J. Vis. Commun. Image R. 21 (2010) 295–310

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Construction of isothetic covers of a digital object: A combinatorial approach

Arindam Biswas [a,*], Partha Bhowmick [b], Bhargab B. Bhattacharya [c]

[a] Department of Information Technology, Bengal Engineering and Science University, Shibpur, India
[b] Computer Science and Engineering Department, Indian Institute of Technology, Kharagpur, India
[c] Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

ARTICLE INFO

Article history:
Received 14 January 2009
Accepted 20 January 2010
Available online 18 February 2010

Keywords:
Connected component
Digital geometry
Digital object
Isothetic cover
Isothetic polygon
Image processing
Pattern recognition
Shape analysis

ABSTRACT

An *isothetic cover* of a digital object not only specifies a simple representation of the object but also provides an approximate information about its structural content and geometric characteristics. When the cover "tightly" encloses the object, it is said to be an *outer isothetic cover*; and when the cover tightly inscribes the object, it is an *inner isothetic cover*. If a set of horizontal and vertical grid lines is imposed on the object plane, then the outer (inner) isothetic cover is defined by a set of isothetic polygons, having their edges lying on the grid lines, such that the effective area corresponding to the object is minimized (maximized). Increasing or decreasing the grid size, therefore, decreases or increases the complexity or the accuracy of the isothetic cover corresponding to a given object, which, in turn, extracts the object information at different resolutions. In this paper, a combinatorial algorithm is presented for varying grid sizes, which is free of any backtracking and can produce the isothetic cover of a connected component in a time linear in the length of the perimeter of the cover. The algorithm has also been extended for finding the isothetic covers of real-world digital objects having multiple components with or without holes. Experimental results that demonstrate applications of an isothetic cover to diverse problems of pattern analysis and computer vision, are reported for various data sets.

1. Introduction

Determination of the (minimum-)maximum-area (outer) inner isothetic cover corresponding to a 2D digital object is a problem of practical relevance to various fields. Given a set of isothetic grid lines under the object plane, an isothetic cover corresponds to a collection of isothetic polygons, which bears a structural and geometric relation with the concerned object, and hence can be useful to many interesting applications, such as VLSI layout design, robot grasping and navigation, rough sets, inner and outer approximations of polytopes [3], and document image analysis. In VLSI layout design, computation of a minimum-area safety region, referred as the classical *safety zone problem* [1], may be necessary to ensure the correctness of design rules before fabricating an integrated circuit. The trade-off lies between the minimization of total area of the fabricated parts (an obvious economic constraint) and the necessary electrical relationship (insulation or contact) in the presence of possible production fluctuations [2]. In robotics, identification of free configuration space (path-planner) is a pertinent problem in robot navigation. For example, a real-time robot motion planner often uses standard graphics hardware to rasterize the configuration space into a series of bitmap slices, and then applies a dynamic programming technique to calculate paths in this rasterized space [4]. The motion paths produced by the planner should be minimal with respect to the Manhattan distance (L_1) metric. Similarly, for grasping a 3D object, its outer isothetic cover may be helpful, as the mechanical fingers of a robot may be constrained by only axis-parallel movements [5–7]. In many applications of rough sets, computation of the lower and upper approximations of a rough set is required [8–11]. For example, in image mining [12,13], a challenging problem is to discover valid, novel, potentially useful, and ultimately understandable knowledge from a large image database, in order to overcome the curse of dimensionality. Solutions, using rough-set concepts, mainly include several partitioning and dimension-reduction algorithms, where the (possibly not equi-spaced) demarcating lines (analogous to the background grid lines used while finding isothetic covers) are specified by the corresponding feature values (low, medium, high, etc.). Subsequently, a tight isothetic cover of the region of interest can be obtained following these demarcating lines. Deriving the electronic version of a paper document for the purpose of storage, retrieval, and interpretation, requires an efficient representation scheme. A document representation involves the steps of skew detection, page segmentation, geometric layout analysis, and logical layout analysis, for which isothetic polygons can be used [14–17]. For example, in the page segmentation method [18], a document page image is cut into polygonal blocks using the inter-column and the inter-paragraph gaps as horizontal and vertical lines, followed by the construction of simple isothetic polygonal blocks using an

* Corresponding author.
E-mail addresses: abiswas@it.becs.ac.in, barindam@gmail.com (A. Biswas).

1047-3203/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2010.02.001

intersection table. Isothetic polygons can also be used for ground truthing of complex documents [19,20].

The problem addressed in this paper is stated as follows. Given a 2D digital object S (Definition 3) registered on a set of horizontal and vertical grid lines $\mathscr{G}$ (Definition 5), the problem is to find the tight (outer and inner) isothetic covers (Definitions 9 and 10) of S. Clearly, the shape of an isothetic cover depends on the resolution of the grid or the grid size (Definition 5) and on registration of the object with the underlying grid. The major difference of our work with the existing ones, therefore, lies in its ability to find an inner/outer isothetic cover that, for a lower grid size, almost grazes the contour of a digital object without touching it, and in its ability to reduce the output complexity (i.e., number of vertices) by producing a rough cover when a higher grid size is specified. Fig. 1 shows the original "bear" image (left), its outer isothetic cover (OIC) and the inner isothetic cover (IIC) in the middle, and the superimposed outer and inner isothetic covers (right). It is seen that the boundary of the object lies in the region of difference given by the OIC minus the interior of IIC.

2. Definitions and preliminaries

Definition 1 (*Digital Plane*). The digital plane, $\mathbb{Z}^2$, is the set of all points having integer coordinates in the real plane $\mathbb{R}^2$. A point in the digital plane is called a digital point, or called a pixel in the case of a digital image. Henceforth, the terms "point" and "digital point" will be used interchangeably.

Definition 2 (*k-connectedness*). The set of four horizontally and vertically adjacent points of a point $p(x,y) \in \mathbb{Z}^2$ is called the 4-neighborhood of p, which is given by $N_4(p) := \{(x',y') : (x',y') \in \mathbb{Z}^2 \wedge |x-x'| + |y-y'| = 1\}$. The set of all eight neighbors, i.e., the four horizontal and vertical neighbors, and the four diagonal neighbors, defines the 8-neighborhood of p, given by $N_8(p) := \{(x',y') : \ (x',y') \in \mathbb{Z}^2 \wedge \max(|x-x'|, |y-y'|) = 1\}$. Each point in $N_k(p)$ is said to be a k-neighbor ($k = 4$ or 8) of p. Two points p and q are k-connected in a digital set $S \subset \mathbb{Z}^2$ (Definition 3) if and only if there exists a sequence $\langle p := p_0, p_1, \ldots, p_n := q \rangle \subseteq S$ such that $p_i \in N_k(p_{i-1})$ for $1 \leqslant i \leqslant n$. For any point $p \in S$, the set of points that are k-connected to $p \in S$ is called a k-connected component of S. In other words, a k-connected component of a nonempty set $S \subseteq \mathbb{Z}^2$ is a maximal k-connected set of S. If S has only one connected component, it is called a k-connected set.

Definition 3 (*Digital object*). A digital object (henceforth referred as an object) is a finite subset of $\mathbb{Z}^2$, which consists of one or more k-connected components.

In this paper, each component (8-connected, i.e., $k = 8$) of an object S may contain one or more holes. A hole of S is a $\bar{k}$-connected component of $\mathbb{Z}^2 \setminus S$ which is completely surrounded by one of the components of S (for $k = 8$, we have $\bar{k} = 4$, and vice versa) [21].

Definition 4 (*Isothetic distance*). The (isothetic) distance between two points, $p(i_p, j_p)$ and $q(i_q, j_q)$, is the Minkowski norm L_∞ [22], which is given by $d_\top(p,q) = \max\{|i_p - i_q|, |j_p - j_q|\}$. The distance of a point p from an object S is $d_\top(p,S) = \min\{d_\top(p,q) : q \in S\}$, and the distance between two connected components S_1 and S_2 is $d_\top(S_1,S_2) = \min\{d_\top(p,q) : p \in S_1, q \in S_2\}$.

Definition 5 (*Digital grid*). A digital grid (henceforth referred simply as a grid) $\mathscr{G} := (\mathscr{H}, \mathscr{V})$ consists of a set $\mathscr{H}$ of horizontal (digital) grid lines and a set $\mathscr{V}$ of vertical (digital) grid lines, where, $\mathscr{H} = \{\ldots, l_H(j-2g), l_H(j-g), l_H(j), l_H(j+g), l_H(j+2g), \ldots\} \subset \mathbb{Z}^2$ and $\mathscr{V} = \{\ldots, l_V(i-2g), l_V(i-g), l_V(i), l_V(i+g), l_V(i+2g), \ldots\} \subset \mathbb{Z}^2$, for a grid size, $g \in \mathbb{Z}^+$. Here, $l_H(j) := \{(i',j) : i' \in \mathbb{Z}\}$ denotes the horizontal grid line and $l_V(i) := \{(i,j') : j' \in \mathbb{Z}\}$ denotes the vertical grid line intersecting at the point $(i,j) \in \mathbb{Z}^2$, called the grid point, where i and j are multiples of g.

Definition 6 (*Grid segment*). A (digital) grid segment (horizontal/vertical) is the set of points on a (horizontal/vertical) grid line between and inclusive of two consecutive grid points. Thus, the horizontal grid segment belonging to $l_H(j)$ and lying between two consecutive vertical grid lines, $l_V(i)$ and $l_V(i+g)$, is $s_j^i = \{(i',j) \in l_H(j) : i \leqslant i' \leqslant i+g\}$. Similarly, the vertical grid segment belonging to $l_V(i)$ between $l_H(j)$ and $l_H(j+g)$ is $s_i^j = \{(i,j') \in l_V(i) : j \leqslant j' \leqslant j+g\}$.

Definition 7 (*Unit grid block (UGB)*). The horizontal digital line $l_H(j)$ divides $\mathbb{Z}^2$ into two half-planes (each closed at one side by $l_H(j)$), which are given by $h_H^+(j) := \{(i',j') \in \mathbb{Z}^2 : j' \geqslant j\}$ and $h_H^-(j) := \{(i',j') \in \mathbb{Z}^2 : j' \leqslant j\}$. Similarly, $l_V(i)$ divides $\mathbb{Z}^2$ into two half-planes, which are $h_V^+(i) := \{(i',j') \in \mathbb{Z}^2 : i' \geqslant i\}$ and $h_V^-(i) := \{(i',j') \in \mathbb{Z}^2 : i' \leqslant i\}$. Then the set, given by $(h_V^+(i) \cap h_V^-(i+g)) \cap (h_H^+(j) \cap h_H^-(j+g))$, is defined as the unit grid block, UGB(i,j). The interior of $U_1 := \text{UGB}(i,j)$ is given by $U_1 \setminus (s_i^j \cup s_{i+g}^j \cup s_j^i \cup s_{j+g}^i)$.

Hence, for a given grid point, $q(i,j)$, there are four neighboring UGBs, namely, $U_1 := \text{UGB}(i,j)$, $U_2 := \text{UGB}(i-g,j)$, $U_3 := \text{UGB}(i-g,j-g)$, and $U_4 := \text{UGB}(i,j-g)$, as shown in Fig. 2. Two adjacent UGBs share a common (horizontal or vertical) grid segment. For example, the vertical grid segment shared between U_1 and U_2, is given by $s_i^j := U_1 \cap U_2$. Thus, U_1, U_2, U_3, and U_4 share a common point, namely the grid point $q(i,j)$.

Definition 8 (*Isothetic polygon*). An isothetic polygon P is a simple polygon (i.e., with nonintersecting sides) of finite size in $\mathbb{Z}^2$ whose alternate sides are subsets of the members of $\mathscr{H}$ and $\mathscr{V}$. The polygon P, hence given by a finite set of UGBs, is represented by the (ordered) sequence of its vertices, which are grid points. The border B_P of P is the set of points belonging to its sides. The interior of P is the set of points in the union of its constituting UGBs excluding the border of P.

An isothetic polygon P can be classified into:

- *Outer polygon:* $P \cap S \neq \emptyset$; for each $p \in B_P$, $0 < d_\top(p, P \cap S) \leqslant g$.
- *Inner polygon:* $P \cap S \neq \emptyset$; for each $p \in B_P$, $0 < d_\top(p, S' \setminus P) \leqslant g$.
- *Outer hole polygon:* for each $p \in B_P$, $0 < d_\top(p, S \setminus P) \leqslant g$.
- *Inner hole polygon:* for each $p \in B_P$, $0 < d_\top(p, P \cap S') \leqslant g$.

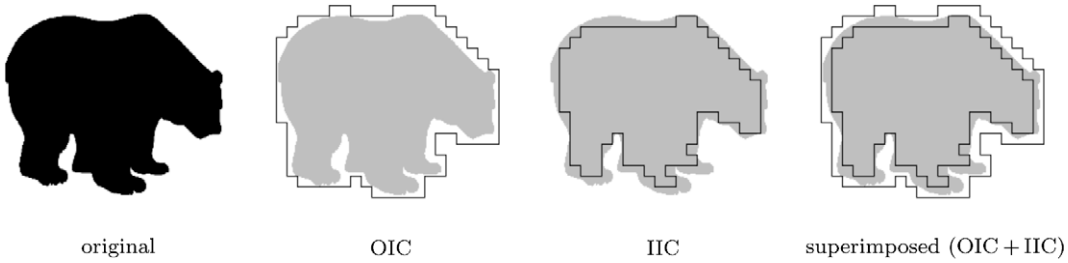
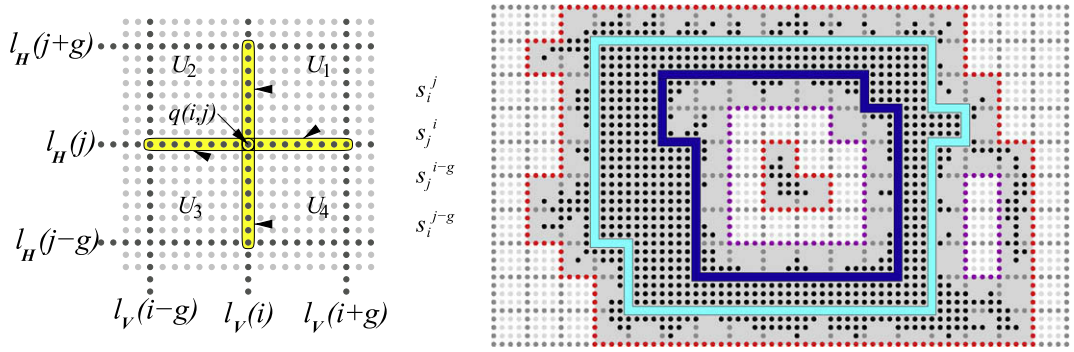


Fig. 1. The outer cover (OIC) and the inner cover (IIC), corresponding to the image "bear".



(a) UGBs incident at $q(i,j)$. (b) OIC and IIC.

Fig. 2. (a) Four neighboring UGBs, namely $U_1 = \mathrm{UGB}(i,j)$, $U_2 = \mathrm{UGB}(i-g,j)$, $U_3 = \mathrm{UGB}(i-g,j-g)$, and $U_4 = \mathrm{UGB}(i,j-g)$, incident at a grid point $q(i,j)$. (b) OIC (in red and magenta) and IIC (in cyan and blue) corresponding to an object S (in black) having two connected components. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

Roughly speaking, an outer polygon contains one or more components of S such that its border is a subset of $S' := \mathscr{Z}^2 \setminus S$ and the number of its constituting UGBs is minimum. Similarly, an inner polygon is contained in exactly one component of S such that its border is a subset of S and the number of its constituting UGBs is maximum. An outer hole polygon lies in a hole (or a sufficiently large background region with a narrow opening in S) and its border is a subset of S', whereas, an inner hole polygon contains a hole and its border is a subset of S. In Fig. 2(a), the outer hole polygon is shown in magenta and the inner hole polygon in blue.

Definition 9 (*Outer isothetic cover*). The outer (isothetic) cover (OIC), denoted by $\overline{P}(S)$, is a set of outer polygons and (outer) hole polygons, such that the region, given by the union of the outer polygons minus the union of the interiors of the hole polygons, contains a UGB if and only if it has object occupancy (i.e., has a nonempty intersection with S).

Definition 10 (*Inner isothetic cover*). The inner (isothetic) cover (IIC), denoted by $\underline{P}(S)$, is a set of inner polygons and (inner) hole polygons, such that the region, given by the union of the inner polygons minus the union the interiors of of the hole polygons, contains a UGB if and only if it is a subset of S.

Hence, the OIC of an object consists of minimum number of UGBs and its IIC consists of maximum number of UGBs. As a consequence, given an object S and a grid imposed on S, the OIC and the IIC are, respectively, of minimum and of maximum areas measured in the grid unit.

The *interior* of an OIC is given by the union of the interiors of its outer polygons minus the union of its hole polygons. A point, therefore, is said to *lie inside the OIC* if only if it is an interior point; it is said to *lie on the OIC* if and only if it lies on the border of one of its outer polygons or hole polygons. The OIC and the IIC of an object S having two components are shown in Fig. 2. The OIC (shown in red) consists of two outer polygons and two hole polygons, where one hole polygon corresponds to an actual hole of the object and the other to a sufficiently large background region with a narrow opening in S. The IIC polygons (in cyan) are two in number, the larger one being the inner polygon and the smaller one being the hole polygon.

3. Related works

As mentioned in Section 1, the existing works are mostly related with border tracking or edge detection of a digital object [21–25]; whereas, the proposed method finds an isothetic cover whose precision can be varied by specifying the grid size. One of the existing algorithms is the crack following algorithm by Rosenfeld [21], which finds the border B of a digital object, S, using the fact that B consists of the points of S that are 4-adjacent to its complement, S'. Depending on how the current pair of 4-adjacent points, (p,p'), where $p \in B$ and $p' \in S'$, is oriented w.r.t. the crack, the next point of B is determined. Fig. 3 shows a digital object S, its border B traced by the crack following algorithm, and the OICs constructed by the proposed algorithm. The red points/pixels (dark and light red points taken together) denote the border B. The dark red points have been traversed twice while following the crack (shown in dotted line); in fact, there are two polygons that are connected by a single-pixel thick curve. The proposed algorithm, on the other hand, can extract the (border-like, for $g = 1$) OIC without retracing the path and can also find a coarser OIC with a larger grid size, as shown in Fig. 3(d) for $g = 2$, which has a smaller number of vertices compared to the OIC for $g = 1$ (Fig. 3(c)).

Isothetic covers of a digital object S are also related with the Jordan digitization of S [22]. The outer Jordan digitization $J^+(S)$ is the union of all 2-cells (grid squares) that have nonempty intersections with S, and the inner Jordan digitization $J^-(S)$ is the union of all 2-cells that are completely contained in S. However, Jordan digitization (inner/outer) does not specify the (inner/outer) cover as a sequence of vertices, which is a characteristic of our algorithm.

Klette and Rosenfeld demonstrated a similar concept called isothetic *frontier* [22]. However, there exist some differences between an isothetic cover and an isothetic frontier. To find the frontier ϑS of an object S in any metric space $[A,d]$, the (open) *ε-neighborhood* of a point p in A is defined as $U_\varepsilon(p) = \{q : 0 \leqslant d(p,q) < \varepsilon\}$. Then, $p \in S$ is a *frontier point* of $S \subseteq A$ if and only if, for any $\varepsilon > 0$, $U_\varepsilon(p)$ contains points of S as well as points of $A \setminus S$. In the digital plane, ε is always a positive integer and $A \subseteq \mathbb{Z}^2$. Hence, if $\varepsilon = 1$, then there exists no point $p \in A$ for which $U_\varepsilon(p)$ contains a point of $A \setminus S$. In the case of an isothetic cover, the metric used is isothetic distance $d_\top$ (Definition 4), and if a point p lies on the cover, then there exist point(s) $q \in S$ as well as point(s) $q' \in S'(:= A \setminus S)$ such that $0 < d_\top(p,q), d_\top(p,q') \leqslant g$, g being the grid size (as in Definition 5). Adopting such a policy ensures that both the outer (or the inner) isothetic covers have minimum (maximum) area. Also, the isothetic frontier of an object is defined in a metric space where the background grid is not required. On the contrary, the isothetic cover of an object is identified assuming that the object is registered on the underlying grid with the goal of controlling the complexity of the cover by changing the grid size. For a higher grid size, we get a coarser description of the object, whereas for a lower one, we get a finer description.

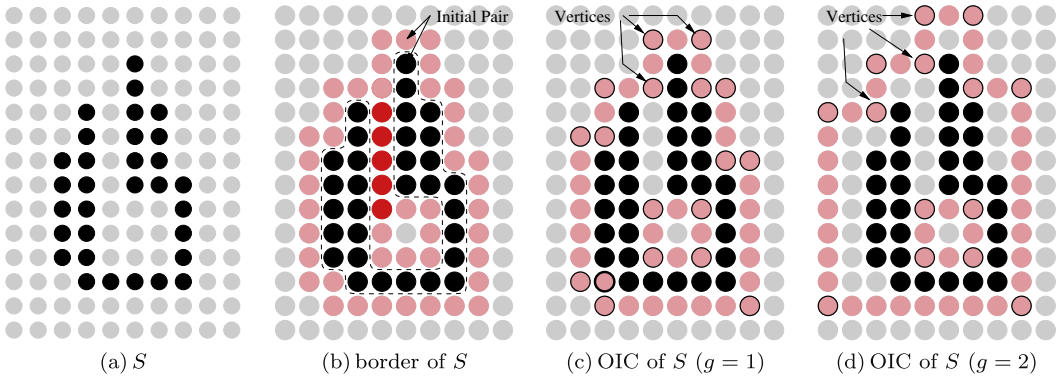


Fig. 3. Output-wise comparison between Rosenfeld's crack following algorithm and the proposed algorithm for a simple digital object, *S*.

In the literature, several other works dealing with boundary approximation [25,26], minimum-perimeter polygons [23,27], and the concavity trees can be found. Sklansky [23] described an algorithm for computing minimum-perimeter polygon (MPP) of digitized silhouettes (often referred as cellular complexes). A technique has been described for describing and measuring the concavities of such cellular complexes [24]. It combines the concept of half-cell expansion and the method of finding the cellular convex hulls in order to determine the concavity tree. The algorithm uses the concept a of stretched string constrained to lie in the cellular boundary of the digitized silhouette. In [26], two other algorithms, namely the square-tracing algorithm and the Moore-neighborhood-tracing algorithm, have been suggested for boundary tracing, which, however, lack the proof of correctness. In particular, counterexamples have been shown in [26] to point out the existence of various cases where the algorithms fail to produce correct results. Sloboda and Zat'ko [25] has presented a method of boundary approximation for finding the minimum-perimeter polygon in a given polygonally-defined annular region.

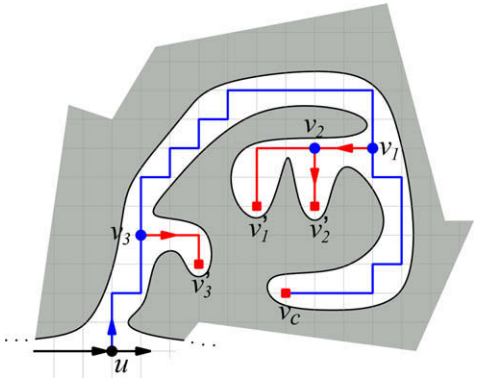


Fig. 4. An example of backtracking in a complex region from the current vertex v_c lying in a *dead end* (object *S* shown in gray and background S' in white). In the proposed algorithm, no such backtracking is required. See text for details.

4. Contribution of our work

The difficulty in finding the outer (inner) isothetic cover as a sequence of vertices (grid points) lies in the fact that, during the traversal of an isothetic polygon, if the traversed path enters a complex region (background region in the case of an outer cover or object region in the case of an inner cover) for which a path of retreat from that region at a later stage is not possible, which marks a *dead end*, then a backtracking is required from an appropriate vertex lying on the traversed path. If the current vertex v_c lies in a *dead end*, then a vertex v has to be found on the path $v_s \rightsquigarrow v_c$, where v_s denotes the start vertex, by tracing back from v_c to v_s, along the path $v_c \rightsquigarrow v_s$. Such a vertex v should have an alternative path that would possibly come out of the complex region. However, the alternative path from v may again lead to a vertex v' lying in a dead end, which again results in backtracking the path $v' \rightsquigarrow v$, and still failing to produce a feasible path coming out of the corresponding region. This, in turn, calls for (recursively or iteratively) backtracking and searching another alternative path from some other vertex in $v \rightsquigarrow v_s$.

An instance where backtracking is required, has been illustrated in Fig. 4. The path (shown in blue) from u leads to the vertex v_c (current vertex) that lies in a dead end (shown as a red square).[1] While backtracking from v_c, none of the nine grid points lying on the path $v_c \rightsquigarrow u$ provides any alternative path, which has to be verified for each of these points. The vertex v_1 is the first vertex found during backtracking, from which an alternative path is possible. While traversing along this alternative path from v_1, however, it is found that the path terminates at the vertex v_1' that also marks a dead end. Next, backtracking is made along $v_1' \rightsquigarrow v_1$, and a vertex v_2 is found from which the alternative path again ends at the vertex v_2' lying in a dead end. No other vertex in the remaining part of $v_1' \rightsquigarrow v_1$ (i.e., $v_2 \rightsquigarrow v_1$) admits an alternative path, and therefore, again a new vertex is looked for by resuming the backtracking along $v_1 \rightsquigarrow u$. One such vertex is v_3, which again does not produce a successful solution. Finally, the backtracking ends at the vertex u, which has an alternative path lying outside the concerned region.

The challenge, therefore, lies in recognizing such a complex region against a background grid, which, if entered, would need backtracking. If the entry point of such a region can be recognized during the traversal of a polygon belonging to an isothetic cover, then instead of following the path entering the complex region, the alternative path along the grid lines can be chosen. Such a strategy would be entirely free of the backtracking process, and hence can construct the isothetic cover in a simpler and faster way.

We have proposed an algorithm for finding the outer and the inner isothetic covers of an arbitrarily shaped digital object. The proposed algorithm does not require any backtracking of the already traversed path. Also, the concept of background grid has been introduced whose resolution can be varied to enable fine or coarse analysis of the underlying shape. Two different approaches on finding the isothetic covers have been discussed in [28,29], the latter for uniform grid and the former for nonuniform grid. In [28], the isothetic covers are obtained using the adjacency labels, namely 'L', 'R', 'B', and 'T', corresponding to the position of the object with

[1] For interpretation of the references to color in the text of Figs. 4 and 23, the reader is referred to the web version of this paper.

respect to a grid point. The approach is different from the proposed algorithm. In [29], the "edge matrix" and the "unit square matrix" have been used to find the isothetic covers. However, in that work, the theoretical formulation, algorithmic statements, and the proof of correctness have not been given. Possible applications of isothetic covers have not been discussed there. All these necessary requirements for the algorithm and its relevant applications have been given in detail in this paper.

The strength of the proposed algorithm lies in the fact that it takes into account the combinatorial arrangement of the grid lines with respect to the object while simultaneously traversing and determining the boundary of the polygon defining an isothetic cover. The algorithm is completely devoid of any backtracking, and therefore, is fast and efficient. Avoidance of backtracking also makes the algorithm output-sensitive, in the sense that its time-complexity becomes linear in the length of the sum of the perimeters of the polygons constituting the isothetic cover (in grid units).

The paper is organized as follows. The preliminaries, including the basic notions of digital geometry, necessary definitions, and review of earlier works have already been given in Sections 2 and 3. The algorithm for constructing the outer isothetic cover (OIC) for a digital object consisting of a single connected component is described in Section 5. The generalized algorithm to construct the isothetic covers (both outer isothetic cover (OIC) and inner isothetic cover (IIC)) of an object with multiple connected components, is given in Section 6. Section 7 contains the relevant experimental results on several databases and elaborates the scope of further applications using isothetic covers. Finally, in Section 8, we conclude the paper with few open problems and with pointers to some possible future directions.

5. Outer isothetic cover for a single connected component

The algorithm for constructing the outer isothetic cover (OIC), $\overline{P}(S)$, corresponding to an object S, which consists of only one connected component without holes, is stated here. The OIC of such an object consists of only one (outer) polygon. The generalized algorithm for an object having multiple connected components with or without holes is stated in Section 6. To construct $\overline{P}(S)$, we consider $\mathscr{I}$ to be a finite rectangular subset of $\mathbb{Z}^2$, which contains the entire object S. Let the height h and the width w of $\mathscr{I}$ be such that the grid size, g, divides both $h-1$ and $w-1$, and the boundary UGBs of $\mathscr{I}$ do not contain any part of S.

5.1. Combinatorial classification of a grid point

An isothetic polygon has two types of vertices with internal angles 90° and 270° (i.e., −90°). The procedure for finding $\overline{P}(S)$ is based on classifying the grid points of $\mathscr{I}$ while constructing $\overline{P}(S)$. The type of any grid point, $q(i,j)$, is determined using the object occupancies of its neighboring UGBs and their combinatorial arrangements, which are obtained as follows.

5.2. Object occupancy

UGB(i,j) is described by two horizontal (s^i_j and s^i_{j+g}) and two vertical (s^j_i and s^j_{i+g}) grid segments (Fig. 2). Since S is 8-connected and a grid segment is 4-connected, the intersection between them is well defined [21]. Hence, S has an intersection with UGB(i,j), or UGB(i,j) is said to have an *object occupancy*, if and only if there exists a point p in $S \cap (s^i_j \cup s^i_{j+g} \cup s^j_i \cup s^j_{i+g})$.

Hence, for each UGB, U_a $(a = 1,2,3,4)$, incident at $q(i,j)$, there are two possibilities: either $U_a \cap S \neq \emptyset$ or $U_a \cap S = \emptyset$, thus giving rise to $(2^4 =)$ 16 possible arrangements of the four UGBs. These 16 possible arrangements are further grouped into five combinatorial classes in our algorithm, where a particular class C_m includes all arrangements for which exactly $m(= 0,1,2,3,4)$ out of the four neighboring UGBs of q has/have object occupancy and the remaining $(4-q)$ ones do not have any object occupancy.

5.3. Combinatorial arrangements of UGBs

Let $f(U_a)$ denote the object occupancy of U_a $(a = 1,2,3,4)$, defined as follows:

$$f(U_a) = \begin{cases} 0 & \text{if } U_a \cap S = \emptyset \\ 1 & \text{if } U_a \cap S \neq \emptyset \end{cases} \quad (1)$$

The number of UGBs having object occupancy is, therefore, given by $m = f(U_1) + \cdots + f(U_4)$. As a result, the grid point q will belong to one of the following five classes corresponding to five different values of m, which are also illustrated in Fig. 5.

1. C_0 $(m = 0)$: None of the four UGBs has object occupancy.
2. C_1 $(m = 1)$: Exactly one out of the four UGBs has object occupancy. Four such arrangements are possible depending on the UGB having occupancy.
3. C_2 $(m = 2)$: There are two arrangements for $m = 2$: UGBs having object occupancy (i) have a grid segment in common (i.e., adjacent); (ii) do not have a common grid segment (i.e., diagonally opposite).
   (i) C_{2A} *(Adjacent):* U_a and U_b $(a,b \in \{1,2,3,4\}, a \neq b)$ have object occupancy, such that $(a+b) \bmod 2 = 1$. Four such arrangements are possible depending on the common grid segment.
   (ii) C_{2B} *(Diagonal)*: Either U_1 and U_3, or U_2 and U_4, have object occupancy.
4. C_3 $(m = 3)$: Three neighboring UGBs have object occupancy. Four arrangements are possible depending on the UGB that has no object occupancy.
5. C_4 $(m = 4)$: All four neighboring UGBs have object occupancy.

The positions of a UGB, of a grid segment, and of a grid point, with respect to the OIC are determined using the following lemmas and theorems.

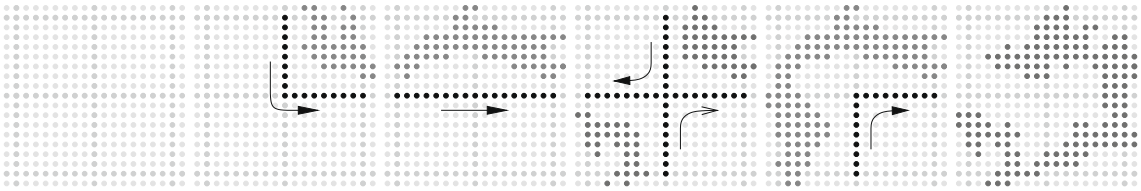
Lemma 1. *The interior of a UGB lies outside $\overline{P}(S)$ if and only if the UGB has no object occupancy.*

Proof. Let, for contradiction, there be a UGB, $U' \in \overline{P}(S)$, with no object occupancy. Then, either U' has at least one grid segment on (the boundary of) $\overline{P}(S)$, or U' has none of its four grid segments on $\overline{P}(S)$. If U' has one or more grid segments on the boundary of $\overline{P}(S)$, then the interior and the segments of U' lying on $\overline{P}(S)$ can be excluded from $\overline{P}(S)$ so that the segments of U' lying inside $\overline{P}(S)$ redefine the corresponding boundary of $\overline{P}(S)$. If U' has no segment on $\overline{P}(S)$, then only the interior of U' is excluded from $\overline{P}(S)$ and the four segments of U' add to the boundary of $\overline{P}(S)$. In either case, $\overline{P}(S)$ with a UGB having no object occupancy is not of minimum area, which contradicts the definition of the OIC (Definition 9).

Conversely, if U' lies outside $\overline{P}(S)$, then U' cannot have an object occupancy as no part of S lies outside $\overline{P}(S)$. □

Lemma 2. *A grid segment s belongs to the boundary of $\overline{P}(S)$ if and only if, out of the two UGBs having s in common, the interior of one lies outside and that of the other lies inside $\overline{P}(S)$.*

Proof. Let the two UGBs with s as the common grid segment, be U and U'. Hence, by Lemma 1, if the interior of U lies inside and that of U' outside $\overline{P}(S)$, then the segment s lies on $\overline{P}(S)$.



(a) C_0 (b) C_1 (c) C_{2A} (d) C_{2B} (e) C_3 (f) C_4

Fig. 5. Vertex classification in an OIC: (a) exterior point, (b) 90° vertex, (c) edge point, (d) cross vertex, (e) 270° vertex, and (f) interior point.

The converse is obvious; if s lies on the boundary, then the interior of one UGB is in $\overline{P}(S)$ and that of the other is not. □

Theorem 1 (*90° vertex*). *A grid point q is a 90° vertex of $\overline{P}(S)$ if and only if q belongs to class C_1.*

Proof. W.l.o.g., let U_1 be occupied by the object S and the other three neighbor UGBs of $q(i,j)$ are not. As, of the two horizontally adjacent UGBs, U_1 and U_2, only U_1 has object occupancy, the grid segment s_i^j, common to U_1 and U_2, are on $\overline{P}(S)$ as stated in Lemma 2. Similarly, s_j^i is on the boundary of $\overline{P}(S)$, since between the two vertically adjacent segments, U_1 and U_4, only U_1 is occupied by S. The grid segment s_j^{i-g} is not on the boundary of $\overline{P}(S)$ as none of its adjacent UGBs, U_2 and U_3, has object occupancy. Same is true for s_i^{j-g}. Hence, s_i^j and s_j^i are two grid segments on $\overline{P}(S)$ incident at q, thereby making it a 90° vertex.

Conversely, if q is a 90° vertex, then exactly two mutually perpendicular grid segments, say s_i^j and s_j^i, of the four incident grid segments at q, are on $\overline{P}(S)$. As s_j^{i-g} and s_i^{j-g} are not on $\overline{P}(S)$, it can be shown that only U_1 has object occupancy. This is true for other three combinations for q being a 90° vertex. Thus, if q is a 90° vertex, then it belongs to class C_1. □

Theorem 2 (*270° vertex*). *A grid point q is a 270° vertex of $\overline{P}(S)$ if and only if q belongs to class C_3 or class C_{2B}.*

Proof. If q belongs to C_3, then exactly three of the UGBs have object occupancy. Let, w.l.o.g., U_1 has no object occupancy. Then, s_i^j $(= U_1 \cap U_2)$ and s_j^i $(= U_1 \cap U_4)$ are on the boundary of $\overline{P}(S)$ by Lemma 2. The included angle at q being 270°, q is a 270° vertex.

If q belongs to C_{2B}, then two diagonal UGBs are occupied by S. By Lemma 2, all the four grid segments incident at q lie on $\overline{P}(S)$. Hence, q occurs twice in $\overline{P}(S)$. However, as each polygon in $\overline{P}(S)$ is simple, q cannot occur twice in a particular polygon of $\overline{P}(S)$. Thus, q occurs in two polygons, once for each. Further, since there is exactly one outer polygon in $\overline{P}(S)$ (S being a single component, with or without holes), q lies on at least one hole polygon. As shown in Fig. 5(d), if the segments s_i^{j-g} and s_j^{i-g} lie on the same polygon, and S lies left of the polygon during its traversal, then, while taking a left turn at q (traversing s_i^{j-g} first and s_j^{i-g} next), the part of S in U_3 lies left but that in U_1 lies right, which is a contradiction. If there is a right turn at q by traversing s_i^{j-g} first and s_j^i next, then S lies left, both in U_1 and U_3. This implies that q is a 270° vertex for one polygon. Clearly, for the other polygon, q will be again a 270° vertex for similar reasons.

Conversely, if q is a 270° vertex, then one UGB, say U_1, has no object occupancy and both of its adjacent UGBs (U_2 and U_4) have object occupancy. If U_3 has object occupancy, then q belongs to C_3, and if U_3 has no object occupancy, then q belongs to C_{2B} by Lemma 2. □

Theorem 3 (*edge point*). *A grid point q is a nonvertex edge point of $\overline{P}(S)$ if and only if q belongs to class C_{2A}.*

Proof. If q belongs to class C_{2A}, then two adjacent UGBs, say U_1 and U_2, are occupied by the object and the other two are not. Thus, by Lemma 2, s_j^{i-g} and s_j^i are on the boundary of $\overline{P}(S)$. As both the grid segments are horizontal and $s_j^{i-g} \cap s_j^i = q$, q is a nonvertex edge point on the boundary of $\overline{P}(S)$.

If q is an edge point, then both the segments incident at q, which are part of $\overline{P}(S)$, are either horizontal or vertical. Thus, only two adjacent UGBs have object occupancy. Hence, by Lemma 2, when q is an edge point, it belongs to class C_{2A}. □

If q belongs to class C_{2B}, then q occurs twice as a vertex in the OIC, which is referred as a *cross vertex* in Fig. 5(d). If a grid point q is neither a vertex nor an edge point of $\overline{P}(S)$, then it lies either inside or outside $\overline{P}(S)$. The classification of such a grid point w.r.t. $\overline{P}(S)$ is done using the following theorems.

Theorem 4 (*exterior point*). *A grid point q lies outside $\overline{P}(S)$ if and only if q belongs to class C_0.*

Proof. If q belongs to class C_0, then no U_a $(a = 1,2,3,4)$, incident at q, has object occupancy, and hence, by Lemma 2, none of the four grid segments incident at q lies on $\overline{P}(S)$. As a result, the intersection of these four grid segments, which is q, does not belong to $\overline{P}(S)$.

If q lies outside $\overline{P}(S)$, then none of the four grid segments incident at q is on the boundary of $\overline{P}(S)$. Hence, none of the neighboring UGBs has object occupancy, which means q belongs to class C_0. □

Theorem 5 (*interior point*). *A grid point q is in the interior of $\overline{P}(S)$ if and only if q belongs to class C_4.*

Proof. When q belongs to class C_4, all four UGBs incident at q have object occupancy. None of the four grid segments incident at q qualifies for being on the boundary of $\overline{P}(S)$ by Lemmas 1 and 2. Thus, q is an interior point of $\overline{P}(S)$.

Conversely, when the grid point q is in the interior of $\overline{P}(S)$, none of the grid segments incident at q is on the boundary of $\overline{P}(S)$. Hence, all the corresponding UGBs have object occupancy, which means, q belongs to class C_4. □

It may be mentioned here that, if S is a subset of the interior of a UGB, which may arise if S is sufficiently small relative to the grid size, then no grid segment intersects S. Such a degenerate case, where all grid points are classified to C_0, can be handled by determining the object occupancy based on the intersection of S with the interior of a UGB.

5.4. Construction of $\overline{P}(S)$

Here we assume that the object S consists of one connected component without any hole, and its top-left point, $p_0(i_0,j_0)$, is given. The point p_0 is the top-left point of S if and only if, for each other point $(i,j) \in S$, either $j < j_0$, or $j = j_0$ and $i > i_0$. The *start point* of traversal, q_s, is first determined from p_0. Then the construction starts from q_s and ends when the traversal reaches q_s (Theorem 7). The traversal is guided by the type of each grid point q, which is obtained from the corresponding class of q. The type t of q

corresponding to its internal angle 90°, 180°, and 270°, w.r.t. an isothetic polygon is considered to be **1**, **0**, and $-\mathbf{1}$, respectively. The algorithm outputs $\overline{P}(S)$ (here, one outer polygon) as an ordered set of vertices. The information stored corresponding to each vertex are its coordinates and type. In the degenerate case in which S lies in the interior of a single UGB, $\overline{P}(S)$ is empty. For, if S lies in the interior of a single UGB, say U, then none of the four edges of U has any intersection with S. Hence, each of the four grid points corresponding to U is classified to class C_0.

5.5. Determination of start point

If p_0 lies in the interior of a UGB (w.l.o.g, say U_1), then the top-left grid point of U_1 is considered as the start point, q_s. If p_0 coincides with a grid point, then the top-left grid point of U_2 (incident at p_0, Fig. 2(a)) is considered as q_s. Otherwise, p_0 lies on a horizontal (vertical) grid segment common to two adjacent UGBs, say U_1 and U_4; then the top-left grid point of U_1 is considered as q_s. The coordinates of q_s, therefore, are given by

$$i_s = (\lceil i_0/g \rceil - 1) \times g, \quad j_s = (\lfloor j_0/g \rfloor + 1) \times g. \tag{2}$$

The intersections of the neighboring UGBs of q_s with S are computed and the vertex type of q_s is determined (Theorems 1–3). Let d denote the direction to be followed from a vertex or an edge point q while constructing an isothetic polygon of $\overline{P}(S)$. The different values of d, namely 0, 1, 2, and 3, denote the directions towards right, top, left, and down, respectively. The starting direction from q_s is given by $(2 + t_s) \bmod 4$. By Eq. (2), neither U_1 nor U_2 (of q_s), and no UGB above U_1, can have object occupancy. However, U_3 may or may not have an object occupancy, thereby giving rise to two distinct cases: q_s is a 90° vertex when only U_4 is occupied; q_s is an edge point when both U_3 and U_4 are occupied. We get the following lemma.

Lemma 3. *The start point is either a 90° vertex or an edge point.*

5.6. Tracing the polygon

During the traversal of (the boundary of the polygon in) $\overline{P}(S)$, the direction d_c from the current grid point q_c, and the coordinates (i_c, j_c) of q_c, being known, we compute the coordinates of the next grid point, $q_n := (i_n, j_n)$, lying on $\overline{P}(S)$, as follows:

$$\begin{array}{l|l} d_c = 0: \quad i_n = i_c + g & d_c = 1: \quad i_n = i_c \\ \qquad\qquad j_n = j_c & \qquad\qquad j_n = j_c + g \\ \hline d_c = 2: \quad i_n = i_c - g & d_c = 3: \quad i_n = i_c \\ \qquad\qquad j_n = j_c & \qquad\qquad j_n = j_c - g \end{array}$$

which is expressed in a simpler way to

$$(i_n, j_n) = d_c \circledast (i_c, j_c). \tag{3}$$

The vertex type t_n of q_n is computed by determining the object occupancy of its neighboring UGBs. The direction of traversal, d_n, from q_n is given by

$$d_n = (d_c + t_n) \bmod 4 \tag{4}$$

Iteratively, the details of each grid point lying on $\overline{P}(S)$ are computed until q_n coincides with q_s. If the grid point is a vertex, then it is added to the sequence of vertices comprising $\overline{P}(S)$. The following theorems justify the correctness of $\overline{P}(S)$ (Definition 9) and how the traversal converges to obtain the sequence of its vertices.

Theorem 6. *If p is a point lying on $\overline{P}(S)$, then $0 < d_\top(p, S) \leqslant g$.*

Proof. Since no point on $\overline{P}(S)$ is in S, $d_\top(p, S) > 0$. To show that $d_\top(p, S) > g$, let, w.l.o.g., p be any point on the horizontal grid segment s_j^i that lies on $\overline{P}(S)$. Let $d_\top(p, S) = h > g$, if possible. For any point q, belonging to U_1 or U_4, $d_\top(p, q)$ is less than or equal to g. Since $h > g$, neither U_1 nor U_4 has any object occupancy, wherefore their interiors are outside $\overline{P}(S)$ (Lemma 1). Hence, s_j^i does not lie on $\overline{P}(S)$ (Lemma 2), which means p cannot be on $\overline{P}(S)$ — a contradiction. □

Theorem 7. *The construction of $\overline{P}(S)$ concludes at the start vertex.*

Proof. S intersects a subset of UGBs comprising $\mathscr{I}$. Note that in the degenerate case where S is a subset of the interior of a UGB, S still intersects a UGB but it does not intersect the border of that UGB and the $\overline{P}(S)$ is empty. The construction of $\overline{P}(S)$ starts from a 90° vertex or an edge point, q_s (Lemmma 3), whose distance from S is at most g (Theorem 6). The construction process continues along those grid segments such that all points belonging to such a grid segment have the maximum distance of g from S (Theorem 6). If q_s is a 90° vertex (edge point), then the downward (leftward) grid segment from q_s is traversed at the first step, and its other grid segment on $\overline{P}(S)$ remains untraversed. Each other vertex or edge point of $\overline{P}(S)$ also has exactly two grid segments lying on $\overline{P}(S)$, which would be traversed in two consecutive steps during the construction. Hence, the untraversed grid segment incident at q_s is finally traversed, which concludes the construction of $\overline{P}(S)$. □

5.7. Algorithm MAKE-OIP

The algorithm MAKE-OIP that constructs the outer polygon P, corresponding to an object S consisting of a single connected component, is shown in Fig. 6 (also see Section 5.8 and Fig. 6). The start point, q_s (u_1 in Fig. 6), is determined by the procedure START using the top-left point, p_0, as input. The subsequent vertices and edge points of P are visited one by one in the **do-while** loop (Steps 5–10) of the algorithm MAKE-OIP. If $|P|$ denotes the length of the perimeter of P, then the number of grid points (i.e., vertices and edge points) lying on the perimeter of P is $|P|/g$, when the grid size is g. In each iteration (**do-while** loop), the type of a grid point q lying on P is checked and added to the list of vertices, L, if q is a vertex of P. The coordinates and the type of q lying on P are computed in each iteration. The procedure VTYPE finds the type of q using the four UGBs incident at q. For each UGB, each point on each of its four defining grid segments is considered one by one to check whether it belongs to S, which needs $O(1)$ time in the best case and $O(g)$ time in the worst case. Thus, in the worst case, each iteration takes $O(g)$ time, finding the start point q_s takes $O(g)$ time, and all other steps of the algorithm MAKE-OIP take $\Theta(1)$ time in total. Hence, the worst-case time-complexity of the algorithm MAKE-OIP for an object S with a single connected component without any hole, is given by $(|P|/g) \times O(g) + O(g) + \Theta(1)$, which simplifies to $(|P|/g) \times O(g) = O(|P|)$. The best-case time-complexity, found in a similar way, is $(|P|/g) \times O(1) + O(1) + \Theta(1) = O(|P|/g)$. The algorithm is, therefore, linear on the perimeter of the outer polygon, and hence output-sensitive.

5.8. Demonstration of MAKE-OIP

Fig. 7 shows a demonstration of the algorithm MAKE-OIP which constructs the outer cover of a digital object. From the given top-left point, p_0, of the digital object, the start vertex u_1 is found by the START procedure (Step 2). The direction from u_1 is decided as $d = (2 + t_s) \bmod 4$, i.e., $d = 3$ (Step 4). Hence, the construction follows the downward direction, and the next grid point is determined in Step 7. This is the grid point vertically below u_1 (say, q'). The type of q' is $t = 0$ (from procedure VTYPE), and the direction of traversal is $d = 3$. In this manner, the traversal proceeds to u_2. The type of u_2 is again determined as $t = 1$, and the subsequent direction from u_2 is $d = 0$ (in a horizontally right direction). Similarly, at each grid point, the type of the grid point is checked and

Algorithm MAKE-OIP (S, p_0)

Steps:

1. $L \leftarrow \emptyset$
2. $q_s \leftarrow$ START (S, p_0)
3. $q \leftarrow q_s$
4. $d \leftarrow (2 + t_s) \mod 4$
5. **do**
6. **if** $t \in \{1, -1\}$ **then** $L \leftarrow L \cup \{q\}$
7. $q \leftarrow d \circledast q \triangleright$ Eqn. 3
8. $t \leftarrow$ VTYPE(S, q)
9. **if** $t = -2$ **then** $t \leftarrow -1$
10. $d \leftarrow (d + t) \mod 4$
11. **while** $(q \neq q_s)$
12. **return** L

Procedure START (S, p_0)

Steps:

1. $i_s \leftarrow (\lceil i_0/g \rceil + 1) \times g$, $j_s \leftarrow (\lfloor j_0/g \rfloor - 1) \times g$
2. $t_s \leftarrow$ VTYPE(S, q_s)
3. **return** $q_s \triangleright (i_s, j_s, t_s)$

Procedure VTYPE (S, q)

Steps:

1. $m \leftarrow 0, r \leftarrow 0$
2. **for** $k \leftarrow 1$ to 4
3. **if** $U_i^q \cap S \neq \emptyset$
4. $m \leftarrow m + 1,\ r \leftarrow r + k$
5. **if** $m = 2$ **and** $r \in \{4, 6\}$ **then** $t \leftarrow -2$
6. **else if** $m \in \{0, 4\}$ **then** $t \leftarrow 0$
7. **else** $t \leftarrow 2 - m$
8. **return** t

Fig. 6. The algorithm and the procedures to construct $\overline{P}(S)$ for an object S having a single connected component.

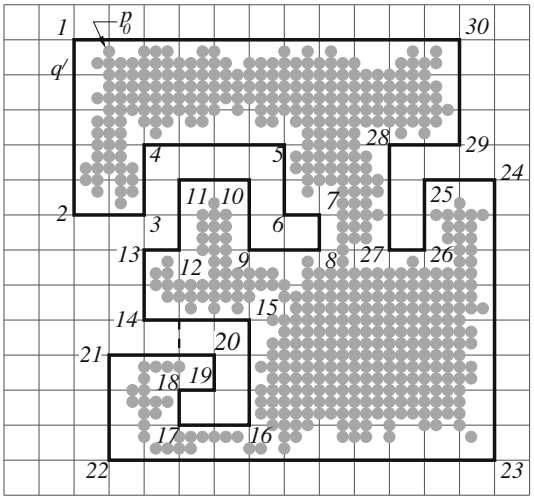


Fig. 7. Demonstration of the algorithm MAKE-OIP on a digital object is shown here. Each vertex u_i of the resulting outer (isothetic) cover is denoted by 'i' for sake of simplicity. Note that the outer cover is constructed in the counterclockwise direction by the algorithm.

the direction of subsequent traversal is determined. For example, at u_5, $d = 0$ and type of u_5 is $t = 3$; hence the next direction is $d = (d + t) \mod 4$, i.e., $d = (0 + 3) \mod 4 = 3$, and so the traversal follows the downward direction. In this manner, the algorithm detects the vertices up to u_{30} and concludes when the traversal reaches the start vertex u_1.

6. Generalized algorithms to find the isothetic covers

6.1. Outer isothetic cover

The algorithm MAKE-OIC to construct the OIC of an object S, having multiple components with or without holes, is given in Fig. 8. Each grid point, $q \in \mathscr{I}$, initialized as unvisited (Steps 2–4), is considered in row-major order (Steps 5–11). If q is not already visited (Step 6) and qualifies as a vertex (Steps 7 and 9), then the construction of a new polygon of $\overline{P}(S)$ starts from q as the start vertex (Steps 8 and 10). The procedure MAKE-IP traces an outer polygon if q is a 90° vertex, and a hole polygon if it is a 270° vertex. The procedure MAKE-IP is similar in nature as the algorithm presented in Fig. 6, except that the grid points traversed in course of constructing the polygon are marked as "visited" in Step 4 of MAKE-IP. In Step 9 of MAKE-IP, the start vertex is added to the ordered list, L, to mark the end of one polygon. The vertices of the subsequent polygons, if any, are appended to L starting at (and ending with) the corresponding start vertices, so that the final form of OIC is given by $L = \{q_{11}, q_{21}, \ldots, q_{11}, q_{21}, q_{22}, \ldots, q_{21}, \ldots\}$, where, q_{i1} is the start vertex and q_{ij} is the jth vertex of the ith polygon, P_i.

6.2. Time complexity

The basic operation used in the algorithm is the checking of each grid point of $\mathscr{I}$, in row-major order, for its possible candidature of being a vertex or an edge point or none, which needs $O(g)$ time in the worst case. This is done exactly once for each grid point, as already-traversed grid points are marked as "visited". The number of grid points in $\mathscr{I}$, excepting the boundary grid points (since they will not lie on the isothetic cover as per our consideration of $\mathscr{I}$), is $((h-1)/g - 1) \times ((w-1)/g - 1) = \Theta(hw/g^2)$. Hence, the worst-case time-complexity of the algorithm MAKE-OIC is $\Theta(hw/g^2) \cdot O(g) = O(hw/g)$.

6.3. Inner isothetic cover

The inner cover, $\underline{P}(S)$, can be constructed in a manner exactly converse to that of $\overline{P}(S)$. A grid point $q \in S$ is classified as a vertex of $\underline{P}(S)$, depending on the intersections of the UGBs (incident at q) with $S' := \mathscr{I} \setminus S$. If m $(0 \leqslant m \leqslant 4)$ denotes the total number of UGBs intersected by S', then q belongs to class C'_m. Class C'_1 and class C'_3 correspond to 90° and 270° vertices, respectively. For class C'_2, if the UGBs intersected by S' has a common grid segment, then q is an edge point; otherwise, q is a 270° vertex. Points in C'_0 and in C'_4 lie, respectively, inside and outside $\underline{P}(S)$.

Fig. 9 states the algorithm MAKE-IIC for constructing the inner cover of a general object S, having one or more components with or

Algorithm MAKE-OIC (S)

Steps:

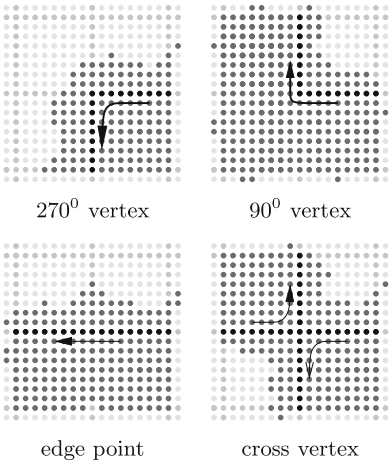
1. $L \leftarrow \emptyset$
2. **for** each grid point $q \in \mathcal{I}$
   ▷ in row-major order
3. $visited[q] \leftarrow$ FALSE
4. $t \leftarrow \text{VTYPE}(S, q)$
5. **for** each grid point $q \in \mathcal{I}$
6. **if** $visited[q] =$ FALSE
7. **if** $t = 1$
8. MAKE-IP $(S, q, 3)$
9. **else if** $t = -1$
10. MAKE-IP $(S, q, 0)$
11. **return** L

Procedure MAKE-IP (S, q, d)

Steps:

1. $q_s \leftarrow q$
2. **do**
3. **if** $t = -2$ **then** $t = -1$
4. **else** $visited[q] \leftarrow$ TRUE
5. **if** $t \in \{1, -1\}$ **then** $L \leftarrow L \cup \{q\}$
6. $q \leftarrow d \circledast q$
7. $d \leftarrow (d + t) \mod 4$ ▷ Eqn. 3
8. **while** $(q \neq q_s)$
9. $L \leftarrow L \cup \{q_s\}$
10. **return**

Fig. 8. The generalized algorithm and the procedures to construct $\overline{P}(S)$ for an object S with multiple connected components (possibly having holes).



Algorithm MAKE-IIC (S', g)

Steps:

1. $L \leftarrow \emptyset$
2. **for** each grid point $q \in \mathcal{I}$
   ▷ in row-major order
3. $visited[q] \leftarrow$ FALSE
4. $t \leftarrow \text{VTYPE}(S', q)$
5. **for** each grid point $q \in \mathcal{I}$
6. **if** $visited[q] =$ FALSE
7. **if** $t = 1$
8. MAKE-IP $(S', q, 0)$
9. **else if** $t = -1$
10. MAKE-IP $(S', q, 3)$
11. return L

Fig. 9. Vertex classification and the algorithm to construct the IIC.

without holes. The algorithm is similar to MAKE-OIC, but it takes the background, S', as the input. The construction of a polygon of $\underline{P}(S)$ starts from a 90° vertex, q_s, keeping S' left during the traversal. The polygon is traced to the next grid point, q_n. The type of q_n is decided and the direction of traversal from q_n is computed. The traversal is continued until q_s is reached. As the basic steps of the algorithm MAKE-IIC are similar to those of MAKE-OIC, the time complexity of the algorithm is same as that of MAKE-OIC.

7. Experimental results

We have implemented the proposed algorithm in C in SunOS Release 5.7 Generic of Sun Ultra 5_10, Sparc, 233 MHz. The algorithm is run on different sets of binary images, such as (i) geometric figures, (ii) logo images, (iii) object-type images, (iv) optical and handwritten characters, and (v) scanned document images. The results and related findings are discussed in the following sections.

7.1. Geometric figures

The isothetic covers for $g = 3$ and $g = 6$ corresponding to a few simple geometric shapes are shown in Fig. 10. For a square-shaped object, as shown in this figure, the number of vertices is four for $g = 3$ and $g = 6$. This is in conformity with the fact that the bounding shapes (or complexities) of the OIC and the IIC of an axis-parallel square should not alter with a change in the grid size. However, if the square is tilted, then the complexity (i.e., the number of vertices) of the OIC and that of the IIC increases with a decrease of the grid size. Further, in the vertex sequence of the OIC (IIC) of a tilted square, as shown in Fig. 10, each vertex of type **1** is followed by a vertex of either type **1** or type $-$**1**, and each vertex of type $-$**1** by a vertex of type **1**. Each pattern **1**($-$**1**) contains a horizontal/vertical segment of length kg or $(k + 1)g$, where $k > 1$, and each pattern ($-$**1**)**1** contains a vertical/horizontal segment of length g. Such a sequence satisfies the properties of digital straightness [30,22], indicating that the longest subsequence of the OIC

having the form $\mathbf{1}((-\mathbf{1})\mathbf{1})^*\mathbf{1}$ corresponds to a straight part of the underlying shape. Two consecutive vertices with type **1** imply the start of a new straight edge, thereby corresponding to a 90° corner of a square.

For a disc, the complexities of both the OIC and the IIC decrease with an increase of the grid size and vice versa, as shown in Fig. 10. The symmetry owing to the circular shape of a disc is also captured in the outer and the inner covers. The symmetry, however, is only present in each of the two covers only about the vertical, horizontal, and two diagonal lines (45° and 135°) passing through the center of the corresponding cover due to the anisotropic nature of an orthogonal grid. Similarly, for an ellipse, the corresponding OIC and IIC also possess the desired horizontal and vertical symmetries, but not the diagonal symmetries.

As a more complicated geometric shape, we have considered a spiral and have shown its OICs in Fig. 11. A series of the pattern $(\mathbf{1}(-\mathbf{1}))^*(-\mathbf{1})(-\mathbf{1})$ occurs while the OIC traverses inwards or "spirals in". The spiral unwinds with a consecutive occurrence of the pattern $(\mathbf{11})(\mathbf{1}(-\mathbf{1}))^*$. The pattern $(-\mathbf{1})(-\mathbf{1})$ indicates a concave region and the pattern **11** indicates a convex region. The asymmetry of a spiral is also captured in its OIC and IIC. The complexity of the cover decreases with an increase of the grid size. When the grid size is sufficiently high, the OIC may not capture the concavity of the spiral as the width of the concavity is less than that of the grid size.

Table 1 shows the number of vertices and the perimeter of OIC, and the CPU time required for the computation of an OIC on different grid sizes for several images. As g increases, the number of vertices of an OIC decreases. The CPU time also decreases with an increase of the grid size. Fig. 12 shows how the number of vertices and the perimeter of an OIC vary with increasing grid size. It may be noted here that the "myth" image referred here has been shown in Fig. 25. Fig. 13 shows that the CPU time decreases as g increases. Fig. 14 shows how the number of vertices, n, of the OIC of a given object decreases with the increase of g. The change in number of vertices n of the OIC, corresponding to $g = 4$, on rotating a given object by an angle θ, from 1° to 90°, is plotted in Fig. 14. The plot shows that the pattern of variation of n is repeated every 45°, the pattern is regular for a symmetric object (e.g., square), and it varies for more complex, irregular objects like logo and spiral. In Fig. 15, we have furnished two 3D plots of the error frequency versus g and $d_\top$ corresponding to the images "spiral" and "myth". For a grid size g, the error frequency, $f(g,\delta)$, is given by the number of points on $\overline{P}(S)$ for which the nearest object points are at a distance δ (measured using $d_\top$).

7.2. Logo and object-type images

The OICs of four logo images for $g = 8$ and $g = 16$ are shown in Fig. 16. The OIC can be coded using the vertex types, **1** and $-\mathbf{1}$, and the edge lengths, to generate a shape code that describes the OIC, and hence to capture the structural information about the underlying object. Such shape codes can be generated for appropriate grid sizes to derive a multi-resolution feature-vector. These feature-vectors can be used to compute the Hamming distance between two or more objects as a measure of similarity for object indexing and retrieval [31].

In Fig. 17, the OICs of four animal images, and in Fig 18, the OICs of two different types of leaves, are shown for $g = 8$ and $g = 16$. Like logo images, the OICs capture the shape of the animal images in a similar way. Usually, the OIC consists of a single polygon. In some cases, however, the OIC also contains a few hole polygons or pseudo-hole polygons. Note that a pseudo-hole polygon is created due to vertex/vertices of class C_{2B} (Fig 5(d)). For example,

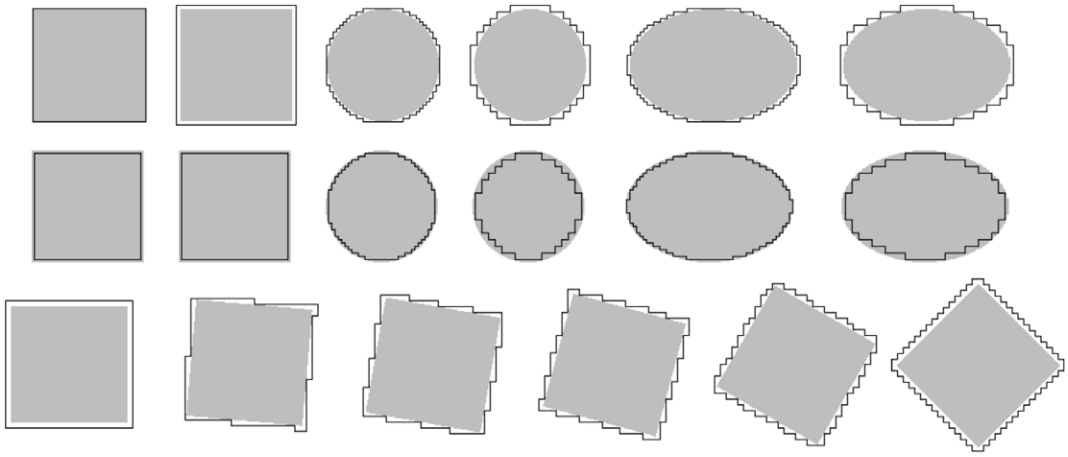


Fig. 10. The first row shows the OICs and the second row shows the IICs of different geometric shapes for $g = 3$ (odd columns) and $g = 6$ (even columns). The third row shows the OICs of a square, rotated at an interval of $7\frac{1^\circ}{2}$.

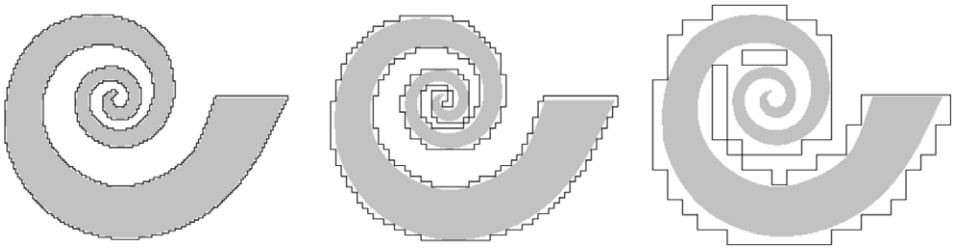


Fig. 11. The OICs of a spiral at different grid sizes: g= 2, 5, and 14.

Table 1
Numbers of vertices (n), perimeters (s), and areas (a) of the isothetic covers, and the respective CPU times (T, in milliseconds) corresponding to different grid sizes.

Image	Image size (object size)	g	OIC				IIC			
			n	s	a	T	n	s	a	T
Logo	200 × 200	4	100	886	8971	6	104	808	5205	2
	(6588)	8	40	704	10,589	6	46	684	3481	2
		16	14	512	14,337	4	12	192	867	2
Leaf	296 × 412	4	346	2400	49,055	14	322	2120	38,844	17
	(42,835)	8	168	2301	54,140	14	148	1936	35,084	8
		16	63	1855	61,855	8	70	1696	26,708	8
Spiral	442 × 442	4	300	2096	28,777	48	286	2021	20,437	20
	(23,776)	8	158	2032	33,009	38	116	1584	16,219	12
		16	60	1472	41,437	25	42	1184	10,322	9
Myth	420 × 710	4	468	4304	55,409	69	464	3792	34,194	66
	(42,773)	8	208	3504	63,634	49	206	3005	27,178	45
		16	82	2592	77,580	38	66	1600	16,168	35

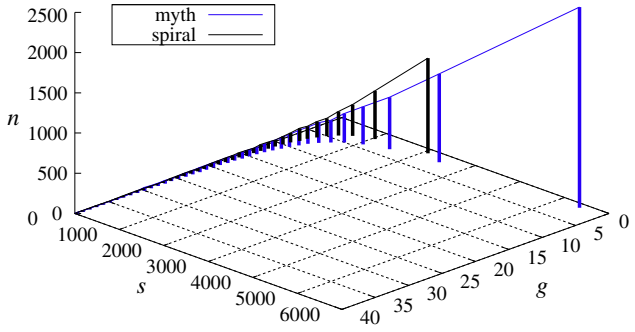


Fig. 12. Plot on the number of vertices (n) and the perimeter (s) of the OICs versus the grid size (g) corresponding to the images "spiral" and "myth".

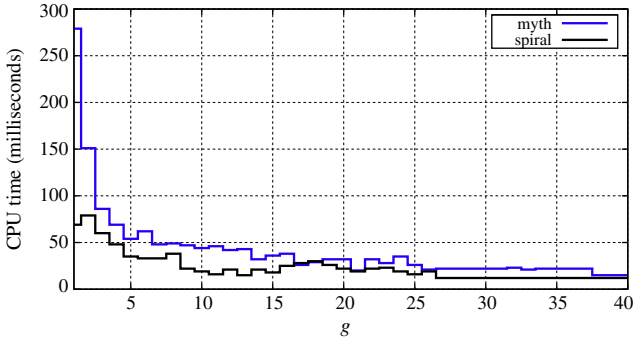


Fig. 13. Plot on CPU time (in milliseconds) for construction of the OICs versus the grid size (g) corresponding to the images "spiral" and "myth".

the OIC of the image "elephant" has one hole polygon and one pseudo-hole polygon for $g = 8$, and has two pseudo-hole polygons for $g = 16$.

The isothetic covers of a digital object can be used for many practical applications. For example, the shape complexity of a digital object can be computed by chain-code encoding [32] of the OIC and reducing the chain code using certain reduction rules [33]. Another example is the polygonal approximation of thick and rough digital curves, which can be derived using their isothetic covers [34]. An example is shown in Fig. 19, where a polygonal approximation of a digital curve representing the map of India is shown. The outer polygon and the hole polygon corresponding to this curve approximates the map of India. Similarly, a real-world object, when digitized, may possess disconnectedness, noisy information, etc. The existing algorithms on component labeling are liable to produce components that are larger in number than sought for. Whereas, using the isothetic covers, two or more components that are spatially not far off, may be reported as a single loosely connected component [35]. Fig. 20 shows how we can get a single loosely connected components corresponding to "a flock of birds" for $g = 16$.

7.3. Optical and handwritten characters

Fig. 21 shows the OICs corresponding to a set of optical and handwritten Bengali characters and numerals. It is evident from Fig. 21 that the OIC corresponding to an optical character has a regular and definite shape. However, the OIC corresponding to the same character in the handwritten set is not as definite and regular as the optical one. For example, a straight segment (horizontal,

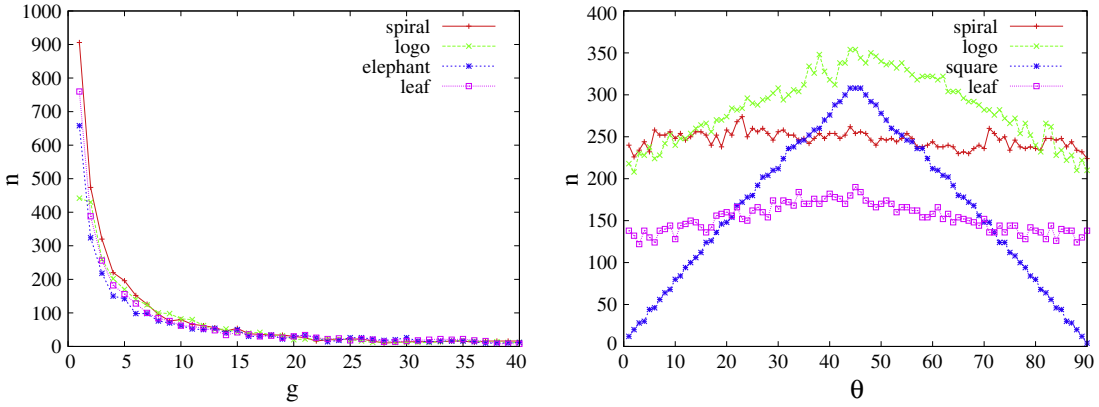


Fig. 14. Plot on the number of vertices, n, versus g, and plot on n versus θ, where θ is the angle of rotation for $g = 4$.

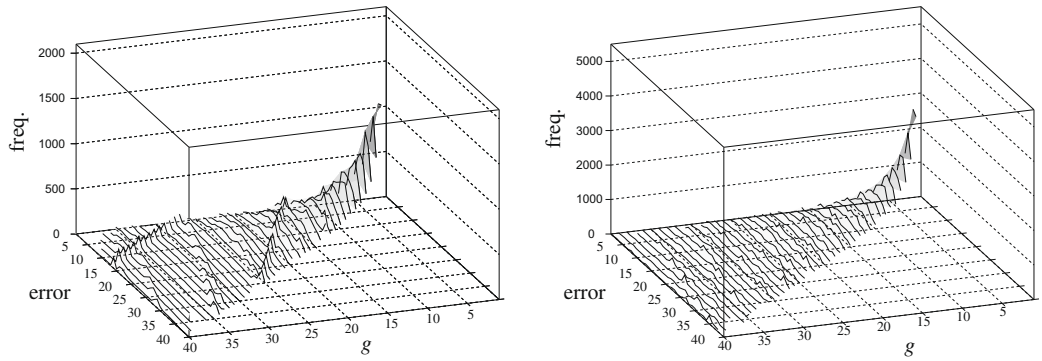


Fig. 15. Frequency of errors $(f(g,d_{\top}))$ plotted against the isothetic error $(d_{\top})$ and the grid size (g in [1,40]), corresponding to the images "spiral" (left) and "myth" (right).

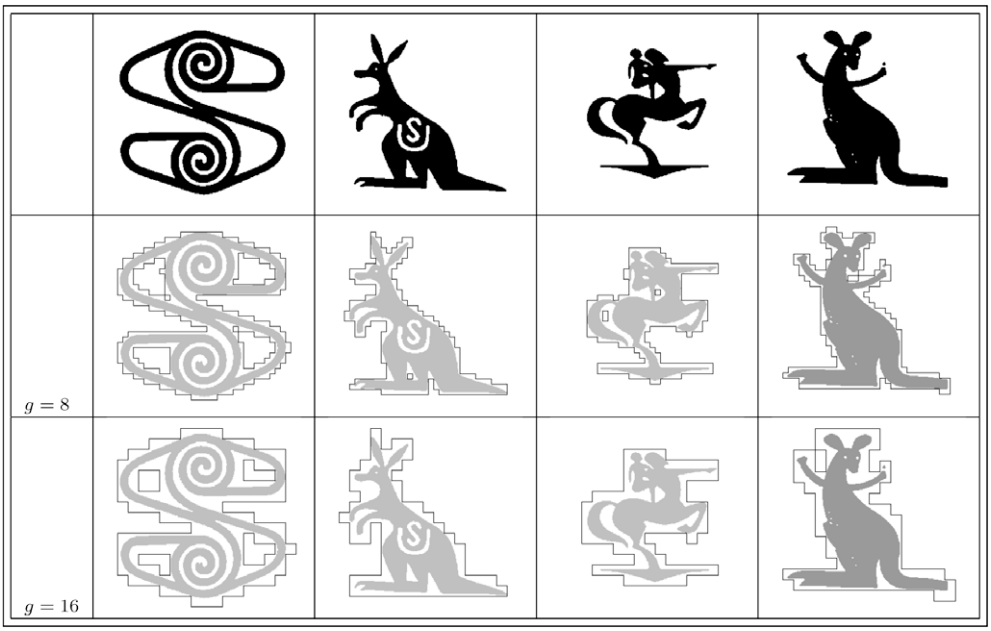


Fig. 16. The outer isothetic covers (OIC) for a set of logo images.

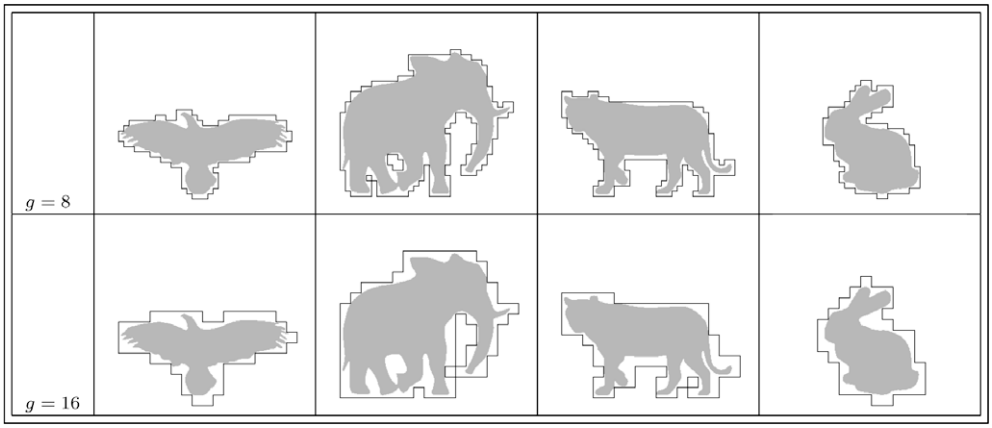


Fig. 17. The outer isothetic covers (OIC) for a set of animal images.

vertical, or inclined) of the Bengali character "kaw" is being conformed by the corresponding vertex pattern of its OIC in a more prominent way in the optical set than in the handwritten set. It is also interesting to note that the OIC of the handwritten "kaw" consists of a hole polygon inside the main polygon — a fact supporting the results corresponding to the optical "kaw". The optical

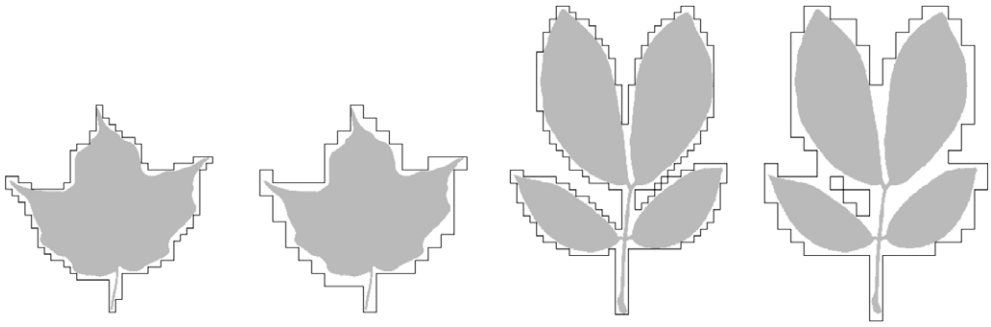


Fig. 18. The outer isothetic covers (OIC) for two leaf images for $g = 8$ (first one) and $g = 16$ (second one).

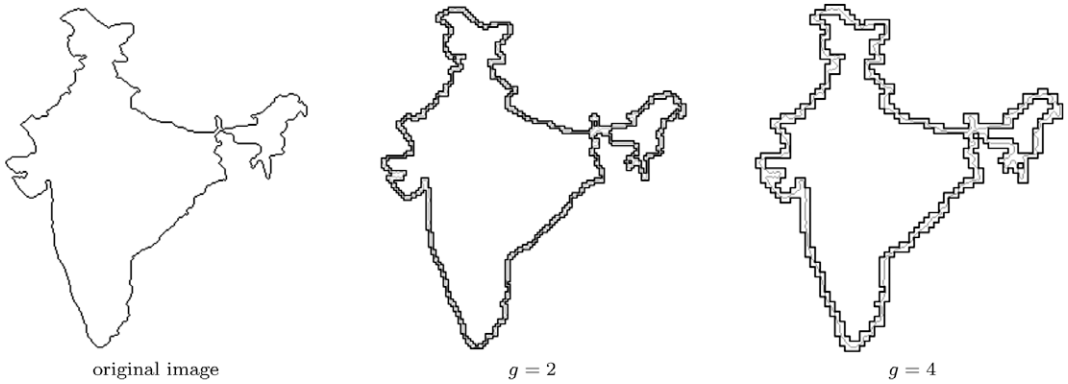


Fig. 19. OICs corresponding to the image "India" for $g = 2$ and $g = 4$.

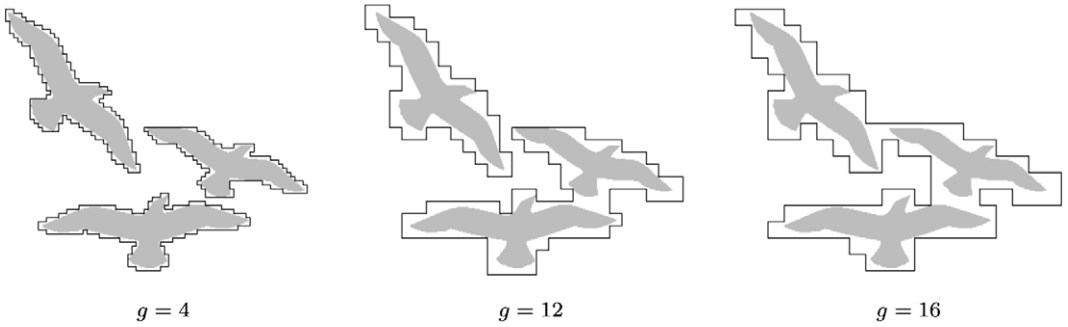


Fig. 20. Numbers of loosely connected components are 3, 2, and 1 for $g = 4$, 12, and 16, respectively.

"kaw" has a pseudo-hole polygon in excess, which is, however, much smaller in size, and hence might play a negligible role in the similarity measure.

Similar results on few English characters and numerals are presented in Fig. 22, and they demonstrate how the OICs capture the structural information of English characters. From these results, it is apparent that the covers of optical and handwritten characters can be used to design an OIC-based system for optical or handwritten character recognition (OCR/HCR).

Another area in which the OICs of handwritten characters can be used, is ranking the character-prototypes in a large database, which has been recently studied [36]. Recognizing a handwritten character involves matching the character with the prototypes or their features — a complex procedure with a large overhead. The process can be made faster if we can arrange the prototypes in some order, using their OICs, so that the recognition time may be reduced. For a given character, a subset of the prototypes, which are almost similar in their OIC-related information, may be replaced by a smaller subset depending on the trade-off between speed and precision. Similarly, to include a new prototype in the database, we can consider its degree of dissimilarity with the existing prototypes corresponding to the concerned character in the database, and take the decision on including that prototype, accordingly.

7.4. Scanned document images

The proposed algorithm can be used for document image segmentation also. To find the OIC of a document image, the object occupancy of an UGB is decided by checking all the pixels constituting the corresponding UGB. The text lines, which are sufficiently close along the vertical direction, would lie within a single OIC. If two consecutive paragraphs lie in the same OIC for a particular grid size, say $g = g_1$, then these paragraphs can be separated into two different OICs by reducing the grid size appropriately, say to $g = g_2 < g_1$. If the inter-paragraph spacing of a document image

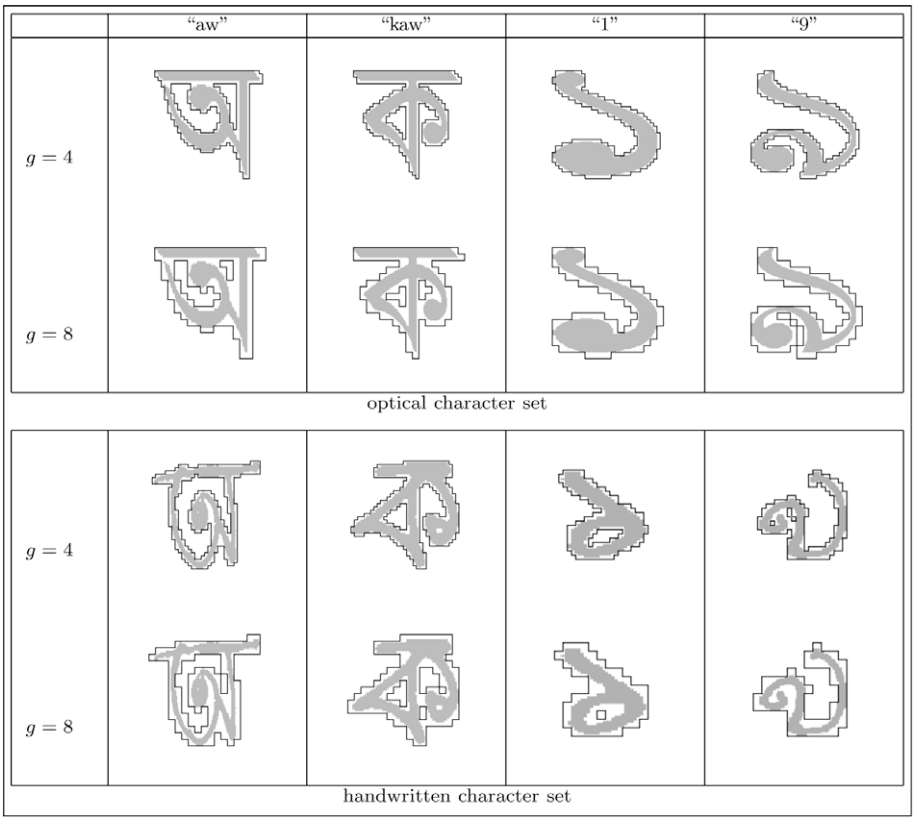


Fig. 21. The OICs of a few Bengali characters (optical and handwritten) for two different grid sizes.

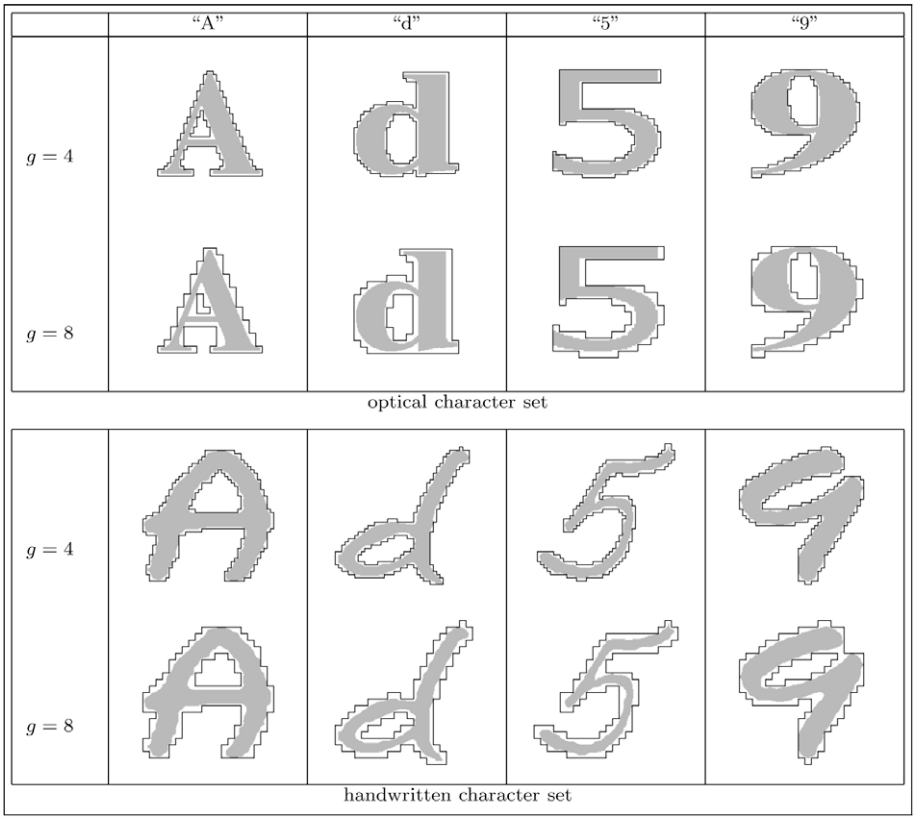


Fig. 22. The OICs of a few English characters (optical and handwritten) for two different grid sizes.

is same as its inter-line spacing, then they can be separated by observing that a paragraph starts with a fixed indentation from the left. A document page can thus be segmented into its paragraphs. Similarly, from a segmented paragraph, we can segment its lines by decreasing the grid size further; from a line, we can extract its words; and so forth. Other document features like mathematical equations, figures and graphical objects, tabular structures, etc., can also be extracted from the image using an appropriate analysis of the OICs.

In Fig. 23, the OICs (in yellow) are shown corresponding to each letter for grid size $g = 1$. Some of the letters are touching each other; in such a situation, one single OIC is derived (shown in blue). For $g = 5$, all the words are segregated. Thus, with the help of a multi-resolution treatment, different parts of a document page can be segmented using its different sets of OICs.

7.5. Inner isothetic cover

Fig. 24 shows the IIC of a logo image for different grid sizes. For grid size $g = 1$ or $g = 2$, the IIC is a single polygon; for $g = 4$, there are three polygons tightly inscribing the object; whereas, for $g = 6$, it is again only one polygon. For $g = 8$, the IIC comprises a total of four polygons. It may be noted that the fractional part of the object lying outside the IIC increases with the increase of the grid size.

As per the definition, an OIC should consist of minimum number of UGBs whose union contains the object without intersecting it, and an IIC should consist of maximum number of UGBs of the object without intersecting the background. Fig. 24 shows how an IIC consists of maximum number of UGBs belonging to the object. The difference of the interior of the IIC from the OIC gives an isothetic region in which the object boundary lies, which has been already illustrated in Fig. 1.

8. Concluding remarks

We have shown how the minimum-(maximum-)area outer (inner) isothetic cover of a digital object can be constructed corresponding to a given grid. The algorithm proposed here does not require any backtracking, and hence is an output-sensitive algorithm. The time complexity is linear on the length of the perimeter

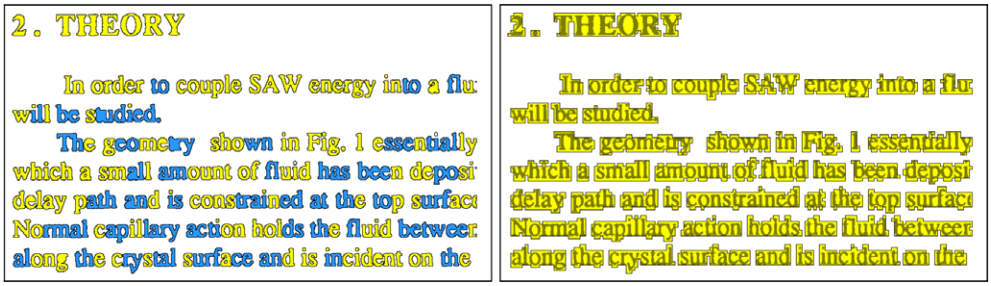


Fig. 23. The OICs of a part of a document image for grid sizes $g = 1$ (left) and $g = 5$ (right) show their abilities in extracting the letters and the words from a document image.

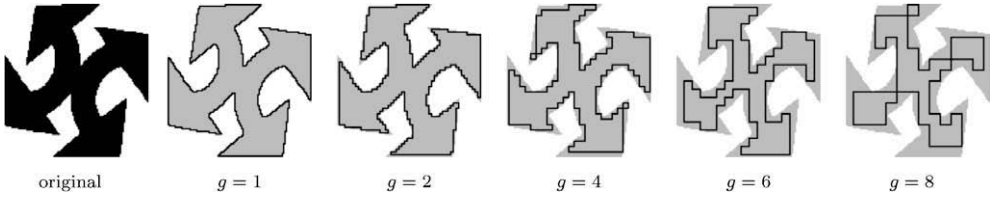


Fig. 24. The IICs of a logo image corresponding to different grid sizes.

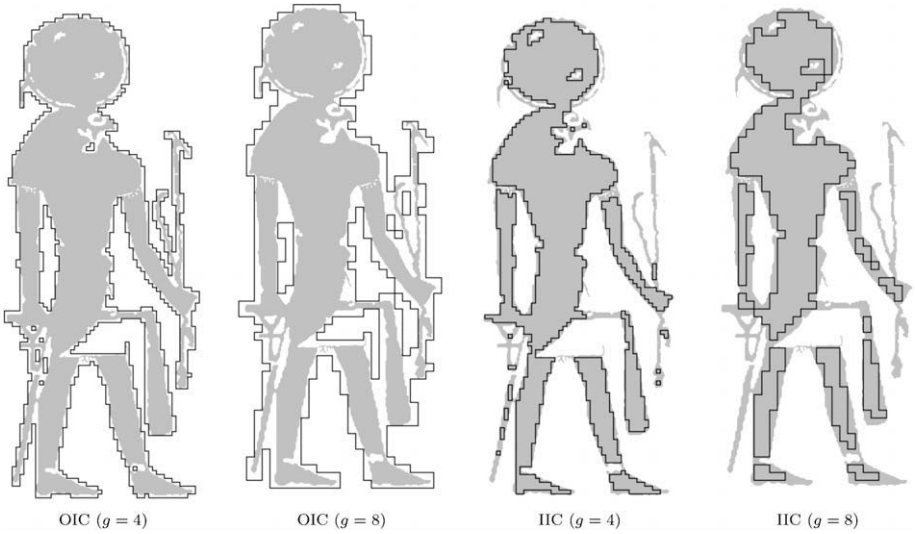


Fig. 25. OICs and IICs corresponding to the image "myth" for $g = 4$ and $g = 20$.

of the cover measured in grid units. Experimental results on various databases justify the efficiency of the algorithm and demonstrate potential applications.

Several open problems may arise in context to an isothetic cover of a digital object. It is evident that such a cover of a digital object depends on how the object is positioned or oriented with respect to the underlying grid. A challenging problem is, therefore, finding the object-grid registration for which the complexity of an isothetic cover is minimum. The complexity measure may be in terms of the number of vertices, or the perimeter, or the area of the isothetic cover. Another interesting problem is, given a collection of isothetic covers corresponding to different positions or orientations of the object for a grid size, to design an algorithm to (approximately) reconstruct the original object. The algorithm can be extended to higher dimensions for modeling 3D objects. Also, the algorithm when extended to the background of nonuniform grid for higher dimensions can be useful in determining the upper and lower approximations in rough sets.

There are several applications of isothetic polygons, which have been mentioned in Section 3 with some results shown in Section 7. One such application is document image analysis, where there still remains ample scope of exploiting isothetic polygons for solving related problems. Extension of the proposed algorithm to 3- and to higher dimensional digital space needs further investigation, which is currently in our research purview.

References

[1] S.C. Nandy, B.B. Bhattacharya, A. Barrera, Safety zone problem, J. Algorithms 37 (2000) 538–569.

[2] F. Preparata, M. Shamos, Computational Geometry, Springer-Verlag, New York, 1985.

[3] A. Bemporad, C. Filippi, F.D. Torrisi, Inner and outer approximations of polytopes using boxes, Comput. Geom. Theory Appl. 27 (2004) 151–178.

[4] J. Lengyel, M. Reichert, B.R. Donald, D.P. Greenberg, Real-time robot motion planning using rasterizing computer, Comput. Graph. ACM 24 (4) (1990) 327–335.

[5] L. Gatrell, CAD-based grasp synthesis utilizing polygons, edges and vertices, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1989, pp. 184–189.

[6] Y. Kamon, T. Flash, S. Edelman, Learning to grasp using visual information, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1995, pp. 2470–2476.

[7] A. Morales, P.J. Sanz, Á.P. del Pobil, Vision-based computation of three-finger grasps on unknown planar objects, in: IEEE International Conference on Intelligent Robots and Systems, 2002, pp. 1711–1716.

[8] S.K. Pal, P. Mitra, Case generation using rough sets with fuzzy representation, IEEE Trans. Knowledge Data Eng. 16 (3) (2004) 292–300.

[9] S.K. Pal, P. Mitra, Pattern Recognition Algorithms for Data Mining, Chapman and Hall/CRC Press, FL, 2004.

[10] J. Tadrat, V. Boonjing, P. Pattaraintakorn, A framework for using rough sets and formal concept analysis in case based reasoning, in: Proceedings of the IEEE International Conference on Information Reuse and Integration, 2007, pp. 227–232.

[11] W. Zhu, F. Wang, On three types of covering-based rough sets, IEEE Trans. Knowledge Data Eng. 19 (8) (2007) 1131–1144.

[12] M. Liu, Y. He, H. Hu, D. Yu, Dimension reduction based on rough set in image mining, in: International Conference on Computer and Information Technology (CIT'04), 2004, pp. 39–44.

[13] Peng-Yeng Yin, Shin-Huei Li, Content-based image retrieval using association rule mining with soft relevance feedback, J. Vis. Commun. Image Represent. 17 (5) (2006) 1108–1125.

[14] B. Gatos, S.L. Mantzaris, A novel recursive algorithm for area location using isothetic polygons, in: Proceedings of the International Conference on Pattern Recognition (ICPR), vol. 3, 2000, pp. 492–495.

[15] B. Gatos, S.L. Mantzaris, K.V. Chandrinos, A. Tsigris, S.J. Perantonis, Integrated algorithms for newspaper page decomposition and article tracking, in: Proceedings of the International Conference on Document Analysis and Recognition, 1999, pp. 559–562.

[16] A.K. Jain, M. Bin Yu, Document representation and its application to page decomposition, IEEE Trans. PAMI 20 (3) (1998) 294–308.

[17] Qixiang Ye, Jianbin Jiao, Jun Huang, Hua Yu, Text detection and restoration in natural scene images, J. Vis. Commun. Image Represent. 18 (6) (2007) 504–513.

[18] O.T. Akindele, A. Belaid, Page segmentation by segment tracing, in: Proceedings of the International Conference Document Analysis and Recognition (ICDAR), 1933, pp. 341–344.

[19] A. Antonacopoulos, H. Meng, A ground-truthing tool for layout analysis performance evaluation, in: Proceedings of the International Workshop Document Analysis Systems, LNCS, vol. 2423, 2002, pp. 236–244.

[20] S. Yacoub, V. Saxena, S.N. Sami, Perfectdoc: a ground truthing environment for complex documents, in: Proceedings of the Eight International Conference on Document Analysis and Recognition, vol. 1, 2005, pp. 452–456.

[21] A. Rosenfeld, A. Kak, Digital Picture Processing, Academic Press, New York, San Francisco, London, 1982.

[22] R. Klette, A. Rosenfeld, Digital geometry: geometric methods for digital image analysis, Morgan Kaufmann Series in Computer Graphics and Geometric Modeling, Morgan Kaufmann, 2004.

[23] J. Sklansky, Minimum-perimeter polygons of digitized silhouettes, IEEE Trans. Comput. 21 (3) (1972) 260–268.

[24] J. Sklansky, Measuring concavity on a rectangular mosaic, IEEE Trans. Comput. 21 (12) (1972) 1355–1364.

[25] F. Sloboda, B. Zat'ko, On boundary approximation, in: CAIP '95: Proceedings of the Sixth International Conference on Computer Analysis of Images and Patterns, 1995, pp. 488–495.

[26] G. Toussaint, Course Notes: Grids, Connectivity and Contour Tracing. Avaiable from: %3chttp://cgm.cs.mcgill.ca/godfried/teaching/pr-notes/contour.ps/%3e, 1997.

[27] J. Sklansky, A theory of non-uniformly digitized binary pictures, IEEE Trans. Systems Man Cybernet. 6 (9) (1976) 637–647.

[28] P. Bhowmick, A. Biswas, B.B. Bhattacharya, Isothetic polygons of a 2d object on generalized grid, in: Proceedings of the First International Conference on Pattern Recognition and Machine Intelligence (PReMI 2005), LNCS, vol. 3776, 2005, pp. 407–412.

[29] A. Biswas, P. Bhowmick, B.B. Bhattacharya, TIPS: on finding a tight isothetic polygonal shape covering a 2d object, in: Proceedings of the 14th Scandinavian Conference on Image Analysis (SCIA 2005), LNCS, vol. 3540, 2005, pp. 930–939.

[30] H. Freeman, Techniques for the digital computer analysis of chain-encoded arbitrary plane curves, in: Proceedings of the National Electronics Conference, vol. 17, 1961, pp. 421–432.

[31] A. Biswas, P. Bhowmick, B.B. Bhattacharya, MUSC: Shape codes Multigrid Shape Codes and their applications to image retrieval, in: International Conference on Computational Intelligence and Security, LNAI, vol. 3801, 2005, pp. 1057–1063.

[32] H. Freeman, On the encoding of arbitrary geometric configurations, IRE Trans. Electron. Comput. EC-10 (1961) 260–268.

[33] A. Biswas, P. Bhowmick, B.B. Bhattacharya, SCOPE: Shape Complexity of Objects using isothetic Polygonal Envelope, in: Proceedings of the Sixth International Conference on Advances in Pattern Recognition (ICAPR-2007), 2007, pp. 356–360.

[34] P. Bhowmick, A. Biswas, B.B. Bhattacharya, PACE: Polygonal Approximation of thick digital Curves using cellular Envelope, in: Proceedings of the Fifth Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP), LNCS, vol. 4338, 2006, pp. 299–310.

[35] P. Bhowmick, A. Biswas, B.B. Bhattacharya, DRILL: Detection and Representation of Isothetic Loosely connected components without Labeling, in: Proceedings of the Sixth International Conferences on Advances in Pattern Recognition (ICAPR), 2007, pp. 343–348.

[36] P. Bhowmick, A. Biswas, B.B. Bhattacharya, Ranking of optical character prototypes using cellular lengths, in: Proceedings of the International Conference on Computing: Theory and Applications (ICCTA), IEEE CS Press, 2007, pp. 422–426.