# Fast Polygonal Approximation of Digital Curves Using Relaxed Straightness Properties

Partha Bhowmick and Bhargab B. Bhattacharya

**Abstract**—Several existing digital straight line segment (DSS) recognition algorithms can be used to determine the digital straightness of a given one-pixel-thick digital curve. Because of the inherent geometric constraints of digital straightness, these algorithms often produce a large number of segments to cover a given digital curve representing a real-life object/image. Thus, a curve segment, which is not exactly digitally straight but appears to be visually straight, is fragmented into multiple DSS when these algorithms are run. In this paper, a new concept of approximate straightness is introduced by relaxing certain conditions of DSS, and an algorithm is described to extract those segments from a digital curve. The number of such segments required to cover the curve is found to be significantly fewer than that of the exact DSS cover. As a result, the data set required for representing a curve also reduces to a large extent. The extracted set of segments can further be combined to determine a compact polygonal approximation of a digital curve based on certain approximation criteria and a specified error tolerance. The proposed algorithm involves only primitive integer operations and, thus, runs very fast compared to those based on exact DSS. The overall time complexity becomes linear in the number of points present in the representative set. Experimental results on several digital curves demonstrate the speed, elegance, and efficacy of the proposed method.

**Index Terms**—Digital geometry, digital straight line, polygonal approximation, shape analysis.

## 1 Introduction

The efficient representation of lines and curves in the digital plane has been an active subject of research for nearly half a century [24], [25], [37]. In particular, digital straight line segments (DSS)[1] have drawn special attention for their challenging nature from the viewpoint of theoretical formulation and also for their potential applications to image analysis and computer graphics. In a digital image containing one or more objects with fairly straight edges, the set of (exact or approximate) DSS captures a strong geometric property that can be used for shape abstraction of the underlying objects, as well as for finding the resemblance among several digital objects.

The necessary and sufficient conditions for a discrete/digital curve (DC) to be a DSS have been stated in the literature in various forms [17], [24], [35], [37]. In [35], it has been shown that a DC is the digitization of a straight line segment if and only if it has the *chord property*. A DC $\mathcal{C}$ has the chord property if, for every $(p, q)$ in $\mathcal{C}$, the chord $\overline{pq}$ (the line segment drawn in the real plane, joining $p$ and $q$) "lies near" $\mathcal{C}$, which, in turn, means that, for any point $(x, y)$ of $\overline{pq}$, there exists some point $(i, j)$ of $\mathcal{C}$ such that $\max(|i - x|, |j - y|) < 1$. A few other definitions related to this work are given below.[2]

*Chain Code*. If $p\ (i, j)$ is a grid point, then the grid point $(i', j')$ is a neighbor of $p$, provided that $\max(|i - i'|, |j - j'|) = 1$. The chain code [16], [17] of $p$ with respect to its neighbor grid point in $\mathcal{C}$ can have a value in $\{0, 1, 2, \ldots, 7\}$, as shown in Fig. 1a.

*Digital Curve* (DC). A DC $\mathcal{C}$ is an ordered sequence of grid points (representable by chain codes) such that each point (excepting the first one) in $\mathcal{C}$ is a neighbor of its predecessor in the sequence (see Figs. 1b, 1c, and 1d).

*Irreducible Digital Curve*. A DC $\mathcal{C}$ is said to be irreducible if and only if the removal of any grid point in $\mathcal{C}$ makes $\mathcal{C}$ disconnected. All the DC shown in Fig. 1 are irreducible. A DSS is essentially an irreducible DC.

An example of an open-end DC is shown in Fig. 1b. The traversal of an open-end DC using Depth First Search (DFS) [10] starts from one of its two endpoints, say, $s$. Thus, the chain code of $\mathcal{C}$ is given by (1, 2)`10756543`, considering $s = (1, 2)$. If $\mathcal{C}$ has some branching (Fig. 1c), then the complete chain code of $\mathcal{C}$ can be written as (1, 2)`10756543`(3, 4)`76`. If $\mathcal{C}$ is a closed curve, then the DFS may start from any suitable grid point of $\mathcal{C}$ (Fig. 1d).

Several intriguing problems and properties related to DSS and digital straight line/ray (DSL) have been studied by various authors [7], [33], [37]. Many attributes of DSS can be interpreted in terms of continued fractions [26], [29], [46]. The most fundamental problem, which is highly relevant to pattern recognition in general and to curve approximation in particular, is to ascertain whether or not a given DC is a DSS. Many solutions to this problem have been reported in the literature [11], [12], [13], [27], [28], [43].

The proposed work introduces a new concept of approximate digital straight line segments (ADSS) by preserving

1. The acronyms "DC," "DSL," "DSS," and "ADSS" have been used in this paper in both singular and plural senses, depending on the context.

2. The definitions and discussions in this paper are with respect to the 8-neighborhood connectivity [24] of the object and are valid for the 4-neighborhood as well with certain modifications.

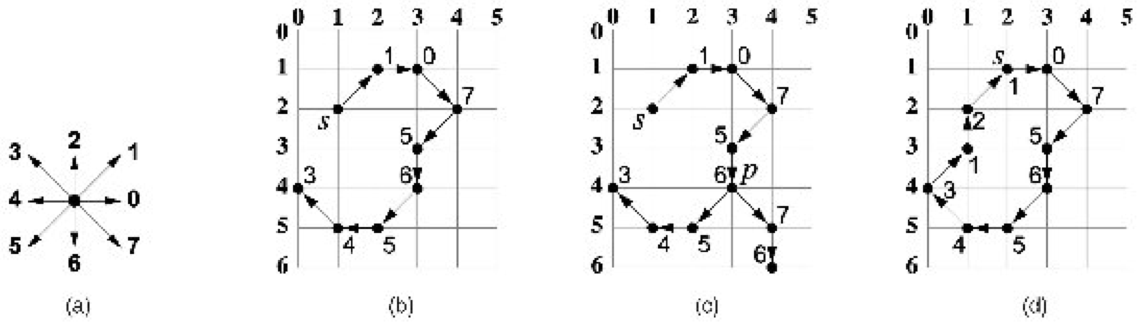


Fig. 1. Chain codes and their enumeration for defining DC. (a) Chain codes in 8-neighborhood connectivity. (b) (1, 2)`10756543`. (c) (1, 2)`10756543`(3, 4)`76`. (d) (2, 1)`0756543121`.

some of the most fundamental properties of a DSS, while relaxing or dropping a few others (see Fig. 3). A procedure is then described for extracting the set of ADSS required to cover a given DC assuming 8-neighborhood connectivity. The number of ADSS extracted from a set of DC in a real-world scenario is likely to be appreciably fewer than that of a DSS cover since many visually straight segments may fail to satisfy all the stringent properties of an exact DSS and, thus, are recognized as multiple DSS by the extraction algorithms. Some examples of DSS and ADSS present in DC have been shown in Fig. 2. It may be observed that the set of DSS in Fig. 2 contains 48 fragments, each of which is "exactly straight," whereas that of ADSS contains only 20, which look "visually straight."

The concept of ADSS can also be used to construct a polygonal approximation of a DC efficiently. Since the set of ADSS provides an elegant and compact representation of DC, it is very effective in producing approximate polygons (or polychains) using a single parameter. The whole process consists of two stages—first, extraction of ADSS and, second, polygonal approximation. The major features of the algorithm are listed as follows:

1. The detection of ADSS in stage 1 is based on simple chain code properties; thus, only primitive integer operations, namely, comparison, increment, shift, and addition (subtraction), are required.
2. The ADSS extraction algorithm does not use any recursion and thus saves execution time.
3. To compute the polygonal approximation in stage 2, only the two endpoints of each ADSS are required as input data and a few integer multiplications as operations; thus, the algorithm runs very fast.
4. The actual approximation of a DC never oversteps the worst case approximation for a given value of a control parameter. That is, the maximum deviation (of an edge) of the resulting polygon from the original curve never exceeds the prescribed value of the approximation (control) parameter.

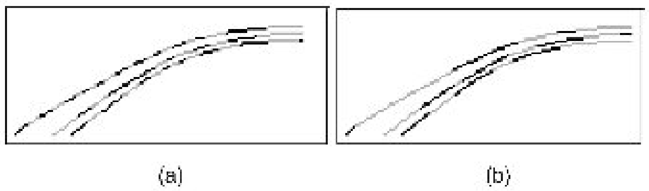


Fig. 2. Set of (a) DSS and that of (b) ADSS extracted from a small set of DC (the segments alternately colored in black and gray).

Several other methods [8], [9], [20], [50] have been proposed recently for (approximate) line detection. Most of the conventional parametric approaches are based on certain distance criteria, usage of masks, eigenvalue analysis, the Hough transform, and so forth. In contrast, the proposed method relies on utilizing some of the basic properties of DSS for extraction of ADSS.

Many algorithms for approximating a given DC or contour are well known [1], [3], [23]. Several variants of efficient but suboptimal algorithms had been proposed later [5], [6], [32], [41], [42]. The class of polygonal approximation algorithms, in general, can be broadly classified into two categories—one in which the number of vertices of the approximate polygon(s) is specified and the other where a distortion criterion (for example, maximum Euclidean distance) is used.

Most of the existing polygonal approximation algorithms, excepting a few, have superlinear time complexities, for example, $\mathcal{O}(N)$ in [47], $\mathcal{O}(MN^2)$ in [32], $\mathcal{O}(N^2)$ in [41] and [42], and $\mathcal{O}(N^3)$ in [34], where $M$ denotes the number of segments, and $N$ denotes the total number of points representing the input set of DC. A comparative study of these algorithms can be found in [51]. Further, in order to analyze curvature, most of them require intensive floating-point operations [2], [15], [18], [44], [49]. For other details, the reader may look at [4], [14], [31], [47], [48], [39], [44], [52], and [53]. The method proposed here uses only integer operations and yields a suboptimal polygonal approximation with linear time complexity. A comparison of the proposed technique with some of the important approaches is shown in Table 1.

## 2 Exact and Approximate Straight Line Segments

In this work, we use some regularity properties of DSS that can be successfully derived from the chord property. Before justifying the rationale of our algorithm, the DSS properties (defined with respect to chain codes [16]) [17], [35] are listed below:

(F1) At most two types of elements can be present and these can differ only by unity, modulo eight.

(F2) One of the two element values always occurs singly.

(F3) Successive occurrences of the element occurring singly are as uniformly spaced as possible.

TABLE 1
A Comparative Study of Some Existing Algorithms with the Proposed One

| | **Algorithm** and its features | Non-Euclidean | Flexibility[a] | Procedural complexity | Difficulty in implementation | Error control | Non-recursive |
|---|---|---|---|---|---|---|---|
| 1. | **curvature maxima:** (i) region of support; (ii) measure of significance;[b] (iii) nonmaxima suppression [44]. | no | no | high[c] | medium | no | no |
| 2. | **ant colony search:** (i) graph representation; (ii) node transition rule; (iii) pheromone updating rule [52]. | no | no | very high[d] | high | yes | no |
| 3. | **area deviation** per unit length of approx. segment [47]. | yes[e] | no | low | low | yes | yes |
| 4. | **perceptual organization:** (i) link classification;[f] (ii) linking-merging; (iii) smoothing; (iv) application of knowledge/rule [22]. | yes | no | high[g] | high | yes | no |
| 5. | **using ADSS** (proposed). | yes | yes | very low | low | yes | yes |

[a]w.r.t. (self-)intersecting/branching curves when the input set of grid points constituting a curve is not ordered
[b]e.g., cosine curvature, $k$ curvature, 1 curvature, etc.
[c]due to multiple iterations for nonmaxima suppression (in 4 passes) and curvature finding
[d]due to complex calculations, e.g., exponentiation in selection problem of a node
[e]by replacing $\sqrt{x^2+y^2}$ with $(|x|+|y|)$ or $\max\{|x|,|y|\}$
[f]parallel links (class 1 & 2), intersection links, and single links.
[g]due to three stages, each using multiplications for the entire set of points on the curve

The Properties (F1-F3) were based on heuristic insights [17]. Further, the Property (F3) is not precise enough for a formal proof, as stated in [30]. A formal characterization of DSS was provided later in [35], stated as follows:

(R1) The runs have at most two directions, differing by 45 degrees, and, for one of these directions, the run length must be 1.

(R2) The runs can have only two lengths, which are consecutive integers.

(R3) One of the run lengths can occur only once at a time.

(R4) For the run length that occurs in runs, these runs can themselves have only two lengths, which are consecutive integers, and so on.

A few instances have been given in Section 2.1 (see Fig. 3) to clarify the significance of (R1-R4) in characterizing a DSS.

## 2.1 Extraction of ADSS

In the proposed algorithm EXTRACT-ADSS, designed for extraction of ADSS from a DC, we have used (R1) along with certain modifications in (R2). However, we have dropped (R3) and (R4), since they impose very tight restrictions on a DC to be recognized as a DSS. Such a policy, with the adoption of (R1), modification of (R2), and omission of (R3) and (R4), has been done in order to successfully extract the ADSS from a DC, and some of the major advantages of this scheme are given as follows:

- avoiding tight DSS constraints, especially for the curves representing the gross pattern of a real-life image with certain digital aberrations/imperfections,
- enabling extraction of ADSS from a DC, thereby straightening a part of the DC when the concerned part is not exactly "digitally straight,"
- reducing the number of extracted segments, thereby decreasing storage requirement and runtime in subsequent applications,
- reducing the CPU time of ADSS extraction, and
- using integer operations only.[3]

3. No floating-point operations are required in EXTRACT-ADSS. Only integer operations for addition, shift, and comparison are necessary. Even multiplications and divisions have been avoided; for example, to compute $\lfloor (p+3)/4 \rfloor$, 3 is added with $p$, followed by two successive right shifts.

Since the chain code of a DC is a one-dimensional list $\mathcal{C}$, the ADSS may be characterized by the following sets of parameters:

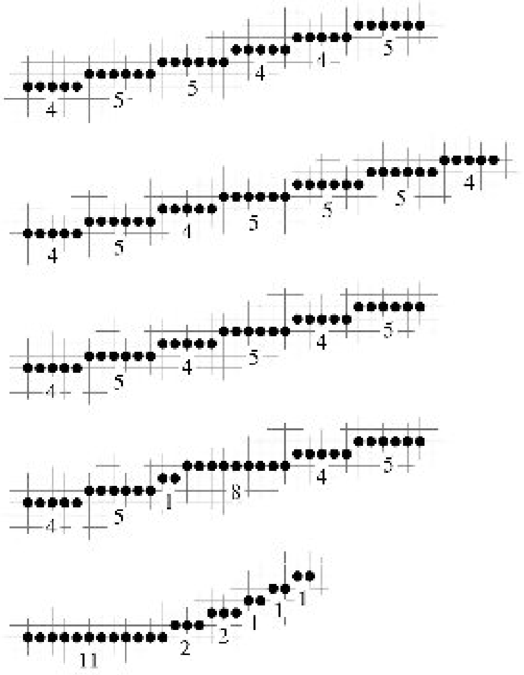
1. *Orientations parameters* given by $\mathtt{n}$ (nonsingular element), $\mathtt{s}$ (singular element), $l$ (length of the leftmost run of $\mathtt{n}$), and $r$ (length of the rightmost run of $\mathtt{n}$), which play decisive roles on the orientation (and the digital composition thereof) of the concerned ADSS. For example, in Fig. 3, the curve $\mathcal{C}_1$ has $\mathtt{n}=0$, $\mathtt{s}=1$, and chain code $0^4 1 0^5 1 0^5 1 0^4 1 0^4 1 0^5$ having $l=4$ and $r=5$.
2. *Run length interval parameters* given by $p$ and $q$, where $[p,q]$ is the range of possible lengths (excepting $l$ and $r$) of $\mathtt{n}$ in $\mathcal{C}$ that determines the level of approximation of the ADSS, subject to the following conditions:

$$\text{(c1)} \quad q-p \leq d = \lfloor (p+1)/2 \rfloor, \tag{1}$$

$$\text{(c2)} \quad (l-p),(r-p) \leq e = \lfloor (p+1)/2 \rfloor. \tag{2}$$

While implementing EXTRACT-ADSS, we have strictly adhered to (R1), as it is directly related to the overall straightness of a DC. However, we have modified the stricture in (R2) by considering that the run lengths of $\mathtt{n}$ can vary by more than unity, depending on the minimum run length of $\mathtt{n}$. The rationale of modifying (R2) to the Condition (c1) lies in the fact that, in order to approximate the extracted line segments from the DC, an allowance of approximation $(d)$ specified by (c1) may be permitted. Given a value of $p$, the amount $d$ by which $q$ is in excess of $p$ indicates the deviation of the ADSS from the actual/real line, since, ideally (for a DSS), $q$ can exceed from $p$ by at most unity, and the significance of $d$ in characterizing an ADSS is as follows.

Without loss of generality, let $\mathbf{L}$ be an ADSS with slope in [0, 1]. Let $p_0 = p$, $p_1 = p_0 + 1$, $p_2 = p_1 + 1, \ldots, p_d = q$ be the run lengths of $\mathtt{0}$s present in $\mathbf{L}$, and let the frequency of the run length $p_i$ in $\mathbf{L}$ be $n_i \ (\geq 0)$ for $i = 0, 1, \ldots, d$. Then, the slope of the real line joining the start point and the endpoint of $\mathbf{L}$ is given by



$\mathcal{C}_1$ with chain code $0^4 10^5 10^5 10^4 10^4 10^5$ (from left to right) ($p = 4, q = 5, l = 4, r = 5$) that does not satisfy (R3), since both the run lengths 4 and 5 have non-singular occurrences in the code of run lengths: 455445. $\mathcal{C}_1$ is not a DSS but an ADSS.

$\mathcal{C}_2$ with chain code $0^4 10^5 10^4 10^5 10^5 10^5 10^4$ ($p = 4, q = 5, l = 4, r = 4$) does not satisfy (R4), since in the run length code 4545554, the runs of 5 have lengths 1 and 3 that are not consecutive. $\mathcal{C}_2$ is not a DSS but an ADSS.

$\mathcal{C}_3$ with chain code $0^4 10^5 10^4 10^5 10^4 10^5$ ($p = 4, q = 5, l = 4, r = 5$) that satisfies (R1-R4) and (c1, c2). $\mathcal{C}_3$ is an ADSS as well as a DSS.

$\mathcal{C}_4$ with chain code $0^4 10^5 1010^8 10^4 10^5$ ($p = 1, q = 8, l = 4, r = 5$) that does not satisfy (R2) and conditions (c1, c2). $\mathcal{C}_4$ is neither a DSS nor an ADSS.

$\mathcal{C}_5$ with chain code $0^{11} 10^2 10^2 101010$ ($p = 1, q = 2, l = 11, r = 1$) that violates (R2) and is, therefore, not a DSS. Further, although it satisfies (c1) (as $q - p\,(= 1) \leq d\,(= 1)$), but since $l - p\,(= 10) \not\leq e\,(= 1)$, it fails to satisfy (c2); and therefore, it is not even recognized as an ADSS.

Fig. 3. Instances of DC showing the significance of properties and conditions related with DSS and ADSS recognition.

$$m = (N-1) \Big/ \sum_{i=0}^{d} n_i(p_i + 1), \tag{3}$$

where $N = n_0 + n_1 + \ldots + n_d$. Hence, using the fact that $p \leq p_i \leq q$ for $i = 0, 1, \ldots, d$, we get

$$\frac{N-1}{N}\frac{1}{q+1} \leq m \leq \frac{N-1}{N}\frac{1}{p+1}. \tag{4}$$

Thus, when the real line intersects the concerned run (if the real line does not intersect the concerned run of $\mathbf{L}$, then the error is greater), the isothetic error of a run length $p_i$ in $\mathbf{L}$ (see Fig. 4) is given by

$$\epsilon \leq m(p_i + 1) \leq \left(1 - \frac{1}{N}\right)\left(1 + \frac{d}{p+1}\right) \leq 1 + \frac{d}{p+1}. \tag{5}$$

Hence, it is evident that the error incurred with an ADSS is controlled by $d$, and, by (5), the lower the run length ($p$) of n, the lower would be this allowance of approximation. Thus, we keep the provision for adaptively changing this allowance of approximation, so that elongation of an ADSS is made as much as possible till it does not lose its overall visual straightness.

Apart from $d$, the other parameter, namely, $e$, is incorporated in (c2), which, along with (c1), ensures that the extracted ADSS is not badly approximated, owing to some unexpected values of $l$ and $r$. The DSS properties (R1-R4), however, do not give any idea about the possible values of $l$ and $r$ (depending on n). Further, in the algorithm for DSS recognition [11], $l$ and $r$ are not taken into account for adjudging the DSS characteristics of a DC. However, we have imposed some bounds on the possible values of $l$ and $r$ in order to ensure a reasonable amount of straightness at either ends of an extracted ADSS. The values of $d$ and $e$ are heuristically chosen so that they become computable with integer operations only. Some other values like $d = \lfloor (p+3)/4 \rfloor$ and $e = \lfloor (p+1)/2 \rfloor$ or so may also be chosen, provided that the computation is realizable in the integer domain and does not produce any undesirable ADSS. For example, in Fig. 3, the curve $\mathcal{C}_5$ has $p = 1, q = 2, l = 11$, and $r = 1$. In our case ((1) and (2)), therefore, we get $d = 1$ and $e = 1$ resulting in a violation of (c2) by $l$; thus, $\mathcal{C}_5$ will not be accepted as an ADSS.

To justify the rationale of (c1) and (c2), we consider a few DC, $\mathcal{C}_1$-$\mathcal{C}_5$, as shown in Fig. 3. It is interesting to observe that, although each of $\mathcal{C}_1$ and $\mathcal{C}_2$ has the appearance of a digital line segment, they fail to hold all the four properties of DSS simultaneously, as shown in their respective figures. The curve $\mathcal{C}_1$ violates (R3), and the curve $\mathcal{C}_2$ violates (R4). However, they satisfy (R1), (c1), and (c2) and, therefore, each of them is declared as an ADSS. Similarly, the curve $\mathcal{C}_3$ satisfies (R1-R4), (c1), and (c2); it is both an ADSS and a DSS. However, none of the curves $\mathcal{C}_4$ and $\mathcal{C}_5$ can be announced as a DSS or an ADSS because of the violation of (R2), (c1), and (c2) (see Fig. 3).

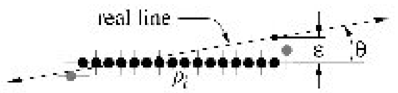


Fig. 4. Maximum (isothetic) error $\epsilon$ corresponding to a run length $p_i$ (of 0s) —with $p_i + 1$ number of grid points in the corresponding row—for an ADSS representing a real line segment with slope $m = \tan\theta$, $0 \leq m \leq 1$. Although the real line intersects the run, in some other instance, it may not.

## 2.2 Algorithm Extract-ADSS

Fig. 5 describes the algorithm EXTRACT-ADSS for extracting ADSS from the chain code of each DC, say, $\mathcal{C}_k$, stored in the list $\mathcal{C}$. This requires $n_k$ repetitions from Step 2 through Step 23, where $n_k$ is the number of ADSS in $\mathcal{C}_k$. Let the $i$th repetition on $\mathcal{C}_k$ produce the ADSS $\mathbf{L}_i^{(k)}$. Recognition of $\mathbf{L}_i^{(k)}$ is prompted by

```
Algorithm EXTRACT-ADSS ($\mathcal{C}$)

Steps:
 1. $\mathcal{A} \leftarrow \{1\}, u \leftarrow 1$
 2. FIND-PARAMS ($\mathcal{C}, u$)
 3. $c \leftarrow l$
 4. if $s - n \pmod 8 \not\equiv 1$ then goto step 20
 5. $p \leftarrow q \leftarrow$ length of next run of $n$
 6. $d \leftarrow e \leftarrow \lfloor (p+1)/2 \rfloor$
 7. $c \leftarrow c + 1 + p$
 8. if $l - p > e$ then goto step 20
 9. while $q - p \leq d$
10.     $k \leftarrow$ length of next run of $n$
11.     if $l - k > e$ then
12.         $c \leftarrow c + 1 + k$
13.         break
14.     if $k - p \leq e$ then $c \leftarrow c + 1 + k$
15.     else $c \leftarrow c + 1 + p + e$
16.     if $k < p$ then
17.         $p \leftarrow k$
18.         $d \leftarrow e \leftarrow \lfloor (p+1)/2 \rfloor$
19.     if $k > q$ then $q \leftarrow k$
20. $u \leftarrow u + c$
21. $\mathcal{A} \leftarrow \mathcal{A} \cup \{u\}$
22. $u \leftarrow u - 1$ ▷ next start point
23. repeat from step 2 until $\mathcal{C}$ is finished
```

Fig. 5. Algorithm EXTRACT-ADSS to find out the ordered list $\mathcal{A}$ of endpoints of ADSS in the input curve $\mathcal{C}$ that contains the chain code for each connected component.

```
Procedure FIND-PARAMS ($\mathcal{C}, u$)

Steps:
 1. $i \leftarrow u$
 2. if $\mathcal{C}[i] = \mathcal{C}[i+1]$ then
 3.     $n \leftarrow \mathcal{C}[i]$
 4.     $s \leftarrow$ element $\neq n$ following $\mathcal{C}[i+1]$
 5.     $l \leftarrow$ leftmost run length of $n$
 6.     return
 7. else
 8.     $n \leftarrow \mathcal{C}[i], s \leftarrow \mathcal{C}[i+1], l \leftarrow 1$
 9.     $i \leftarrow i + 1$
10.     while $\mathcal{C}[i+1] \in \{n, s\}$
11.         if $\mathcal{C}[i] = \mathcal{C}[i+1]$ then
12.             if $\mathcal{C}[i] = s$ then
13.                 swap $n$ and $s$
14.                 $l \leftarrow 0$
15.             return
16.         else
17.             $i \leftarrow i + 1$
18.     return
```

Fig. 6. The procedure FIND-PARAMS that finds $n$, $s$, and $l$ from $\mathcal{C}$.

finding its corresponding parameters $(n, s, l)$ using the FIND-PARAMS procedure (Fig. 6) in Step 2 of EXTRACT-ADSS. This is followed by checking/validation of

1. Property (R1): Step 4 and Step 10;
2. Condition (c1): **while** loop check at Step 9; and
3. Condition (c2): on the leftmost run length $l$ in Step 8 and Step 11, and on the rightmost run length $r$ in Step 14.

*Proof of correctness.* For each ADSS $\mathbf{L}_i^{(k)}$, we show that Property (R1) and Conditions (c1) and (c2) are simultaneously satisfied. We also show that $\mathbf{L}_i^{(k)}$ is maximal in length in $\mathcal{C}_k$ in the sense that inclusion of the character ($n$ or $s$ or any other in $\{0, 1, \ldots, 7\}$) (or a substring of characters) that immediately precedes or follows the part of DC corresponding to $\mathbf{L}_i^{(k)}$ in $\mathcal{C}_k$ does not satisfy the ADSS property/conditions.

While checking (R1) in Step 4 or Step 10, if an expected $n$ or $s$ is not found at the desired place in $\mathcal{C}_k$, then the current ADSS $\mathbf{L}_i^{(k)}$ ends with the previously checked valid characters. This is explicit in Step 4 and implicit in Step 10. Thus, $\mathbf{L}_i^{(k)}$ satisfies (R1) and is maximal from its starting point and finishing end, since either it is the first ADSS in $\mathcal{C}_k$ or the previous ADSS $\mathbf{L}_{i-1}^{(k)}$ was maximal.

Now, for each new run (of $n$), (c1) is verified in Step 9—excepting the leftmost run $l$, which is not required since $p$ (and $q$) does not exist for a single run—after appropriately updating $p$ and $q$ in Step 17 and Step 19, respectively, whenever necessary. In Step 9, if it is found that $q$ is unacceptably large (that is, $q \not\leq p + d$), then the **while** loop (Steps 10-19) is not executed, and the current ADSS $\mathbf{L}_i^{(k)}$ ends with the truncated part of that run (truncated maximally, that is, up to length $p + e$, in Step 15 of the previous iteration) as its rightmost run $r$.

In checking (c2), however, we have to be more careful. For the second run (that is, the run immediately following $l$) of the current ADSS, (c2) is checked (with respect to $l$) in Step 8. It may be noted that if $l - p > e$, then (c2) is not satisfied and, so, the first two runs ($l$ and its successor) trivially constitute an ADSS by Step 7, because, for two runs, we get only $l$ and $r$ (and no $p$ or $q$), and no relation is imposed between $l$ and $r$ to define an ADSS.

For the third and the subsequent run(s), if any, the corresponding run length is stored in $k$ (Step 10). If some (small enough) $k$ violates (c2), then that $k$ is treated as $r$ Steps 11-13), and the current ADSS ends with that run as the rightmost run (of run length $k$), whereby the maximality criterion of the ADSS is fulfilled. Otherwise, if $k$ does not exceed the maximum possible length of the rightmost run (checked in Step 14), then we consider $k$ as a valid run of the current ADSS (Step 14); else, we truncate it to the maximum permissible length $(p + e)$ as the rightmost run (Step 15). Note that, if $k > p + e$, then $k > q$ (for $p + e \geq p + d \geq q$), and Step 19 updates $q$ to $k$; hence, (c1) will be false in Step 9 in the next iteration and, so, the ADSS will end here with the (maximally) truncated part $(p + e)$ as its rightmost run.

*Time Complexity.* Determination of the parameters $(n, s, l)$, in FIND-PARAMS consists of two cases—the first one (Steps 2-6) being easier than the second (Steps 7-18). In either of these two cases, the procedure searches linearly in $\mathcal{C}$ for two distinct (but not necessarily consecutive) chain code values and determines the parameters accordingly. As evident from the loop in either case, the three parameters are obtained using only a few integer comparisons. The number of comparisons is $l - 1$ for the first case, and that for the second case is the number of characters in $\mathcal{C}$ until two consecutive nonsingular characters are found.

The parameters $n$, $s$, and $l$ obtained in FIND-PARAMS are successively passed through a number of check points, as mentioned earlier, which take constant time as evident in Steps 3-8 of EXTRACT-ADSS. In Step 5 of EXTRACT-ADSS, the first run length of $n$ is measured immediately after the leftmost run length of $n$, if any, and it starts from the first nonsingular character out of the two consecutive characters detected in FIND-PARAMS. In Step 10 of EXTRACT-ADSS, we have another simple (and silent) loop that determines in linear time each valid run of $n$ in $\mathcal{C}$, the validity criteria being verified and updated in Steps 9-19, with each of these steps

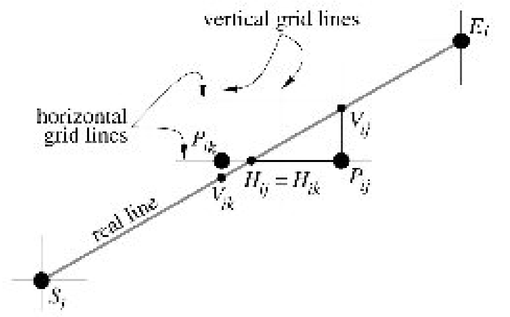


Fig. 7. Isothetic distance of $\mathrm{P}_{ij}$ (from $\overline{\mathrm{S}_i\mathrm{E}_i}$) is $\overline{\mathrm{P}_{ij}\mathrm{V}_{ij}}$, which is greater than $\frac{1}{2}$, thereby making $\mathrm{P}_{ij}$ an error point, whereas $\mathrm{P}_{ik}$ is not an error point, since the isothetic distance of $\mathrm{P}_{ik}$ is $\overline{\mathrm{P}_{ik}\mathrm{V}_{ik}}$, which is lesser than $\frac{1}{2}$.

taking constant time. Hence, for the ADSS $\mathbf{L}_i^{(k)}$, the algorithm EXTRACT-ADSS, together with the procedure FIND-PARAMS, takes linear time; therefore, the time complexity for extraction of all ADSS in $\mathcal{C}$ is strictly linear on the number of points in $\mathcal{C}$.

### 2.3 Error Points

An ADSS extracted from an input DC may not be a perfect DSS. There may occur erroneous points. An erroneous point or error point is that whose isothetic distance (that is, the minimum of the vertical distance and the horizontal distance) from the real straight line corresponding to the concerned ADSS is greater than $\frac{1}{2}$. To check whether a point is an error point or not, we use (6), stated as follows.

Let $\mathrm{S}_i$ and $\mathrm{E}_i$ be the start point and the end point of the $i$th ADSS and let $\mathrm{P}_{ij}$ be the $j$th digital point on the $i$th ADSS, as shown in Fig. 7. Let $\overline{\mathrm{S}_i\mathrm{E}_i}$ denote the real line segment joining $\mathrm{S}_i$ and $\mathrm{E}_i$. Then, it can be shown that $\mathrm{P}_{ij} := (x_p, y_p)$ is an error point corresponding to the line $\overline{\mathrm{S}_i\mathrm{E}_i}$ if and only if

$$2|\mathbf{x}_{\mathrm{ES}}\mathbf{y}_{\mathrm{EP}} - \mathbf{x}_{\mathrm{EP}}\mathbf{y}_{\mathrm{ES}}| - \max\{|\mathbf{x}_{\mathrm{ES}}|, |\mathbf{y}_{\mathrm{ES}}|\} > 0, \qquad (6)$$

where $\mathbf{x}_{\mathrm{ES}} = x_e - x_s$, $\mathbf{y}_{\mathrm{ES}} = y_e - y_s$, and so forth. Although (6) is not required at any stage in our algorithm, it enables us to check whether or not $\mathrm{P}_{ij}$ is an error point without using any floating-point arithmetic.

## 3 Polygonal Approximation

Extraction of the ADSS for each curve $\mathcal{C}_k$ in the given set (binary image) $\mathcal{I} := \{\mathcal{C}_k\}_{k=1}^{K}$ of DC generates an ordered set of ADSS, namely, $\mathcal{A}_k := \langle \mathbf{L}_i^{(k)} \rangle_{i=1}^{n_k}$, corresponding to $\mathcal{C}_k$. In each such set $\mathcal{A}_k$, several consecutive ADSS may occur, which are approximately collinear and, therefore, may be combined together to form a single segment.

Let $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$ be the maximal (ordered) subset of the ADSS starting from $\mathbf{L}_{j_1}^{(k)}$ that conforms to some approximation criterion. Then, these $j_2 - j_1 + 1$ segments in $A_k$ are combined together to form a single straight line segment starting from the start point of $\mathbf{L}_{j_1}^{(k)}$ and ending at the endpoint of $\mathbf{L}_{j_2}^{(k)}$. This procedure is repeated for all such maximal subsets of $\mathcal{A}_k$ in succession to obtain the polygonal approximation (in case $\mathcal{C}_k$ is a closed curve) or polychain approximation (in case $\mathcal{C}_k$ is open), namely, $\mathcal{P}_k$, corresponding to $\mathcal{C}_k$.

In the proposed algorithm, depending on the approximation criterion, we have used a greedy method of approximating the concerned curve $\mathcal{C}_k$ starting from the very first ADSS in $\mathcal{A}_k$. The determination of a minimal set of DSS (and a minimal set $\mathcal{A}_k$ of ADSS, thereof) corresponding to a given curve $\mathcal{C}_k$ is known to be computationally intensive [24], [37]; thus, for real-time applications, a near-optimal but speedy solution is often preferred than the optimal one.

### 3.1 Approximation Criterion

There are several variants of approximation criteria available in the literature [39]. We have tested our algorithms with two variants of the approximation measures using area deviation by Wall and Danielsson [47] (Table 1), which are realizable in the purely integer domain, subject to few primitive operations only. The approximation criterion is defined with respect to the *approximation parameter* or *error tolerance*, denoted by $\tau$, as follows.

#### *3.1.1 Cumulative Error (Criterion $\mathrm{C}_\Sigma$)*

Let $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$ be an ordered subset of $A_k$ as discussed above. Then, the ADSS ($j_2 - j_1 + 1$ in number) in $A_k$ are replaced by a single straight line segment starting from the start point of $\mathbf{L}_{j_1}^{(k)}$ and finishing at the endpoint of $\mathbf{L}_{j_2}^{(k)}$ if

$$\sum_{j=j_1}^{j_2-1} \left| \triangle\left(s\left(\mathbf{L}_{j_1}^{(k)}\right), e\left(\mathbf{L}_{j}^{(k)}\right), e\left(\mathbf{L}_{j_2}^{(k)}\right)\right) \right| \leq \tau d_\top\left(s\left(\mathbf{L}_{j_1}^{(k)}\right), e\left(\mathbf{L}_{j_2}^{(k)}\right)\right), \qquad (7)$$

where $s(\mathbf{L}_j^{(k)})$ and $e(\mathbf{L}_j^{(k)})$ represent the respective start point and the end point of the ADSS $\mathbf{L}_j^{(k)}$, and so forth The start point of $\mathbf{L}_j^{(k)}$ coincides with the end point of the preceding ADSS, if any, in $\mathcal{A}^k$, and the end point of $\mathbf{L}_j^{(k)}$ coincides with the succeeding one, if any. In (7), $|\triangle(p,q,r)|$ denotes twice the magnitude of the area of the triangle with vertices $p := (x_p, y_p)$, $q := (x_q, y_q)$, and $r := (x_r, y_r)$, and $d_\top(p,q)$ is the maximum isothetic distance between two points $p$ and $q$. Since all these points are in 2D digital space, the above-mentioned measures are computable in the integer domain, as shown in the following equations:

$$d_\top(p,q) = \max\{|x_p - x_q|, |y_p - y_q|\}, \qquad (8)$$

$$\triangle(p,q,r) = \begin{vmatrix} 1 & 1 & 1 \\ x_p & x_q & x_r \\ y_p & y_q & y_r \end{vmatrix}. \qquad (9)$$

From (9), it is evident that $\triangle(p,q,r)$ is a determinant that gives twice the signed area of the triangle with vertices $p$, $q$, and $r$. Hence, the ADSS in the given subset are merged to form a single straight line segment, say, $\widetilde{\mathbf{L}}$, provided that the cumulative area of the triangles ($j_2 - j_1$ in number), having $\widetilde{\mathbf{L}}$ as base and the third vertices being the endpoints of the ADSS (excepting the last one) in the subset $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$, does not exceed the area of the triangle with base $\widetilde{\mathbf{L}}$ (isothetic length) and height $\tau$.

#### *3.1.2 Maximum Error (Criterion $\mathrm{C}_{\max}$)*

With similar notations as mentioned above, using the maximum error criterion, the ADSS in $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$ would be replaced by a single piece, provided that the following condition is satisfied:

$$\max_{j_1 \le j \le j_2 - 1} \left| \triangle\left( s\left(\mathbf{L}^{(k)}_{j_1}\right), e\left(\mathbf{L}^{(k)}_{j}\right), e\left(\mathbf{L}^{(k)}_{j_2}\right) \right) \right| \le \tau d_\top \left( s\left(\mathbf{L}^{(k)}_{j_1}\right), e\left(\mathbf{L}^{(k)}_{j_2}\right) \right). \tag{10}$$

The rationale of considering two such criteria is explained as follows: Since we would be replacing a number of ADSS, which are almost straight and, more importantly, are not ordinary DC of arbitrary patterns and arbitrary curvatures, the end point of each ADSS makes a triangle with the replacing segment, namely, $\widetilde{\mathbf{L}}$; therefore, the sum of the areas of triangles formed by the endpoints of these ADSS in combination with the replacing line $\widetilde{\mathbf{L}}$ gives a measure of error due to the approximation of *all ADSS in* $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$ by $\widetilde{\mathbf{L}}$. Alternatively, if we are guided by the worst case approximation, that is, if the mostly digressing ADSS is considered to estimate the error, then the maximum of the areas of these triangles should be considered as the error measure for approximation of the *worst ADSS in* $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$ by $\widetilde{\mathbf{L}}$.

Empirical observations, as reported in Section 4, reveal that the above-mentioned two criteria are essentially similar in the sense that they produce almost identical polygons for different DC for different values of the error tolerance (that is, $\tau$). This is quite expected as far as the output is concerned.

As mentioned earlier in Section 3, to construct polygonal approximation, we consider the start point of the first ADSS (that is, $\mathbf{L}^{(k)}_{j_1}$) and the end point of the last ADSS (that is, $\mathbf{L}^{(k)}_{j_2}$). This can be justified as follows:

**Fact 1.** The sum (for criterion $\mathrm{C}_\Sigma$) or the maximum (for criterion $\mathrm{C}_{\max}$) of the isothetic distances of the endpoints of each ADSS from the replacing line $\widetilde{\mathbf{L}}$ never exceeds the specified error tolerance $\tau$. This follows easily on expansion of the left-hand side of the corresponding equations ((7) and (10)) and from the fact that the term $d_\top(s(\mathbf{L}^{(k)}_{j_1}), e(\mathbf{L}^{(k)}_{j_2}))$ represents the isothetic length of $\widetilde{\mathbf{L}}$.

**Fact 2.** Since each ADSS $\mathbf{L}^{(k)}_j$ is approximately a DSS, we consider that $\nexists p \in \mathbf{L}^{(k)}_j$ such that the isothetic distance of $p$ from the DSL passing through the endpoints of $\mathbf{L}^{(k)}_j$ exceeds unity (as testified in our experiments). Although, for sufficiently long ADSS, this may not hold for the underlying Conditions (c1) and (c2), as stated in Section 2, however, in our experiments with real-world images, this was found to hold. In case of any violation, some heuristics may be employed to find the error points and to find smaller ADSS to resolve the problem.

## 3.2 Algorithm for Polygonal Approximation

The algorithm for polygonal approximation of a sequence of ADSS in the set $\mathcal{A}$, using the approximation criterion of (7), is described in Fig. 8. To take care of the criterion $\mathrm{C}_{\max}$ of (10), a similar procedure may be written.

*Final Time Complexity.* As explained in Section 2.2, the time complexity for extracting the ADSS in a set of DC $\mathcal{I} := \{\mathcal{C}_k\}_{k=1}^{K}$ is given by $\Theta(N_1) + \Theta(N_2) + \ldots + \Theta(N_K) = \Theta(N)$, where $N(= N_1 + N_2 + \ldots + N_K)$ is the total number of points representing $\mathcal{I}$. Now, in the algorithm MERGE-ADSS, we have considered only the ordered set of vertices of the ADSS corresponding to the curves, so that the worst-case time complexity in this stage is linear in $N$. Hence, the overall time complexity is given by $\Theta(N) + \mathcal{O}(N) = \Theta(N)$, whatsoever may be the error of approximation $\tau$.

## 3.3 Quality of Approximation

The goodness of an algorithm for polygonal approximation is quantified, in general, by the amount of discrepancy

```
Algorithm MERGE-ADSS($\mathcal{A}, n, \tau$)
Steps:
1.  for $m \leftarrow 1$ to $n$
2.      for $S \leftarrow 0, i \leftarrow 1$ to $(n - m - 1)$
3.          $S \leftarrow S + \triangle(\mathcal{A}[m], \mathcal{A}[m+i], \mathcal{A}[m+i+1])$
4.          $dx \leftarrow |\mathcal{A}[m].x - \mathcal{A}[m+i+1].x|$
5.          $dy \leftarrow |\mathcal{A}[m].y - \mathcal{A}[m+i+1].y|$
6.          $d \leftarrow \max(dx, dy)$
7.          if $S \le d\tau$
8.              delete $\mathcal{A}[m+i]$ from $\mathcal{A}$
9.          else
10.             break
11.     $m \leftarrow m + i - 1$
```

Fig. 8. Algorithm MERGE-ADSS for polygonal approximation of a sequence of ADSS in $\mathcal{A}$ using criterion $\mathrm{C}_{\max}$.

between the approximate polygon(s) (or polychain(s)) and the original set of DC. There are several measures to assess the approximation of a curve $\mathcal{C}_k$, such as

1. compression ratio $\mathrm{CR} = N_k / M_k$, where $N_k$ is the number of points in $\mathcal{C}_k$, and $M_k$ is the number of vertices in the approximate polygon $\mathcal{P}_k$, and
2. the integral square error (ISE) between $\mathcal{C}_k$ and $\mathcal{P}_k$.

Since there is always a trade-off between CR and ISE, other measures may also be used [21], [38], [40]. These measures, however, may not always be suitable for some intricate approximation criterion. For example, the figure of merit [40], given by $\mathrm{FOM} = \mathrm{CR}/\mathrm{ISE}$, may not be suitable for comparing approximations for some common cases, as shown in [39]. In [45], the percentage of relative difference, given by $((E_{approx} - E_{opt})/E_{opt}) \times 100$, has been used, where $E_{approx}$ is the error incurred by a suboptimal algorithm under consideration, and $E_{opt}$ is the error incurred by the optimal algorithm under the assumption that the same number of vertices are produced by both algorithms. Similarly, one may use two components, namely, fidelity and efficiency, given by $(E_{opt}/E_{approx}) \times 100$ and $(M_{opt}/M_{approx}) \times 100$, respectively, where $M_{approx}$ is the number of vertices in the approximating polygon produced by the suboptimal algorithm, and $M_{opt}$ is the same as that produced by the optimal algorithm subject to the same $E_{approx}$ as the suboptimal one [39].

The algorithm proposed here is not constrained by the number of vertices $M_k$ of the output polygon $\mathcal{P}_k$ and, therefore, the measures of approximation where $M_k$ acts as an invariant are not applicable. Instead, we have considered the error of approximation, namely, $\tau$, as the sole parameter in our algorithm, depending on the number of vertices $M_k$ corresponding to $\mathcal{P}_k$ that will change. A high value of $\tau$ indicates a loose or slacked approximation; hence, the number of vertices $M_k$ decreases automatically, whereas a low value of $\tau$ implies a tight approximation, thereby increasing the number of vertices in the approximate polygon. Hence, in accordance with the usage of $\tau$ in both of our proposed methods, one based on criterion $\mathrm{C}_\Sigma$ and the other on $\mathrm{C}_{\max}$, the total number of vertices $M := M_1 + M_2 + \ldots + M_K$ in the set of approximate polygons $\{\mathcal{P}_k\}_{k=1}^{K}$ corresponding to the input set of DC, namely, $\mathcal{I} := \{\mathcal{C}_k\}_{k=1}^{K}$, versus $\tau$, provides the necessary quality of approximation. Since the total number of points lying on all the points in $\mathcal{I}$ characterizes (to some extent) the

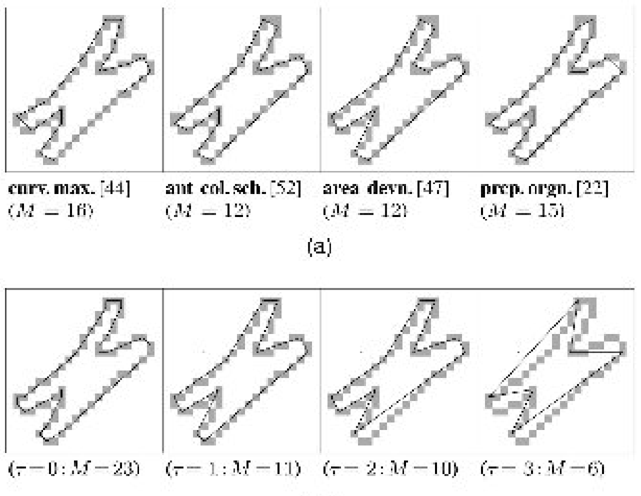


Fig. 9. Results on "chromosome." (a) By some existing methods (Table 1). (b) By the proposed method.

complexity of $\mathcal{I}$, we consider the CR as a possible measure of approximation.

Another measure of approximation is given by how much a particular point $(x, y) \in \mathcal{C}_k \in \mathcal{I}$ has deviated in the corresponding polygon $\mathcal{P}_k$. If $\widetilde{p} := (\widetilde{x}, \widetilde{y})$ is the point in $\mathcal{P}_k$ corresponding to $p := (x, y)$ in $\mathcal{I}$, then, for all points in $\mathcal{I}$, this measure is captured by the variation of the number of points with isothetic deviation $d_\perp$ with respect to $d_\perp$, where the *(isothetic) deviation from* $p$ *to* $\widetilde{p}$ is given by

$$dev_\perp(p \to \widetilde{p}) = \min\{|x - \widetilde{x}|, |y - \widetilde{y}|\}. \tag{11}$$

Further, since $dev_\perp(p \to \widetilde{p})$ depends on the chosen value of $\tau$ in our algorithm, the fraction of the number of points in $\mathcal{I}$ with deviation $d_\perp$ varies plausibly with $\tau$. Therefore, the *isothetic error frequency* (IEF) (or, simply, *error frequency*), given by

$$f(\tau, d_\perp) = \frac{1}{N} |\{p \in \mathcal{I} : dev_\perp(p \to \widetilde{p}) = d_\perp\}| \tag{12}$$

versus $\tau$ and $d_\perp$, acts as the second measure that provides the error distribution for the polygonal approximation of $\mathcal{I}$.

## 4 Implementation, Experiments, and Results

We have implemented one algorithm for DSS extraction based on DSS recognition [11] and the proposed algorithm EXTRACT-ADSS in $\mathcal{C}$ on the Sun OS Release 5.7 Generic of Sun_Ultra 5_10, Sparc, 233 MHz. The results of the two algorithms on several binary image files of curves and contours are reported here.

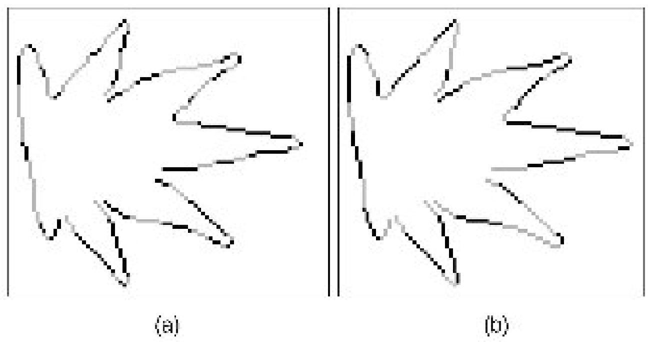


Fig. 11. Set of DSS and that of ADSS extracted from the image of a leaf. (a) DSS. (b) ADSS.

We have also compared the results of our method with those of some existing methods as shown in Table 1. In Fig. 9 and in Fig. 10, we have presented the comparative results on polygonal approximation of two benchmark curves, namely, "chromosome" (given in Appendix B of [44]) and "semicir," respectively. The results show that our method compares favorably with others when we consider $\tau = 1$ in MERGE-ADSS.

It may be noted that, for $\tau = 2, 3, \ldots$, the approximation obtained by our method requires a smaller number of vertices ($M$) but at the cost of some error incurred and may not be profitable for small curves like "chromosome" (and the other test/benchmark curves considered in [39], [44], [47], [48], [52]). However, for sufficiently large curves (a few being presented in Figs. 15 and 16), a slackening of $\tau$ reduces the number of vertices to the desired limit, as explained later in this section.

In our implementation, the procedure for extraction of the straight line segments (DSS/ADSS) from a given curve starts from one end of the curve if it is an open end or starts from a point with a chain code (with respect to its neighbors) not satisfying Property (R1) if it is a closed one; in case of failure of the above-mentioned two criteria, the start point is chosen arbitrarily.

The results for the algorithm on DSS extraction and that on ADSS extraction for an image of a leaf (cropped and zoomed for pixelwise clarity) are shown in Fig. 11. Earlier, in Fig. 2, similar results have been shown for another image (cropped). It is evident from the extracted set of DSS and that

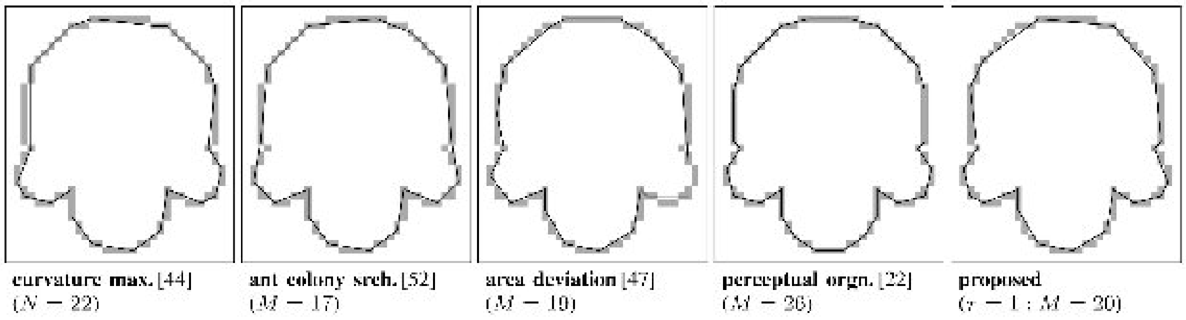


Fig. 10. Results on "semicir."

TABLE 2
Comparison of DSS and ADSS Extraction Algorithms on Different Images

| Image | Image size | No. of | P.A. | No. of segs. | | Average Length | | CPU time (secs.) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Name | row×col | Points | (secs.)$^{\#}$ | DSS | ADSS | DSS | ADSS | DSS | ADSS | $\{\mathcal{P}\}^*$ |
| bird-nestlings | 480×320 | 3041 | 6.38 | 902 | 327 | 3.37 | 9.30 | 5.42 | 0.17 | 0.05 |
| climber | 320×330 | 2750 | 5.92 | 1170 | 419 | 2.35 | 6.56 | 6.74 | 0.20 | 0.05 |
| leaf | 240×256 | 1312 | 3.88 | 341 | 106 | 3.85 | 12.38 | 2.17 | 0.08 | 0.02 |
| spider | 292×286 | 1767 | 4.20 | 583 | 157 | 3.03 | 11.25 | 3.93 | 0.11 | 0.03 |
| test-001 | 140×1050 | 2809 | 6.26 | 858 | 276 | 3.27 | 10.18 | 4.48 | 0.14 | 0.04 |
| vase | 408×333 | 6066 | 10.81 | 1972 | 681 | 3.07 | 8.91 | 14.52 | 0.43 | 0.10 |

$^{\#}$CPU time for polygonal approximation using area deviation [47].

$^*$average CPU time for $\tau = 1 - 20$ with criteria $C_{\sum}$ and $C_{\max}$.

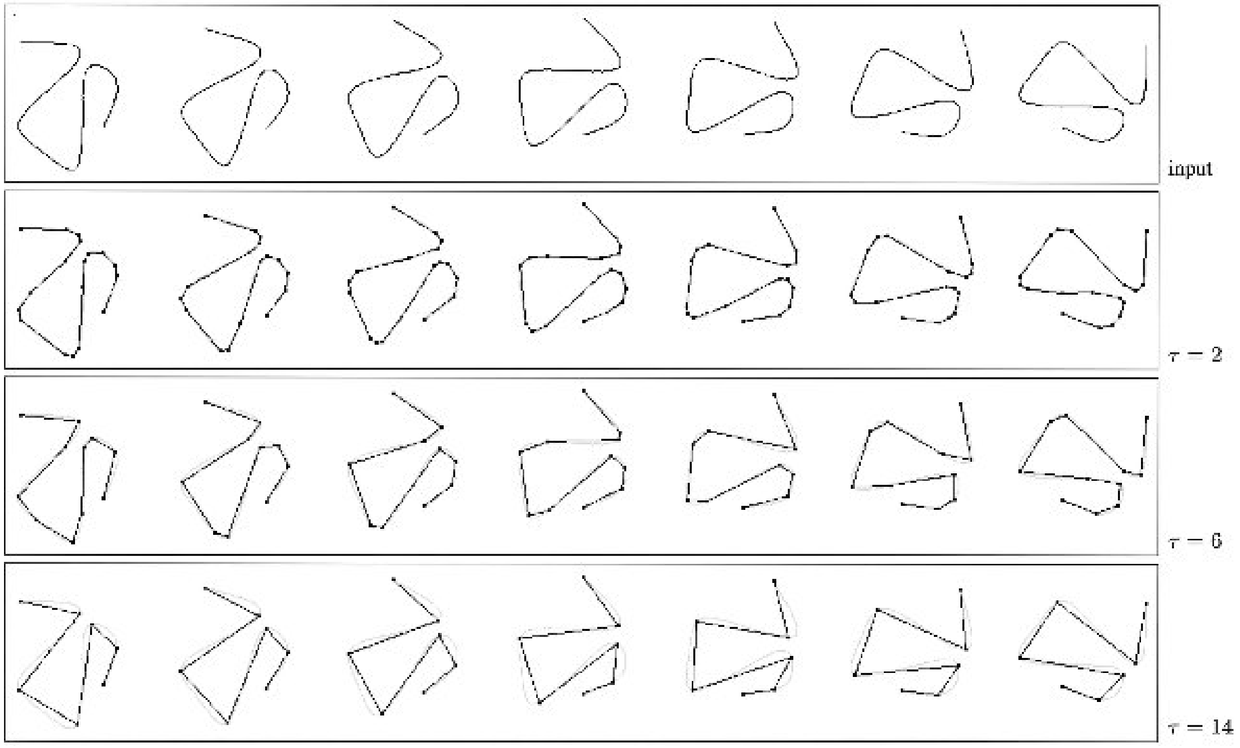


Fig. 12. Polygonal approximations for a set of small test images, namely, "test-001," using criterion $C_{\Sigma}$.

of ADSS that not only is an extracted ADSS reasonably straight but the number of ADSS is also appreciably smaller than that of DSS, a result that is used to expedite the subsequent algorithm for polygonal approximation. Because of the recursive nature (apropos the run of run lengths) of the DSS extraction algorithms, they are inherently much slower than the ADSS algorithm, as reflected by the CPU times reported in Table 2. In this table, we have also given the CPU time required for polygonal approximation by the algorithm based on area deviation [47] (by considering *all points* in the input set of curves instead of the endpoints of their ADSS) to show how ADSS extraction prior to polygonal approximation accelerates the process. We have also furnished some other significant parameters to justify the use of ADSS in our polygonal approximation algorithm. Since such an algorithm will run faster for a smaller set of segments (whether exact or approximate), the set of ADSS may be used instead of that of DSS.

We have tested our polygonal approximation algorithms on two classes of test images—one with 500 sets containing randomly generated DC of lengths varying from $N = 500$ to $N = 2,000$ and another class with 20 sets of hand-drawn curves. One such hand-drawn test set is shown in Fig. 12. In addition, results on 150 real-world binary images (edge maps) of various natures have been generated.

Fig. 12 demonstrates the polygonal approximation with the criterion $C_{\Sigma}$. The input image consists of a single (hand-drawn) curve with seven orientations at 15 degrees increments such as 0 degrees, 15 degrees, ..., 90 degrees. It is interesting to notice that, as the approximation

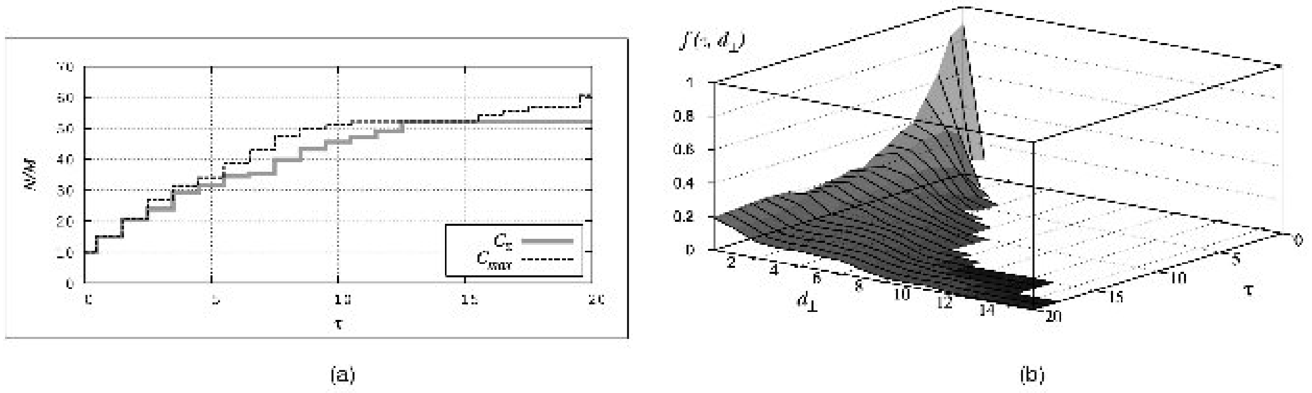


Fig. 13. Quality of approximation for the "test-001" image shown in Fig. 12. In (a), two histogram profiles on $N/M$ versus the error tolerance $\tau$ have been shown, where one histogram corresponds to criterion $\mathrm{C}_\Sigma$ and the other to criterion $\mathrm{C}_{\max}$. In (b), the plot of $\mathrm{IEF} := f(\tau, d_\perp)$ versus the error tolerance $\tau$ and the isothetic distance $d_\perp$ corresponding to criterion $\mathrm{C}_\Sigma$ has been shown, and that corresponding to criterion $\mathrm{C}_{\max}$ is very much similar. It may be noted that, here, $\tau = 0$ corresponds to the number of vertices without any polygonal approximation (that is, with ADSS only).

tolerance $\tau$ continues to increase, the sets of vertices tend to become increasingly similar, indicating the invariance of polygonal approximation to rotation (or orientation) when $\tau$ becomes sufficiently high. Further, due to the anisotropic nature of the square grid and the rounding-off error in digitization and thinning [19], [36] applied on a rotated curve, the difference chain code [36] of the rotated curve is likely to differ from that of the original curve. Hence, for smaller values of $\tau$, the difference in such chain codes representing two orientations plays a critical role in ascertaining approximate straightness of a small component (whose "smallness" is decided by $\tau$). A larger value of $\tau$, however, compensates for such local anomalies in a DC and thus produces almost identical polygons.

The two measures on quality of approximation for the "test-001" image have been rendered in Fig. 13. In Fig. 13a, the profiles of two histograms demonstrating the distribution of the $\mathrm{CR} = N/M$ with respect to the error tolerance $\tau$ have been shown corresponding to criteria $\mathrm{C}_\Sigma$ and $\mathrm{C}_{\max}$. The stability of the algorithm on polygonal approximation is evident from the nature of the plots shown for both the criteria. For a sufficiently high value of $\tau$, the CR becomes almost asymptotically constant, owing to the fact that nearly identical sets of vertices are produced for two close (and sufficiently high) values of $\tau$, as shown in Fig. 12.

In Fig. 13b, the plot on IEF using criterion $\mathrm{C}_\Sigma$ only has been shown, since that with criterion $\mathrm{C}_{\max}$ is very much similar. It may be observed from this plot that, for higher values of $\tau (\geq 6)$, the amount of maximum (isothetic) deviation (say, $d_{\perp(\max)}$) in all the seven curves in Fig. 12 falls quite short of the permissible limit, that is, $\tau$, and no less importantly, for a given value of $\tau$, the number of points (say, $N_{d_\perp}$) with deviation $d_\perp$ (and IEF, thereof) decreases almost monotonically with $d_\perp$. A small subset of the numerical figures in our experiment with the "test-001" curve is furnished below:

| $\tau$ | 6 | 7 | $\cdots$ | 16 | 18 | 20 |
|---|---|---|---|---|---|---|
| $d_{\perp(\max)}$ | 4 | 5 | $\cdots$ | 13 | 15 | 15 |
| $N_{d_{\perp(\max)}}$ | 35 | 15 | $\cdots$ | 2 | 9 | 13 |

The above-mentioned data clearly shows that, for the curves given in Fig. 12, for $\tau \geq 6$, we get $d_{\perp(\max)}$ appreciably smaller than $\tau$ and, more importantly, the IEF for $d_\perp = d_{\perp(\max)}$ is very small (for example, for $\tau = 20$, $\mathrm{IEF} = 13/2809 = 0.0046$, and so forth), which indicates that error values nearing $d_{\perp(\max)}$ have very low frequency, thereby reinforcing the expectation of high quality in the approximation process. This assertion is true as well for the other images, whether synthetic or real world.

Since most of the images in practical applications involve real-world images, we have presented here results of few such images considered in our experiment. Fig. 15 exhibits the approximate polygons for a "bird-nestlings" image for a few values of $\tau$ corresponding to each of the approximation criteria. It is apparent from this figure that the resultant sets of polygons for the two criteria differ marginally (the set corresponding to criterion $\mathrm{C}_\Sigma$ is marginally better than $\mathrm{C}_{\max}$) and that the number of vertices falls off drastically in the lower end of the spectrum (of $\tau$) and gradually steadies in the upper end of the spectrum. Further, our algorithm in the case of this real-world "bird-nestlings" image has very closely similar characteristics with the test image (Fig. 12) as evidenced by the resemblance of its quality measures given in Fig. 14 with those in Fig. 13, certifying the efficiency and robustness of our algorithm, irrespective of the nature and complexity of the curves.

In Fig. 16, approximate polygons for few other images with $\tau = 4$, corresponding to the approximation criterion $\mathrm{C}_\Sigma$, have been shown. The quality measures for these images are not, however, included in this paper for their almost similar patterns with those given in Figs. 13 and 14.

## 5 Conclusions and Future Work

It is evident from the discussions and the algorithms proposed here that a set of ADSS extracted from DC is significantly smaller in size than that of DSS extracted from the same, although each ADSS can be treated as sufficiently straight for various practical applications. Furthermore, the CPU time needed for ADSS extraction is remarkably lesser than that for DSS extraction.

Regarding polygonal approximation, the proposed method has been found to work well to determine a suboptimal solution from an arbitrary set of DC. It is evident from the experimental result and analysis that the polygon vertices are densely located in and around the regions with high

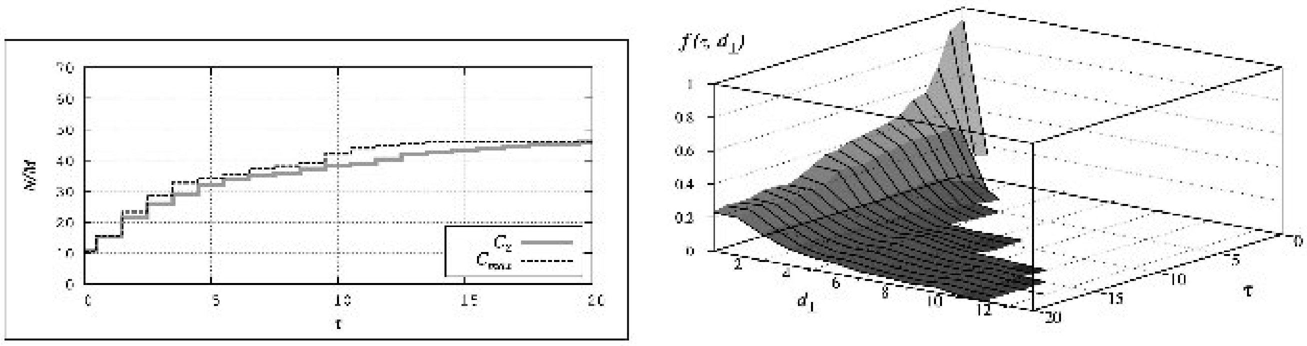


Fig. 14. Quality of approximation for the "bird-nestlings" image shown in Fig. 15. See text and Fig. 13 for the significance of the plots.

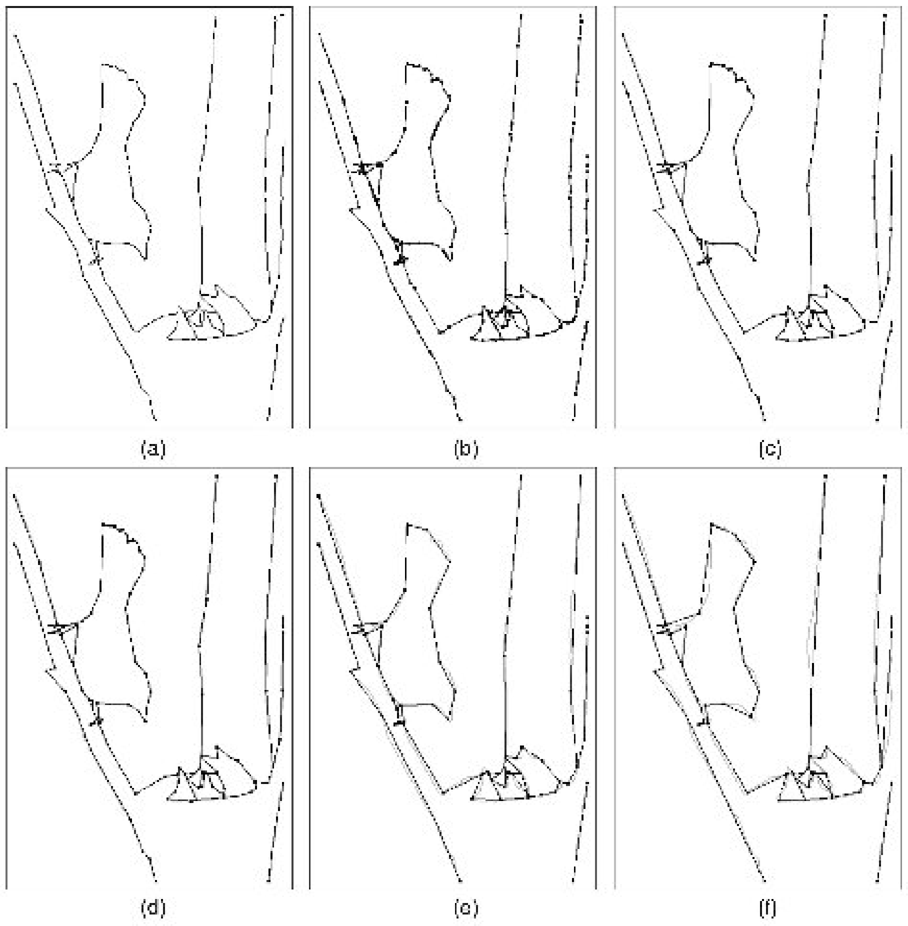


Fig. 15. Results on "bird-nestlings" image including extraction of ADSS. (a) **input** set of DC. (b) ADSS. (c) $C_\Sigma : \tau = 2$. (d) $C_{\max} : \tau = 2$. (e) $C_\Sigma : \tau = 8$. (f) $C_{\max} : \tau = 8$.

curvature and sparsely in the regions with low curvature, owing to the fact that the length of an ADSS (alternatively, a DSS) is small in the former region but high in the latter.

The major contributions of the proposed work are summarized as follows.

**Approximate straightness**. This is the major strength of the algorithm and marks its difference from the existing algorithms. The proposed algorithm utilizes the basic concepts of digital geometry and outputs the polygon efficiently and successfully using low-level operations.

**Primitive integer operations**. As no floating-point operations are needed, the algorithms run very fast. Herein lies one of the major differences from the existing approaches.

**Convergence property**. The algorithm converges to a quasi-static set of polygons for sufficiently high values of $\tau$, as evident from the analysis and observations.

**Robustness**. The proposed algorithms are robust because of the fact that the basic properties of approximate straightness are inherited from those of the exact digital straightness.

**Minor dependence on optimality criterion**. As the set of ADSS is generated first, the final stage of merging the collinear ADSS to a single edge of the approximate polygon becomes relatively simple when either of the two approximation criteria proposed here is used.

In the future, optimizing the set of ADSS to cover a given DC would be a promising area of research, since the output

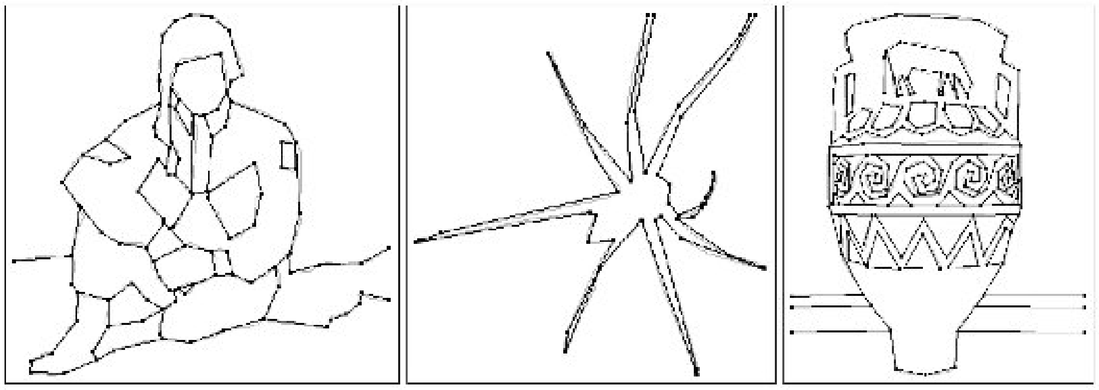


Fig. 16. Approximate polygons for few more images with $\tau = 4$ and criterion $C_\Sigma$.

set of the algorithm EXTRACT-ADSS depends on the starting point and the direction of the traversal. A theoretical analysis of the error distribution in an ADSS extraction algorithm, as well as investigation of other underlying properties of ADSS, is another field for conducting further research.

## Acknowledgments

The authors would like to acknowledge the anonymous reviewers for their valuable suggestions that helped improve this paper.

## References

[1] J.R.V. Aken and M. Novak, "Curve-Drawing Algorithms for Raster Display," *ACM Trans. Graphics*, vol. 4, no. 2, pp. 147-169, 1985.

[2] I.M. Anderson and J.C. Bezdek, "Curvature and Tangential Deflection of Discrete Arcs: A Theory Based on the Commutator of Scatter Matrix Pairs and Its Application to Vertex Detection in Planar Shape Data," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, pp. 27-40, 1984.

[3] F. Attneave, "Some Informational Aspects of Visual Perception," *Psychological Rev.*, vol. 61, no. 3, pp. 183-193, 1954.

[4] J.C. Bezdek and I.M. Anderson, "An Application of the $c$-Varieties Clustering Algorithms to Polygonal Curve Fitting," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 15, pp. 637-641, 1985.

[5] P. Bhowmick, A. Biswas, and B.B. Bhattacharya, "Isothetic Polygons of a 2D Object on Generalized Grid," *Proc. First Int'l Conf. Pattern Recognition and Machine Intelligence*, pp. 407-412, 2005.

[6] A. Biswas, P. Bhowmick, and B.B. Bhattacharya, "TIPS: On Finding a Tight Isothetic Polygonal Shape Covering a 2D Object," *Proc. 14th Scandinavian Conf. Image Analysis*, pp. 930-939, 2005.

[7] J. Bresenham, "An Incremental Algorithm for Digital Plotting," *Proc. ACM Nat'l Conf.*, 1963.

[8] T.C. Chen and K.L. Chung, "A New Randomized Algorithm for Detecting Lines," *Real Time Imaging*, vol. 7, pp. 473-481, 2001.

[9] S. Climer and S.K. Bhatia, "Local Lines: A Linear Time Line Detector," *Pattern Recognition Letters*, vol. 24, pp. 2291-2300, 2003.

[10] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*. Prentice Hall of India, 2000.

[11] E. Creutzburg, A. Hübler, and V. Wedler, "On-Line Recognition of Digital Straight Line Segments," *Proc. Second Int'l Conf. Artificial Intelligence and Information Control Systems of Robots*, pp. 42-46, 1982.

[12] L.S. Davis, A. Rosenfeld, and A.K. Agrawala, "On Models for Line Detection," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 6, pp. 127-133, 1976.

[13] I. Debled-Rennesson and J.P. Reveilles, "A Linear Algorithm for Segmentation of Digital Curves," *Int'l J. Pattern Recognition and Artificial Intelligence*, vol. 9, pp. 635-662, 1995.

[14] J.G. Dunham, "Optimum Uniform Piecewise Linear Approximation of Planar Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, pp. 67-75, 1986.

[15] M.A. Fischler and H.C. Wolf, "Locating Perceptually Salient Points on Planar Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, pp. 113-129, Feb. 1994.

[16] H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," *IRE Trans. Electronic Computers*, vol. 10, pp. 260-268, 1961.

[17] H. Freeman, "Techniques for the Digital Computer Analysis of Chain-Encoded Arbitrary Plane Curves," *Proc. Nat'l Electronics Conf.*, vol. 17, pp. 421-432, 1961.

[18] H. Freeman and L.S. Davis, "A Corner Finding Algorithm for Chain-Coded Curves," *IEEE Trans. Computers*, vol. 26, pp. 297-303, 1977.

[19] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*. Addison-Wesley, 1993.

[20] D.S. Guru, B.H. Shekar, and P. Nagabhushan, "A Simple and Robust Line Detection Algorithm Based on Small Eigenvalue Analysis," *Pattern Recognition Letters*, vol. 25, pp. 1-13, 2004.

[21] A. Held, K. Abe, and C. Arcelli, "Towards a Hierarchical Contour Description via Dominant Point Detection," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 24, pp. 942-949, 1994.

[22] J. Hu and H. Yan, "Polygonal Approximation of Digital Curves Based on the Principles of Perceptual Organization," *Pattern Recognition*, vol. 30, no. 5, pp. 701-718, 1997.

[23] H. Imai and M. Iri, "Computational Geometric Methods for Polygonal Approximations of a Curve," *Computer Vision, Graphics, and Image Processing*, vol. 36, pp. 31-41, 1986.

[24] R. Klette and A. Rosenfeld, *Digital Geometry: Geometric Methods for Digital Image Analysis*. Morgan Kaufmann, 2004.

[25] R. Klette and A. Rosenfeld, "Digital Straightness: A Review," *Discrete Applied Math.*, vol. 139, nos. 1-3, pp. 197-230, 2004.

[26] J. Koplowitz, M. Lindenbaum, and A. Bruckstein, "The Number of Digital Straight Lines on an $n \times n$ Grid," *IEEE Trans. Information Theory*, vol. 36, pp. 192-197, 1990.

[27] V.A. Kovalevsky, "New Definition and Fast Recognition of Digital Straight Segments and Arcs," *Proc. Int'l Conf. Pattern Recognition*, pp. 31-34, 1990.

[28] J.A.V. Mieghem, H.I. Avi-Itzhak, and R.D. Melen, "Straight Line Extraction Using Iterative Total Least Squares Methods," *J. Visual Comm. Image Representation*, vol. 6, pp. 59-68, 1995.

[29] F. Mignosi, "On the Number of Factors of Sturmian Words," *Theoretical Computer Science*, vol. 82, no. 1, pp. 71-84, 1991.

[30] T. Pavlidis, *Structural Pattern Recognition*. Springer, 1977.

[31] T. Pavlidis, "Algorithms for Shape Analysis and Waveforms," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 2, pp. 301-312, 1980.

[32] J.C. Perez and E. Vidal, "Optimum Polygonal Approximation of Digitized Curves," *Pattern Recognition Letters*, vol. 15, pp. 743-750, 1994.

[33] I. Povazan and L. Uher, "The Structure of Digital Straight Line Segments and Euclid's Algorithm," *Proc. Spring Conf. Computer Graphics*, pp. 205-209, 1998.

[34] J. Rocha and R. Bernardino, "Singularities and Regularities on Line Pictures via Symmetrical Trapezoids," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 4, pp. 391-395, Apr. 1998.

[35] A. Rosenfeld, "Digital Straight Line Segments," *IEEE Trans. Computers*, vol. 23, no. 12, pp. 1264-1268, Dec. 1974.

[36] A. Rosenfeld and A.C. Kak, *Digital Picture Processing*, second ed. Academic Press, 1982.

[37] A. Rosenfeld and R. Klette, "Digital Straightness," *Electronic Notes in Theoretical Computer Science*, vol. 46, 2001, http://www.elsevier.nl/locate/entcs/volume46.html.

[38] P. Rosin and G. West, "Non-Parametric Segmentation of Curves into Various Representations," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, pp. 1140-1153, 1995.

[39] P.L. Rosin, "Techniques for Assessing Polygonal Approximation of Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 659-666, June 1997.

[40] D. Sarkar, "A Simple Algorithm for Detection of Significant Vertices for Polygonal Approximation of Chain-Coded Curves," *Pattern Recognition Letters*, vol. 14, pp. 959-964, 1993.

[41] K. Schröder and P. Laurent, "Efficient Polygon Approximations for Shape Signatures," *Proc. IEEE Int'l Conf. Image Processing*, pp. 811-814, 1999.

[42] G.M. Schuster and A.K. Katsaggelos, "An Optimal Polygonal Boundary Encoding Scheme in the Rate Distortion Sense," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, pp. 13-26, 1998.

[43] A.W.M. Smeulders and L. Dorst, "Decomposition of Discrete Curves into Piecewise Segments in Linear Time," *Contemporary Math.*, vol. 119, pp. 169-195, 1991.

[44] C.-H. Teh and R.T. Chin, "On the Detection of Dominant Points on Digital Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 8, pp. 859-872, Aug. 1989.

[45] J. Ventura and J. Chen, "Segmentation of Two-Dimensional Curve Contours," *Pattern Recognition*, vol. 25, pp. 1129-1140, 1992.

[46] K. Voss, "Coding of Digital Straight Lines by Continued Fractions," *Computers and Artificial Intelligence*, vol. 10, pp. 75-80, 1991.

[47] K. Wall and P.-E. Danielsson, "A Fast Sequential Method for Polygonal Approximation of Digitized Curves," *Computer Vision, Graphics, and Image Processing*, vol. 28, pp. 220-227, 1984.

[48] L.-D. Wu, "A Piecewise Linear Approximation Based on a Statistical Model," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, pp. 41-45, 1984.

[49] D.M. Wuescher and K.L. Boyer, "Robust Contour Decomposition Using a Constant Curvature Criterion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 1, pp. 41-51, Jan. 1991.

[50] Y. Xie and Q. Ji, "Effective Line Detection with Error Propagation," *Proc. IEEE Int'l Conf. Image Processing*, pp. 181-184, 2001.

[51] P.-Y. Yin, "A New Method for Polygonal Approximation Using Genetic Algorithms," *Pattern Recognition Letters*, vol. 19, no. 11, pp. 1017-1026, 1998.

[52] P.Y. Yin, "Ant Colony Search Algorithms for Optimal Polygonal Approximation of Plane Curves," *Pattern Recognition*, vol. 36, pp. 1783-1797, 2003.

[53] P.Y. Yin, "A Discrete Particle Swarm Algorithm for Optimal Polygonal Approximation of Digital Curves," *J. Visual Comm. Image Representation*, vol. 15, no. 2, pp. 241-260, 2004.

**Partha Bhowmick** received the BTech degree from the Indian Institute of Technology, Kharagpur, India, and the MTech degree from the Indian Statistical Institute, Kolkata, India. Presently, he serves on the Faculty of Computer Science and Technology, Bengal Engineering and Science University (BESU), Howrah, India. His research interests include digital geometry, low-level image processing, approximate pattern matching, shape analysis, and biometrics. He has published several research papers in international journals and refereed conference proceedings and has filed four US patents.

**Bhargab B. Bhattacharya** received the BSc degree in physics from Presidency College, Kolkata, India. He also received the BTech and MTech degrees in radiophysics and electronics and the PhD degree in computer science all from the University of Calcutta, Kolkata. Since 1982, he has been on the faculty of the Indian Statistical Institute, Kolkata, where he is a full professor. He held visiting positions in the Department of Computer Science and Engineering, University of Nebraska-Lincoln, during 1985-1987 and 2001-2002, and in the Fault-Tolerant Computing Group, Institute of Informatics, the University of Potsdam, Germany, during 1998-2000. In 2005, he visited the Indian Institute of Technology, Kharagpur, as the Videsh Sanchar Nigam Limited (VSNL) chair professor. His research interests include logic synthesis and testing of VLSI circuits, physical design, digital geometry, and image processing architecture. He has published more than 180 papers in archival journals and refereed conference proceedings and holds nine US patents. He is a fellow of the Indian National Academy of Engineering, a fellow of the National Academy of Sciences, India, and a fellow of the IEEE and the IEEE Computer Society. He is on the editorial board of the *Journal of Circuits, Systems, and Computers* (World Scientific, Singapore) and the *Journal of Electronic Testing—Theory and Applications* (Springer).