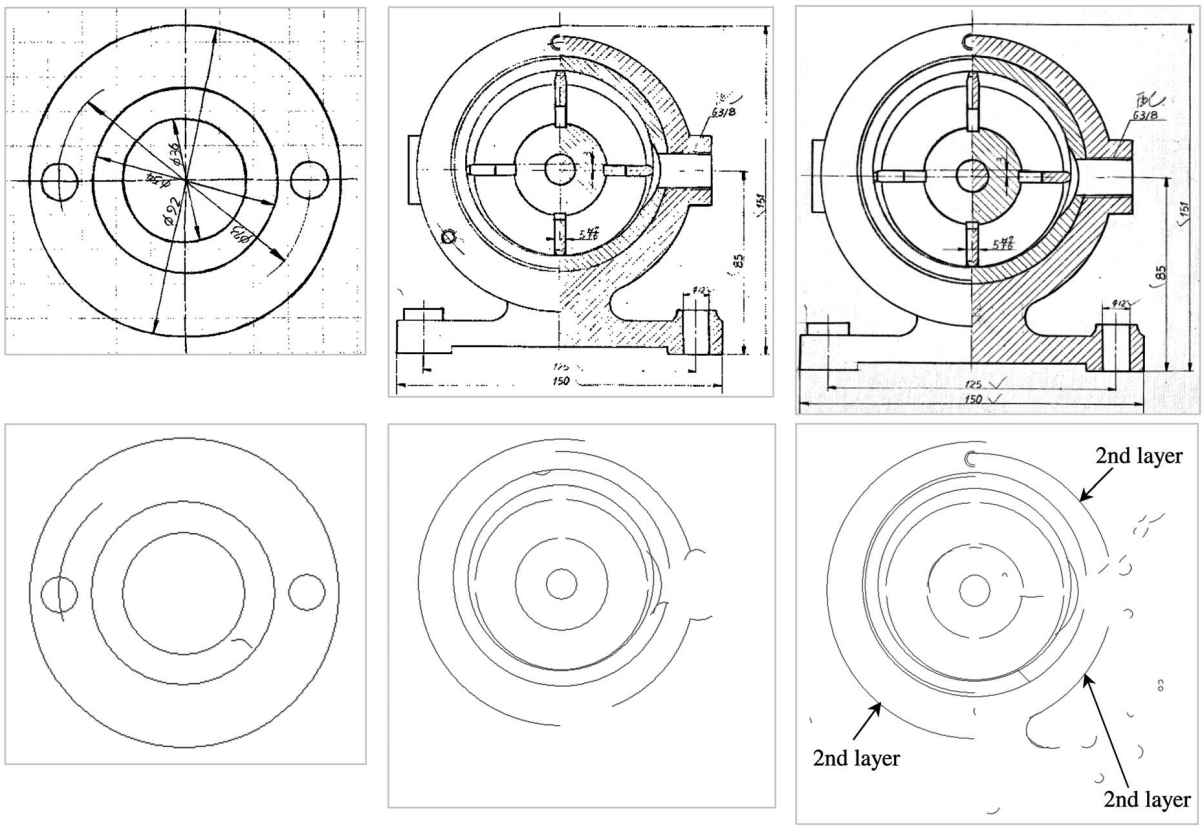


Fig. 15. Performance curves of arc-line intersection images.

We use the same set of contest images to compare the proposed MAS method with the other three methods, including the two methods participating in the GREC '01 contest [20], [21] and the IAS method [15], which are all vectorization-based methods. The comparison result is shown in Table 1, where the performance scores of [20], [21] are obtained from the contest report [24]. It should be mentioned that Hilaire's system crashed on the scanned image P3.tif and, therefore, obtained a zero score. Table 1 shows that Hilaire's method and the MAS method have similar good performance on synthesized images. Hilaire's method obtained the highest scores on the synthetic image with high frequency noise and that with geometry noise, while our method got the highest scores on the other two synthetic images. For the average performance on the synthetic images, our method outperforms all other methods.

On the other hand, for the scanned images, the performance of the other three methods drops dramatically, especially for large images, while our method always keeps relatively high scores due to using the MAS paradigm. Note P1.tif and P2.tif are small so that the arc segmentation only performs at the original resolution. For P3.tif, two resolution levels are generated and, therefore, those big arcs (indicated in Fig. 16) are correctly segmented from the second layer. Consequently, the average score of our method on scanned images is significantly higher than those of the other methods. As a summary, the overall score (VRI_{all}) is computed as the average of the two averaged scores of



P1.tif
$D_v = 0.798$; $F_v = 0.164$;
$VRI = 0.817$.

P2.tif
$D_v = 0.722$; $F_v = 0.228$;
$VRI = 0.747$.

P3.tif
$D_v = 0.868$; $F_v = 0.225$;
$VRI = 0.822$.

Fig. 16. Experimental results on real scanned images.

synthesized images and scanned images. The VRI_{all} of our method far exceeds those of the other methods.

The GREC '03 Arc Segmentation Contest [31] introduces a set of test images emphasizing on the segmentation of tangent connection of arcs (Fig. 17). The sizes of these images are small; therefore, the MAS method only works in the original resolution. The performance comparison between our method and the other two methods is reported in Table 2. Overall, our method outperforms Elliman's method [20], but lags behind the IAS method [15]. Some typical failure modes of our method can be seen from 1_n4.tif, 3_100.tif, and 3_n4.tif. For 1_n4.tif, the noise-added version of 1.tif, our method missegments the arc and the circle located closely in the center of the image into one thick circle because they are blurred by noises. For 3_100.tif and 3_n4.tif, which are the low-resolution version and the noise-added version of 3.tif, respectively, our method fails to locate the connection points between tangent arcs correctly, resulting in inaccurate arc parameters. Another reason for the VRI performance drop of our method is that these images contain some small and short arcs, such as the cross arcs on arrows, which are mostly discarded by the arc verification.

We also use the large-sized scanned test images of GREC '99 contest [33] and other real scanned images to test the proposed MAS method. Fig. 18 shows a real scanned image named "test1.tif," which contains many filled areas. The proposed method segments all circles, including those in filled areas, correctly from test1.tif. Fig. 18 also shows the segmentation result of a cloud-shape overlapping lines and texts. Our method segments four pieces of arcs correctly, misses two others, and produces three false positives. The top-center arc is missed due to failure in detecting arc seeds, while the top-right arc is rejected by the verification. The false positives are caused by curve strokes in characters. For the same image, the IAS method breaks three circles into arcs and produces much more false positives. For these large images, we employ a goal-directed performance evaluation protocol proposed by Chhabra and Philips [34], [35], [36], which estimates the manual editing cost to correct the mistakes generated by automatic graphics recognition. They defined the editing cost as follows:

$$EditingCost = w1 \cdot false\_alarms + w2 \cdot misses + w3 \cdot one2many + w4 \cdot many2one,$$

where $one2many$ is the count of one ground truth object being converted into many objects, and $many2one$ is the converse. Editing a $false\_alarm$ (or a $miss$) needs one removal (or adding) operation, while the editing costs of $one2many$ (or $many2one$) errors are more complex and depend on the editing tool used. In [34], $w1$ through $w4$ are set to be 1 for simplicity. However, we set $w3$ and $w4$ to be 2 in our evaluation since a $one2many$ (or $many2one$) error needs at least one removal plus one adding operations. Table 3 reports the evaluation result of the proposed MAS

TABLE 1
Performance Comparison of Three Methods with GREC '01 Contest Test Images

Test Images	Synthesized Images (*.tif)				Scanned Images (*.tif)		
	Gau_05	**Frq_03**	**Pen_03**	**Geo_03**	**P1**	**P2**	**P3**
D. Elliman [20]'s *VRI*	0.904	0.853	0.896	0.927	0.547	0.482	0.371
	Average: 0.895				Average: 0.467		
	Overall (VRI_{all}): 0.681						
X. Hilaire [21]'s *VRI*	0.891	**0.889**	0.944	**0.958**	0.707	0.311	*0*
	Average: 0.921				Average: 0.339		
	Overall (VRI_{all}): 0.630						
IAS [15]'s *VRI*	0.892	0.850	0.808	0.918	**0.836**	0.692	0.420
	Average: 0.867				Average: 0.649		
	Overall (VRI_{all}): 0.758						
Our *VRI*	**0.951**	0.870	**0.954**	0.945	0.817	**0.747**	**0.822**
	Average: **0.930**				Average: **0.795**		
	Overall (VRI_{all}): **0.863**						

method compared with the other two methods, SCT [22] and IAS [15].

Table 3 shows that the false alarms of MAS and IAS are significantly higher than that of SCT, while the misses of SCT are significantly higher than the other two. The former is because the minimum detectable radius of both MAS and IAS are smaller than that of SCT. The arc verification of the MAS method greatly reduces false positives compared with the IAS method. The remaining false positives are mostly caused by curve strokes of text. Since the current arc verification algorithm does not consider environmental evidence, it cannot discover such false positives. The latter is due to the fact that SCT misses many big arcs/circles and very small circles. Although SCT enjoys the lowest editing cost, its subjective performance is lower than that of the MAS method since big arcs/circles are missing.

Since the proposed MAS method improves the segmentation capability by a complex multiresolution mechanism, we need to analyze its time complexity. The size of the largest scanned image we tested reaches $16,215 \times 11,856$ pixel^2, requiring four resolution levels for the arc segmentation. Table 4 shows the processing time and its distribution for large-sized images, which are tested on a PC with P4/2.4G CPU and 1G RAM.

From Table 4, we find that the processing time spent on $Layer[0]$ takes on average 66 percent of the total processing time. The most time-consuming part is the recognition, including arc seed detection and circular tracking. When R_{max} (the maximum detectable radius) is large, the tracking is slow. If R_{max} is changed from 600 to 300, the total processing time decreases by 30 percent. The time for localization is high in $Layer[0]$ where most arcs/circles are segmented. From $Layer[1]$ to $Layer[N]$, the time for localization increases gradually. For large images, the time for preparation for the next layer, i.e., downsizing images, is not neglectable. The clearing-up process eliminates overlapping arcs caused by the following case: An arc/circle is segmented incompletely from one seed, and then its residual in the $Work\_Image$ forms another seed for a complete segmentation since the tracking works in the Bak_Image. We can see that the speed of the proposed MAS

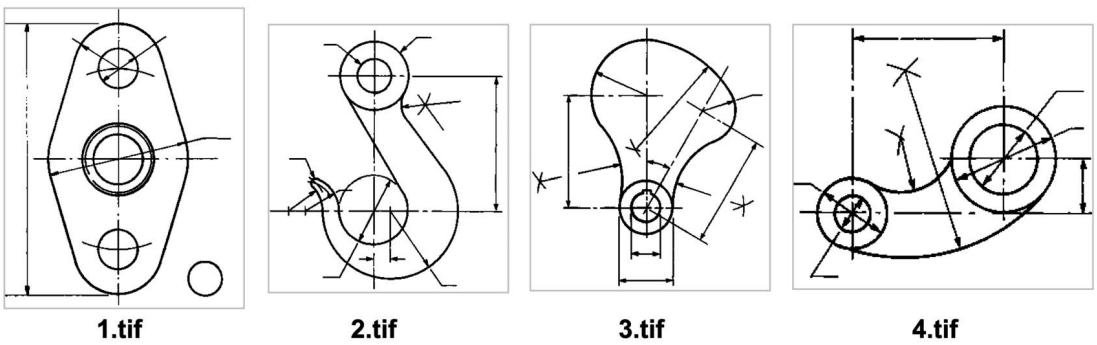


Fig. 17. Part of GREC '03 contest test images.

TABLE 2
Performance Comparison on GREC '03 Contest Test Images

Image	Our scores			D. Elliman [20]'s scores			IAS [15]'s scores		
(*.tif)	*Dv*	*Fv*	*VRI*	*Dv*	*Fv*	*VRI*	*Dv*	*Fv*	*VRI*
1	0.553	0.272	0.641	0.462	0.329	0.567	0.781	0.317	**0.732**
2	0.789	0.283	0.753	0.359	0.482	0.439	0.895	0.247	**0.824**
3	0.482	0.417	0.532	0.197	0.653	0.272	0.462	0.293	**0.584**
4	0.742	0.272	0.735	0.000	0.000	0.500	0.810	0.219	**0.796**
1_230	0.553	0.273	0.640	0.481	0.302	0.589	0.773	0.315	**0.729**
2_100	0.780	0.208	0.786	0.426	0.400	0.513	0.794	0.159	**0.817**
3_100	0.290	0.688	0.301	0.417	0.379	0.519	0.486	0.336	**0.575**
4_230	0.803	0.223	0.790	0.188	0.543	0.323	0.874	0.137	**0.869**
1_n4	0.446	0.428	0.509	0.488	0.159	0.664	0.835	0.223	**0.806**
2_n4	0.700	0.294	0.703	0.506	0.282	0.612	0.801	0.359	**0.721**
3_n4	0.199	0.750	0.224	0.327	0.425	0.451	0.487	0.512	**0.488**
4_n4	0.680	0.305	0.688	0.283	0.485	0.399	0.821	0.155	**0.833**
Average	0.585	0.368	0.609	0.345	0.370	0.487	0.735	0.273	**0.731**

method is not fast but still acceptable. We represent the total processing time as follows:

$$T_{all} \approx \sum_{i=0..k-1} [T_{dt}(i) + T_{tr}(i) + T_{er}(i)] + \sum_{i=0..k-2} T_{ds}(i) + T_{lc} + T_{vr},$$

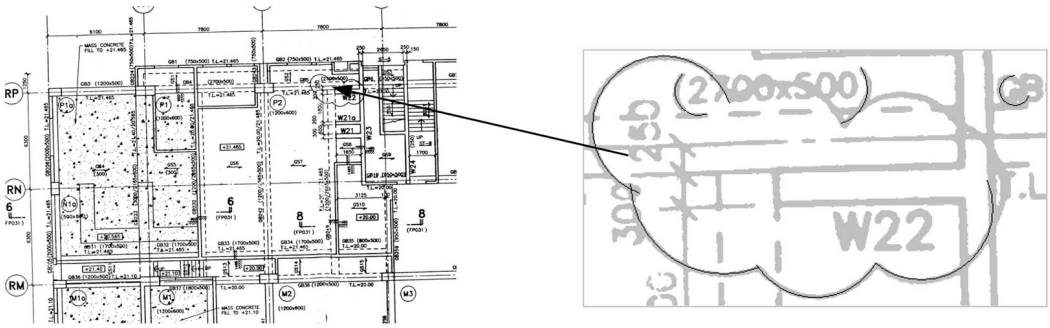
where k is the number of layers and T_{dt}, T_{tr}, T_{er}, T_{ds}, T_{lc}, and T_{vr} stand for the time for detection, tracking, erasing, downsizing, localization, and verification, respectively. Thus, the order of computational complexity of the MAS method in terms of accessing pixels can be evaluated as follows:

$$\begin{aligned} O(T_{all}) &= \sum_{i=0}^{k-1} \left[O\left(\frac{N_b}{4^i}\right) + O\left(\frac{N_g}{4^i}\right) + O\left(\frac{N_g}{4^i}\right) \right] + \\ &\quad \sum_{i=0}^{k-2} \left[2 \cdot O\left(\frac{N_p}{4^i} + \frac{N_p}{4^{i+1}}\right) \right] + \sum_{i=1}^{k-1} O\left(\frac{8 \cdot N_g}{4^i}\right) + O(N_g) \\ &= \sum_{i=0}^{k-1} \left[O\left(\frac{N_b}{4^i}\right) + O\left(\frac{2 \cdot N_g}{4^i}\right) \right] + \\ &\quad \sum_{i=1}^{k-1} \left[O\left(\frac{10 \cdot N_p}{4^i}\right) + O\left(\frac{8 \cdot N_g}{4^i}\right) \right] + O(N_g), \end{aligned}$$

where N_p, N_b, and N_g denote the numbers of all pixels, black pixels, and ground-truth arc/circle pixels in an image, respectively. Therefore, the computational complexity depends much on N_g. This explains why the median-size image "2200.tif" takes the longest processing time in Table 4. Note when $k = 1, O(T_{all}) = O(N_b) + O(3 \cdot N_g)$. When $k > 1$, N_p begins to affect the computational complexity.

Table 5 shows the processing times of the other three methods on the same set of images. All of them are more time-efficient than the MAS method. The first two approaches are vectorization-based approaches, whose false alarms are more than ours. The SCT approach is a direct recognition approach, which is fast but does not handle big arcs/circles well.

Besides the probability parameters in the arc verification, the MAS method introduces nine parameters, whose relationship is shown in Fig. 19. First, R_{min} and W_{max} are set according to the domain-specific knowledge, and then other parameters can be set as Fig. 19, with rules described in Sections 3.1 and 3.2. The parameters directly affecting performance include k, R_{inner}, R_{outer}, and T_{size}. Their influence on performance based on the experimental results is shown in Table 6, where "+" denotes "positively related," "−" denotes "negatively related," and "∧" denotes "nonmonotonically related." The



(a) (b)

Fig. 18. (a) A fraction of test1.tif and (b) a fraction of segementation result.

TABLE 3
Evaluation of Editing Cost

Image	Approaches	*False alarms*	*Misses*	*One2many*	*Many2one*	*Editing Cost*
Pal152.tif	MAS	157	4	1	0	163
	SCT [22]	6	28	0	0	34
	IAS [15]	328	10	9	0	356
2012_02.tif	MAS	193	5	0	0	198
	SCT [22]	27	14	0	0	41
	IAS [15]	392	1	3	0	399
P3.tif	MAS	18	2	1	0	22
	SCT [22]	1	20	0	0	21
	IAS [15]	136	4	11	0	162
Test1.tif	MAS	121	3	0	0	124
	SCT [22]	1	5	0	0	6
	IAS [15]	377	3	5	0	390

parameter values reported in Sections 3.1 and 3.2 are set according to the given rules, and they almost achieve peak performance in the experiments.

5 Conclusion and Future Work

This paper proposes a novel multiresolution arc segmentation method (MAS), which handles arcs/circles with a large range of radius well by detecting them in multiple resolution levels. A systematic performance evaluation of the MAS method has been conducted using the third-party evaluation tool, test images, and ground truth files obtained from the GREC arc segmentation contests. The MAS method demonstrates robust and satisfactory performance over different arc angles, arc lengths, line thickness, noise types, noise levels, arc-arc intersections, and arc-line

TABLE 4
Processing Time and Distribution for Large Images

Image	Size (pixel^2)	Layer	Multi-resolution processes (sec)			Clearing up (sec)	Verification (sec)	Total (sec)
			Recognition	Localization	Preparation for the next layer			
Pal152.tif	8256× 10885	0	34.78	4.34	12.97	4.56	0.27	80.67
		1	6.38	1.64	3.17			
		2	2.64	4.97	0.78			
		3	0.72	3.44	0.0			
Pal168.tif	8256× 10884	0	24.19	4.36	12.95	5.44	0.55	73.49
		1	4.63	1.80	3.14			
		2	3.61	5.47	0.78			
		3	1.63	4.94	0.0			
2032_01.tif	10776× 10989	0	40.23	7.86	17.11	3.30	0.25	91.34
		1	6.31	2.09	4.63			
		2	3.92	1.92	1.03			
		3	0.80	1.88	0.0			
2012_02.tif	10776× 13128	0	31.38	3.59	20.13	3.89	0.58	95.39
		1	9.58	2.20	4.91			
		2	5.55	3.74	1.22			
		3	3.09	5.53	0.0			
2200.tif	5712× 10874	0	47.27	12.08	8.95	7.88	0.38	102.67
		1	7.99	4.25	2.19			
		2	5.52	6.17	0.0			
P3.tif	2125× 2113	0	10.27	1.16	0.66	0.77	0.14	16.81
		1	2.61	1.22	0.0			
Test1.tif	3508× 2464	0	17.30	3.44	1.25	1.77	0.28	29.99
		1	4.28	1.67	0.0			

TABLE 5
Processing Times (Seconds) of Three Other Approaches

	Pal152.tif	Pal168.tif	2032_01.tif	2012_02.tif	2200.tif	P3.tif	Test1.tif
IAS [15]	7	4	5	3	2	3	3
Tif2Vec[20]	7	6	9	6	4	3	4
SCT [22]	2.86	2.56	2.50	2.88	1.95	1.16	1.58

intersections. We compare the MAS method with other methods, including those participating in the GREC arc segmentation contests, in terms of arc segmentation performance, editing cost, and time complexity analysis on both synthetic and real scanned images. The comparison result confirms the clear advantages of the proposed MAS method on handling arcs/circles with a large range of radius, even in complex environments. The experiment results also show that the MAS method tends to reject short and nearly flat arcs, and the endpoint precision needs further improvement.

A known limitation of the MAS method is the high time complexity, which needs further simplification. Future work will also develop the capability in segmenting dashed arcs/circles. The arc seed detection will distinguish curves from disconnected segments since one single segment of a dashed arc may not appear curvature. The detectable scope will be flexible to detect various dashed patterns. The circular tracking, arc localization, and arc verification will be improved to tolerate regularly distributed gaps, where heuristics on dashed patterns should apply.

ACKNOWLEDGMENTS

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong SAR, China (Project No. CUHK4182/03E).

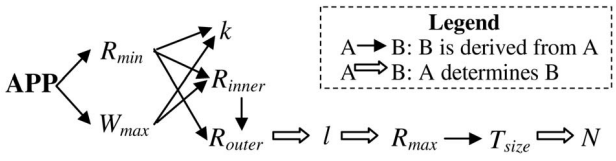


Fig. 19. Relationship of the parameters used. R_{min}: minimum detectable radius, W_{max}: maximum line thickness, R_{max}: maximum detectable radius, T_{size}: downsizable threshold, k: scanning interval, l: detectable scope, N: layer number, R_{inner}: inner window radius, and R_{outer}: outer window radius.

TABLE 6
Influence on Performance of Four Parameters

	Processing time	False alarms	Misses	*VRI*
k	−	−	+	∧
R_{inner}	+	−	+	∧
R_{outer}	∧	+	∧	∧
T_{size}	+*	−	+	∧

*Note: It depends on whether there exist large arcs/circles.

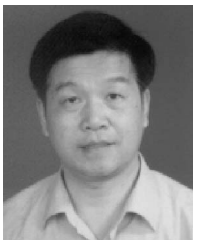
REFERENCES

[1] D.A. Anderson, "The Circular Structural Model," *J. Royal Statistical Soc., B*, vol. 43, pp. 131-141, 1981.
[2] D.H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111-122, 1981.
[3] M. Berman and D. Culpin, "The Statistical Behavior of Some Least Squares Estimator of the Center and Radius of a Circle," *J. Royal Statistical Soc., B*, vol. 48, pp. 183-196, 1986.
[4] P.L. Rosin and G.A. West, "Segmentation of Edges into Lines and Arcs," *Image and Vision Computing*, vol. 7, no. 2, pp. 109-114, May 1989.
[5] I. Amir, "Algorithm for Finding the Center of Circular Fiducials," *Computer Vision, Graphics, Image Process*, vol. 49, no. 3, pp. 398-406, 1990.
[6] V.A. Kovalevsky, "New Definition and Fast Recognition of Digital Straight Segments and Arcs," *Proc. 10th Int'l Conf. Pattern Recognition (ICPR '90)*, vol. 2, pp. 31-34, 1990.
[7] L. Moura and R. Kitney, "A Direct Method for Least-Squares Circle Fitting," *Computer Physics Comm.*, vol. 64, no. 1, pp. 57-63, 1991.
[8] V.F. Leavers, "The Dynamic Generalized Hough Transform: Its Relationship to the Probabilistic Hough Transforms and an Application to the Concurrent Detection of Circles and Ellipses," *Computer Vision, Graphics, Image Understanding*, vol. 56, no. 3, pp. 381-398, 1992.
[9] K.K.Y. Raymond, K.S.T. Peter, and N.K.L. Dennis, "Modification of Hough Transform for Circles and Ellipses Detection Using a 2-Dimensional Array," *Pattern Recognition*, vol. 25, no. 9, pp. 1007-1022, 1992.
[10] B.B. Chaudhori and P. Kundu, "Optimum Circular Fit to Weighted Data in Multi-Dimensional Space," *Pattern Recognition Letters*, vol. 14, no. 1, pp. 1-6, 1993.
[11] C. Ho and L. Chen, "A Fast Ellipse/Circle Detector Using Geometric Symmetry," *Pattern Recognition*, vol. 28, no. 1, pp. 117-124, 1995.
[12] Z. Wu, L. Wu, and A. Wu, "The Robust Algorithms for Finding the Center of an Arc," *Computer Vision and Image Understanding*, vol. 62, no. 3, pp. 269-278, 1995.
[13] D. Dori, "Vector-Based Arc Segmentation in the Machine Drawing Understanding System Environment," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 11, pp. 1057-1068, Nov. 1995.
[14] D. Dori and W. Liu, "Stepwise Recovery of Arc Segmentation in Complex Line Environments," *Int'l J. Document Analysis and Recognition*, vol. 1, no. 1, pp. 62-71, 1998.
[15] W. Liu and D. Dori, "Incremental Arc Segmentation Algorithm and Its Evaluation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 424-431, Apr. 1998.
[16] Y. Wei, "Circle Detection Using Improved Dynamic Generalized Hough Transform (IDGHT)," *Proc. IEEE Int'l Symp. Geoscience and Remote Sensing*, vol. 2, pp. 1190-1192, 1998.
[17] Y. Wei, "A Fast Finding and Fitting Algorithm to Detect Circles," *Proc. IEEE Int'l Symp. Geoscience and Remote Sensing*, vol. 2, pp. 1187-1189, 1998.
[18] P. Dosch, G. Masini, and K. Tombre, "Improving Arc Detection in Graphics Recognition," *Proc. 15th Int'l Conf. Pattern Recognition (ICPR '00)*, vol. 2, pp. 243-246, 2000.
[19] J. Song, F. Su, C.-L. Tai, J. Chen, and S. Cai, "Line Net Global Vectorization: an Algorithm and Its Performance Evaluation," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition (CVPR '00)*, vol. 1, pp. 383-388, June 2000.
[20] D. Elliman, "TIF2VEC, An Algorithm for Arc Segmentation in Engineering Drawings," *Graphics Recognition—Algorithms and Applications*, D. Blostein and Y.-B. Kwon, eds., vol. 2390, *Lecture Notes in Computer Science*, Berlin, Germany: Springer-Verlag, pp. 350-358, 2002.

[21] X. Hilaire, "RANVEC and the Arc Segmentation Contest," *Graphics Recognition—Algorithms and Applications,* D. Blostein and Y.-B. Kwon, eds., vol. 2390, *Lecture Notes in Computer Science,* Berlin, Germany: Springer-Verlag, pp. 359-364, 2002.
[22] J. Song, F. Su, C.-L. Tai, J. Chen, and S. Cai, "An Object-Oriented Progressive-Simplification Based Vectorization System for Engineering Drawings: Model, Algorithm and Performance," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 8, pp. 1048-1060, Aug. 2002.
[23] http://www.cs.cityu.edu.hk/~liuwy/ArcContest/ArcSegContest.htm, 2002.
[24] W. Liu, J. Zhai, and D. Dori, "Extended Summary of the Arc Segmentation Contest," *Graphics Recognition—Algorithms and Applications,* D. Blostein and Y.-B. Kwon, eds., vol. 2390, *Lecture Notes in Computer Science,* Berlin, Germany: Springer-Verlag, pp. 343-349, 2002.
[25] R.D.T. Janssen and A.M. Vossepoel, "Adaptive Vectorization of Line Drawing Images," *Computer Vision and Image Understanding,* vol. 65, no. 1, pp. 38-56, 1997.
[26] O. Hori and S. Tanigawa, "Rastor-to-Vector Conversion by Line Fitting Based on Contours and Skeletons," *Proc. Int'l Conf. Document Analysis and Recognition (ICDAR '93),* pp. 272-281, 1993.
[27] J.B. Roseborough and H. Murase, "Partial Eigenvalue Decomposition for Large Image Sets Using Run-Length Encoding," *Pattern Recognition,* vol. 28, no. 3, pp. 421-430, 1995.
[28] D. Dori and W. Liu, "Sparse Pixel Vectorization: An Algorithm and Its Performance Evaluation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 3, pp. 202-215, Mar. 1999.
[29] J.Y. Chiang, S.C. Tue, and Y.C. Leu et al., "A New Algorithm for Line Image Vectorization," *Pattern Recognition,* vol. 31, no. 10, pp. 1541-1549, 1998.
[30] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice.* Reading, Mass.: Addison-Wesley, 1990.
[31] http://www.cvc.uab.es/grec2003/contest.htm, 2004.
[32] W. Liu, J. Zhai, D. Dori, and L. Tang, "A System for Performance Evaluation of Arc Segmentation Algorithms," *Proc. Third CVPR Workshop Empirical Evaluation Methods in Computer Vision,* http://www.cs.cityu.edu.hk/liuwy/ArcContest/NoiseModels.pdf, 2001.
[33] http://www.iapr-tc10.or.kr/file/contest99.htm, 2004.
[34] I.T. Phillips and A.K. Chhabra, "Empirical Performance Evaluation of Graphics Recognition Systems," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 9, pp. 849-870, Sept. 1999.
[35] A.K. Chhabra and I.T. Phillips, "A Benchmark for Graphics Recognition Systems," *Proc. IEEE Computer Soc. Workshop Empirical Evaluation Methods in Computer Vision,* pp. 28-38, June 1998.
[36] A.K. Chhabra and I.T. Phillips, "Performance Evaluation of Line Drawing Recognition Systems," *Proc. 15th Int'l Conf. Pattern Recognition,* vol. 4, pp. 864-869, Sept. 2000.

Jiqiang Song received the BS degree in computer science and the PhD degree in computer science and application from Nanjing University in 1996 and 2001, respectively. Currently, he is a postdoctoral fellow in the Department of Computer Science and Engineering at the Chinese University of Hong Kong. His research interests include graphics recognition, automatic interpretation of engineering drawings, image processing, and video processing. He has published more than 20 papers in these areas. He won the first place in the GREC 2003 Arc Segmentation Contest. He is a member of the IEEE.

Michael R. Lyu received the BS degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1981, the MS degree in computer engineering from the University of California, Santa Barbara, in 1985, and the PhD degree in computer science from the University of California, Los Angeles, in 1988. He is currently a professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong. He was with the Jet Propulsion Laboratory as a technical staff member from 1988 to 1990. From 1990 to 1992, he was with the Department of Electrical and Computer Engineering, The University of Iowa, Iowa City, as an assistant professor. From 1992 to 1995, he was a member of the technical staff in the applied research area of Bell Communications Research (Bellcore), Morristown, New Jersey. From 1995 to 1997, he was a research member of the technical staff at Bell Laboratories, Murray Hill, New Jersey. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, wireless communication networks, Web technologies, digital libraries, and E-commerce systems. He has published more than 170 refereed journal and conference papers in these areas. He received Best Paper Awards in ISSRE '98 and ISSRE '2003. He has participated in more than 30 industrial projects, and helped to develop many commercial systems and software tools. He was the editor of two book volumes: *Software Fault Tolerance* (New York: Wiley, 1995) and *The Handbook of Software Reliability Engineering* (Piscataway, NJ: IEEE and New York: McGraw-Hill, 1996). Dr. Lyu initiated the First International Symposium on Software Reliability Engineering (ISSRE) in 1990. He was the program chair for ISSRE '96, and has served in program committees for many conferences, including ISSRE, SRDS, HASE, ICECCS, ISIT, FTCS, DSN, ICDSN, EUROMICRO, APSEC, PRDC, PSAM, ICCCN, ISESE, and WWW. He was the general chair for ISSRE 2001 and the WWW10 program cochair. He has been frequently invited as a keynote or tutorial speaker to conferences and workshops in the US, Europe, and Asia. He served on the editorial board of *IEEE Transactions on Knowledge and Data Engineering* and has been an associate editor of *IEEE Transactions on Reliability and Journal of Information Science and Engineering*. Dr. Lyu is a fellow of the IEEE.



Shijie Cai graduated from the Department of Mathematics from Nanjing University in 1967. He is a professor in the Department of Computer Science and Technology, Nanjing University. His research interests include computer graphics, graphics recognition, image processing, and document analysis and recognition.