TIPS: On Finding a **T**ight **I**sothetic **P**olygonal **S**hape Covering a 2D Object

Arindam Biswas[1], Partha Bhowmick[1], and Bhargab B. Bhattacharya[2]

[1] Computer Science and Technology Department,
Bengal Engineering and Science University, Shibpur, Howrah, India
{abiswas, partha}@becs.ac.in
[2] Center for Soft Computing Research,
Indian Statistical Institute, Kolkata, India
bhargab@isical.ac.in

Abstract. The problem of constructing a tight isothetic outer (or inner) polygon covering an arbitrarily shaped 2D object on a background grid, is addressed in this paper, and a novel algorithm is proposed. Such covers have many applications to image mining, rough sets, computational geometry, and robotics. Designing efficient algorithms for these cover problems was an open problem in the literature. The elegance of the proposed algorithm lies in utilizing the inherent combinatoral properties of the relative arrangement of the object and the grid lines. The shape and the relative error of the polygonal cover can be controlled by changing the granularity of the grid. Experimental results on various complex objects with variable grid sizes have been reported to demonstrate the versatility, correctness, and speed of the algorithm.

1 Introduction

The problem of finding an optimal outer (or inner) polygonal envelope, imposed by isothetic grid lines, for an object, is a grave and critical one, and its solution can be useful to many interesting applications, such as image mining [5], grasping objects by a robot [2, 3, 6], deriving free configuration space (path-planner) for robot navigation [4], lower and upper approximations in rough sets [8, 9], VLSI layout design [7], etc.

The challenge of the problem lies in the fact that, at each grid point, a decision has to be made for the next path to be followed. It may so happen that a current decison based on the local arrangement, may eventually lead to a situation where no further advancements can be made. As a result, it may become mandatory to revoke the earlier decisions in order to backtrack and branch out iteratively until the final solution is found, which would incur excessive computational cost.

Proposed in this paper is a fast, efficient, and elegant algorithm for finding the optimal isothetic inner and outer polygons of an object, where the object can have any arbitrary shape. The elegance and novelty of our algorithm lies in the fact that it takes into account the spatial arrangement of the grid lines

H. Kalviainen et al. (Eds.): SCIA 2005, LNCS 3540, pp. 930–939, 2005.

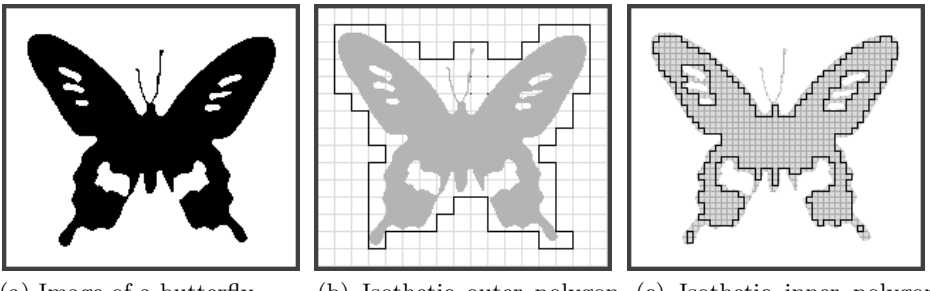
with respect to the object, and constructs the polygon to avoid backtracking completely. Further, this algorithm, with suitable modifications, will also work if non-uniform isothetic grid lines are imposed on the object plane.

It may be mentioned that, in contrast to the classical safety zone problem [7], which computes a minimum area safety region for an input polygon using the Minkowski sum, and also to the problem on inner and outer approximations of polytopes [1], which approximates a convex polytope by a collection of (hyper)boxes, the proposed algorithm can be applied on an arbitrarily shaped object/image, independent of whether or not it is a polygon. Since it works on isothetic grid lines, it can be used for VLSI design rule checking by adjusting the grid space as dictated by the minimum clearance zone required to be maintained. In the present form, this algorithm works on uniform grid spacing; however, this can be easily extended to non-uniform grid spacing which delineates the outer approximation of rough sets of very complex types. Futhermore, this algorithm, in a converse form, can extract the area-maximized isothetic inner polygon, thereby enabling the determination of inner approximation and boundary region of rough sets.

In this paper, after stating the problem definition in Sec. 2, the major steps of the algorithm to find the isothetic outer polygon (area-minimized) are narrated and explained in Sec. 3. In Sec. 4, we have shown experimental results on several objects. Since the algorithm of finding the inner polygon of an object will be very much similar to that of finding the outer one, we have not discussed the details of finding the inner polygon; however, results have been shown for both inner and outer polygons.

2 Problem Definition

Given a region (object) $\mathcal{R}$ defined in the two-dimensional real plane $\mathbf{R}^2$, and a set of uniformly spaced horizontal and vertical grid lines, $\mathcal{G} = (\mathcal{H}, \mathcal{V})$, where $\mathcal{H}$ and $\mathcal{V}$ represent two sets of equispaced horizontal and vertical grid lines respectively



(a) Image of a butterfly. (b) Isothetic outer polygon for grid size 12. (c) Isothetic inner polygon for grid size 4.

Fig. 1. A sample 2D object and its isothetic polygons

(uniform grid), the problem is to construct the corresponding isothetic polygonal envelope, $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$, such that the following conditions are satisfied:

(**c1**) $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$ should not have any self-intersection and should not contain any hole (although $\mathcal{R}$ may be self-intersecting and may contain holes);
(**c2**) no point $p \in \mathbf{R}^2$, lying in the region $\mathcal{R}$, should lie outside $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$;
(**c3**) each vertex of $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$ is the point of intersection of some line in $\mathcal{H}$ and some line in $\mathcal{V}$;
(**c4**) area of $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$ is minimized.

Before going into the discussion about the algorithm, a sample object $\mathcal{R}$, mapped to the 2D discrete plane, has been shown in Fig. 1(a), and the corresponding isothetic outer polygon, completely "containing" $\mathcal{R}$, has been displayed in Fig. 1(b). In Fig. 1(c), the entire set of isothetic inner polygons, that completely "fills' $\mathcal{R}$, has been shown.

3 Proposed Algorithm

Let $\mathcal{I}$ be the smallest two-dimensional image plane that contains the entire object $\mathcal{R}$. Let g be the underlying *grid size*, which is equal to the length (i.e. height or width) of each unit grid square in $\mathcal{G}$, defined over $\mathcal{I}$ (Fig. 2). It may be noted that the height h and the width w of the plane $\mathcal{I}$ are chosen appropriately to suit the requirement that g divides both h and w. Let $\alpha(i, j)$ be the point of intersection of the horizontal grid line $l_H : x = i$ and the vertical grid line $l_V : y = j$, where, $l_H \in \mathcal{H}$ and $l_V \in \mathcal{V}$. It may be observed that, since $\alpha(i, j)$ is a point on grid with grid size g, g always divides i and j, i.e., $i \ (\mathsf{mod}\ g) = 0$ and $j \ (\mathsf{mod}\ g) = 0$. Let $\mathcal{S}_{\mathrm{LT}}(i, j)$, $\mathcal{S}_{\mathrm{RT}}(i, j)$, $\mathcal{S}_{\mathrm{LB}}(i, j)$, and $\mathcal{S}_{\mathrm{RB}}(i, j)$ represent the four unit grid squares surrounding the common grid node α, and lying at the left-top, right-top, left-bottom, and right-bottom block respectively, as shown in Fig. 2.

We define a function φ to construct a binary matrix M_e (*unit edge matrix*) that stores the information regarding the intersection of each of the unit grid

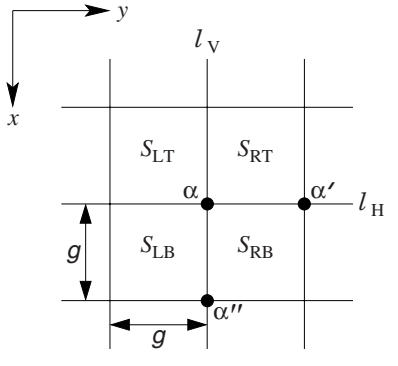


Fig. 2. Four unit grid squares with common vertex α

edges (of length g) with the object $\mathcal{R}$. It may be noted that the total number of unit horizontal edges is $\frac{w}{g}(\frac{h}{g}+1)$, and total number of unit vertical edges is $\frac{h}{g}(\frac{w}{g}+1)$, which togetherly decide the size of the unit edge matrix, M_e. Let $\alpha'(i, j+g)$ be the grid point lying immediate right of $\alpha(i,j)$, and $e(\alpha,\alpha')$ be the unit horizontal grid edge connecting α and α' (Fig. 2). Similarly, let $\alpha''(i+g, j)$ be the grid point lying immediate below $\alpha(i,j)$, and $e(\alpha,\alpha'')$ be the unit verctial grid edge connecting α and α''. Then the function φ, defined as follows, indicates the entry place, $\langle i_{e(\alpha,\beta)}, j_{e(\alpha,\beta)}\rangle$, in M_e where the binary information about the intersection of the unit edge $e(\alpha,\beta)$ with the object $\mathcal{R}$ is stored:

$$\varphi : e(\alpha,\beta) \mapsto \begin{cases} \langle 2i/g, j/g\rangle, \text{ if } \beta = \alpha'; \\ \langle i/g, 2j/g\rangle, \text{ if } \beta = \alpha''. \end{cases} \tag{1}$$

Depending on the intersection of the edge $e(\alpha,\beta)$ with the object $\mathcal{R}$, the corresponding entry $M_e[i_{e(\alpha,\beta)}][j_{e(\alpha,\beta)}]$ is decided as follows:

$$M_e[i_{e(\alpha,\beta)}][j_{e(\alpha,\beta)}] = \begin{cases} 1, \text{ if edge } e(\alpha,\beta) \text{ intersects } \mathcal{R}; \\ 0, \text{ otherwise.} \end{cases} \tag{2}$$

Now from the unit edge matrix, M_e, we construct another binary matrix, called the *unit square matrix*, M_s, having h/g rows and w/g columns, as follows.

$$M_s[i_s][j_s] = \begin{cases} 1, \text{ if } ((M_e[2i_s][j_s] \texttt{ OR} \\ \qquad\quad M_e[2i_s+1][j_s] \texttt{ OR} \\ \qquad\quad M_e[2i_s+1][j_s+1] \texttt{ OR} \\ \qquad\quad M_e[2i_s+2][j_s]) = 1) \\ 0, \qquad \text{otherwise} \end{cases} \tag{3}$$

It may be observed that, if an entry in the unit square matrix, M_s, is unity, then the corresponding unit square grid in $\mathcal{I}$ contains some part of the object $\mathcal{R}$. More precisely, if $M_s[i_s][j_s] = 1$, then the unit grid square, formed by the four grid lines (two horizontal and two vertical), namely $x = gi_s$, $x = g(i_s+1)$, $y = gj_s$, and $y = g(j_s+1)$, contains some part of $\mathcal{R}$. Furthermore, as evident from Eqn. 3, $M_s[i_s][j_s] = 1$ if and only if at least one among $M_e[2i_s][j_s]$, $M_e[2i_s+1][j_s]$, $M_e[2i_s+1][j_s+1]$, and $M_e[2i_s+2][j_s]$ is unity, since the object $\mathcal{R}$ must penetrate at least one of the four edges of the unit grid square in order that it lies inside the concerned square.

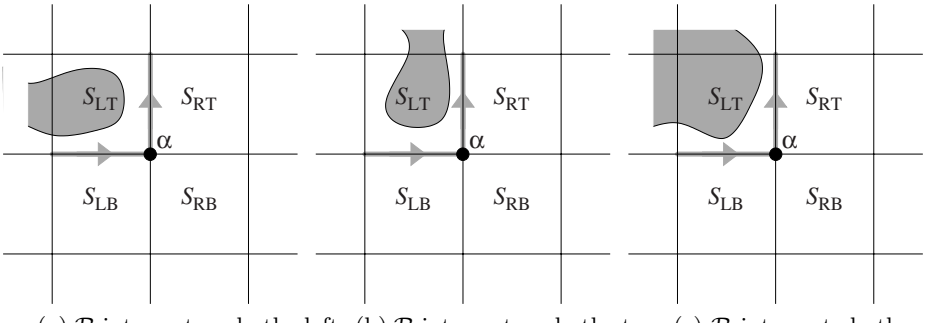
3.1 Finding the Vertices of $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$

In the proposed algorithm, the candidature of α as a vertex of $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$ is checked by looking at the combinatorial arrangements (w.r.t. object containments) of the four unit grid squares having common vertex α (Fig. 2). It may be noted that, each of these four unit squares has a binary entry at the corresponding locations in the *unit square matrix*, M_s, which, in turn, is derived from Eqns. 1, 2, 3, as discussed above. These four entries together form a 2×2 submatrix in M_s. Now, there exist $2^4 = 16$ different arrangements of these 4 unit grid squares, since each square have 2 possibilities ('0'/'1'). These 16 arrangements

have been further reduced to 5 cases in this algorithm, where, a particular case $\mathbf{C}_q$, $q = 0, 1, \ldots, 4$, includes all the arrangements where exactly q out of these 4 squares have object containments (i.e. contain parts of the object $\mathcal{R}$), and the remaining (i.e. $4 - q$) ones have not. That is, the case in which the sum of the 4 bits in the corresponding entries in M_s is equal to q is represented by $\mathbf{C}_q$. Further, out of these 5 cases, only two cases, namely $\mathbf{C}_1$ and $\mathbf{C}_3$, always represent vertices of $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$, and one case, namely $\mathbf{C}_2$, may conditionally give rise to a vertex of $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$, as discussed below.

Case $\mathbf{C}_1$:
In this case, exactly one of the four unit grid squares contains some part of the object $\mathcal{R}$. W.l.g., let $\mathcal{S}_{\mathrm{LT}}(i, j)$ be the unit grid square that contains some part of $\mathcal{R}$, as shown in Fig. 3. Hence the isothetic envelope, $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$, will have its one horizontal edge ending at α and the next vertical edge starting from α, if we consider traversal along the edges of $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$ in a way such that the region/object $\mathcal{R}$ always lies to the left of each edge while being traversed (shown by the respective arrows in Fig. 3). This argument holds for each of the 4 arrangements where exactly one of the corresponding four binary entries in M_s is unity and the remaining three is zero. This observation leads to the fact that α is a 90^0 vertex (i.e. a vertex with 90^0 internal angle) of $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$, if and only if $q = 1$.

Case $\mathbf{C}_2$:
Here, exactly two of the four unit grid squares contain parts of $\mathcal{R}$. If the two unit grid squares, having object containments, do not have any unit grid edge in common, only then α will be vertex of $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$. It is easy to observe that, in case $\mathbf{C}_2$, only two different arrangements are possible for which α is a vertex;



(a) $\mathcal{R}$ intersects only the left edge of $\mathcal{S}_{\mathrm{LT}}$.

(b) $\mathcal{R}$ intersects only the top edge of $\mathcal{S}_{\mathrm{LT}}$.

(c) $\mathcal{R}$ intersects both the left edge and the top edge of $\mathcal{S}_{\mathrm{LT}}$.

Fig. 3. Three possible instances for one of the 4 arrangements of case $\mathbf{C}_1$, where α is a 90^0 vertex. The edges (right edge and bottom edge of $\mathcal{S}_{\mathrm{LT}}$), which would belong to $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$, have been highlighted and directed to show their directions of traversal with $\mathcal{R}$ always lying at the left

these are: (i) $\mathcal{S}_{\mathrm{RT}}(i,j)$ and $\mathcal{S}_{\mathrm{LB}}(i,j)$ contain object parts, and (ii) $\mathcal{S}_{\mathrm{LT}}(i,j)$ and $\mathcal{S}_{\mathrm{RB}}(i,j)$ contain object parts. An instance of arrangement (i) is shown in Fig. 4. It can be shown that, in each of these two arrangements, α becomes a 270^0 vertex, since $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$ is considered to be non-self-intersecting, as stated in condition (**c1**) in Sec. 2. It should be carefully observed in Fig. 4 that two different styles of arrow heads indicate the two possibilities of formation of edges of $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$, since α may be visited along either of the two possible paths during construction of $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$, and only the traced one is included in the final solution of $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$.

For all other (four) arrangements with $q = 2$, α is just an ordinary point lying on some edge of $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$.

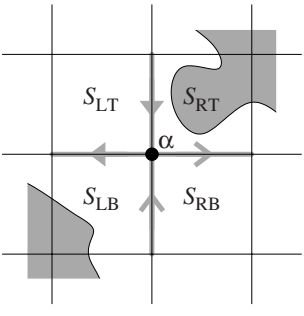


Fig. 4. An instance of one of the 2 arrangements of case $\mathbf{C}_2$, where α is a 270^0 vertex

Case $\mathbf{C}_3$:
If $q = 3$, then out of the four unit grid squares, only one square is free, which will have 4 different arrangements. In each of these arrangements, one of which is shown in Fig. 5, α would be a 270^0 vertex (i.e. a vertex with 270^0 internal angle).

For the two other cases, namely case $\mathbf{C}_0$ and case $\mathbf{C}_4$, it can be proved that α can never be a vertex of $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$. Case $\mathbf{C}_0$ implies that α is just an ordinary grid point that lies in $\mathcal{I} \setminus \mathcal{R}$, and can be shown to by lying outside $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$. Case $\mathbf{C}_4$ indicates that α is a grid point that either lies in $\mathcal{R}$, or lies in a hole of $\mathcal{R}$ and is surrounded by parts of $\mathcal{R}$ in all four unit grid squares with common vertex α, whence α can be shown to be a grid point lying inside $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$.

3.2 Storing the Vertices of $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$

It is easy to see that there are two types of vertices of the isothetic polygon, $\mathcal{P}_{out}(\mathcal{R},\mathcal{G})$: 90^0 vertex (obtained from case $\mathbf{C}_1$), and 270^0 vertex (obtained from case $\mathbf{C}_2$ and case $\mathbf{C}_3$), whose nature are discussed in Sec. 3.1. During the process of extraction of these vertices, each of them is dynamically inserted simultaneously in two temporary link lists, L_x and L_y, such that L_x always remains

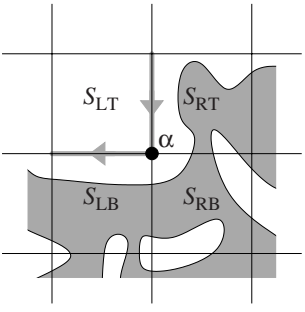


Fig. 5. A typical instance of one of the 4 arrangements of case $\mathbf{C}_3$, where α is a 270^0 vertex

lexicographically sorted in an increasing order w.r.t. x (primary key) and y (secondary key), and L_y always lexicographically sorted in an increasing order w.r.t. y (primary key) and x (secondary key). In addition to the grid coordinates of these vertices, one bit (*type-bit*) is stored for each vertex v in L_x and in L_y to denote its type ('0' denotes a 90^0 vertex and '1' denotes a 270^0 vertex). Furthermore, when the first 90^0 vertex ($v_1^{(0)}$) is detected, the way in which its two edges should be traversed, such that the object $\mathcal{R}$ lies left during traversal, is decided and stored accordingly. After extraction of all the vertices, the link lists L_x and L_y are processed, starting from any one of the 90^0 vertices, in order to construct the isothetic polygonal envelope, $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$, as discussed in Sec. 3.3.

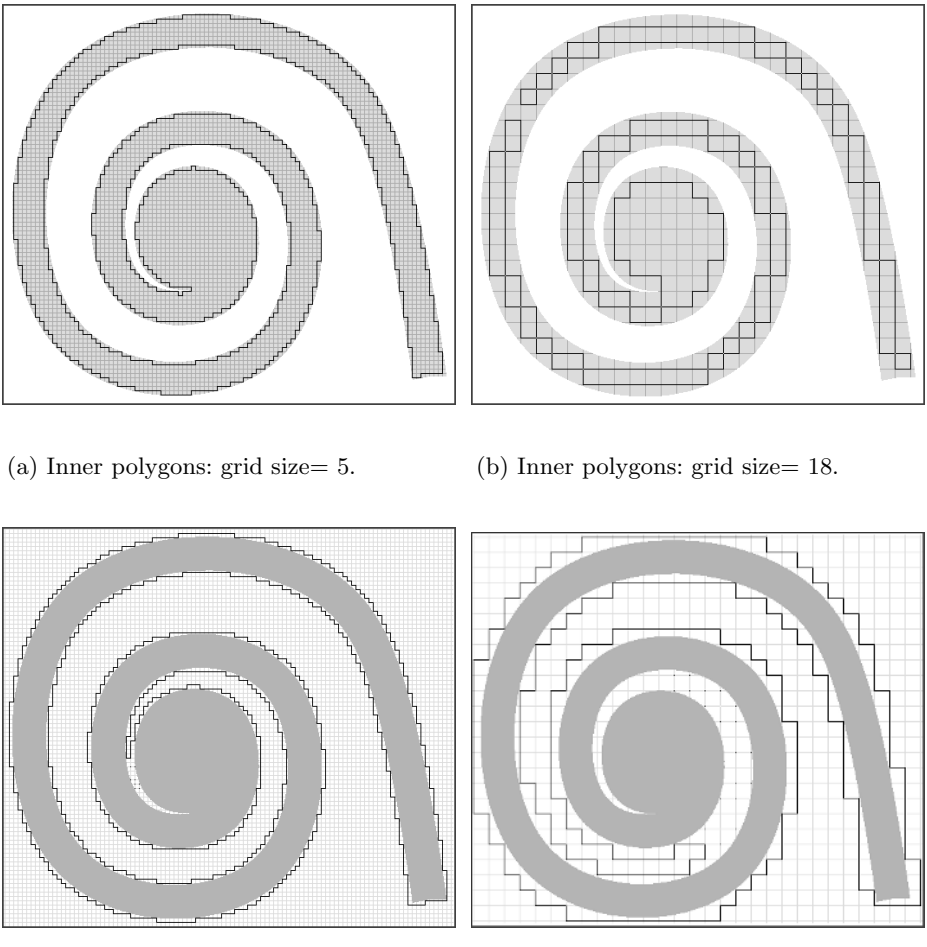
3.3 Construction of $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$

The construction of the isothetic polygonal envelope, $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$, starts from $v_1^{(0)}$, which is the start vertex, as discussed in Sec. 3.2. It may be noted that, considering any 270^0 vertex as a start vertex is risky, since that vertex may be derived as a result of case $\mathbf{C}_2$, which has a dubious nature, as discussed in Sec. 3.1 and illustrated in Fig. 4. Further, if, for example, the isothetic envelope is merely a rectangle, then there exist 90^0 vertices and there does not exist any 270^0 vertex. Hence a vertex with internal angle 90^0 should always be considered as a start vertex.

Now, if the outgoing edge from $v_1^{(0)}$ is vertical and directed towards top (as shown in Fig. 3, then the preceding vertex in the list L_x is the next vertex, v_{next}, of $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$, since the points in L_x are ordered w.r.t. x. Similarly, if the outgoing edge from $v_1^{(0)}$ is vertical and directed towards bottom, then the succeeding vertex in the list L_x becomes the next vertex, v_{next}, of $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$. For the other two possible arrangements, the decisions are similarly taken, and v_{next} is obtained using L_y. Once v_{next} is found, then depending on the type-bit ('0' or '1') of v_{next}, the outgoing edge from v_{next} is decisively selected using the rule that $\mathcal{R}$ always lies left during traversal, and the process continues until the start vertex $v_1^{(0)}$ is reached (after traversing the incident edge of $v_1^{(0)}$ as the last edge of $\mathcal{P}_{out}(\mathcal{R}, \mathcal{G})$).

4 Results

The above algorithm for constructing the outer polygon of any object in a 2D real plane has been tested on several objects of various shapes and sizes in 2D discrete plane. The algortihm requires slight modification for finding the inner polygons. By definition (Sec. 2), for a single object, we will always have a single outer polygon. However, if holes and self-intersections of outer polygon are allowed, the algorithm can be modified to produce the desired results. In the case of inner polygons, a single polygon may not be tight, i.e., complete "fill" the given object. Hence, we have allowed multiple inner polygons, whenever necessary. The CPU times on two typical sample images (shown in Figs. 6 and 7) have been given in



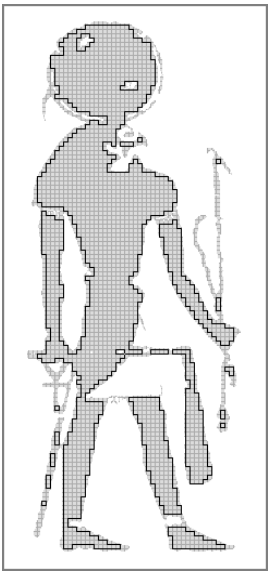
(a) Inner polygons: grid size= 5.

(b) Inner polygons: grid size= 18.

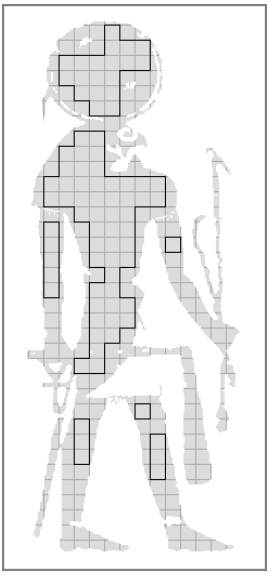
(c) Inner polygons: grid size= 5.

(d) Outer polygon: grid size= 18.

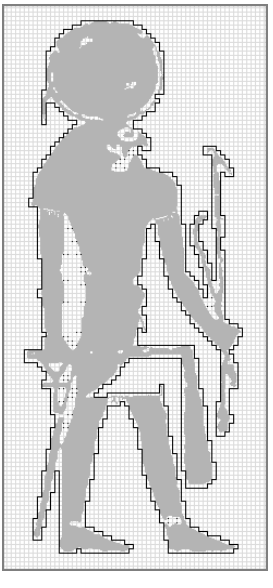
Fig. 6. Inner and outer polygons of a spiral



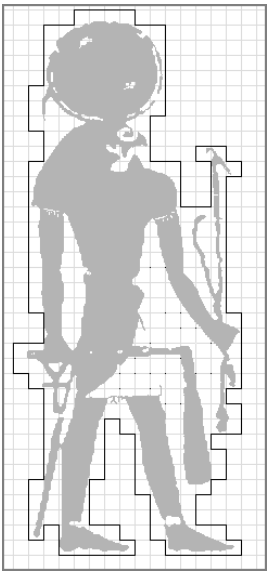
(a) Inner polygons: grid size= 4.



(b) Inner polygons: grid size= 14.



(a) Outer polygon: grid size= 4.



(b) Outer polygon: grid size= 14.

Fig. 7. Inner and outer polygons of a mythological figure

Table 1. CPU times in millisecs. for objects given in Figs. 6 and 7 for various grid sizes

Grid Size	CPU Time			
	spiral		myth. fig.	
	Inner	Outer	Inner	Outer
1	895	687	1285	451
2	235	201	280	203
4	64	62	67	40
8	15	20	13	15
16	5	9	4	9

Table 1 for varying grid sizes. From Table 1, it is evident that, there is a sharp decrease in CPU time with the increase in grid size.

5 Conclusion and Future Works

The proposed algorithm has been tested on several 2D objects in discrete domain, and has produced fast successful results in all cases. The proof of correctness of the algorithm is under preparation and will be reported in a subsequent paper. The algorithm will be tested on a non-uniform grid as a future work. Further, extension of the algorithm to 3D objects is also under our consideration.

References

1. A. Bemporad, C. Filippi, and F. D. Torrisi, *Inner and outer approximations of polytopes using boxes*, Computational Geometry - Theory and Applications, Vol. 27, (2004) 151–178
2. L. Gatrell, *Cad-based grasp synthesis utilizing polygons, edges and vertices*, Proc. IEEE Intl. Conf. Robotics and Automation (1989) 184–189
3. Y. Kamon, T. Flash, and S. Edelman, *Learning to grasp using visual information*, Proc. IEEE Intl. Conf. Robotics and Automation (1995) 2470–2476
4. J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg, *Real-time Robot Motion Planning Using Rasterizing Computer*, Computer Graphics, ACM, Vol. 24(4), (1990) 327–335
5. M. Liu, Y. He, H. Hu, and D. Yu, *Dimension Reduction Based on Rough Set in Image Mining*, Intl. Conf. on Computer and Information Technology (CIT'04) (2004) 39–44
6. A. Morales, P. J. Sanz, and Á. P. del Pobil, *Vision-Based Computation of Three-Finger Grasps on Unknown Planar Objects*, IEEE Intl. Conf. on Intelligent Robots and Systems (2002) 1711–1716
7. S. C. Nandy and B. B. Bhattacharya, *Safety Zone Problem*, Journal of Algorithms, Vol. 37 (2000) 538–569
8. S. K. Pal and P. Mitra, *Case Generation Using Rough Sets with Fuzzy Representation*, IEEE Trans. on Knowledge and Data Engg., Vol. 16(3), (2004) 292–300
9. S. K. Pal and P. Mitra, *Pattern Recognition Algorithms for Data Mining*, Chapman and Hall/CRC Press, Bocan Raton, FL (2004)