



A linear time combinatorial algorithm to compute the relative orthogonal convex hull of digital objects



Md A.A.A. Aman^a, Apurba Sarkar^a, Mousumi Dutt^b, Arindam Biswas^{c,*}

^a Department of Computer Science and Technology, Indian Institute of Engineering Science and Technology, Shibpur, India

^b Department of Computer Science and Engineering, Kolkata, India

^c Department of Information Technology, Indian Institute of Engineering Science and Technology, Shibpur, India

ARTICLE INFO

Article history:

Received 3 August 2019

Received in revised form 15 September 2020

Accepted 28 September 2020

Available online 15 October 2020

Keywords:

Isothetic covers

Convex hull

Relative convex hull

Orthogonal convex hull

Relative orthogonal convex hull

ABSTRACT

A linear time combinatorial algorithm to construct the relative orthogonal convex hull of the inner isothetic cover of a digital object (hole-free) with respect to its outer isothetic cover is presented in this paper. The algorithm first constructs the inner and the outer isothetic covers of the digital object which is the input to the algorithm and then constructs the relative orthogonal convex hull from the inner isothetic cover and the outer isothetic cover. The algorithm can also be used to construct the relative orthogonal convex hull of one orthogonal polygon with respect to another orthogonal polygon which encloses the first polygon entirely where the input will be two orthogonal polygons one containing the other instead of the digital object. The proposed algorithm first finds the concavities of the inner polygon and checks for intruded concavities of the outer polygon. A combinatorial technique is used to obtain the relative orthogonal convex hull. The efficiency and the correctness of the algorithm are verified as it is tested on various types of digital objects.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

The convex hull of a set of points S in the Euclidean plane is the intersection of all half-planes that contain it. It is denoted by $CH(S)$. There is a rich set of algorithms that compute the convex hull of a set of points in the literature. In 1972, Ronald Graham [5] proposed an algorithm to compute the convex hull of a set of n points in $\mathcal{O}(n \log n)$ time, popularly known as Graham Scan algorithm. In 1973, R.A. Jarvis proposed another algorithm for computing the convex hull of a given set of points that runs in $\mathcal{O}(nh)$ time, where n is the cardinality of the set and h is the number of points on the convex hull [7]. This algorithm is familiar in the literature as Jarvis March algorithm. By combining best of both Jarvis March's gift wrapping and Graham Scan algorithm, T.M. Chan [4] proposed a divide and conquer based output sensitive algorithm which is faster than both of the algorithms. Chan's algorithm starts by partitioning the point set of size n into $\lceil n/m \rceil$ partitions, each of size at most m . Subsequently, it applies Graham's scan method to find out the convex hull of each partition and combines them to get the desired hull. Other than the point set and the parameter m , the algorithm also takes a parameter H , which is a guess on the number of convex hull vertices. The running time of the algorithm is $\mathcal{O}(n \log H)$.

Melkman [16] presented a simple on-line algorithm to compute the convex hull of any simple polyline with n vertices in $\mathcal{O}(n)$. However, F.P. Preparata [18] has shown that for any non-simple polyline, the construction takes $\mathcal{O}(n \log n)$ time.

* Corresponding author.

E-mail addresses: mdaman.iest@gmail.com (M.A.A. Aman), as.besu@gmail.com (A. Sarkar), duttmousumi@gmail.com (M. Dutt), barindam@gmail.com (A. Biswas).

<https://doi.org/10.1016/j.tcs.2020.09.043>

0304-3975/© 2020 Elsevier B.V. All rights reserved.

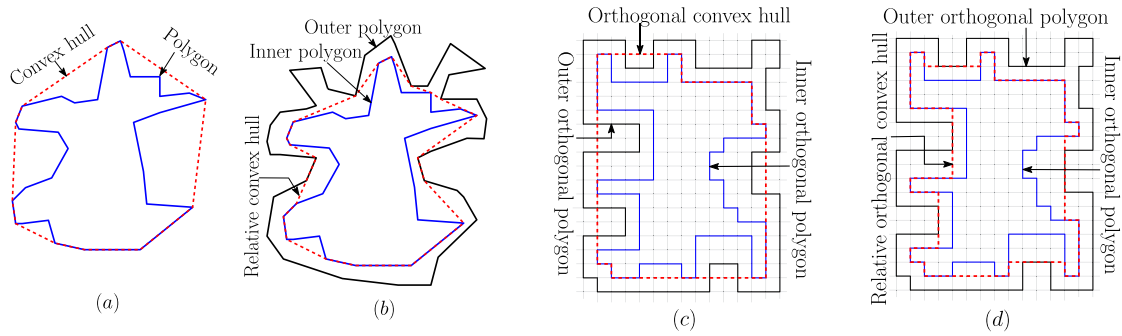


Fig. 1. (a) Euclidean convex hull, (b) Euclidean relative convex hull, (c) orthogonal convex hull, (d) relative orthogonal convex hull.

A. Biswas et al. proposed a linear time algorithm linear w.r.t. the number of pixels on the border of the given digital object for calculating the orthogonal convex hull (OCH) of an isothetic cover of a digital object [3]. The algorithm traverses along the outer isothetic cover of the digital object [1,2] and uses combinatorial reduction rules to get rid of concavities present in the outer isothetic cover of the digital object.

Studies on Relative Convex Hull (RCH) started in 1970. The notion of the relative convex hull was first proposed by Sklansky in [20] when digital imaging was still in its infancy. An optimal algorithm proposed by G.T. Toussaint computes relative convex hull of a set of n points in a polygon in $O(n \log n)$ time [24]. The relative convex hull of polygon A relative to a polygon B ($CH_B(A)$) coincides with Minimum Perimeter Polygon (MPP) and Minimum Length Polygon (MLP) [11,25] under the special condition that $A \subset B$. The Minimum Length Polygon (MLP) is a polygon whose frontier is the shortest Jordan curve among all Jordan curves which circumscribe polygon A but are contained in polygon B . Fig. 1(a), (b), (c) and (d) respectively shows the Euclidean convex hull, the Euclidean relative convex hull, the orthogonal convex hull and the relative orthogonal convex hull.

The relative convex hull finds many applications such as the problem of finding the shortest path in a constrained area which may be solved by computing the RCH. Relative convex hull can be useful for robot motion planning, network optimization, and geographical information processing [17]. There are several works discussing the application of RCH in digital image analysis in a 2D plane, surface area calculation in the 3D digital object, multi-grid convergent estimation of curve length and related structure based on the geodesic metric in [8,10,11,13–15,21,22,26]. There are some reported works on the relative convex hull in the literature. In [22,23], geometric properties of relative convex hull are discussed but no algorithm is given there. A linear-time algorithm for computing the minimum length polygon for constrained polygons is proposed by X. Provençal et al. [19]. The first dynamic algorithm for computing the MLP is presented in [12] by Lachaud et al. G. Klette proposed a recursive algorithm for calculating the relative convex hull in [9]. Whenever the algorithm encounters a concavity in the inner polygon it checks whether there is also any concavity in the outer polygon that goes inside the concavity of the inner polygon. If so then it finds out the intersections of these two concavities to find out the edges of the relative convex hull. The algorithm does the same thing recursively if there are nested concavities of an inner and outer polygon. The running time of the algorithm is quadratic in the worst-case scenario and linear in best case (i.e., if the depth of concavities of the inner polygon is small) with respect to the number of points on the boundary of the digital object. There is another work on the relative convex hull in semi-dynamic subdivisions by M. Ishaque et al. [6] which is based on sweep line but the sweep wavefront may have an arbitrary polygonal shape. The proposed algorithm allows insertions of $O(m)$ barriers and $O(n)$ site deletions with running time of $O((m+n) \text{ polylog}(mn))$ for m barriers and n sites and can answer convex hull queries in $O(\text{polylog}(mn))$ time. P. Wiederhold et al. [25] have recently proposed an algorithm for calculating the relative convex hull of a simple polygon. A brief summary of related work on convex hull and relative convex hull is presented in Table 1.

In this paper, we have presented an algorithm to find out the Relative Orthogonal Convex Hull (ROCH) of an orthogonal polygon with respect to another orthogonal polygon. The input to the algorithm is a simple digital object from which inner and outer isothetic covers are obtained (described in Sec. 2). The algorithm specifically considers the concavities of inner orthogonal polygon and the outer orthogonal polygon and uses reduction rules to obtain the ROCH. The proposed algorithm takes linear time to construct the ROCH w.r.t. the number of points on the boundary of the digital object.

The rest of the paper is organized as follows. The required definitions and preliminaries are given in Sec. 2. The combinatorial rules to find the relative orthogonal convex hull are described in Sec. 3. The algorithm, a demonstration, proof of correctness of the algorithm, and the time complexity are presented in Sec. 4. A discussion on experimental results is given in Sec. 5. Finally, the concluding remarks are given in Sec. 6.

2. Definitions and preliminaries

In this work we restrict ourselves to simple hole-free orthogonal polygons. Let us define few terminologies that are required to explain the proposed algorithm.

Table 1

Summary of related research work on Convex Hull (CH), Orthogonal Convex Hull (OCH), Relative Convex Hull (RCH) and Relative Orthogonal Convex Hull (ROCH).

Author(s)	Year	Types of hull	Techniques applied	Input type	Running time
R.L. Graham [5]	1972	CH	Sorts the points and applies linear search	Finite set of n points in the plane	$\mathcal{O}(n \log n)$
R.A. Jarvis [7]	1973	CH	Gift wrapping	Finite set of n points in the plane	Best case: $\mathcal{O}(nh)$ Worst case: $\mathcal{O}(n^2)$
F.P. Preparata [18]	1979	CH	Real-time algorithm with successive updates of data structure	Finite set of n points in the plane	$\mathcal{O}(n \log n)$
A. Melkman [16]	1987	CH	Ordered traversal of vertices	Simple polygon	$\mathcal{O}(n)$
T.M. Chan [4]	1993	CH	Combination of divide-conquer and gift wrapping	set of points	$\mathcal{O}(n \log H)$
A. Biswas et al. [3]	2012	OCH	Combinatorial	Simple (isothetic) polygon with n vertices	$\mathcal{O}(n \log n)$
G. Toussaint [24]	1986	RCH	Triangulation method	Simple polygon	$\mathcal{O}(n \log \log n)$, n being the number of vertices in the polygon
G. Klette [9]	2010	RCH	Recursive method	Two simple polygons A and B ($A \subset B$)	Best case: $\mathcal{O}(n + m)$ Worst case: $\mathcal{O}((n + m)^2)$, n and m being the vertices of the polygon A and polygon B respectively
M. Ishaque et al. [6]	2008	RCH	Polygonal sweep wavefront	Set of points	insertion/deletion: $\mathcal{O}(n + m) \text{polylog}(mn)$ query: $\mathcal{O}(\text{polylog}(mn))$, n being the sites and m being the barriers
P. Wiederhold et al. [25]	2015	RCH	Iterative method	Two simple polygons A and B ($A \subset B$)	Best case: $\mathcal{O}(n + m)$ Worst case: $\mathcal{O}((n + m)^2)$, n and m being the vertices of the polygon A and polygon B respectively
Our method	–	ROCH	Combinatorial method	Simple digital object	$\mathcal{O}(n_{iic} + n_{oic})$ where n_{iic} and n_{oic} are the vertices of IIC and OIC

Definition 1 (*Digital object*). A digital object (henceforth referred to as an object) is a finite subset of \mathbb{Z}^2 consisting of one or more k -connected components.

In this work we are considering only digital objects having a single 8-connected component.

Definition 2 (*Digital grid*). A digital grid (henceforth referred simply as a grid) $\mathcal{G} := (\mathcal{H}, \mathcal{V})$ consists of a set \mathcal{H} of horizontal (digital) grid lines and a set \mathcal{V} of vertical (digital) grid lines, where, $\mathcal{H} = \{\dots, l_H(j - 2g), l_H(j - g), l_H(j), l_H(j + g), l_H(j + 2g), \dots\} \subset \mathbb{Z}^2$ and $\mathcal{V} = \{\dots, l_V(i - 2g), l_V(i - g), l_V(i), l_V(i + g), l_V(i + 2g), \dots\} \subset \mathbb{Z}^2$, 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 [2].

A 8-connected digital object is shown in Fig. 2(a) in black where the background digital grid is 4-connected.

Definition 3 (*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 $s_H^j = \{(i', j) \in l_H(j) : i \leq i' \leq i + g\}$. Similarly, the vertical grid segment belonging to $l_V(i)$ between $l_H(j)$ and $l_H(j + g)$ is $s_V^i = \{(i, j') \in l_V(i) : j \leq j' \leq j + g\}$ [2]. Fig. 2(b) shows different grid segments.

Definition 4 (*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 $h_H^+(j) := \{(i', j') \in \mathbb{Z}^2 : j' \geq j\}$ and $h_H^-(j) := \{(i', j') \in \mathbb{Z}^2 : j' \leq j\}$. Similarly, $l_V(i)$ divides \mathbb{Z}^2 into two half-planes, which are $h_V^+(i) := \{(i', j') \in \mathbb{Z}^2 : i' \geq i\}$ and $h_V^-(i) := \{(i', j') \in \mathbb{Z}^2 : i' \leq i\}$. 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 $U_1 = UGB(i, j)$

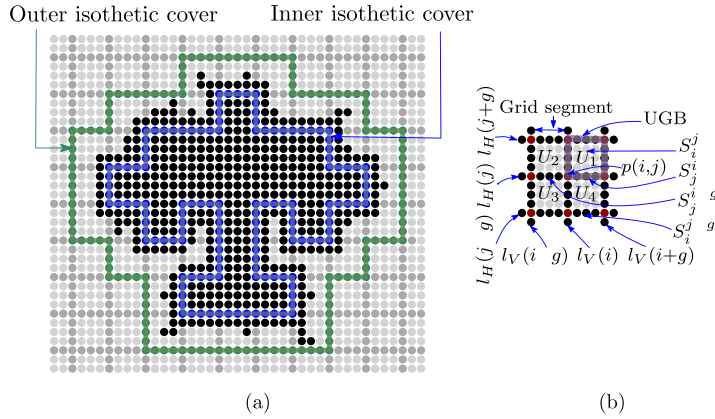


Fig. 2. (a) The digital object (black pixels), Outer isothetic cover (marked in green), and Inner isothetic cover (marked in blue) are shown. The borders of UGBs are marked in dark gray and other digital points (pixels) are marked in light gray. (b) The enlarged version of the four unit grid blocks (UGBs) is shown (one UGB is marked in pink) w.r.t. a grid point $p(i, j)$ where (i, j) are coordinates of p . The grid points are marked in dark red. Here the grid size is 4. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

is given by $U_1 \setminus (s_i^j \cup s_{i+g}^j \cup s_j^i \cup s_{j+g}^i)$ [2]. For a given grid point, $p(i, j)$, there are four neighboring UGBs, namely, $U_1 := UGB(i, j)$, $U_2 := UGB(i - g, j)$, $U_3 := UGB(i - g, j - g)$, and $U_4 := UGB(i, j - g)$ as shown in Fig. 2(b).

The grid segment and UGBs are shown in Fig. 2(b) for $g = 4$. The four UGBs are shown w.r.t. a grid point $p(i, j)$ where (i, j) are the coordinates of p .

Definition 5 (Isothetic polygon). An isothetic polygon P is a simple polygon (i.e., with non-intersecting sides) of finite size in \mathbb{Z}^2 whose alternate sides are subsets of the members of \mathcal{H} and \mathcal{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 BP_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 [2].

In this paper, the sides of polygons (considered here) are always either parallel to vertical or horizontal axes, so the terms orthogonal and isothetic are used interchangeably. The vertices of an isothetic polygon are grid points and the edges of the polygon are lying on grid lines.

Definition 6 (Outer isothetic cover). The outer (isothetic) cover (OIC) of a digital object Q , denoted by $\overline{P}(Q)$, is the minimum-area isothetic polygon covering Q .

Definition 7 (Inner isothetic cover). The inner (isothetic) cover (IIC) of a digital object Q , denoted by $\underline{P}(Q)$, is the maximum-area isothetic polygon inscribed in Q .

The outer (isothetic) cover (OIC) and the inner (isothetic) cover (IIC) of a digital object are also isothetic polygons shown in Fig. 2(a) (shown in green and blue polygons respectively). It is to be noted here that the input digital object is such that its OIC and IIC are hole-free and simple polygons having only one connected component.

Classification of vertices for deriving the outer isothetic cover (OIC). The algorithm to obtain outer isothetic cover produces an ordered list of vertices of $\overline{P}(Q)$ using a combinatorial classification of the grid points lying on/outside the object boundary. As shown in Fig. 2(b), there are four UGBs w.r.t. a grid point $p(i, j)$. Depending on the object occupancy in four UGBs w.r.t. $p(i, j)$, the grid point $p(i, j)$ is classified as follows.

1. None of the four UGBs of $p(i, j)$ has object occupancy (Fig. 3(a)).
2. Exactly one out of the four UGBs has object occupancy. There are four such possible arrangements (Fig. 3(b)). It is called *type 1 vertex*.
3. Exactly two out of the four UGBs have object occupancy. Two of these occupied UGBs can be adjacent (i.e., sharing a grid segment) (Fig. 3(c)) or they are not adjacent (i.e., diagonal) (Fig. 3(d)); and as a result there are two sub-cases as follows.
 - (a) There are four such arrangement depending on the common grid segment. The grid point is not a vertex, it is called *edge point* or *type 2* (Fig. 3(c)).
 - (b) There are two such arrangements either UGBs U_1 and U_3 have object occupancy or UGBs U_2 and U_4 have object occupancy. Such grid point is called *cross vertex* and it is considered as *type 3 vertex* (Fig. 3(d)).

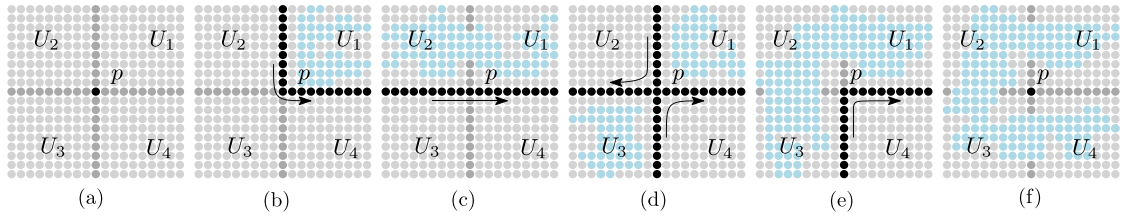


Fig. 3. Different types of vertices for outer isothetic cover. (a) The grid point, p , is outside the object, (b) p is a type 1 vertex, (c) p is not a vertex, it is an edge point, (d) p is a cross vertex (type 3 vertex), (e) p is a type 3 vertex, (f) p is an interior grid point not a vertex.

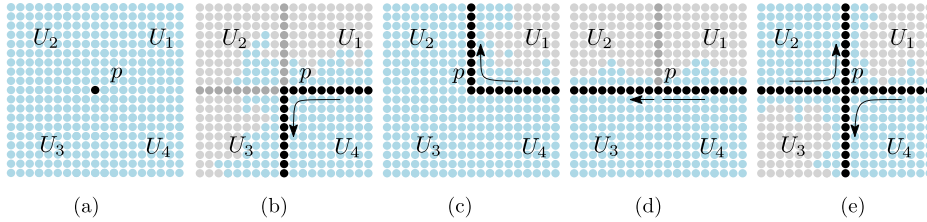


Fig. 4. Different types of vertices for inner isothetic cover. (a) p is an interior grid point not a vertex, (b) p is a type 1 vertex, (c) p is a type 3 vertex, (d) p is not a vertex, it is an edge point, (e) p is a cross vertex (type 3 vertex).

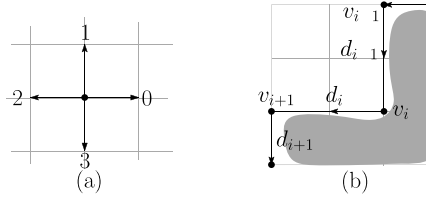


Fig. 5. (a) Direction codes and (b) direction w.r.t. vertices.

4. Three out of the four UGBs have object occupancy. Four such arrangements are possible in this class (Fig. 3(e)). This type of grid points is called *type 3 vertex*.
5. All four UGBs have object occupancy (Fig. 3(f)).

Classification of vertices for deriving the inner isothetic cover (IIC). The procedure for obtaining inner isothetic cover, $\underline{P}(Q)$, is same as the outer isothetic cover with one exception in the classification of the vertices. In case of OIC, it is checked whether an UGB contains at least one object pixel, whereas for IIC, it is checked whether an UGB is fully occupied with object pixels. When an UGB is fully contained with object pixel, then only it is considered as occupied with object for IIC. The classification of the vertices for inner isothetic cover is depicted in Fig. 4. When all four UGBs w.r.t. a grid point $p(i, j)$ are fully occupied, then $p(i, j)$ is not a vertex, a grid point is lying inside the object Fig. 4(a). The type 1, type 3, type 2, and cross vertex are shown in Fig. 4(b)-(e) respectively.

Procedure for obtaining OIC or IIC. The digital object is imposed on a background grid. The starting point of traversal is determined which is the top left grid point of the OIC and IIC (stated in [1,2]) and its type is considered as type 1 and direction as 3 (see Fig. 5(a)). The object is traversed anticlockwise. Let v_i be the current vertex and v_{i-1} be the previous vertex in $\underline{P}(Q)$ ($\underline{P}(Q)$), also let d_{i-1} be the direction of v_{i-1} and t_i be the type of v_i , then the next direction of traversal from v_i is obtained by the formula $d_i = (d_{i-1} + t_i) \bmod 4$, where $d_i \in [0, 3]$ indicating the direction along right, top, left, and downward respectively as shown in Fig. 5(a). The next grid point of traversal will be the point along d_i . This procedure continues until it reaches the start point, thereby concluding at the start point of the outer (inner) isothetic cover (see Fig. 5(b)). In Fig. 5(b), v_{i-1} is the start vertex which is type 1 and direction is 3.

Definition 8 (Concavity). A *concavity* in an isothetic polygon arises when there are two or more consecutive type 3 vertices. The edge between two consecutive type-3 vertices is called a *concave edge*.

The concavity with two consecutive type 3 vertices and more than two consecutive type 3 vertices are shown in Fig. 6(a), (b) respectively. The concave edges are $v_i v_{i+1}$ and $v_{i+1} v_{i+2}$.

Definition 9 (Convexity). A *convexity* in an isothetic polygon arises when there are two or more consecutive type 1 vertices. The edge between two consecutive type-1 vertices is called a *convex edge*.

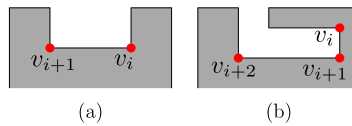


Fig. 6. a) Concavity with two consecutive type 3 vertices, b) Concavity with more than two consecutive type 3 vertices.

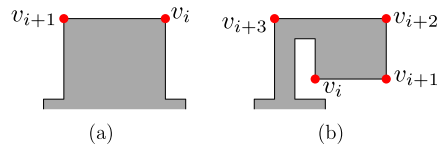


Fig. 7. a) Convexity with two consecutive type 1 vertices, b) Convexity with more than two consecutive type 1 vertices.

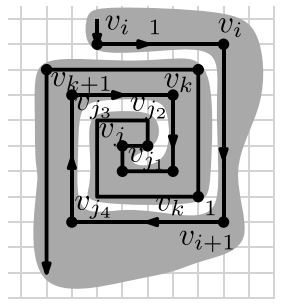


Fig. 8. The convoluted region.

The convexity with two consecutive type 1 vertices and more than two consecutive type 1 vertices are shown in Fig. 7(a), (b) respectively. The convex edges are $v_i v_{i+1}$ and $v_{i+1} v_{i+2}$.

Definition 10 (*Convoluting region*). When there are three or more consecutive type 3 vertices, it gives rise to a convoluted region.

The example of convoluted region is shown in Fig. 8. The consecutive type 3 vertices are $v_i \dots v_j$ followed by consecutive type 1 vertices ($v_{j_1} \dots v_{k+1}$) to get out of the convoluted region in Fig. 8.

Definition 11 (*Orthogonal hull*). An orthogonal convex hull of an orthogonal (isothetic) polygon P is the smallest area orthogonal polygon such that every grid point of P lies inside the hull and intersection of the hull with any horizontal or vertical line is either empty or a line segment.

A polygon is said to be convex if it does not contain any concavity. Let us consider two isothetic polygons A and B with vertices at the borders be $\{v_{A1}, v_{A2}, \dots, v_{An}\}$ and $\{v_{B1}, v_{B2}, \dots, v_{Bm}\}$ such that $A \subset B$. It is to be noted here that isothetic polygon contains the border pixels of the polygon along with its interior pixels.

Definition 12 (*Orthogonal B -convex polygon*). An orthogonal polygon A is orthogonally B -convex if and only if any straight line segment (either horizontal or vertical) in B that has both end points in A , is also contained in A . For example, the polygon A_1 in Fig. 9(a) is an orthogonal B -convex polygon and the polygon A_2 in Fig. 9(b) is not an orthogonal B -convex polygon.

Definition 13 (*Relative orthogonal convex hull*). The relative orthogonal convex hull of A relatively to B (in short, $ROCH_B(A)$ or simply $ROCH$) is the intersection of all orthogonal B -convex polygons containing A . For example, the dashed (red) polygon in Fig. 1(d) represents the relative orthogonal convex hull while the dashed polygon in Fig. 1(b) represents its Euclidean counterpart.

3. Obtaining the relative orthogonal convex hull

While constructing the inner isothetic cover (IIC) and the outer isothetic cover (OIC) of the digital object using the algorithm in [1,2], the lexicographically sorted lists of vertices for the two are prepared in the preprocessing steps. L_{nx}^{IIC} is

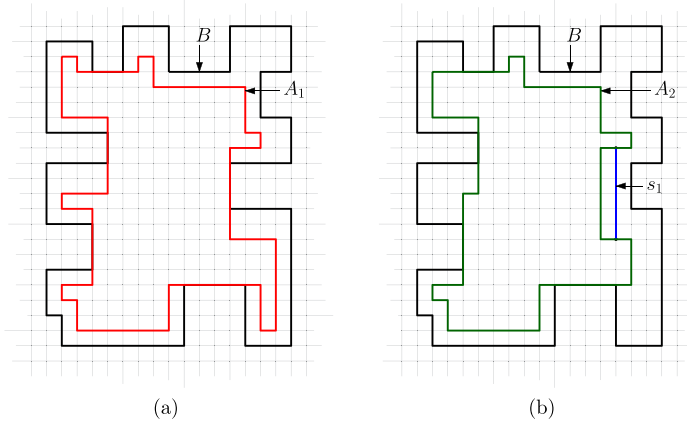


Fig. 9. (a) Orthogonal B-Convex polygon (marked in red). (b) An orthogonal polygon which is not B-Convex (marked in green).

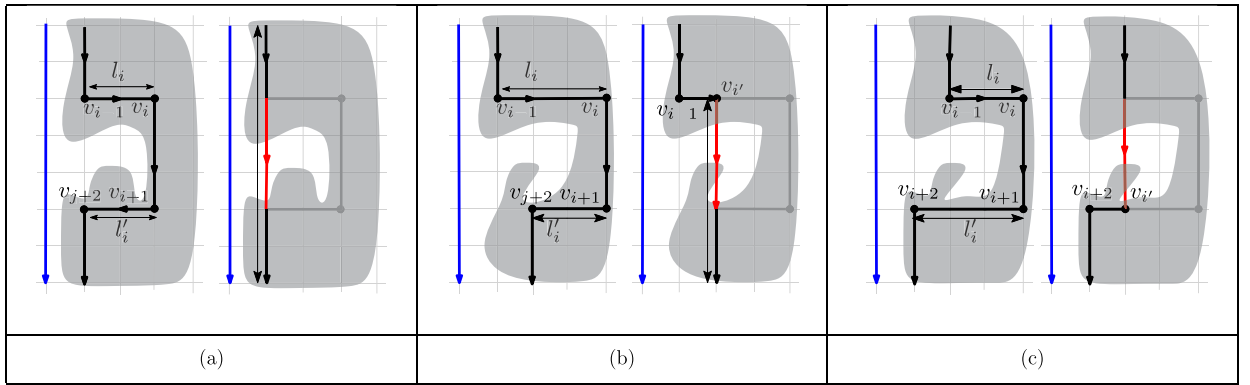


Fig. 10. Illustration of Reduction Rule R1.

lexicographically sorted lists for IIC with x as primary key and y as secondary key and L_{py}^{iic} is lexicographically sorted lists for IIC with y as primary key and x as secondary key. L_{px}^{oic} is lexicographically sorted lists for OIC with x as primary key and y as secondary key and L_{py}^{oic} is lexicographically sorted lists for OIC with y as primary key and x as secondary key.

To obtain the relative orthogonal convex hull (ROCH), the inner isothetic cover (IIC) is traversed from the top-left corner in anticlockwise manner. During the traversal, whenever a concavity is detected, it is checked which rules can be applied. The rules are discussed in details in Sec. 3.1. The rules are applied to obtain the ROCH and they reduce the number of vertices of inner isothetic cover. The rules are termed as *reduction rules*.

3.1. Reduction rules

The reduction rules are applied whenever there is a concavity in the inner isothetic cover (IIC). The concavity comprises of either two consecutive type 3 vertices or more than two consecutive type 3 vertices. It may also happen that a concavity consisting of either two consecutive type 3 vertices or more than two consecutive type 3 vertices in OIC may be intruded in the concavity of IIC. Based on these the three rules are formulated which are discussed as follows.

Rule R1. This rule is applied when there are two consecutive type 3 vertices in IIC and no intruded concavity in OIC. The three instances of this case are depicted in Fig. 10. Let the two type 3 vertices in IIC be v_i and v_{i+1} . Let l_i and l'_i be the length of the edges of $v_{i-1}v_i$ and $v_{i+1}v_{i+2}$. The three instances of this rule will be $l_i = l'_i$, $l_i > l'_i$, $l_i < l'_i$ (shown in Fig. 10(a), (b), (c) respectively). When $l_i = l'_i$, the vertices v_{i-1} and v_{i+2} are joined using a straight line (Fig. 10(a)) and in the resulting ROCH the four vertices $v_{i-1}, v_i, v_{i+1}, v_{i+2}$ are deleted. When $l_i > l'_i$, a new vertex v'_i is created on the edge $v_{i-1}v_i$ and the three vertices v_i, v_{i+1}, v_{i+2} are deleted. An edge is incident on v_{i+2} from v'_i in the resulting ROCH as shown in Fig. 10(b). When $l_i < l'_i$, a new vertex v'_i is created on the edge $v_{i+1}v_{i+2}$ and the three vertices v_{i-1}, v_i, v_{i+1} are deleted. An edge is incident on v_{i-1} from v'_i in the resulting ROCH as shown in Fig. 10(c).

Rule R2. This rule is applied when there is a concavity with more than two type 3 vertices in IIC and no concavity of OIC is intruded. The four instances of this case are depicted in Fig. 11. Let the type 3 vertices in IIC be v_i, \dots, v_j which gives rise to a convoluted region. The edge $(v_i v_{i+1})$ corresponding to the first two consecutive type 3 vertices may be horizontal or vertical. The edge $v_{k-1}v_k$ in the convoluted region is such edge whose distance from $v_i v_{i+1}$ is minimum w.r.t. all other

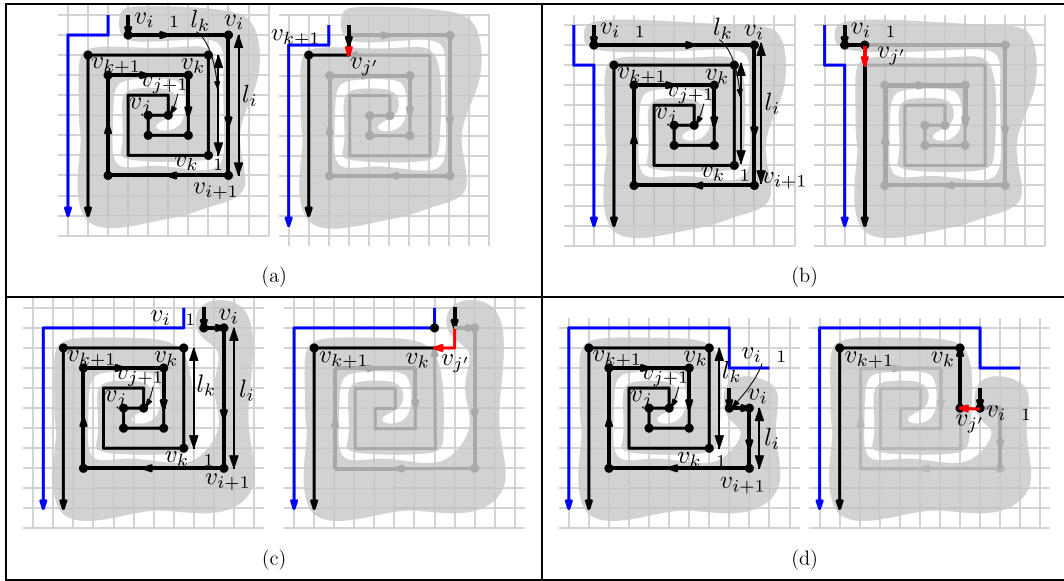


Fig. 11. Illustration of Reduction Rule R2.

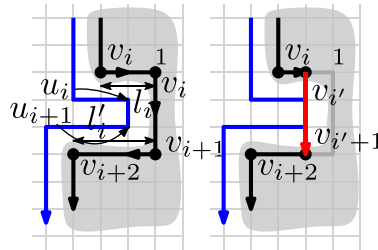


Fig. 12. Illustration of Reduction Rule R3.

edges (from v_{i+2} to v_k) in the convoluted region. It is to be noted that when the edge $v_i v_{i+1}$ is vertical (horizontal) all vertical (horizontal) edges are considered in the convoluted region. This minimum distance edge, $v_{k-1} v_k$, is termed here as “nearest edge” of $v_i v_{i+1}$. The distance of all vertical (horizontal) edges followed by v_j having range within $v_i v_{i+1}$ and having direction opposite to $v_i v_{i+1}$ are measured from $v_i v_{i+1}$ to determine the nearest edge. In Fig. 8, $v_i v_{i+1}$ is vertical. Thus the distance of all the vertical edges from v_{j_1} in the convoluted region is measured from $v_i v_{i+1}$ to find the nearest edge. It is to be noted that after the vertex v_{k+1} , the edge range is beyond the position of $v_i v_{i+1}$. Thus $v_{k-1} v_k$ is the nearest edge w.r.t. $v_i v_{i+1}$. It is explained in steps 2-6 in the Procedure APPLY-R2. When the edge, $v_{k-1} v_k$, is fully contained within the edge, $v_i v_{i+1}$, i.e., $l_k < l_i$, there are three cases as shown in Fig. 11(a), (b), (c). Here, the resulting ROCH depends on the position of the edge $v_k v_{k+1}$ w.r.t. the vertex v_{i-1} . Accordingly, a new vertex, $v_{j'}$, is inserted and the required vertices are deleted. In Fig. 11(a), the position of v_{i-1} is between the vertices v_k and v_{k+1} . The vertex v_{i-1} lies at left w.r.t. the edge $v_k v_{k+1}$ in Fig. 11(b). The vertex v_{i-1} lies at right w.r.t. the edge $v_k v_{k+1}$ in Fig. 11(c). Otherwise, the position of v_{i-1} is checked w.r.t. the edge $v_{k-1} v_k$ and accordingly $v_{j'}$ is inserted and the required vertices are deleted as shown in Fig. 11(d).

Rule R3. This rule is applied when there is a concavity of OIC intruded in the concavity in IIC as shown in Fig. 12. The concave edge $v_i v_{i+1}$ of IIC is reduced in such a way that it passes through the intruded concavity in OIC. To find the intruded concavity in OIC the lexicographically sorted lists are used. When the concave line is vertical L_{px}^{oic} is searched, otherwise L_{py}^{oic} is searched. For the concavity ($v_i v_{i+1}$) shown in the Fig. 12, L_{px}^{oic} is traversed until a vertical line in OIC is found ($u_i u_{i+1}$) which is totally contained in it. When the concavity consists of two or more consecutive type 3 vertices in IIC and there is intruded concavity of OIC, the Rule R3 is applied step by step. This instance is shown in Fig. 13 in step by step.

4. Algorithm to determine ROCH

The algorithm to obtain the relative orthogonal convex hull (ROCH) is discussed in Sec. 4.1. A demonstration of the algorithm is also proposed in Sec. 4.2. The proof of correctness is stated in Sec. 4.3. The time complexity is presented in Sec. 4.4.

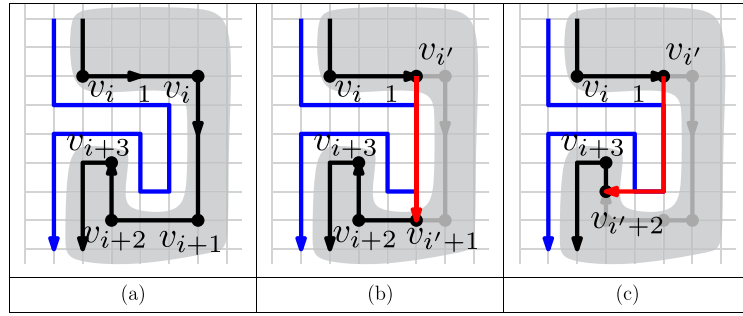


Fig. 13. Illustration of reduction rule R3 with intruded concavity having more than two type 3 vertices.

Algorithm 1: ROCH.

Input: Q, g
Output: L_R

- 1 $L_i, L_{p_x}^{iic}, L_{p_y}^{iic} \leftarrow \text{CONSTRUCT-IIC}(Q)$
- 2 $L_o, L_{p_x}^{oic}, L_{p_y}^{oic} \leftarrow \text{CONSTRUCT-OIC}(Q)$
- 3 \triangleright Let $L_i, L_{p_x}^{iic}, L_{p_y}^{iic}, L_o, L_{p_x}^{oic}, L_{p_y}^{oic}$ be globally accessible
- 4 $L_R \leftarrow \text{ADD}(L_R, v_0) \triangleright v_0$ be the start vertex of IIC
- 5 $v_i \leftarrow v_0.\text{next}, \text{flag} \leftarrow 0$
- 6 **while** $v_i \neq v_0$ **do**
- 7 **if** $\text{flag} = 0$ **then**
- 8 $L_R \leftarrow \text{ADD}(L_R, v_i)$
- 9 $\text{count} \leftarrow 1, v_{i+1} \leftarrow v_i.\text{next}$
- 10 **if** $v_i.\text{type} = 3$ **and** $v_{i+1}.\text{type} = 3$ **then**
- 11 $v_j \leftarrow v_{i+1}$
- 12 **while** $v_j.\text{type} = 3$ **do**
- 13 $\text{count} \leftarrow \text{count} + 1$
- 14 $v_j \leftarrow v_j.\text{next}$
- 15 $v_i, L_R \leftarrow \text{APPLYRULE}(v_i, v_j, \text{count}, L_R)$
- 16 $\text{flag} \leftarrow 1$
- 17 **else**
- 18 $\text{flag} \leftarrow 0, v_i \leftarrow v_i.\text{next}$
- 19 **return** L_R

4.1. Algorithm

The Algorithm 1: ROCH takes the digital object, Q , and the grid size, g as input and the list of vertices of ROCH, L_R , is the output of the algorithm. In the preprocessing steps, IIC and OIC are constructed and the vertex list along with lexicographically sorted lists are prepared in Steps 1, 2 respectively using the algorithm stated in [1,2]. All these lists are globally accessible, i.e., accessible through all the procedures (step 3 as comment). Let v_0 be the start vertex of IIC and it is inserted in L_R using the procedure ADD (Step 4). v_i is set as the following vertex of v_0 and a variable flag is initialized to '0' (Step 5). From steps 6-18, all the vertices of IIC are accessed from top-left corner in anticlockwise manner until the start vertex is reached. When the reduction rule is applied in the loop, flag is set to '1', otherwise it is set to '0'. Each vertex v_i is added to L_R (Step 7). The variable count is initialized to '1' to keep a track on the number of consecutive type 3 vertices (step 9). v_{i+1} is the following vertex of v_i in L_i (step 9). The number of consecutive type 3 vertices are counted in steps 10-14. v_j is set to the following vertex of v_i (Step 11). Thus the consecutive type 3 vertices can be represented as v_i, \dots, v_j . The consecutive type 3 vertices are accessed in the loop shown in steps 12-14. The procedure APPLYRULE is called in step 15 with the parameters $v_i, v_j, \text{count}, L_R$ and returns the updated L_R and the vertex v_i from which the traversal will start again. The flag is set to '1' in step 16 as reduction rule is applied, otherwise it is set to '0' and v_i is set to its following vertex in L_i (steps 17-18). Finally the list of vertices of ROCH in anticlockwise order starting from the top-left corner of A is returned in L_R (step 19).

In step 1 of the procedure APPLYRULE, it is checked whether any OIC concavity is intruded in the corresponding concavity ($v_i v_i.\text{next}$) in IIC using the procedure CHECK-OIC-CONCAVITY which returns two variables, val and L_c . The value of val is TRUE when there is intruded concavity in OIC, otherwise it returns FALSE. When val is TRUE, L_c contains the consecutive type 3 vertices of OIC for the intruded concavity, otherwise L_c is NULL. If val is FALSE and there are two consecutive type 3 vertices then Rule R1 is applied using the procedure APPLY-R1 (Step 3-4). If val is FALSE and there are more than two consecutive

Procedure APPLYRULE($v_i, v_j, count, L_R$).

```

1   $val, L_C \leftarrow \text{CHECK-OIC-CONCAVITY}(v_i, v_i.next)$ 
2  if  $val = \text{FALSE}$  then
3    if  $count = 2$  then
4       $v', L_R \leftarrow \text{APPLY-R1}(v_i, v_j, L_R)$ 
5    else
6       $v', L_R \leftarrow \text{APPLY-R2}(v_i, v_j, L_R)$ 
7  else
8     $v_{i'+1}, L_R \leftarrow \text{APPLY-R3}(v_i, v_i.next, L_C, L_R)$ 
9    if  $count > 2$  then
10      $v', L_R \leftarrow \text{APPLYRULE}(v_{i'+1}, v_j, count - 1, L_R)$ 
11 return  $v', L_R$ 

```

Procedure APPLY-R1(v_i, v_j, L_R).

```

1   $v_{i-1} \leftarrow v_i.prev, v_{j+1} \leftarrow v_j.next$ 
2  if  $\max \{|v_i.x - v_{i-1}.x|, |v_i.y - v_{i-1}.y|\} > \max \{|v_j.x - v_{j+1}.x|, |v_j.y - v_{j+1}.y|\}$  then
3     $v_{i'} \leftarrow \text{SETVERTEX-R1}(v_{i-1}, v_i, v_j, v_{j+1}, L_R)$ 
4     $v' \leftarrow v_{i-1}$ 
5     $L_R \leftarrow \text{DEL}(L_R, v_i)$ 
6     $L_R \leftarrow \text{ADD}(L_R, v_{i'})$ 
7  else if  $\max \{|v_i.x - v_{i-1}.x|, |v_i.y - v_{i-1}.y|\} = \max \{|v_j.x - v_{j+1}.x|, |v_j.y - v_{j+1}.y|\}$  then
8     $v' \leftarrow v_{i-1}.next$ 
9     $L_R \leftarrow \text{DEL}(L_R, v_{i-1})$ 
10    $L_R \leftarrow \text{DEL}(L_R, v_i)$ 
11 else
12    $v_{j'} \leftarrow \text{SETVERTEX-R1}(v_{i-1}, v_i, v_j, v_{j+1}, L_R)$ 
13    $v' \leftarrow v_{i-1}.next$ 
14    $L_R \leftarrow \text{DEL}(L_R, v_{i-1})$ 
15    $L_R \leftarrow \text{DEL}(L_R, v_i)$ 
16    $L_R \leftarrow \text{ADD}(L_R, v_{j'})$ 
17 return  $v', L_R$ 

```

Procedure APPLY-R2(v_i, v_j, L_R).

```

1   $v_{i-1} \leftarrow v_i.prev, v_{i+1} \leftarrow v_i.next$ 
2   $v_r \leftarrow v_j.next, dist \leftarrow \infty$ 
3  while  $v_i.d \neq v_r.d$  and  $v_{i+1}.y > v_r.y$  do
4     $dist \leftarrow \text{MIN}(dist, v_i.x - v_r.x)$ 
5     $v_{k-1} \leftarrow v_r, v_k \leftarrow v_r.next$ 
6     $v_r \leftarrow (v_r.next).next$ 
7  if  $(v_{k-1}).y \geq v_i.y$  then
8     $v_{k+1} \leftarrow v_k.next$ 
9    if  $v_{k+1}.x > v_{i-1}.x$  and  $v_k.x > v_{i-1}.x$  then
10      $v_{j'}.y \leftarrow v_{i-1}.y, v_{j'}.x \leftarrow v_k.x$ 
11      $v' \leftarrow v_{i-1}$ 
12  else
13      $v_{j'}.x \leftarrow v_{i-1}.x, v_{j'}.y \leftarrow v_k.y$ 
14      $v' \leftarrow v_{i-1}.prev$ 
15      $L_R \leftarrow \text{DEL}(L_R, v_{i-1})$ 
16 else
17    $v_{j'}.y \leftarrow v_{i-1}.y, v_{j'}.x \leftarrow v_k.x$ 
18    $v' \leftarrow v_{i-1}$ 
19  $L_R \leftarrow \text{DEL}(L_R, v_i)$ 
20  $L_R \leftarrow \text{ADD}(L_R, v_{j'})$ 
21 return  $v', L_R$ 

```

type 3 vertices then Rule R2 is applied using the procedure APPLY-R2 (Step 5-6). If *val* is TRUE then Rule R3 is applied using the procedure APPLY-R3 for the first two consecutive type 3 vertices (Step 8). If there are more than two consecutive type 3 vertices then the procedure APPLYRULE is called recursively by reducing the value of *count* by '1' and for the consecutive type 3 vertices $v_{i'+1}, \dots, v_j$ (Steps 9-10). After applying each of the rules, the list L_R is updated. v' denotes the position of the vertex from which next traversal will start. In step 11, the updated L_R is returned.

Procedure APPLY-R3(v_i, v_{i+1}, L_C, L_R).

```

1  $v_{i'}, v_{i'+1} \leftarrow \text{SETVERTEX-R3}(v_i, v_{i+1}, L_C, L_R)$ 
2  $L_R \leftarrow \text{DEL}(L_R, v_i)$ 
3  $L_R \leftarrow \text{ADD}(L_R, v_{i'})$ 
4  $L_R \leftarrow \text{ADD}(L_R, v_{i'+1})$ 
5 return  $v_{i'+1}, L_R$ 

```

The procedure APPLY-R1 uses reduction Rule R1 (see Sec. 3.1 for details). v_{i-1} is the previous vertex of v_i and v_{j+1} is the following vertex of v_j in L_i (step 1). It checks the length of the edges $v_{i-1}v_i$ and v_jv_{j+1} which are l_i and l'_i respectively. In step 2, it is checked whether $l_i > l'_i$ (Fig. 10(b)). Accordingly, a new vertex $v_{i'}$ is introduced by calling the procedure SETVERTEX-R1 (step 3). In step 4, v' is set to v_{i-1} from where the next traversal will start. The vertex, v_i , is deleted using the procedure DEL from L_R (step 5) and $v_{i'}$ is added using the procedure ADD (step 6) in L_R . In step 7, it is checked whether $l_i = l'_i$ (Fig. 10(a)). In step 8, v' is set to the following vertex of v_{i-1} in L_i from where the next traversal will start. The vertices v_{i-1} and v_i are deleted using the procedure DEL from L_R in steps 9-10. In step 11, it is checked whether $l_i < l'_i$ (Fig. 10(c)). In step 12, $v_{j'}$ is set using the procedure SETVERTEX-R1. In step 13, v' is set to the following vertex of v_{i-1} in L_i from where the next traversal will start. The vertices v_{i-1} and v_i are deleted using the procedure DEL from L_R in steps 14-15 and $v_{j'}$ is added using the procedure ADD (step 16) in L_R . The updated L_R is returned in step 17.

The procedure APPLY-R2 applies Rule R2 (see Fig. 11). It is to be noted here that the procedure given here is for the concavity having the direction of the first type 3 vertex '3'. For the concavities in other directions, the procedure will be slightly modified due to the different orientations of the concavities. The procedure takes the first two consecutive type 3 vertices, v_i, v_{i+1} , of IIC and the list of vertices of ROCH, L_R , as input. v_{i-1} is set to the previous vertex of v_i and v_{i+1} is set to the following vertex of v_i in L_i (step 1). In step 2, v_r is set to the following vertex of v_j and a variable, *dist*, is set to ∞ to keep track of nearest distance. While the direction of v_i and v_r are opposite to each other and $v_{i+1}.y$ is greater than $v_r.y$ (step 3), the distance between the edges $v_i v_{i+1}$ and $v_r v_r.next$ is measured and minimum distance value is set to *dist* (step 4). v_{k-1} is set to v_r and v_k is set to $v_r.next$ in step 5 and next v_r is set to $(v_r.next).next$ in step 6. Thus in steps 3-6, the nearest edge $v_{k-1}v_k$ is determined. In step 7, it is checked whether the edge, $v_k v_{k-1}$, is fully contained within the edge $v_i v_{i+1}$. Then the solution depends on the position of v_{k-1} w.r.t. the edge $v_{i-1}v_i$ (steps 8-15) as shown in Fig. 11(a)-(c). In step 8, v_{k+1} is set to $v_k.next$. If the edge $v_k v_{k+1}$ is fully contained in the edge $v_{i-1}v_i$ as shown in Fig. 11(b) (step 9), the solution is stated in steps 10-11. A new vertex $v_{j'}$ is set in step 10. v' is set to v_{i-1} (step 11). If the edge $v_k v_{k+1}$ is not fully contained in the edge $v_{i-1}v_i$ as shown in Fig. 11(a), (c) (step 12), the solution is stated in steps 13-15. A new vertex $v_{j'}$ is set in step 13. v' is set to the previous vertex of v_{i-1} in L_i (step 14). The vertex v_{i-1} is deleted from L_R using the procedure DEL in step 15. When the edge, $v_k v_{k-1}$, is not fully contained within the edge $v_i v_{i+1}$ (step 16), the solution is stated in steps 17-18. A new vertex $v_{j'}$ is set in step 17. v' is set to v_{i-1} (step 18). The vertex v_i is deleted from v_i using the procedure DEL in step 19 and the vertex $v_{j'}$ is added in L_R using the procedure ADD in step 20. The vertex v' and L_R are returned in step 21.

The procedure APPLY-R3 applies Rule R3 (see Fig. 12 and Fig. 13). The two new vertices, $v_{i'}$ and $v_{i'+1}$, have to be introduced as explained in Sec. 3.1 (Fig. 12) using the procedure SETVERTEX-R3 in step 1. The vertex v_i is deleted from L_R using the procedure DEL in step 2. The two vertices, $v_{i'}$ and $v_{i'+1}$, are added to L_R using the procedure ADD in steps 3-4. The vertex, $v_{i'+1}$, and updated L_R are returned.

4.2. Demonstration of the algorithm

In Fig. 14(a), the digital object is shown in gray color and its OIC and IIC are shown in black and blue color. The OIC has thirty four vertices (u_1 to u_{34}) and IIC has forty two vertices (v_1 to v_{42}). L_{px}^{iic} is the sequence of vertices as $\{v_1 v_2 v_{21} v_{22} v_{42} v_{41} v_9 v_{10} v_4 v_3 v_{19} v_{20} v_7 v_8 v_{16} v_{15} v_{39} v_{40} v_5 v_6 v_{12} v_{11} v_{18} v_{17} v_{23} v_{24} v_{13} v_{14} v_{38} v_{37} v_{13} v_{14} v_{28} v_{27} v_{36} v_{35} v_{31} v_{32} v_{34} v_{33} v_{26} v_{25}\}$. L_{py}^{iic} is $\{v_1 v_{42} v_{39} v_{38} v_{41} v_{40} v_{37} v_{36} v_4 v_5 v_{35} v_{34} v_2 v_3 v_7 v_6 v_{28} v_{29} v_9 v_8 v_{12} v_{13} v_{31} v_{30} v_{10} v_{11} v_{32} v_{33} v_{19} v_{18} v_{16} v_{17} v_{21} v_{20} v_{15} v_{14} v_{27} v_{26} v_{22} v_{23} v_{24} v_{25}\}$. Similarly, the lists L_{px}^{oic} and L_{py}^{oic} are prepared. The IIC is traversed from top left vertex (L_i is accessed). The first consecutive type 3 vertices appear at $v_4 v_5 v_6$. Now, $L_R = \{v_1 v_2 v_3 v_4\}$. Thus, Rule 2 is applied (see Fig. 14(b)). Here, v_4, v_5 are equivalent to v_i, v_{i+1} in the Fig. 11(c). The nearest edge of $v_4 v_5$ is $v_6 v_7$. Here, v_7 is equivalent to v_k in the Fig. 11(c). After application of this rule, $L_R = \{v_1 v_2 v_3 v_5\}$. The next consecutive type 3 vertices detected are $v_5 v_7$ and there is an intruded concavity of OIC ($u_3 u_4$). So, Rule R3 is applied here and a new vertex $v_{8'}$ is introduced. Now, $L_R = \{v_1 v_2 v_3 v_{8'}\}$ (Fig. 14(c)). Next, consecutive type 3 vertices are detected at v_{12} . Now, $L_R = \{v_1 v_2 v_3 v_{8'} v_9 v_{10} v_{11} v_{12}\}$. As there are no intruded concavity of OIC, Rule R2 is applied. The

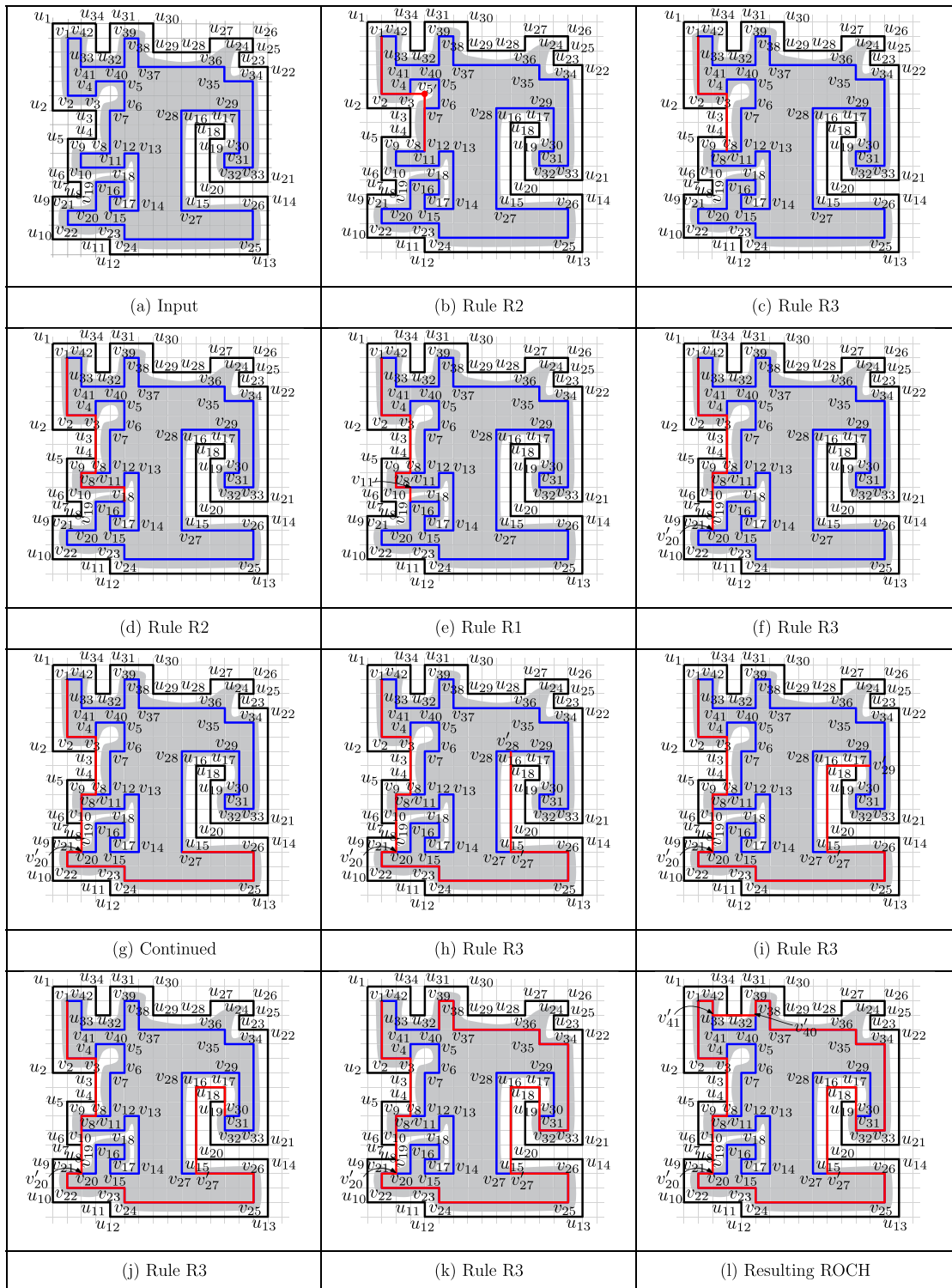


Fig. 14. The demonstration to obtain ROCH is shown in several steps (a)-(l). The digital object, OIC, IIC, and ROCH are shown in gray, black, blue, and red color respectively.

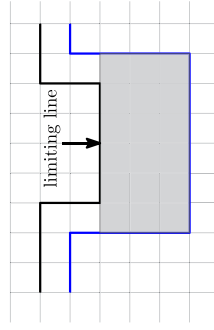


Fig. 15. Illustration of Rule R3.

nearest edge of $v_{12}v_{13}$ is $v_{18}v_{19}$. v_k of Fig. 11(d) is v_{19} here. $L_R = \{v_1v_2v_3v_8'v_9v_{10}v_{11}v_{18}\}$. The next detected concavity is for the two consecutive type 3 vertices v_{11} and v_{18} and no OIC concavity is intruded into it. So, Rule R1 is applied here. The intermediate solution is shown in Fig. 14(e). Now, $L_R = \{v_1v_2v_3v_8'v_{11}'\}$. The next detected concavity is for the two consecutive type 3 vertices $v_{11}'v_{20}$ and the OIC concavity, u_7u_8 , is intruded into it. The intermediate result is shown in Fig. 14(f). Now, $L_R = \{v_1v_2v_3v_8'v_9v_{20}'\}$. The next concavity is detected for the vertices $v_{27}v_{28}v_{29}v_{30}$ (Fig. 14(g)). The OIC concavity $u_{15}u_{16}u_{17}u_{18}$ is intruded into it. Thus Rule R3 is applied here. The two new vertices are added v_{27}' and v_{28}' (Fig. 14(h)). $L_R = \{v_1v_2v_3v_8'v_9v_{20}'v_{21}v_{22}v_{23}v_{24}v_{25}v_{26}v_{27}'v_{28}'\}$. In the next step of Rule R3, v_{28}' gets deleted and v_{29}' is added (Fig. 14(i)). In the following step of the application of Rule R3, the resulting L_R becomes $\{v_1v_2v_3v_8'v_9v_{20}'v_{21}v_{22}v_{23}v_{24}v_{25}v_{26}v_{27}'u_{16}u_{17}v_{31}\}$ (Fig. 14(j)). It is to be noted here that the vertices u_{16} and u_{17} are added in ROCH as new vertices. Since the position matches exactly with the OIC vertices, no renaming of the vertices is done here. The traversal continues and $L_R = \{v_1v_2v_3v_8'v_9v_{20}'v_{21}v_{22}v_{23}v_{24}v_{25}v_{26}v_{27}'u_{16}u_{17}v_{31}v_{32}v_{33}v_{34}v_{35}v_{36}v_{37}v_{38}v_{39}v_{40}\}$ as shown in Fig. 14(k). The next detected concavity is $v_{40}v_{41}$ and the OIC concavity $u_{32}u_{33}$ is intruded into it. Rule R3 is applied and the resulting list of vertices of ROCH becomes $L_R = \{v_1v_2v_3v_8'v_9v_{20}'v_{21}v_{22}v_{23}v_{24}v_{25}v_{26}v_{27}'u_{16}u_{17}v_{31}v_{32}v_{33}v_{34}v_{35}v_{36}v_{37}v_{38}v_{39}v_{40}'v_{41}'v_{42}\}$ (Fig. 14(l)).

4.3. Proof of correctness

By the definition of $ROCH_B(A)$, any vertical or horizontal line segments having end points in A , it will lie within A and B . The definition of the relative orthogonal convex hull is violated whenever there is a concavity in the inner polygon. The algorithm uses three types of reduction rules, R1, R2, R3. The Rule R1 is applied when there is a concavity in IIC with two consecutive type 3 vertices and no intruded concavity in OIC. The Rule R2 is applied when there is a concavity in IIC with two or more consecutive type 3 vertices and no intruded concavity in OIC. The Rule R3 is applied when there is a concavity in IIC and also an intruded concavity in OIC.

Thus, there will be no vertical or horizontal line in OIC and having end points in IIC, which is not contained in IIC. In case of the concavities of the later type, the concavity of the IIC limited by the horizontal line (or the vertical line as the case may be) of the protruded concavity of the OIC is removed. It may be noted that beyond this horizontal (vertical) line of the protruded concavity there can not be any horizontal line lying inside the OIC. Thus, the removal of the shaded portion (Fig. 15) is sufficient for deriving the $ROCH_{OIC}(IIC)$. The rules remove the concavities present in a convoluted concave region of the IIC by adjusting the length and types of vertices as shown in Fig. 11. The algorithm completes when the traversal of the IIC is completed thereof removing all concavities (or partial removal of concavities in case of protruded concavity of the OIC) of the IIC. The result is given as the ROCH.

4.4. Complexity analysis

The algorithm first constructs the inner and outer isothetic cover ([1,2]) of the digital object which takes $O(g) * O(n/g) = O(n)$ time, where n is number of points on the boundary of the digital object and g is the grid size. These steps are mentioned in steps 1,2 in the Algorithm ROCH. The lexicographically sorted lists are prepared which takes $O(n \log n)$. The total preprocessing steps takes $O(n \log n)$. The algorithm traverses along IIC from top-left corner in anticlockwise manner and checks whether there is a concavity. The loop mentioned in the Algorithm ROCH takes $O(n)$ time. The procedure APPLYRULE traverses only the concavity part of IIC and OIC. The procedure CHECK-OIC-CONCAVITY searches in L_{px}^{OIC} , L_{py}^{OIC} , L_{px}^{IIC} , and L_{py}^{IIC} to find intruded concavity which takes linear time as it searches only a part of the lists. The procedures DEL and ADD delete and add a vertex in L_R which takes linear time. The procedure SETVERTEX-R1 mentioned in the procedure APPLY-R1, takes constant time as it checks the length of two edges along the two sides of concavity with two consecutive type 3 vertices to find v_i' or v_j' . Thus the procedure APPLY-R1 takes linear time. The procedure APPLY-R2 also takes linear time. It checks nearest convex edge w.r.t. $v_i v_{i+1}$ in linear time. The two procedures DEL and ADD take linear time. The procedure SETVERTEX-R3 checks the position of intruded concavity of OIC using the lists L_{px}^{OIC} or L_{py}^{OIC} w.r.t. the concavity in IIC for the edge

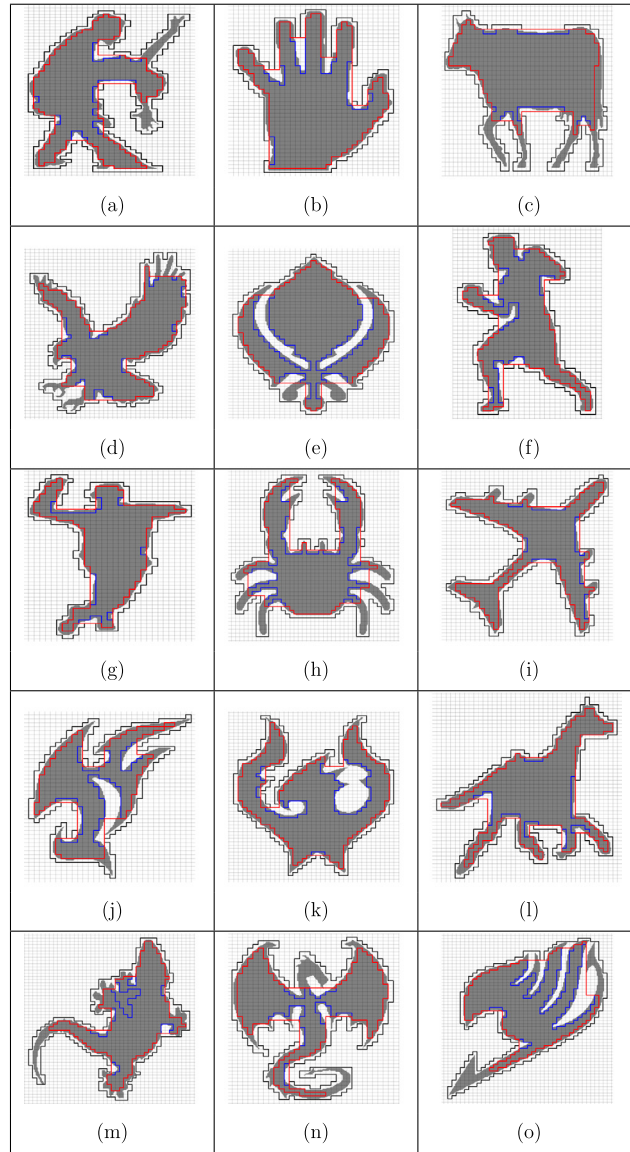


Fig. 16. Experimental results of ROCH for different objects where ROCH is marked by red, IIC is marked by blue, and OIC is marked by black for grid size $g = 4$.

$v_i v_{i+1}$ which takes linear time. Other two procedures DEL and ADD take linear time. Thus the total time taken by the procedure APPLY-R3 takes linear time. Thus the total time complexity to obtain the ROCH is $O(n)$, i.e., linear. Let us explain the time complexity w.r.t. the number of vertices of OIC and IIC. All the vertices of IIC are traversed and some part of OIC is traversed when there is intruded concavity. Let n_{iic} , n_{oic} , and n_{roch} be the total number of vertices in IIC, OIC, and resulting ROCH respectively. It is to be noted that the ROCH is obtained from some of the vertices of IIC and the number of vertices in ROCH is less than that of in IIC. Since there are linear traversal of IIC and part of OIC, the total time complexity to obtain the ROCH is $O(n_{iic} + n_{oic})$. It may be noted that $O(n_{iic} + n_{oic})$ is equivalent to $O(n)$.

5. Experimental results

The results of ROCH are generated in the computer system with i5-3230M CPU, 2.60 GHZ $\times 4$ processor and OS Ubuntu 16.04 LTS 64-bit. The experimental results on different digital objects are shown in Fig. 16, Fig. 17, and Fig. 18. The Table 2 shows the comparison of the number of vertices and area of the outer and inner isothetic covers, orthogonal convex hull, and relative orthogonal convex hull for the results in Fig. 16. It is seen that the number of vertices is always less in OCH compared to others (i.e., OIC, IIC, ROCH) and that of ROCH is always less than OIC and IIC but greater than OCH. The area of ROCH is greater than IIC but less than OIC and OCH. When an image is considered for image analysis or shape analysis

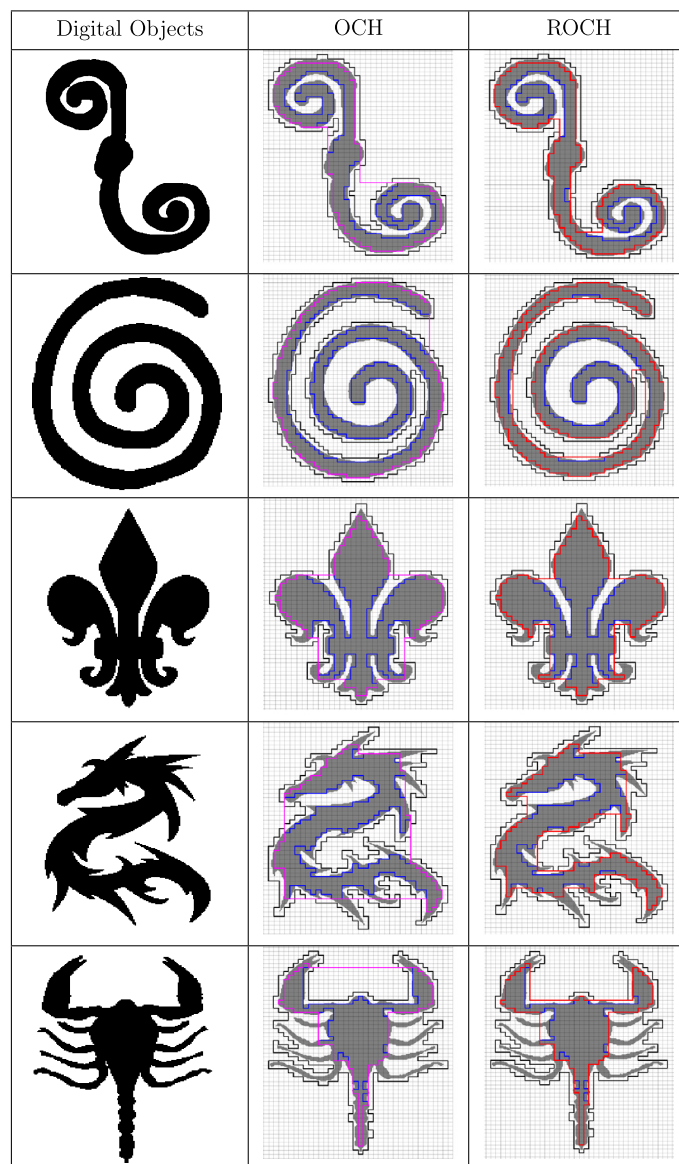


Fig. 17. The experimental results of OCH and ROCH where the outer and inner isothetic covers are marked by black and blue, the OCH is marked by magenta, and ROCH is marked by red for grid size $g = 4$.

Table 2

The comparison of the number of vertices and area of OIC, IIC, OCH, and ROCH for the objects given in Fig. 16 for grid size $g = 4$.

Digital objects	Number of vertices				Area			
	OIC	IIC	UCH	ROCH	OIC	IIC	UCH	ROCH
<i>a</i>	310	234	146	206	13472	7968	13152	8688
<i>b</i>	208	230	136	186	12496	8720	11136	9696
<i>c</i>	254	150	120	132	12992	8416	10032	9024
<i>d</i>	256	224	143	189	13904	8560	13200	9280
<i>e</i>	166	304	154	154	16320	9824	12768	12768
<i>f</i>	216	238	140	190	9936	6080	11632	7120
<i>g</i>	188	178	134	160	8672	5600	8576	6128
<i>h</i>	302	272	136	188	14560	7456	13136	9552
<i>i</i>	272	252	148	234	10944	6512	17200	7232
<i>j</i>	224	216	120	158	8976	4512	7536	5920
<i>k</i>	266	300	153	223	15056	8944	15264	11328
<i>l</i>	280	264	158	240	11648	7152	14160	7952
<i>m</i>	256	232	158	174	12384	6992	8768	7968
<i>n</i>	320	248	140	206	12576	6320	12896	7296
<i>o</i>	236	308	144	164	15600	8000	12016	42432

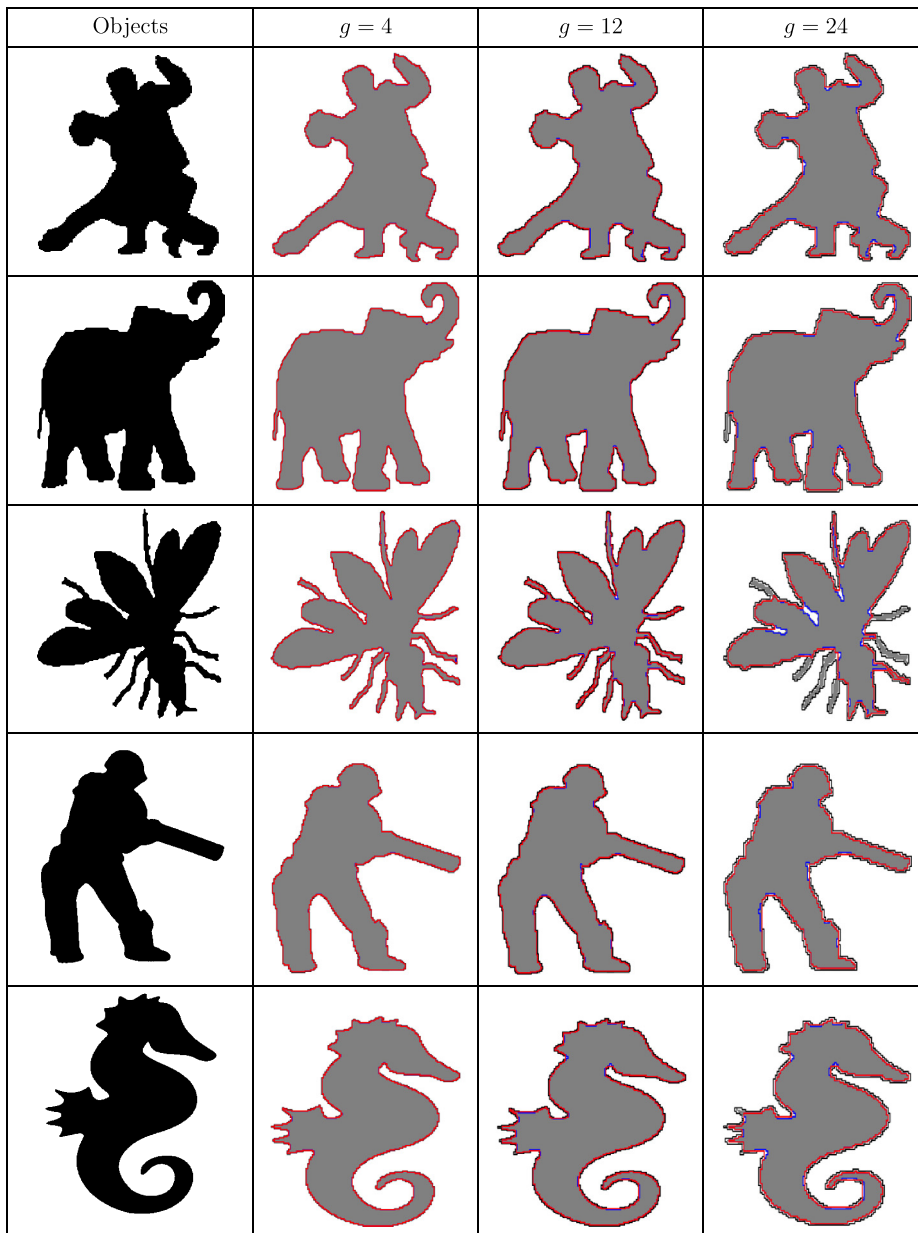


Fig. 18. Experimental results of ROCH for another set of objects of size $3k \times 3k$ for various grid sizes where relative convex hull is marked by red.

relative orthogonal convex hull will be very useful compared to orthogonal convex hull as it does not include much area outside the object. The ROCH is better compared to OIC and IIC as ROCH contains less concavities compared to OIC and IIC.

Table 3 presents the number of vertices and area of OCH and ROCH for the figures shown in Fig. 17. It is seen that OCH occupies much more extra area outside the objects compared to ROCH. Since the area of ROCH is much less compared to OCH, it is better for further processing for different applications in image analysis. Table 4 presents the number of vertices and area of OIC, IIC, OCH and ROCH for the figures shown in Fig. 18. This table contains the CPU execution time for the mentioned digital objects of size $3k \times 3k$. From the data, it can be said that ROCH lies between OIC and IIC. The algorithm is tested on different digital objects for various grid sizes which are shown in Fig. 18. The mpeg7 database is used here. It is obvious that for lower grid sizes the more finer details of the objects on the contour is captured thus the number of vertices and area are more in lower grid sizes compared to higher grid sizes. These are depicted in the Table 4. Similarly, the CPU execution time is higher in lower grid sizes and vice versa.

The plots presented in Fig. 19 show the variation of number of vertices and area when grid size increases for the digital objects shown in Fig. 18. It is seen that the number of vertices decreases when grid size increases for OIC, IIC, and ROCH. The area decreases as grid size increases for IIC and ROCH. The area of OIC increases as grid size increases as it includes

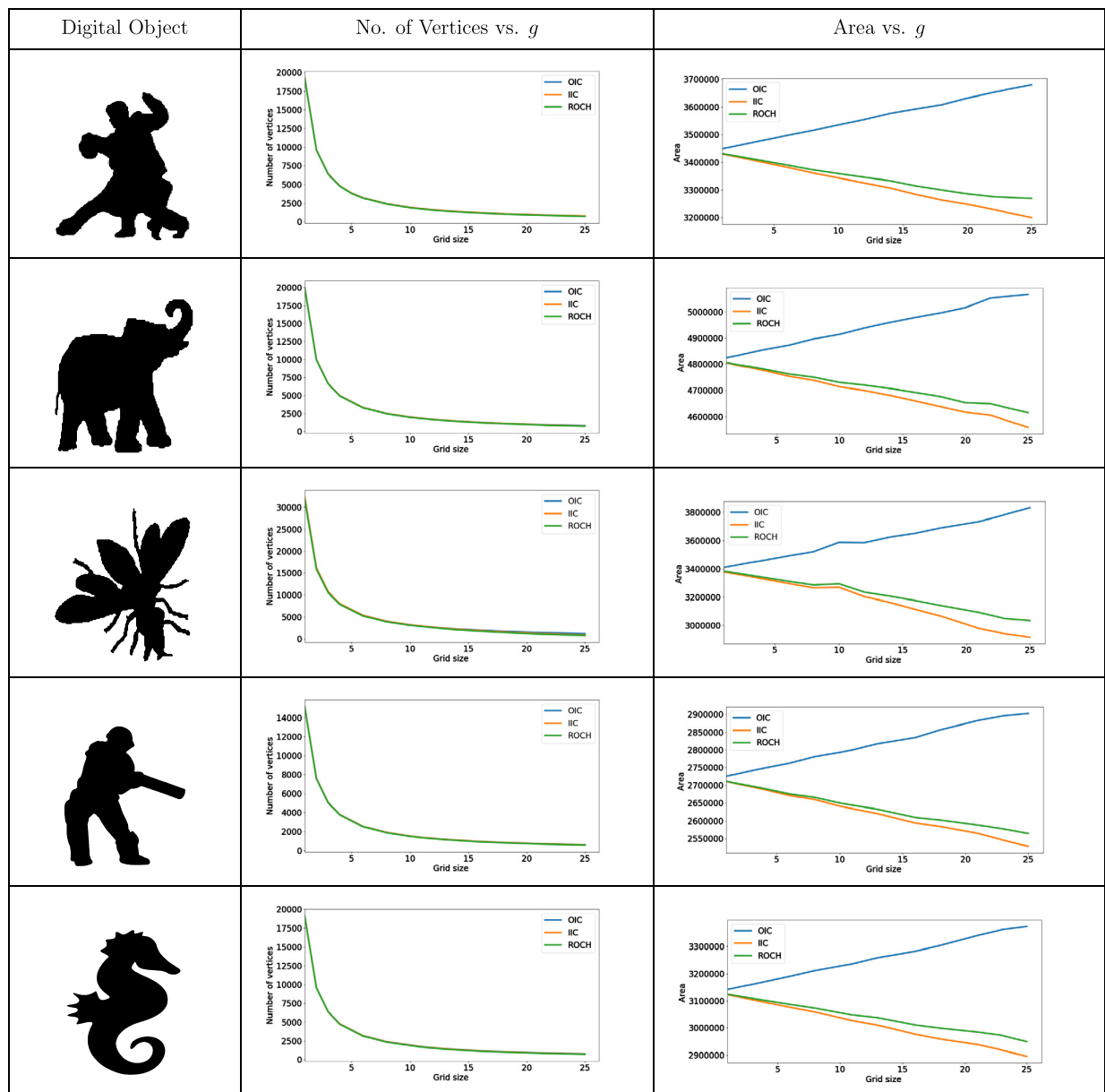


Fig. 19. Plots for the digital objects shown in Fig. 18. The number of vertices of OIC, IIC, and ROCH vs grid size are presented in left column and the area of OIC, IIC, and ROCH vs grid size is presented in right column.

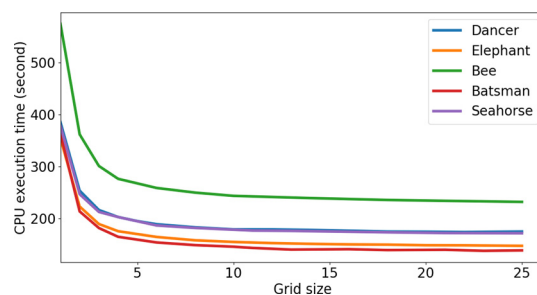


Fig. 20. The CPU execution time (in seconds) to obtain ROCH vs grid size is shown for the digital objects given in Fig. 18.

Table 3

Comparison of the number of vertices and area of OCH, and ROCH for the figures in Fig. 17.










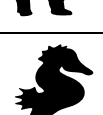
Digital Objects	Number of vertices		Area	
	<i>OCH</i>	<i>ROCH</i>	<i>OCH</i>	<i>ROCH</i>
	154	186	8544	7088
	172	390	22512	14208
	149	173	10704	9680
	140	216	12368	8258
	152	170	8192	5520

Table 4

The number of vertices, area for various grid sizes for OIC, IIC, and ROCH are presented for the digital objects (see Fig. 18). The CPU time (including preprocessing) in seconds is measured for ROCH. Here the object size is $3k \times 3k$.

Digital Objects	Grid size	Number of vertices			Area			CPU Time (Second)
		<i>OIC</i>	<i>IIC</i>	<i>ROCH</i>	<i>OIC</i>	<i>IIC</i>	<i>ROCH</i>	
	5	3852	3844	3807	3487325	3391125	3397800	225.83
	10	1922	1922	1877	3535600	3343000	3358700	211.96
	15	1272	1272	1223	3584250	3296250	3325050	206.71
	20	962	956	913	3632800	3247600	3285600	200.61
	25	758	760	709	3680000	3199375	3269375	201.79
	5	3998	3984	3942	4862525	4762250	4769225	169.30
	10	1996	1992	1950	4914400	4713900	4730600	155.82
	15	1328	1326	1286	4967775	4667850	4695750	151.66
	20	996	988	954	5016400	4617200	4653600	149.75
	25	794	748	716	5066250	4558750	4615625	149.25
	5	6372	6376	6258	3486500	3327075	3338950	265.71
	10	3180	3170	3072	3588500	3269500	3293900	248.85
	15	2092	2094	1983	3684600	3204225	3252150	236.60
	20	1510	1400	1252	3794800	3107600	3207200	232.95
	25	1176	884	759	3832500	2915625	3034375	230.76
	5	3034	3026	2999	2755900	2680150	2683825	159.73
	10	1514	1514	1480	2793000	2641000	2649800	145.27
	15	1010	1004	976	2830050	2602800	2616750	140.50
	20	754	748	716	2870800	2566800	2596000	137.36
	25	598	592	568	2902500	2528125	2564375	138.14
	5	3822	3814	3775	3180050	3084600	3093375	194.71
	10	1908	1888	1846	3228300	3037500	3056600	180.84
	15	1268	1246	1202	3277125	2991150	3022875	178.01
	20	950	922	882	3324000	2942800	2987600	174.29
	25	764	732	694	3373125	2895625	2949375	174.07

more area outside the digital object. The overall efficiency of ROCH is depicted in the plot Fig. 20. The plot represents the variation of CPU execution time to obtain ROCH while grid size varies. When grid size is less more detailed description of digital objects are captured in OIC and IIC, i.e., the number of concavities are more at lower grid sizes, thus CPU execution time to obtain ROCH increases.

6. Conclusions

In this paper, a combinatorial algorithm is presented to compute the relative orthogonal convex hull of the digital object. The time complexity of the proposed algorithm is found to be linear. Most of the algorithms available in the literature are recursive. However, the proposed algorithm uses a combinatorial technique to avoid recursion. The proposed algorithm is simple and it traverses through the inner cover once to produce the relative orthogonal convex hull. The efficacy and correctness of the algorithm tested exhaustively on different objects available in the data mpeg7 data set and are substantiated by a few examples in the figures. The algorithm also runs on large size digital objects. Although the concept of the relative orthogonal convex hull is mainly theoretical, many image processing applications might find it useful in the shape analysis of digital objects. It can also be used in robotics to plan the path of a robot.

Declaration of competing interest

There is no conflict of interest.

References

- [1] Arindam Biswas, Partha Bhowmick, Bhargab B. Bhattacharya, **TIPS**: on finding a tight isothetic polygonal shape covering a 2D object, in: 14th Scandinavian Conference on Image Analysis (SCIA), Springer, Berlin, Heidelberg, 2005, pp. 930–939.
- [2] Arindam Biswas, Partha Bhowmick, Bhargab B. Bhattacharya, Construction of isothetic covers of a digital object: a combinatorial approach, *J. Vis. Commun. Image Represent.* 21 (4) (2010) 295–310.
- [3] Arindam Biswas, Partha Bhowmick, Moumita Sarkar, Bhargab B. Bhattacharya, A linear-time combinatorial algorithm to find the orthogonal hull of an object on the digital plane, *Inf. Sci.* 216 (2012) 176–195.
- [4] Timothy M. Chan, Optimal output-sensitive convex hull algorithms in two and three dimensions, *Discrete Comput. Geom.* 16 (4) (1996) 361–368.
- [5] Ronald L. Graham, An efficient algorithm for determining the convex hull of a finite planar set, *Inf. Process. Lett.* 1 (4) (1972) 132–133.
- [6] Mashhood Ishaque, Csaba D. Tóth, Relative convex hulls in semi-dynamic subdivisions, in: Algorithms – European Symposium on Algorithms (ESA 2008), in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2008, pp. 780–792.
- [7] Ray A. Jarvis, On the identification of the convex hull of a finite set of points in the plane, *Inf. Process. Lett.* 2 (1) (1973) 18–21.
- [8] Yukiko Kenmochi, Reinhard Klette, Surface area estimation for digitized regular solids, in: International Symposium on Optical Science and Technology, San Diego, CA, United States, in: Vision Geometry IX, vol. 4117, 2000, pp. 100–111.
- [9] Gisela Klette, A recursive algorithm for calculating the relative convex hull, in: Proceedings of 25th International Conference of Image and Vision Computing New Zealand, IEEE, Queenstown, 2010, pp. 1–7.
- [10] Reinhard Klette, Multigrid convergence of geometric features, in: Digital and Image Geometry, Advanced Lectures [Based on a Winter School Held at Dagstuhl Castle, Germany in December 2000], Springer-Verlag, 2001, pp. 318–338.
- [11] Reinhard Klette, Aziel Rosenfeld, Digital Geometry: Geometric Methods for Digital Picture Analysis, Morgan Kaufmann, San Francisco, 2004.
- [12] Jacques-Olivier Lachaud, Xavier Provençal, Dynamic minimum length polygon, in: 14th International Workshop on Combinatorial Image Analysis (IW-CIA'11), Springer, Berlin, Heidelberg, 2011, pp. 208–221.
- [13] Christian Lantuejoul, Serge Beucher, On the use of geodesic metric in image analysis, *J. Microsc.* 121 (1) (1981) 39–49.
- [14] Christian Lantuejoul, Florian Maisonneuve, Geodesic methods in quantitative image analysis, *Pattern Recognit.* 17 (2) (1984) 177–187.
- [15] Fajie Li, Reinhard Klette, Euclidean Shortest Paths – Exact or Approximate Algorithms, Springer, 2011.
- [16] Avraham A. Melkman, On-line construction of the convex hull of a simple polyline, *Inf. Process. Lett.* 25 (1) (1987) 11–12.
- [17] Joseph S.B. Mitchell, Geometric shortest paths and network optimization, in: Handbook of Computational Geometry, Elsevier Science Publishers B. V. North-Holland, 1998, pp. 633–701.
- [18] Franco P. Preparata, An optimal real-time algorithm for planar convex hulls, *Commun. ACM* 22 (7) (1979) 402–405.
- [19] Xavier Provençal, Jacques-Olivier Lachaud, Two linear-time algorithms for computing the minimum length polygon of a digital contour, in: Proceedings of 15th International Conference on Discrete Geometry for Computer Imagery (DGCI'09), in: Lecture Notes in Computer Science, vol. 5810, Springer, Heidelberg, 2009, pp. 104–117.
- [20] Jack Sklansky, Robert L. Chazin, Bruce J. Hansen, Minimum-perimeter polygons of digitized silhouettes, *IEEE Trans. Comput.* 21 (3) (1972) 260–268.
- [21] Jack Sklansky, Dennis F. Kibler, A theory of non-uniformly digitized binary pictures, *IEEE Trans. Syst. Man Cybern.* SMC-6 (9) (1976) 637–647.
- [22] Fridrich Sloboda, Josef Stoer, On piecewise linear approximation of planar Jordan curves, *Comput. Appl. Math.* 55 (3) (1994) 369–383.
- [23] Fridrich Sloboda, Bedrich Zat'ko, Josef Stoer, On approximation of planar one-dimensional continua, in: Advances in Digital and Computational Geometry, 1998, pp. 113–160.
- [24] Godfried T. Toussaint, An optimal algorithm for computing the relative convex hull of a set of points in a polygon, in: Signal Processing III: Theories and Applications, 1986, pp. 853–856.
- [25] Petra Wiederhold, Hugo Reyes, Relative convex hull determination from convex hulls in the plane, in: Proceedings of 17th International Workshop on Combinatorial Image Analysis, (IWCI'15), in: Lecture Notes in Computer Science, vol. 9448, Springer-Verlag, 2015, pp. 46–60.
- [26] Linjiang Yu, Reinhard Klette, An approximative calculation of relative convex hulls for surface area estimation of 3d digital objects, in: Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02), in: ICPR '02, vol. 1, IEEE Computer Society, USA, 2002, pp. 131–134.