IJDAR (1998) 1: 62–71

International Journal on IJDAR
Document Analysis and Recognition

Stepwise recovery of arc segmentation in complex line environments

Dov Dori, Wenyin Liu

Faculty of Industrial Engineering and Management, Technion-Israel Institute of Technology, Haifa 32000, Israel
e-mail: {dori, liuwy}@ie.technion.ac.il; http://iew3.technion.ac.il:8080/{Home/Users/dori.phtml;~liuwy}

Received May 30, 1997 / Revised August 31, 1997

Abstract. Accurate arc segmentation, essential for high-level engineering drawing understanding, is very difficult due to noise, clutter, tangency, and intersections with other geometry objects. We present an application of a generic methodology for recognition of multicomponent graphic objects in engineering drawings to the segmentation of circular arcs. The underlying mechanism is a sequential stepwise recovery of components that are segmented as wire fragments during the sparse-pixel vectorization process and meet a set of continuity conditions. Proper threshold selection and consistent checking of cocircularity of the assumed arc pieces result in an accurate arc segmentation method. Experimental results are presented, evaluated by an objective protocol, and discussed.

Key words: Arc segmentation – Vectorization – Graphics recognition – Engineering drawings interpretation – Document analysis and recognition

1 Introduction

A typical line drawing is made up of lines with various geometric forms. Correct recognition of these geometry forms is an essential task of line drawing interpretation. Circular arcs are very common in line drawings and therefore require correct and accurate recognition just as bars (straight line segments). The recognition of circular arcs is more difficult than that of bars because of the higher degree of the circle equation. A free form curve, on the other hand, can usually be recognized as (or expressed by) a polyline.

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.

Research on arc segmentation from digitized drawings began in the early 1980s. *Hough Transform (HT)* was the first technique to be employed in arc segmentation (Hallard 1981; Hunt and Nolte 1988; Dori 1997). HT is a conventional method for object extraction from binary images. HT is normally used for arc segmentation in the 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. Although a number of works, including Kimme et al. (1975), O'Gorman and Sanderson (1984), Illingworth and Kittler (1987), Conker (1988), and Haralick and Shapiro (1992), have attempted to lower the complexity of HT, the time and space requirement have still remained too demanding for the method to be of practical use.

Other arc segmentation methods, including Asada and Brady (1986), O'Gorman (1988a, 1988b), and Rosin and West (1989), belong to the curvature estimation group. 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. Asada and Brady (1986) propose the curvature primal sketch to represent significant changes in curvature along the bounding contour of a planar shape. They define a set of primitive parametrized curvature discontinuities and derive expressions for their convolution with the first and second derivatives of a Gaussian. They interpret the significant changes in curvature at various scales. The input is a bounding contour of the object to be recognized, while the output may be a se-

Correspondence to: D. Dori

mantic network or a filtered response graph. Rosin and West (1989) describe a method of segmenting curves in images into a combination of circular arcs and straight lines. It is an extension of a method proposed by Lowe (1987), which analyzes arbitrary curves and produces a high-level straight line description. In Lowe (1987), each curve is segmented by splitting it at the maximum deviation from the approximating straight line. Rosin and West (1989) find the best fit of a combination of arcs and straight lines to the data. The input to the algorithm is a digital line, hence the image must be preprocessed by an edge detector to extract connected pixels or use boundary descriptions from chain-coded binary images. O'Gorman (1988a, 1988b) proposes to carry out feature extraction by estimating the curvature along digital lines and determining the features from the curvature plots. For the difference of slopes (DOS) approach, curvature at a point is estimated as the angular difference between the slopes of two line segments fitted to the data before and after the points. After finding the curvature of each point in the data, the curvature plot is usually smoothed to reduce noise. The specialized difference of slopes method DOS^+ (O'Gorman 1988b) proposes to use a nonzero, small positive gap between the two segments that estimate the lines between which the angle is measured. DOS^+ has been shown to be effective for estimating curvature in terms of signal detectability. In O'Gorman (1988a), two approaches for curvature estimation are compared: the DOS^+ method and the Gaussian smoothing of the second derivative. DOS^+ was shown to outperform Gaussian Smoothing for small signal angle and high noise. Here, too, the input is assumed to a be a chain of data points.

Both groups of arc segmentation algorithms discussed above do not provide adequate means for extracting arcs from engineering drawings. First, none of these methods detects line width. Second, their main focus is either on detecting full circles or wide-angle arcs from noisy images or extracting features from curvature estimation for object recognition.

Perpendicular Bisector Tracing (PBT) (Dori 1995) 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 uses the knowledge about the bars detected in the OZZ vectorization (Chai and Dori 1992; Dori 1997) to find bar chains 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 till the first black pixel of the arc image is encountered, and the entry point is recorded. The tracing continues till 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 to a considerable extent. 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 PBT algorithm is not fully vector-based in a strict sense, since the tracing done to find the third point on the arc requires pixel visiting, which may be fooled by pixels of other graphic objects. This is a major source for pitfalls, as tracing along a black pixel area (of some annotation line) requires more processing, while concentric arcs are often missed.

In this paper we present a vector-based arc segmentation algorithm which is incorporated into the Machine Drawing Understanding System (MDUS) (Liu and Dori 1996b), that utilizes a generic methodology for recognition of multicomponent objects in engineering drawings through a stepwise recovery of their vectorized components (Liu et al. 1995; Liu and Dori 1996b). It takes as input the vector pieces of the arc image, generated by the Sparse Pixel Vectorization (SPV) algorithm (Liu and Dori 1996a). SPV outputs a set of vectorized wire (bar and/or polyline) fragments, which preserve the original shape of the arc image. 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, we extend the segmented arc to both ends by finding the fragments one at a time. The algorithm is capable of segmenting arcs from mixtures of lines, even in highly complex cases, such as tangency, co-circularity, intersection with lines at small angles, etc. The performance evaluation of the algorithm, based on a protocol proposed in (Liu and Dori 1997), is demonstrated.

2 The Stepwise Recovery Arc segmentation algorithm

Being a specialization of the generic graphic object recognition algorithm (Liu et al. 1995; Liu and Dori 1996b), our Stepwise Recovery Arc Segmentation (SRAS) algorithm starts with finding a key component of an arc. The Vectorized Arc Fragments (VAFs) of the arc image may be bars and polylines (Liu and Dori 1996a). We first select a polyline as the first key component of the arc to be segmented. After all polyline VAFs are examined, and no first key component can be found, we select a bar VAF from the vector list, and, according to the conditions discussed below, try to link it to a neighboring bar to produce a two-edge polyline. If such a polyline can be formed, it is taken as the key component of the arc to be segmented. The reason for this is that an arc that has a small open angle (e.g., $\leq \pi/2$) and a small radius, which is approximately symmetrically divided by a 45° diagonal radius vector, may be vectorized into only two bars, rather than polylines. Since the arc may only be broken at the critical positions, 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 45° diagonal radius vector should be within a range around 45°, e.g., 30°–60°, as

used in this paper. In addition, the difference between the widths should be less than a small number of pixels (2 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.

2.1 Initial polyline-to-arc conversion

After a polyline is selected as the key component of the expected arc in the hypothesis generation phase, the main work of the SRAS algorithm is to construct the initial arc and extend it, if possible.

The approximating polyline of an arc can be calculated using the circle equation

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \quad (1)$$

where (x, y) is any point on circle, (x_c, y_c) is the circle center, and r is the corresponding radius.

The discrete sample arc points are taken from the arc's medial axis and are separated by a fixed small scan angle. Since the polyline approximates an arc, the lengths of its edges are about equal. These edges are also chords of the original arc. The fixed small scan angle is determined by the *chord height*, which is 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 from the arc center to the chord. The relation among the chord height, the radius and the open angle of the chord arc is expressed in (2).

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

where θ is the open angle, r is the radius of the chord arc, and h is the chord height.

According to (2), the selected, small fixed scan angle, which is also the open angle of the arcs corresponding to these chords, determines the precision of the polyline. The smaller the angle, the shorter the edges and the chords, and the smaller the chord heights, therefore, the more precise the polyline. Usually the chord height of these edges can be up to half the line width to yield a relatively satisfactory polyline. Therefore, the angle can be up to $2\arccos(1-0.5w/r)$, where w is the line width.

Unlike the arc-to-polyline conversion described above, the inverse process, namely polyline-to-arc conversion, requires more effort. 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 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 not to 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. The characteristics of the vectorized piece of an arc image by the SPV algorithm (Liu and Dori 1996a) are such that (1) the feature points (endpoints and intermediate points) lie on the arc image, and (2) the distance between any single segment of the polyline and the arc is less than or equal to 1 pixel after the basic SPV procedure, and less than or equal to half the line width after the SPV post processing. Hence, the arc converted from a polyline closely approximates the original arc as it meets the following two center distance conditions

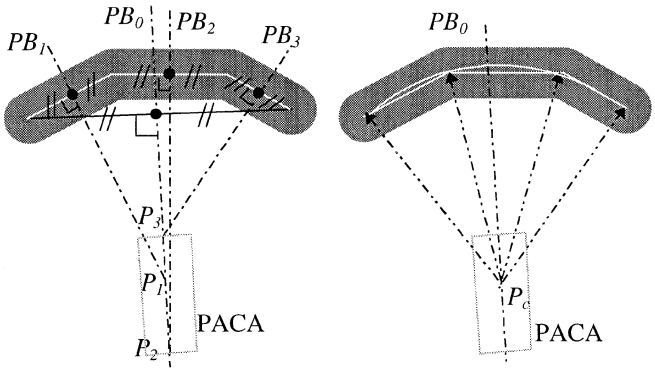

Fig. 1. Illustration of polyline-to-arc conversion. **a.** a polyline, auxiliary PBs, and the Potential Arc Center Area (PACA) **b.** center determination

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

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

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.

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 whose length is R and whose width is equal to the polyline's width and is bisected by PB_0, as shown in Fig. 1a. 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 equations (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 Potential Arc Center Area, we check every pixel within this area 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 square difference (ASD) $ASD(x, y)$ of $dp_i(x, y)$, using (5) and (6), respectively.

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

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

The point whose ASD is minimal is taken as the potential center P_c of the segmented arc, as shown in Fig. 1b. If the center also obeys the two conditions defined in (3) and (4), the initial arc is constructed with the calculated attributes, namely the center and the radius. 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 defined counterclockwise from endpoint 1 to endpoint 2.

2.2 Stepwise 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 several iterations. In each extension iteration the arc is extended by one VAF, the arc endpoint is updated to be the far end of this VAF, and the other 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 and whose side is twice the arc width, as shown in Fig. 2. 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, that is, the difference between them should be less than some predefined threshold, such as 2 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 filled with enough black pixels. This condition requires that the ratio of black pixels to the total number of pixels in the gap area be greater than some predefined threshold, e.g., 80%, as used in this paper. The condition is optional, and is used to increase the likelihood of correct segmentation when the original image 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 Potential Arc Center Area (PACA) is defined, based on the current arc attributes, as shown in Fig. 2. 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 l_{PACA} and w_{PACA} are determined by (7)–(8) and illustrated in Fig. 2a. P_w in Fig. 2a is the farthest point in the width direction of the potential center area. Hence, the condition in (7) should be met. P_l is the farthest 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).

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

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

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

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

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

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

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

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

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

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

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

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 simply 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 simply taken as a square, whose sides are w, centered at the current arc center.

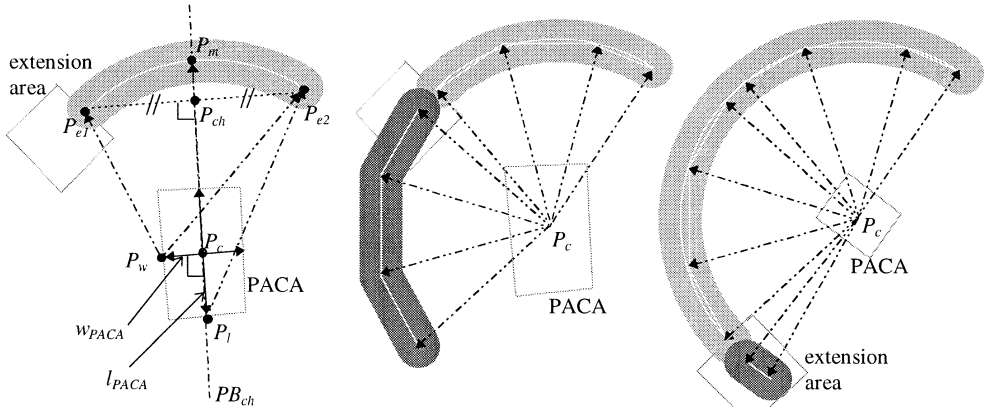

Fig. 2. Illustration of arc extension. **a** the extension area and the Potential Arc Center Area (PACA) **b** center decision when extended to a polyline **c** center decision when extended to a bar

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 Sect. 2.1, 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. 2b,c. 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 arc attributes (the new endpoint, center, and radius) are modified accordingly, and the current extension iteration stops. Otherwise, the next candidate in the queue is tested. This test cycle is repeated until a candidate passes the test or the candidate 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 as to whether the two arc endpoints can be extended to form a circle is done by calculating the distance between them. If the distance is less than twice the line width and the gap area between them is occupied with more than 80% black pixels, the gap between them is bridged and the arc becomes a full circle.

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

2.3 Complexity analysis

The most time consuming operation in the SRAS algorithm is the calculation of the average radius and the averaged square difference (ASD) for each point in the center area, defined in equations (5) and (6), and the consequent condition testing, defined in equations (3) and (4). The time spent on searching for candidates is negligible, because we use the *Planar Position Index* data structure (Liu et al. 1995). The number of candidates at each extension cycle is usually very small. It is only one at the joint, where there is 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. 2. Generally, both the length and the width of the PACA are directly proportional to the arc width, w, as shown in equations (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 equations (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 1, the total number of edges of all VAFs, denoted by N_e, is $N_e = N_p - N_s$. A full circle may be broken into four quarter polylines. Shorter bars may appear at each one of the joints of these polylines. At the positions where the arc is intersected by, and/or is tangent to other lines, the potential segment may be broken into two or more VAFs. However, N_s, the total number of vectorized segments, is usually 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 at least 1 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 SRAS algorithm for a single arc image is therefore $O\left(w^2\theta\sqrt{r}\right)$. 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 $O\left(w^{1.5}\theta\sqrt{r}\right)$. The space required for the algorithm is linear to N_p, i.e., $O\left(\theta\sqrt{r}\right)$.

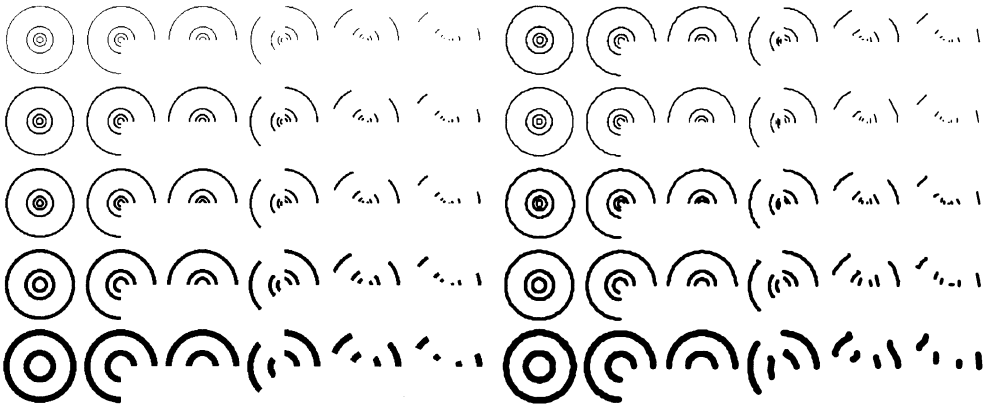

Fig. 3. A synthetic drawing (720X600 pixels) of arcs. **a** synthetic image **b** vectors

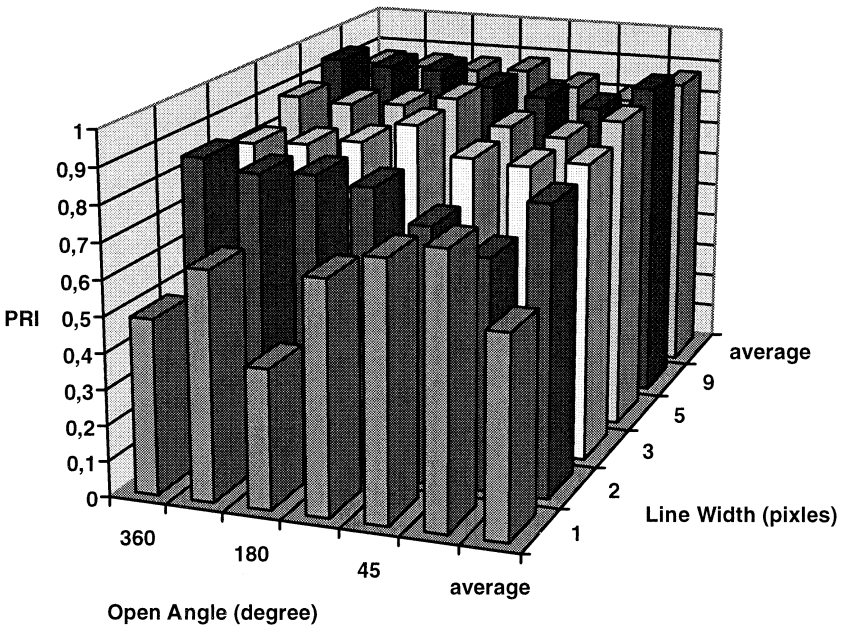

Fig. 4. 3D plot of PRI of Fig. 3

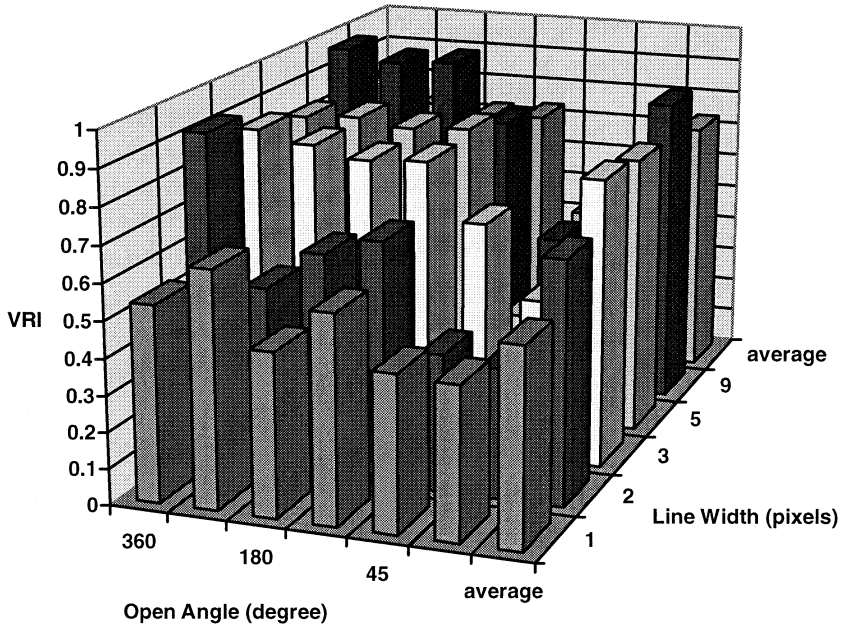

Fig. 5. 3D plot of VRI of Fig. 3

Table 1. Ground truth of the tested circular arcs of Fig. 3a displayed in a matrix, in each cell of which the radii are 5 (only for width = 1, 2, and 3), 10 (only for width = 1, 2, 3, and 5), 20, and 50 pixels

	Column		1	2	3	4	5	6
Row	Width	Open angle	Full Circle $0 \sim 2\pi$	3/4 Circle $0 \sim 3\pi/2$	1/2 Circle $0 \sim \pi$	1/4 Circle $0 \sim \pi/2$ and $3\pi/4 \sim 5\pi/4$	1/8 Circle $0 \sim \pi/4$ and $5\pi/8 \sim 7\pi/8$	1/16 Circle $0 \sim \pi/8$ and $11\pi/16 \sim 13\pi/16$
1	1	Center	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

Table 2. Pixel level evaluation of the Stepwise Recovery Arc Segmentation algorithm

	Column																				
	1			2			3			4			5			6			Total		
Row	D_p	F_p	PRI	D_p	F_p	PRI	D_p	F_p	PRI	D_p	F_p	PRI	D_p	F_p	PRI	D_p	F_p	PRI	D_p	F_p	PRI
1	0.51	0.54	0.49	0.66	0.38	0.64	0.41	0.63	0.39	0.69	0.39	0.65	0.75	0.30	0.72	0.80	0.29	0.76	0.59	0.46	0.56
2	0.86	0.17	0.85	0.83	0.18	0.82	0.83	0.18	0.83	0.80	0.19	0.81	0.75	0.31	0.72	0.66	0.36	0.65	0.82	0.20	0.81
3	0.84	0.19	0.82	0.84	0.19	0.83	0.87	0.17	0.85	0.93	0.12	0.91	0.86	0.19	0.83	0.92	0.29	0.82	0.87	0.18	0.84
4	0.97	0.19	0.89	0.97	0.20	0.88	0.98	0.20	0.89	0.95	0.12	0.92	0.89	0.20	0.85	0.92	0.25	0.83	0.96	0.19	0.89
5	0.97	0.08	0.94	0.97	0.09	0.93	0.98	0.12	0.93	0.95	0.16	0.89	0.89	0.21	0.87	0.92	0.25	0.85	0.96	0.12	0.92
Total	0.91	0.16	0.87	0.91	0.16	0.87	0.91	0.18	0.87	0.92	0.16	0.88	0.89	0.22	0.84	0.89	0.27	0.81	0.91	0.18	0.87
Global	$D_p = 0.91$						$F_p = 0.18$						$PRI = 0.87$								

Table 3. Vector level evaluation of the Stepwise Recovery Arc Segmentation algorithm

	Column																				
	1			2			3			4			5			6			Total		
Row	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
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								

3 Experiments, evaluation, and discussion

The Stepwise Recovery Arc Segmentation (SRAS) algorithm is implemented in C++ within the Machine Drawing Understanding System (MDUS) (Liu and Dori 1996b). The running platform may be **SGI IRIX 5.3** as well as **SUN Solaris 2.5**, and above.

In order to get an objective and comprehensive evaluation of this algorithm, we have developed a performance evaluation protocol (Liu and Dori 1997), which is briefly summarized below. The pixel level evaluation is done using the pixel level detection rate, D_p, and the pixel level false alarm rate, F_p. D_p is the ratio of the number of black pixels in the original image that are detected to the number of original black pixels in the drawing. F_p is the ratio of the number of pixels that are detected but are not in the original to the number of detected pixels. The pixel recovery index is $PRI = (D_p + 1 - F_p)/2$.

The vector level evaluation is carried out as follows. The first step is to match ground truth lines with detected lines. This is done by finding for each line in the ground truth set the subset of detected lines that overlap it 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 the length weighted sum of the Total Quality (Q_v) of all ground truth lines, the Vector False Alarm Rate, F_v, which is the length weighted sum of $1 - Q_v$

Table 4. The time performance (on SGI Indigo 2) of SRAS

Figure #	Size (pixels)	# of segmented arcs	Time (s)	Time/pixel (10^{-6} s)	Time/arc (s)
6	1056×486	20	2.37	4.6	0.12
7	824×600	34	1.82	3.9	0.06
8	344×644	33	1.36	6.1	0.04

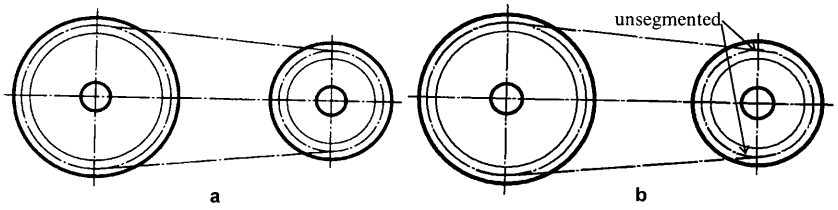

Fig. 6a,b. SRAS applied to a real life drawing taken from ISO (1982). **a** image **b** arc segmentation

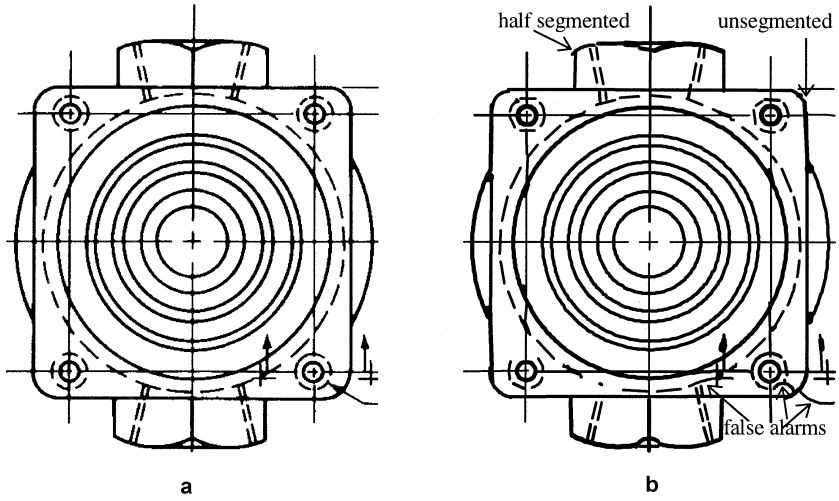

Fig. 7a,b. SRAS applied to a real life drawing from ANSI (1982). **a** image **b** arc segmentation

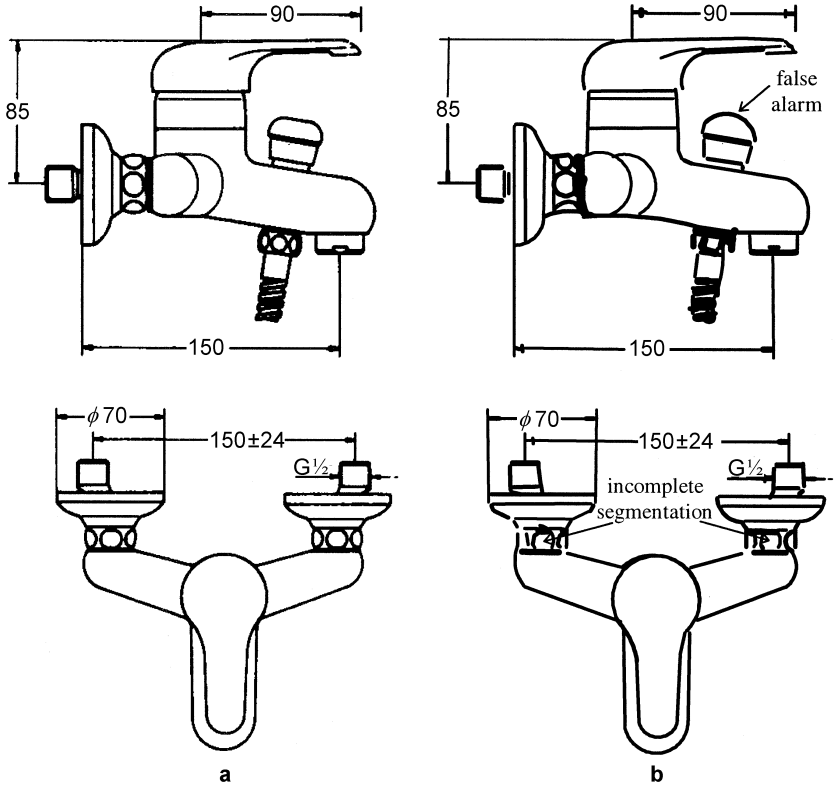

Fig. 8a,b. SRAS applied to a real life drawing from an Israeli tap manual. **a** image **b** arc segmentation

of all detected lines, and the resulting Vector Recovery Index, $VRI = (D_v + 1 - F_v)/2$, are calculated.

To automatically test SRAS's performance, we have manually generated the drawings in Fig. 3a using Microsoft Paint Brush software with the ground truth data shown in Table 1. The vectorized results output by SRASs are displayed in Fig. 3b and the evaluation results are shown in Tables 2 and 3. The arcs are drawn in a 5×6 matrix, the widths of lines in the rows numbered 1–5 from top to bottom are 1, 2, 3, 5, and 9 pixels, the open angles in the columns numbered 1–6 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 5, 10, 20, and 50 pixels. In general, both PRI and VRI increase along with the open angle and the width of the arc, as shown in the 3D plots in Figs. 4 and 5. The highest PRI and VRI are 0.97 for a 9 pixel wide circle, the worst PRIs are around 0.50 for 1 pixel wide arcs, and the worst VRIs are around 0.40 for $\pi/8$ arcs.

We have also applied SRAS to a host of real life drawings taken from various standard definitions with circular arcs in complex line environments. Sample drawings and their arc segmentation results are shown in Figs. 6–8. The results show that SRAS successfully performs arc segmentation from complex line environments, such as concentric arcs, tangency, and intersection of other lines. The cases of intersection with small angle and tangency are the most difficult in arc segmentation, but SRAS usually overcomes these problems. In Fig. 6, which is an ISO (1982) drawing, almost all arcs, including those segments in the dash-dotted circles, are correctly segmented, except for the two small angle ones that are tangent to two straight lines marked on the right hand side of Fig. 6b. There is no false alarm in Fig. 6. In Fig. 7, which is an ANSI (1982) drawing, all the central concentric circles, and most arcs, especially three out of the four small open angle arcs tangent to straight bars and three out of the four quarter arcs on the corners, are correctly segmented, but there are several false alarms. Figure 8 is scanned from an Israel tap manual. There are many small angle arcs, most of which are segmented by SRAS. The time performance (on SGI Indigo 2) of SRAS applied on Figs. 6–8 is shown in Table 4.

4 Summary

A vector-based Stepwise Recovery Arc Segmentation algorithm, which is an application of a generic approach to recognition of graphic objects in engineering drawings, has been presented, implemented, and evaluated. It is based on the hypothesize-and-test paradigm, in which we find a first key component that is a polyline. We try to convert it to an initial arc and extend it to recognize the entire circular arc. We have tested this algorithm with various cases. Experimental results show high detection rate even for complex real life drawings, such as those with concentric arcs, and intensive tangency and intersections.

The vector-based algorithm performs better than those that work at the pixel level. The first advantage is the time efficiency, which has been shown the experiments to be about 0.1 sec/arc. The second advantage is that being a vector-based algorithm, SRAS has a more global view than the pixel-based ones, and therefore it is more robust to noise and clutter found in real life drawing. However, since it is vector-based, its performance is dependent on the quality of vectorization.

The SRAS algorithm is better than the other vector-based algorithm PBT (Dori 1995), which finds all of the components at once, because the clue that a wire fragment is a vectorized arc fragment (VAF) is very weak and can be easily shattered by an interfering element. Since SRAS finds one component at a time, it can select the best candidate and not be mislead by clutter in the arc's neighborhood. SRAS has been incorporated into the MDUS environment (Liu and Dori 1996b) and is very instrumental in segmenting arcs from engineering drawings processed by our system.

References

1. ANSI (1982) Dimensioning and tolerancing ANSI Y14.5M
2. Asada H, Brady M (1986) The curvature primal sketch. IEEE Trans. on PAMI 8:1–14
3. Chai I, Dori D (1992) Orthogonal Zig-Zag: An efficient method for extracting lines from engineering drawings. In: Arcelli C, Cordella LP, Sanniti di Baja G (eds) Visual Form. Plenum Press, New York, pp 127–136
4. Conker RS (1988) A dual plane variation of the Hough Transform for detecting nonconcentric circles of different radii. Computer Vision, Graphics, and Image Processing 43:115–132
5. Dori D (1995) Vector-based arc segmentation in the machine drawing understanding system environment. IEEE Trans. on PAMI 17(11):1057–1068
6. Dori D (1997) Orthogonal Zig-Zag: an algorithm for vectorizing engineering drawings compared with Hough Transform. Advances in Engineering Software 28(1):11–24
7. Hallard DH (1981) Generalizing the Hough Transform to detect arbitrary shapes. Pattern Recognition 13(2):111–122
8. Haralick RM, Shapiro L (1992) Computer and robot vision. Addison Wesley, Reading, MA
9. Hunt DJ, Nolte LW (1988) Performance of the Hough Transform and its relationship to statistical signal detection theory. Computer Vision, Graphics, and Image Processing 43:221–238
10. Illingworth J, Kittler J (1987) The Adaptive Hough Transform. IEEE Trans. on PAMI 9(5):690–697
11. ISO (1982) Technical Drawings. ISO Standard Book 12
12. Kimme C, Ballard DH, Sklansky J (1975) Finding circles by an array of accumulators. Communications of the ACM 18:120–122
13. Liu W, Dori D, Tang L, Tang Z (1995) Object recognition in engineering drawings using planar indexing. In: Proc. 1st IAPR Workshop on Graphics Recognition, Penn. State Univ., PA, pp 53–61
14. Liu W, Dori D (1996a) Sparse pixel tracking: a fast vectorizaiton algorithm applied to engineering drawings. In: Proc. 13th ICPR, Vienna, Vol III (Robotics and Applications) pp 808–811

15. Liu W, Dori D (1996b) Automated CAD conversion with the machine drawing understanding system. In: Proc. 2nd IAPR Workshop on Document Analysis Systems, Malvern, PA, pp 241–259
16. Liu W, Dori D (1997) A protocol for performance evaluation of line detection algorithms. Machine Vision Applications 9:240–250
17. Lowe DG (1987) Three dimensional object recognition from single two dimension images. Artificial Intelligence 31:355–395
18. O'Gorman L, Sanderson AC (1984) The converging squares algorithm: an efficient method for locating peaks in multidimensions. IEEE Trans. on PAMI 6:280–288
19. O'Gorman L (1988a) Curvilinear feature detection from curvature estimation. In: Proc. 9th ICPR, Rome, pp 116–119
20. O'Gorman L (1988b) An analysis of feature detectability from curvature estimation. In: Proc. IEEE Conference on CVPR, Ann Arbor, MI, pp 235–240
21. Rosin PL, West GAW (1989) Segmentation of edges into lines and arcs. Image and Vision Computing 7(2):109–114

Dov Dori is Head of the Area of Information Systems Engineering at The William Davidson Faculty of Industrial Engineering and Management, Technion, Israel Institute of Technology. He received his B.Sc. in Industrial Engineering and Management from the Technion in 1975, M.Sc. in Operations Research from Tel Aviv University in 1981, and Ph.D. in Computer Science from Weizmann Institute of Science, Rehovot, Israel, in 1988. Between 1987 and 1990 he was Assistant Professor at the University of Kansas, Lawrence, Kansas. His research interests include Document Analysis and Recognition, Computer Vision and Pattern Recognition, Information Systems Engineering, Systems Development Methodologies, and Computer-Aided Software Engineering. Dov Dori has developed the Machine Drawing Understanding System (MDUS) and the Object-Process Methodology (OPM). In 1995 he won First Place in the Dashed Line Detection Contest, held at Pennsylvania State University as part of the International Workshop on Graphics Recognition. In 1997 he won the Hershel Rich Technion Innovation Award for the development of OPCAT - Object-Process CASE Tool, that supports OPM. Dov Dori is on the editorial board of International Journal of Pattern Recognition and Artificial Intelligence and author of over 40 journal papers and 60 conference publications. He is co-editor of Shape Structure and Pattern Recognition and a member of IEEE, IEEE Computer Society, ACM and IAPR.

Liu Wenyin was born in Jilin Province, People's Republic of China, 1966. He received his BEngg (1988) and MEngg (1992) degrees from the Department of Computer Science and Technology, the TsingHua University, Beijing, People's Republic of China. Currently he is a DSc candidate at the Faculty of Industrial Engineering and Management, Technion, Israel Institute of Technology, Haifa, Israel. Between 1992 and 1995 he was lecturer at the Department of Computer Science and Technology, the Tsinghua University, Beijing, People's Republic of China. His research interests include automated engineering drawing interpretation, pattern recognition, software engineering, object-oriented programming, object-process methodology, and artificial intelligence. Liu Wenyin was a senior member of the team that won first place in the Dashed Line Recognition Contest held during the First IAPR Workshop on Graphics Recognition at Pennsylvania State University, 1995. He also won a Third Prize in the First International Java Programming Contest (ACM Quest for Java'97), sponsored by the Association of Computing Machinery (ACM) and IBM, 1997. He is a student member of ACM.