Incremental Arc Segmentation Algorithm and Its Evaluation

Liu Wenyin and Dov Dori

Abstract—We present an incremental arc segmentation algorithm, which recovers the vectorized arc fragments obtained by the vectorization of a piece of arc image in a stepwise fashion. Due to proper threshold selection and consistent checking of cocircularity of the assumed arc pieces, the algorithm accurately constructs arcs from the vector input. Nearly 200 synthetic arcs, ranging in radius from five to 50 pixels, in open angle from $1/8\pi$ to 2π, and in thickness from one to nine pixels, are used in the experiments and evaluation. Parts of six real drawings, containing about 200 arcs, are also processed. The algorithm works well for arc segments greater than 10 pixels in radius, $\pi/4$ in angle, and one pixel in width.

Index Terms—Arc segmentation, vectorization, performance evaluation, graphics recognition.

1 Introduction

THE problem of recognizing circular arcs from scanned line drawings, known as *arc segmentation*, is an open problem in the area of line drawing interpretation. The difficulty of arc segmentation stems from a number of sources. One difficulty is due to the relative complexity of the circle equation. Second, intersecting bars and arcs (especially those that intersect at small angles), tangent bars, and tangent arcs often smear the arc's neat image, creating larger black blobs that are hard to segment correctly. Third, the raster image of an arc may shrink disproportionately in different directions to an ellipse or another distorted form. Fourth, circular arcs are very frequently not full circles, and partial arcs, especially those with small angles, are more difficult to segment than full circles. Finally, noise at various levels and types adds to the difficulty of arc segmentation.

Early arc segmentation methods, e.g., [1], are based on Hough Transform (HT), which is a conventional method for object extraction from binary images. HT is normally used for arc segmentation in case of isolated points that potentially lie on circles or circular arcs. However, HT is generally too costly in both time and space to be applicable. Another group of arc segmentation methods, e.g., [2], is based on curvature estimation. Motivated by object recognition, the aim of these algorithms is to extract meaningful features from objects by estimating their edge curvature. This requires heavy pixel-level preprocessing, such as edge detection or thinning, to produce the required input of one-pixel-wide digital curves.

Perpendicular Bisector Tracing (PBT) [3] is the first vector-based arc segmentation method. It utilizes the fact that three different points determine a unique circle, and that the circle center is the intersection of the three perpendicular bisectors of the triangle formed by the three points. It finds a bar chain which may approximate the arc image. First, the two endpoints of the bar chain are selected as two of the three required points to uniquely determine the circle. The third point is found by tracing along the perpendicular bisector of the segment formed by the two endpoints. The tracing starts from the midpoint of the line segment connecting the two endpoints until the first black pixel of the arc image is encountered, and the entry point is recorded. The tracing continues until an exit point, which is the first white pixel, and visits few image pixels. Basing the arc segmentation on intermediate vectorization results inevitably increases the time efficiency. The space requirement is also very low, since the algorithm detects arcs one at a time. However, although the efficiency of both time and space of PBT is guaranteed, it may get trapped in some pitfalls, and a few points can be improved. First, the algorithm is not fully vector-based in a strict sense, since the tracing done to find the third point on the arc requires consecutive pixel visiting, which may be fooled by pixels of other graphic objects along the track. This is a major source of pitfalls, as tracing along a block pixel area (of some annotation line) requires more processing, while concentric arcs are often missed. Moreover, special cases, including full circles, ovals, and round corners, need special treatment.

In this paper, we present and evaluate an improved vector-based Incremental Arc Segmentation (IAS) algorithm. IAS takes as input the vector pieces of the arc image, which are bars and polylines, generated by a vectorization procedure, such as SPV [4]. These are clustered and used to reconstruct the original shape of the arc and determine its parameter values. Rather than finding all the fragments concurrently, it extends the segmented arc incrementally to both ends by finding the fragments one at a time while consistently checking the arc's components cocircularity. The algorithm yields accurate arc segmentation, as proved by the experiments and evaluation.

2 Vectorized Arc Fragments

Like PBT [3], IAS is vector-based, i.e., it starts immediately following vectorization, which yields bars and polylines from arc images. Any vectorization method can be used, as long as its output is a list of wires—bars and polylines that closely approximate the arc shape—which IAS accepts as input. The vectorization method we use is the Sparse Pixel Vectorization (SPV) algorithm [4].

2.1 Term Definitions

To describe the algorithm, we use a planar Cartesian coordinate system, with the x-axis directed left-right and the y-axis top-down. The following terms, shown in Fig. 1, are based on this coordinate system and are used throughout the paper.

Arc—a solid circular line segment with non-zero width, whose image is a contiguous set of black pixels occupying an area, the medial axis of which is approximately constrained by the circle equation:

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \tag{1}$$

where (x, y) is any point on the arc's medial axis, (x_c, y_c) is the center of the circle on which the arc lies, and r is the corresponding radius. The arc is limited by the two endpoints $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$, going counterclockwise along the arc from p_1 to p_2. If the two endpoints coincide, the arc is a full circle. Fig. 1 illustrates an ideal arc image, whose two endpoints are displayed using the round cap style.

Open angle—the angle subtended by an arc.

Chord—a straight line segment formed by linking any two points on an arc.

Chord height—the distance from the chord's midpoint to the corresponding arc's midpoint. Since a chord splits a circle into two arcs, we refer to the shorter one. Therefore, the chord height is the value obtained by subtracting from the radius the distance between the arc center and the chord, as shown in Fig. 1. The relation among the chord height (h), the open angle (θ), the radius (r) of the chord arc is expressed in (2).

• *The authors are with the Faculty of Industrial Engineering and Management, Technion—Israel Institute of Technology, Haifa 32000, Israel. E-mail: {liuwy, dori}@ie.technion.ac.il.*

Manuscript received 15 Oct. 1996. Recommended for acceptance by R. Kasturi. For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 106434.

$$\cos(\theta/2) = \frac{r-h}{r};\ \sin(\theta/2) = \frac{\sqrt{2rh-h^2}}{r} \quad (2)$$

Scan angle—the angle sweeping from the x-axis of the coordinate system to the radius vector directed from the center to a point on the arc's medial axis.
Critical angle—a scan angle whose value is $\pi/4$, $3\pi/4$, $5\pi/4$, or $7\pi/4$ radians.
Critical position—a point along the arc whose scan angle is a critical angle, as shown in Fig. 1.
Bar—a solid straight line segment with non-zero width, which occupies a (possibly slanted) black pixel area in the drawing image. The ideal area is an oval consisting of a rectangle and two semicircles, resulting from the round cap endpoint style, as shown in Fig. 2a.
Polyline—a chain of two or more equal-width bars linked end to end, as shown in Fig. 2b.
Edge—a bar in the polyline chain.
Characteristic point—an endpoint of a polyline edge.
Critical point—a characteristic point whose distance to the straight line segment formed by linking its two neighboring characteristic points in a polyline is less than some defined threshold.
Wire—a solid (continuous) line segment, which is a generalization of bar, arc, and polyline.
Vectorized arc fragment (*VAF*)—a bar or a polyline obtained by vectorization of a piece of an arc image.
Potential arc center area (*PACA*)—a small rectangle in which the potential arc center may be located.

The following abbreviations are used in the paper:

- PBT—the Perpendicular Bisector Tracing arc segmentation algorithm [3];
- SPV—the Sparse Pixel Vectorization algorithm [4];
- IAS—the Incremental Arc Segmentation algorithm;
- ASD—the Averaged Squared Difference, which is similar to the variance; and
- VRI—the Vector Recovery Index [5].

2.2 Features of the Vectorized Arc Fragments

Usually, a VAF generated by a vectorization algorithm, e.g., SPV [4], is a polyline. If it has only two endpoints and no intermediate point, it is approximated as a bar. In order to construct an arc precisely, its VAFs should closely approximate the original arc. First, each VAF should keep a consistent curvature direction, i.e., each edge of the polyline should turn in the same direction as its preceding edge, either clockwise or counterclockwise, thereby maintaining its convexity. Second, all VAFs of the same arc image should be associated with approximately equal line widths. Third, all the characteristic points and the medial axis of the VAF should lie inside the arc image, i.e., the following conditions should be met:

$$\left|r - dp_i\right| \le \frac{w}{2};\ i = 0, 1, \ldots, N \quad (3)$$

$$\left|r - de_i\right| \le \frac{w}{2};\ i = 1, 2, \ldots, N \quad (4)$$

where r is the radius of the arc, w is the line width of the arc and the polyline, dp_i is the distance from the center to the ith characteristic point of the polyline, de_i is the distance from the center to the ith edge of the polyline, and N is the total number of edges of the polyline. Fig. 3 illustrates the VAFs of the arc image of Fig. 1 yielded by SPV. The resulting polylines of the arc image are broken at two critical positions due to the change of the length direction from horizontal to vertical or vice versa. The VAF may be just a bar if the arc is very small.

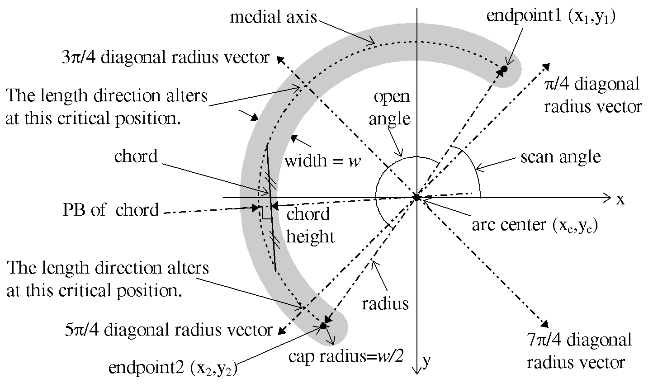
medial axis
endpoint1 (x1,y1)
3π/4 diagonal radius vector
π/4 diagonal radius vector
The length direction alters at this critical position.
open angle
width = w
chord
scan angle
x
PB of chord
chord height
arc center (xc,yc)
The length direction alters at this critical position.
radius
5π/4 diagonal radius vector
7π/4 diagonal radius vector
endpoint2 (x2,y2)
cap radius=w/2
y

Fig. 1. Illustration of an ideal arc image.

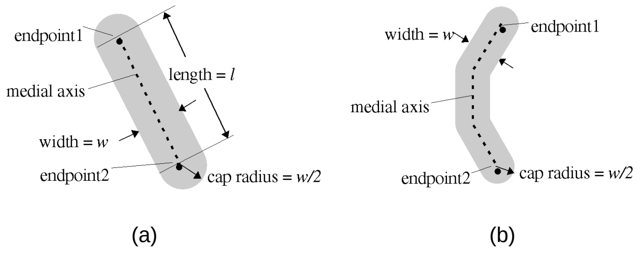

Fig. 2. (a) The ideal oval area occupied by a bar. (b) An ideal polyline image.

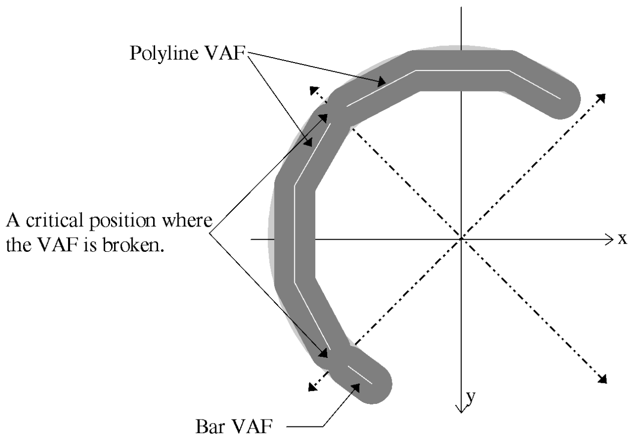

Fig. 3. Illustration of arc vectorization: arc image (light gray) and its VAFs (dark gray).

3 The Incremental Arc Segmentation Algorithm

3.1 Initial Polyline-to-Arc Conversion

The first step of IAS is polyline-to-arc conversion, in which a polyline is selected and converted to an arc. If no polyline can be found, two short neighboring bars can be linked into a polyline and used in the conversion. The reason for this is that an arc that has a small open angle (i.e., $\le \pi/2$) and a small radius and is approximately symmetrically divided at a critical position, may be vectorized into only two bars, rather than a polyline. In this case, the two bars that constitute the polyline should be approximately symmetric. In other words, the ratio of their lengths should be within a range around 1.0, e.g., 0.5 ~ 2.0, as used in this paper. The angle they form with the critical angle should be within a range around 45°, e.g., 30° ~ 60°, as used in this paper. In addition, the difference between their widths should also be less than a small

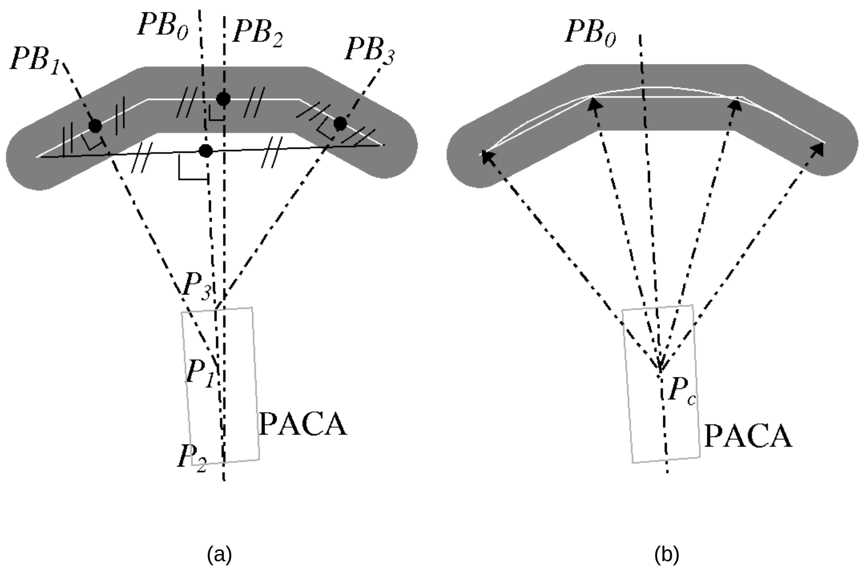

Fig. 4. Illustration of polyline-to-arc conversion. (a) A polyline, auxiliary PBs and the Potential Arc Center Area (PACA). (b) Center decision.

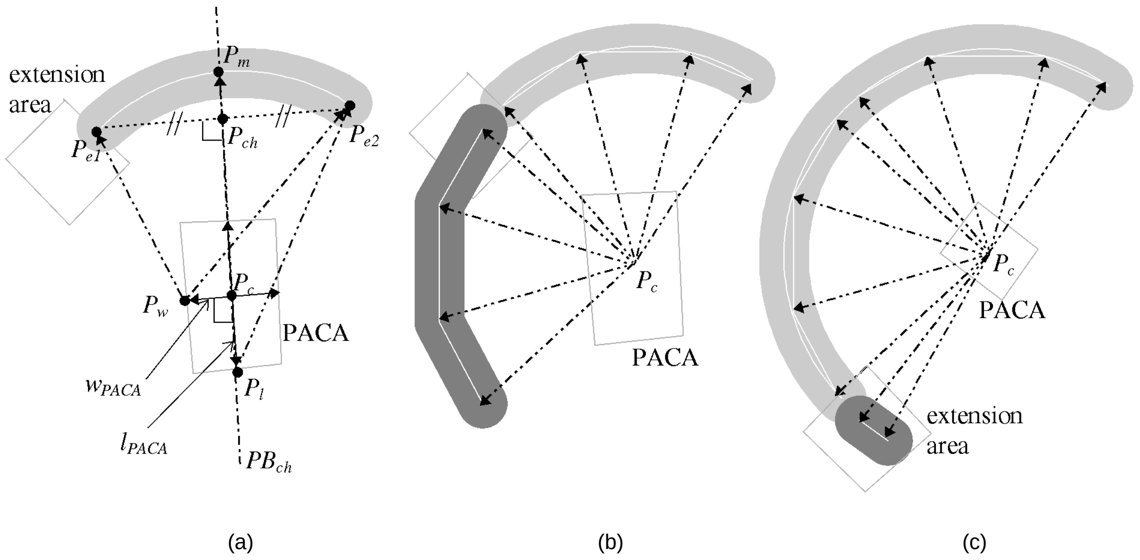

Fig. 5. Illustration of arc extension. (a) The extension area and the Potential Arc Center Area (PACA). (b) Center decision when extended to a polyline. (c) Center decision when extended to a bar.

number of pixels (two in this paper). Such arcs appear mostly at the corners of geometric contours in mechanical drawings. If only one bar is detected from an arc, the arc cannot be segmented by this algorithm.

From plane geometry, we know that the perpendicular bisector (PB) of any chord on a circle passes through the circle or arc center. The intersections of the PBs of all the polyline's edges should therefore coincide at exactly one point, which is the center of the circle. However, this only happens in the ideal case, in which all points of the polyline are precisely located on the circle's medial axis. Even then, errors occur due to the discrete sampling of the grid points. Hence, the polyline resulting from any vectorization method, including SPV, even from an ideal arc image, is usually not that precise, due to the curve digitization and image vectorization procedures. The imprecision of the polyline causes the intersections of the PBs of its edges to not coincide at one point. However, the precision degree of the polyline confines the PBs' intersection points to a small area. This area contains the point which is closest to the expected circle center.

For any polyline which may be an approximation of an arc, we use the following method to define the area in which the potential arc center may be located. We first construct PB_0, the PB of the segment joining the two endpoints of the polyline. For every edge i (i = 1..N) of the polyline, we then construct its PB, PB_i, and calculate the intersection P_i of PB_i and PB_0. This way, all the P_is are located along PB_0 except for those whose PB_i are parallel to PB_0. We define the smallest range R along PB_0 that contains all the P_is. We then construct a rectangular area with length R, width equal to the polyline's width, and which is bisected by PB_0, as shown in Fig. 4a. This rectangle is the Potential Arc Center Area (PACA), within which the potential arc center is sought.

A curvature test is performed as each edge of the polyline is visited. If some edge turns in a direction opposite to the former ones, the polyline is concave, and cannot be converted into an arc. After each P_i is calculated, we take the average distance from it to both polyline endpoints as a temporary radius value and test the two conditions defined in (3) and (4). If either condition is violated, the polyline is most likely not a VAF, and the polyline-to-arc conversion stops.

Having determined the PACA, we check every pixel within it as a center candidate $c(x, y)$. We calculate the radius $r(x, y)$ by taking the average of all $dp_i(x, y)$, and the averaged squared difference $ASD(x, y)$, which is similar to the variance of $dp_i(x, y)$, using (5) and (6), respectively.

$$r(x,y) = \frac{1}{N+1}\sum_{i=0}^{N} dp_i(x,y) \tag{5}$$

$$ASD(x,y) = \frac{1}{N+1}\sum_{i=0}^{N}(r(x,y) - dp_i(x,y))^2 \tag{6}$$

The point whose ASD is minimal is taken as the potential center P_c of the segmented arc, as shown in Fig. 4b. If the center also obeys the two conditions defined in (3) and (4), the initial arc is successfully constructed with its center and radius determined. The line width of the constructed arc is the same as that of the polyline. The two endpoints of the arc are taken as the polyline endpoints, and their order is such that the arc is counterclockwise from Endpoint 1 to Endpoint 2. Otherwise, the polyline is rejected as an arc, and the next polyline is selected to undergo the same conversion procedure. If the conversion succeeds, it further undergoes the following stepwise extension to join as many VAFs as possible.

3.2 Incremental Recovery of Arc Pieces

The initial arc constructed above is further extended to both ends as far as possible. An arc has two extension directions, one from each endpoint. The extension in each direction is completed by applying several extension iterations. In each iteration, the arc is extended by one VAF, the arc endpoint is updated to be the far end of this VAF, and the new arc attributes are modified accordingly. In each iteration, an *extension area* is first constructed as a square that stretches away from the current arc endpoint along the arc tangent direction with its side length twice the arc width, as shown in Fig. 5. To select VAF candidates, we have devised the *VAF candidacy test*, which consists of the following three conditions:

1) Width similarity: The line widths of both the arc and the VAF candidate should be about the same, i.e., the difference between them should be less than some predefined threshold, e.g., two pixels, as used in this paper.
2) Collinearity: The tangent to the arc at the endpoint should form a small angle with the candidate, which is less than some predefined threshold, such as $\pi/3$, as used in this paper.
3) Continuity: The gap between the arc and the candidate at their close endpoints should be less than twice the arc width and filled with enough black pixels, i.e., the percentage of black pixels in the gap area is greater than some predefined threshold, e.g., 80 percent, as used in this paper. This condition is optional, and is used to increase the likelihood of correct segmentation when the original image pixel data is readily available.

All wire fragments that pass through the extension area and also pass the VAF candidacy test are selected as extending VAF candidates. The extending VAF candidates are sorted by increasing distances of their closest endpoint to the current arc endpoint and put into a queue. Each candidate from the queue is tested in turn for passing the following *VAF test*, which proves that it is a real VAF of the arc.

In the VAF test, a dynamic PACA is defined, based on the current arc attributes, as shown in Fig. 5. The PACA is a rectangle, whose center is the current arc center. If the current open angle is less than π, we use the two arc endpoints to construct a chord. We then construct PB_{ch}, the PB of the chord. The PACA's length is l_{PACA}, which is parallel to PB_{ch}, and the PACA's width is w_{PACA}. The values of w_{PACA} and l_{PACA} are determined by (7) and (8), respectively, and illustrated in Fig. 5a. P_w in Fig. 5a is the furthest point of the potential center area in the width direction. Hence, the condition in (7) should be met. P_l is the furthest point in the length direction of the center area, and here the condition in (8) should be met.

$$\left|\overline{P_wP_{e1}} - \overline{P_wP_{e2}}\right| \le w \tag{7}$$

$$\left|\overline{P_lP_m} - \overline{P_lP_{e2}}\right| \le w \tag{8}$$

In these equations, w is the arc width, and the distances of the segments are calculated using (9)-(14).

$$\overline{P_wP_{e1}} = \sqrt{\overline{P_cP_{ch}}^2 + (\overline{P_{e1}P_{ch}} - \overline{P_wP_c})^2} \tag{9}$$

$$\overline{P_wP_{e2}} = \sqrt{\overline{P_cP_{ch}}^2 + (\overline{P_{ch}P_{e2}} + \overline{P_wP_c})^2} \tag{10}$$

$$\overline{P_lP_m} = r + \overline{P_lP_c} \tag{11}$$

$$\overline{P_lP_{e2}} = \sqrt{(\overline{P_lP_c} + \overline{P_cP_{ch}})^2 + \overline{P_{ch}P_{e2}}^2} \tag{12}$$

$$\overline{P_{e1}P_{ch}} = \overline{P_{ch}P_{e2}} = \overline{P_{e1}P_{e2}} / 2 \tag{13}$$

$$r^2 = \overline{P_{e1}P_{ch}}^2 + \overline{P_cP_{ch}}^2 \tag{14}$$

From (9)-(14), we obtain $\overline{P_lP_c}$, $\overline{P_wP_c}$, l_{ac}, and w_{ac} by (15)-(18).

$$\overline{P_lP_c} = \frac{w(2r - w)}{2(r - w - \overline{P_cP_{ch}})} \tag{15}$$

$$\overline{P_wP_c} = w\sqrt{\frac{4r^2 - w^2}{16\overline{P_{e1}P_{ch}}^2 - 4w^2}} \tag{16}$$

$$l_{PACA} = 2\overline{P_lP_c} \tag{17}$$

$$w_{PACA} = 2\overline{P_wP_c} \tag{18}$$

From (7) and (8), we know that the PACA confined by l_{PACA}, and w_{PACA} is the smallest rectangular area that contains the center of the potential arc. If the calculated value of l_{PACA} or w_{PACA} is larger than the radius, it is set as the value of the radius.

If the current open angle is larger than or equal to π, the diameter that is parallel to the chord formed by the endpoints of the arc is used. In this case, $\overline{P_{e1}P_{ch}} = r$ and $\overline{P_cP_{ch}} = 0$, and w_{PACA} is equal to w. If the current open angle is larger than or equal to $3\pi/2$, there are two perpendicular diametric chords. l_{PACA} can then be calculated in the same way as w_{PACA}. Therefore, l_{PACA} is also equal to w. In this case, the potential arc center area can be taken as a square with side w centered at the current arc center.

After the potential arc center area is determined, the new center is found within it, applying the same procedure used to find the initial arc center during the polyline-to-arc conversion, presented in Section 3.1, except that all the characteristic points and edges of the current arc are used in the calculation of (5) and (6), and in the examination that uses (3) and (4), as shown in Fig. 5b and Fig. 5c. If such a center can be found, the candidate passes this VAF test.

If some VAF candidate passes the VAF test, the arc is extended with that VAF, the new endpoint, center, and radius are modified accordingly, the current extension iteration stops, and another iteration follows. Otherwise, the next candidate in the queue is tested. This test cycle is repeated until a candidate passes the test or the queue becomes empty. If the queue is empty, the extension attempt in the current direction halts.

At the end of each extension iteration, a test is done to check to whether the two arc endpoints can be extended to form a circle. If the two endpoints meet the continuity condition in the VAF candidacy test, a circle is formed. After the extension in the first end-

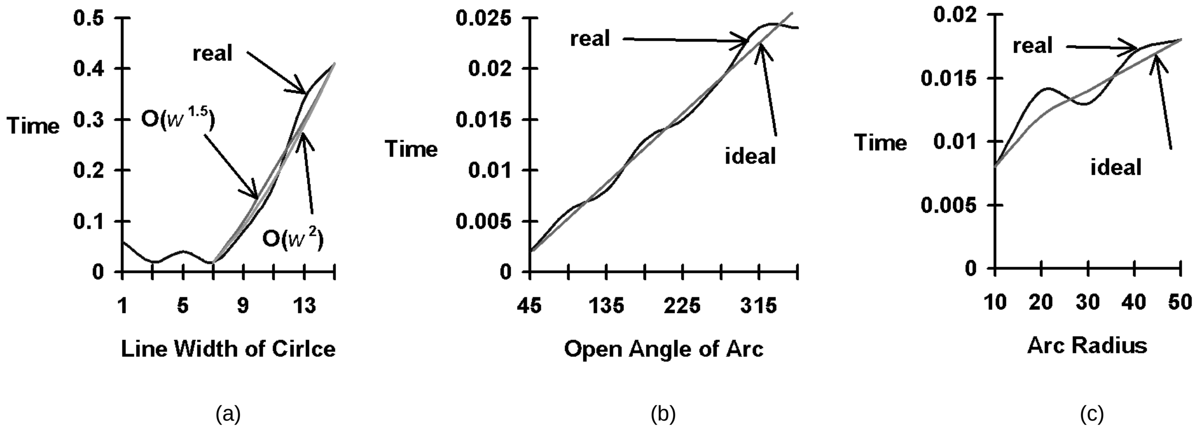

Fig. 6. Time (in seconds) curves with respect to (a) arc widths, (b) open angles, and (c) radii.

point direction is complete, the same extension procedure is carried out from the second endpoint. Finally, to reduce the false alarm rate, all of the VAFs are tested again to verify that they meet the conditions in (3) and (4).

3.3 Complexity Analysis

The most time-consuming operation in IAS is the calculation of the average radius and the ASD for each point in the center area, defined in (5) and (6), and the consequent condition testing, defined in (3) and (4). The time spent on searching for candidates is negligible, because we use the *Position Index* data structure [6]. The number of candidates at each extension cycle is usually very small. It is only one at the joint where there are no other interfering lines. Even if there are several lines passing through the extension area, the number of candidates requiring test is limited, because the extension area is very small, as shown in Fig. 5. Generally, both the length and the width of the PACA are directly proportional to the arc width, w, as shown in (15)-(18). The time complexity is therefore quadratic to w. Another important factor that affects the complexity of this algorithm, in addition to w, is the total number of characteristics points of all VAFs of the arc image, as this is the number of repetitions of the computation defined in (3)-(4).

Let us denote the total number of all vectorized segments of the arc image by N_s and the total number the characteristic points of all these segments by N_p. Since the number of edges in each segment is the number of its characteristic points minus one, the total number of edges of all VAFs, denoted by N_e, is $N_e = N_p - N_s$. Usually, N_s, the total number of vectorized segments, is not large and can therefore be considered as constant.

N_p is determined by the radius, the width, the open angle, and the allowed chord height of the edges of these VAFs. Since the chord height h is usually at least one pixel (and at most w/2, for $w \geq 2$), we introduce it into (2). The maximum open angle for each edge's arc, $\delta\theta$, is defined in (19). Since $\delta\theta$ is a small angle, $\sin(\delta\theta) \approx \delta\theta$, and a simple form is obtained on the right-hand side of (19). Assuming all these characteristic points are evenly distributed, as in the ideal case, (20) holds, where θ is the arc open angle.

$$\delta\theta \geq 2\arcsin\left(\frac{\sqrt{2r-1}}{r}\right) \approx 2\sqrt{\frac{2}{r}} \tag{19}$$

$$N_p \approx \frac{\theta}{\delta\theta} \leq \frac{\theta\sqrt{r}}{2\sqrt{2}} \tag{20}$$

The time complexity of the IAS algorithm for a single arc image is therefore $O(w^2\theta\sqrt{r})$. If we introduce h = w/2 into (2), we get that $\delta\theta \leq 2\sqrt{2w/r}$, approximately. Therefore, the lower bound on the time complexity is $O(w^{1.5}\theta\sqrt{r})$. This may be confirmed in Fig. 6. The space IAS requires is linear to N_p, i.e., $O(\theta\sqrt{r})$.

4 Experiments, Evaluation, and Discussion

The IAS algorithm is implemented in C++ within the Machine Drawing Understanding System (MDUS) [7]. The running platform may be **SGI IRIX5.3** as well as **SUN Solaris2.5** and above. The executable codes are available from the ftp address in [8]. We also evaluate IAS using the protocol in [5], whose main idea is described as follows. We consider each pair of matched ground truth and detected line if they overlap each other either fully or partially. The vector detection qualities of the overlapping segments of every detected line and the ground truth line are measured by the detection accuracy using a number of criteria, including the endpoints, the location offset, the line width, the geometry form, and the line style. The vector detection qualities of these overlapping segments, as well as their lengths, are accumulated for both the ground truth lines and the corresponding detected lines. The Basic Quality, (Q_b), which is the length weighted sum of the vector detection qualities of overlapping segments, the Fragmentation Quality, (Q_{fr}), which is the measurement of the detection fragmentation and/or consolidation, and the Total Quality, (Q_v), which is the product of Q_b and Q_{fr}, for both the ground truth lines and their corresponding detected lines are calculated, respectively. After the qualities of all the ground truth lines and the detected lines are calculated, the total Vector Detection Rate, D_v, which is length weighted sum of Q_v of all ground truth lines, the Vector False Alarm Rate, F_v, which is the length weighted sum of $1 - Q_v$ of all detected lines, and the resulting Vector Recovery Index, $VRI = (D_v + 1 - F_v)/2$, are calculated.

To automatically evaluate IAS, we manually prepared a set of arcs in pixels, whose ground truth data are shown in Table 1. The arcs are drawn in a 5 × 6 matrix, the widths of lines in the rows numbered one to five from top to bottom are one, two, three, five, and nine pixels, the open angles in the columns numbered one to six from left to right are 2π, $3\pi/2$, $\pi/2$, $\pi/4$, and $\pi/8$, and the radii of the concentric arcs in each matrix cell are five, 10, 20, and 50 pixels. The performance evaluation result of the IAS is presented in Table 2, which shows that in general the VRI increases along with the open angle and the width of the arc. The highest VRI is 0.97 for a nine-pixel-wide circle, and the worst VRIs are around 0.40 for $\pi/8$ arcs. The ground truth images (in TIFF) and segmented arcs (in IGES) are also available from the ftp address in [8].

To test IAS's time efficiency as related to radii, open angles, and widths, we have manually generated a matrix of five-pixel-wide

TABLE 1
GROUND TRUTHS OF THE TESTED SYNTHETIC ARCS

	column	1	2	3	4	5	6
row	open angle / width / center	Full Circle $0\sim2\pi$	3/4 Circle $0\sim3\pi/2$	1/2 Circle $0\sim\pi$	1/4 Circle $0\sim\pi/2$ and $3\pi/4\sim5\pi/4$	1/8 Circle $0\sim\pi/4$ and $5\pi/8\sim7\pi/8$	1/16 Circle $0\sim\pi/8$ and $11\pi/16\sim13\pi/16$
1	1	60,60	180,60	300,60	420,60	540,60	660,60
2	2	60,180	180,180	300,180	420,180	540,180	660,180
3	3	60,300	180,300	300,300	420,300	540,300	660,300
4	5	60,420	180,420	300,420	420,420	540,420	660,420
5	9	60,540	180,540	300,540	420,540	540,540	660,540

In each cell, the radii are five (only for width = 1, 2, and 3), 10 (only for width = 1, 2, 3, and 5), 20, and 50 pixels.

TABLE 2
PERFORMANCE EVALUATION OF THE IAS ALGORITHM

column / row	1			2			3			4			5			6			column total		
	D_v	F_v	VRI	D_v	F_v	VRI	D_v	F_v	VRI	D_v	F_v	VRI	D_v	F_v	VRI	D_v	F_v	VRI	D_v	F_v	VRI
1	0.54	0.46	0.54	0.64	0.34	0.65	0.45	0.54	0.45	0.55	0.40	0.57	0.37	0.50	0.43	0.38	0.54	0.42	0.53	0.44	0.54
2	0.91	0.07	0.92	0.51	0.49	0.51	0.61	0.37	0.62	0.65	0.30	0.67	0.34	0.57	0.38	0.35	0.63	0.36	0.66	0.32	0.67
3	0.85	0.13	0.86	0.83	0.16	0.83	0.79	0.19	0.80	0.79	0.17	0.81	0.61	0.31	0.65	0.44	0.53	0.45	0.79	0.18	0.80
4	0.84	0.17	0.83	0.84	0.16	0.84	0.82	0.18	0.82	0.82	0.15	0.83	0.26	0.65	0.30	0.39	0.55	0.42	0.77	0.22	0.78
5	0.97	0.03	0.97	0.93	0.06	0.94	0.95	0.04	0.95	0.73	0.18	0.78	0.33	0.46	0.44	0.43	0.49	0.47	0.85	0.11	0.87
row total	0.82	0.18	0.82	0.74	0.25	0.75	0.71	0.28	0.72	0.70	0.24	0.73	0.38	0.50	0.44	0.40	0.55	0.42	0.71	0.26	0.73
Global	D_v =0.71							F_v =0.26							VRI=0.73						

TABLE 3
TIME (SECONDS) USED IN SEGMENTING ARCS OF VARIOUS RADII AND OPEN ANGLES

	column	1	2	3	4	5	6	7	8	average
row	open angle / radius	$\pi/4$	$\pi/2$	$3\pi/4$	π	$5\pi/4$	$3\pi/2$	$7\pi/4$	2π	
1	10	0.000	0.002	0.007	0.009	0.007	0.013	0.015	0.011	0.008
2	20	0.000	0.008	0.005	0.014	0.020	0.017	0.023	0.021	0.014
3	30	0.002	0.006	0.006	0.014	0.011	0.020	0.021	0.026	0.013
4	40	0.001	0.009	0.015	0.016	0.016	0.022	0.027	0.027	0.017
5	50	0.007	0.005	0.007	0.014	0.019	0.025	0.033	0.035	0.018
	average	0.002	0.006	0.008	0.013	0.015	0.019	0.024	0.024	

TABLE 4
TIME (SECONDS) USED IN SEGMENTING ARCS OF VARIOUS WIDTHS

Width	1	3	5	7	9	11	13	15
Time (seconds)	0.06	0.02	0.04	0.02	0.08	0.17	0.34	0.56

arcs, whose arc segmentation time on SGI Indy is listed in Table 3, and a list of 50 arcs whose radii are 50 pixels and whose arc segmentation time on SGI Indy is listed in Table 4. The 2D curves of the time relative to the arc widths, open angles, and radii are displayed in Fig. 6. These experimental results confirm the time complexity analysis done in Section 3.3.

We have also applied IAS to a host of real-life engineering drawings of various drafting standards. A sample drawing and its arc segmentation result is shown in Fig. 7. The results show that IAS successfully performs arc segmentation from complex line environments, such as concentric arcs as well as tangency and intersection of other lines. The cases of intersection with small angle and tangency are the most difficult in arc segmentation, but IAS usually overcomes these problems. In Fig. 7, which is a drawing from the Indian Standard, most arcs are correctly segmented. Time performance data of IAS applied to these drawings (on **SGI Indigo2**) is presented in Table 5.

5 SUMMARY

A vector-based incremental arc segmentation algorithm has been presented, analyzed, and evaluated. We try to convert a polyline to an initial arc and extend it incrementally to join as many VAFs as possible. Due to proper threshold selection and consistent checking of cocircularity of the assumed arc pieces, the algorithm accurately constructs arcs from the vector input. Experiments and evaluation show high detection rates even for complex real-life drawings, such as those with concentric arcs, and intensive tangency and intersections. Nearly 200 synthetic arcs, ranging in radius from five to 50 pixels, in open angle from $1/8\pi$ to 2π, and in thickness from one to nine pixels, are used in the experiments and evaluation. Parts of six real-life drawings, containing about 200 arcs, are also processed. The algorithm works well for arc segments greater than 10 pixels in radius, $\pi/4$ in angle, and one pixel in width.

The arc center is determined as a point in the PACA with the minimal ASD (6). Although this is similar to the HT idea, which locates the arc center at concentrations of intersecting perpendicu-

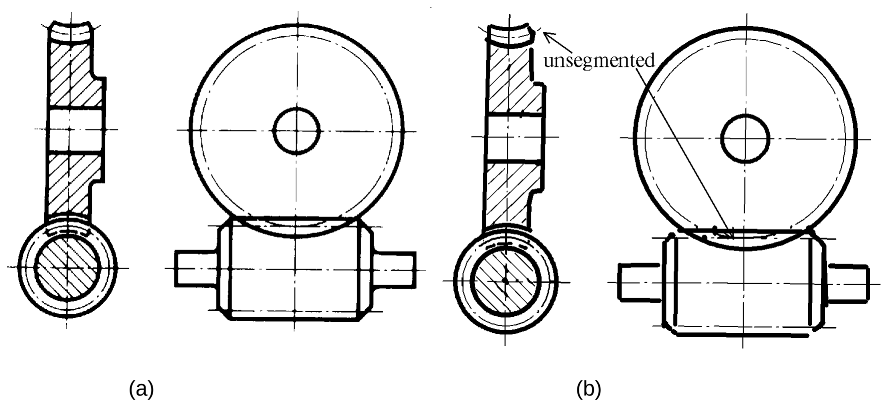

Fig. 7. Incremental Arc Segmentation (IAS) applied to a real-life drawing from the Indian Standard. (a) Image. (b) Arc segmentation.

TABLE 5
THE TIME EFFICIENCY (ON SGI INDIGO2) OF IAS APPLIED ON REAL-LIFE DRAWINGS

Drawing Type	Size (pixels)	# of segmented arcs	Time (seconds)	Time/pixel (10^{-6} seconds)	Time/arc (seconds)
ISO	1056X486	20	2.37	4.6	0.12
ANSI	824X600	34	1.82	3.9	0.06
Indian (fig. 7)	1144X900	23	3.52	3.4	0.15
Israel	344X644	33	1.36	6.1	0.04
ISO	1232X810	83	4.16	4.2	0.05
ISO	1520X1111	82	7.25	4.3	0.09

lars, IAS is superior to the pixel-based HT. The first advantage is the time efficiency, which has been shown in the experiments to be about 0.1 sec/arc. The second advantage is that being a vector based algorithm, IAS has a more global view than the pixel-based ones, and therefore it is more robust to noise and clutter found in the real-life drawing. Pixel-based algorithms are much more vulnerable to noise and other kinds of perturbations, which are prevalent in most real-life drawings.

The IAS algorithm is also better than PBT [3], which finds all of the arc components at once. First, the clustering of the components is blind to a certain extent, since the clustering rule usually does not manage to cluster all the components that comprise the detected object. Thus, many groups of clustered components may be either false alarms or parts of the same object. Second, since the VAFs can be easily shattered by interfering elements, the clustered components may be either redundant, i.e., include more elements than what is required, or incomplete, i.e., contain fewer elements than what is required. Third, although all the components of a graphic object may be found, their roles within the graphic object may be difficult to determine. Since IAS finds one component at a time, it can select the best candidate and not be mislead by clutter in the arc's neighborhood.

Additional improvements of IAS relative to PBT are as follows:

1) IAS is completely vector-based, i.e., it uses the characteristic points of the VAFs as the medial axis points of the arc, while PBT employs a perpendicular bisector tracing (PBT) procedure, which visits the raster image to find the medial axis points of the arc image. Since IAS uses more points than PBT to construct the arc, IAS is more likely to generate a more accurate arc than PBT.
2) IAS first defines a Potential Arc Center Area (PACA) and then finds the best point among the possible centers, while PBT only uses the mass center of the triangle formed by the three centers determined by three groups of triple points on the arc image. Hence, the center found by IAS is likely to be more accurate.
3) Full circles, ovals, and round corners are treated as special cases in PBT, while in IAS they are treated uniformly.

The complexity of the IAS algorithm is also analyzed. The time complexity of IAS for a single arc image is between $O\left(w^2\theta\sqrt{r}\right)$ and $O\left(w^{1.5}\theta\sqrt{r}\right)$. This was confirmed by the experiments shown in Fig. 6. The space required for the IAS algorithm is $O(\theta\sqrt{r})$.

Theoretically, the number of sample points of the arc medial axis determines the precision with which the arc can be reconstructed. The number of sample points is determined by the open angle subtended by an edge of the polyline generated by some vectorization algorithm. In order to preserve the arc shape to a satisfactory extent, according to (2) and (4), this angle should be less than $2arccos(1 - 0.5\ w/r)$, and the smaller the better. In this case, the number of the sample points is proportional directly to the open angle and the radius, but inversely to the width of the entire arc. Therefore, small arcs are more difficult to segment than large ones. The experiments and evaluation in this paper provide empirical evidence to this theoretic observation.

However, while very thick arcs with small open angles are most likely to be segmented as bars, it is not counterintuitive that for arcs whose width-to-radius ratio is smaller than some threshold, the thicker the arc, the easier it is to segment. This is so because according to (3) and (4), thicker arcs allow bigger chord height threshold than thinner ones. The accuracy evaluation uses relative thickness difference rather than absolute one. Hence greater line width improves the characterization accuracy.

IAS has been incorporated into the MDUS environment [7], and it is instrumental in segmenting arcs from engineering drawings processed by our system.

REFERENCES

[1] D.H. Hallard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111-122, 1981.

[2] H. Asada and M. Brady, “The Curvature Primal Sketch,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 1, pp. 1-14, 1986.

[3] 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. 1,057-1,068, Nov. 1995.

[4] W. Liu and D. Dori, “Sparse Pixel Tracking: A Fast Vectorization Algorithm Applied to Engineering Drawings,” *Proc. ICPR96*, vol. 3 (Robotics and Applications), pp. 808-812, Vienna, 1996.

[5] W. Liu and D. Dori, “A Protocol for Performance Evaluation of Line Detection Algorithms,” *Machine Vision Applications*, vol. 9, no. 5, pp. 240-250, 1997.

[6] W. Liu, D. Dori, L. Tang, and Z. Tang, “Object Recognition in Engineering Drawings Using Planar Indexing,” *Proc. GREC95*, pp. 53-61, Philadelphia, 1995.

[7] W. Liu and D. Dori, “Automated CAD Conversion With the Machine Drawing Understanding System,” *Proc. DAS96*, pp. 241-259, Malvern, Penn., 1996.

[8] ftp.technion.ac.il/pub/supported/ie/dori/MDUS/sgimdus.gz, sunmdus.gz, and gtruth.tar.gz.