# Recognizing S-Star Polygons

Laxmi P Gewali
Department of Computer Science
University of Nevada, Las Vegas
4505 Maryland Parkway, Las Vegas, Nevada 89154

# Recognizing S-Star Polygons

**Abstract**

We consider the problem of recognizing star-polygons under stair-case visibility (**s-visibility**). We show that the s-visibility polygon from a point inside a simple orthogonal polygon of $n$ sides can be computed in $O(n)$ time. When the polygon contains holes the algorithm runs in $O(n \log n)$ time, which we prove to be optimal by linear time reduction from sorting. We present an algorithm for computing the s-kernel of a polygon in $O(n)$ time for simple orthogonal polygons and in $O(n^2)$ time for orthogonal polygons with holes; both complexities are optimal in the worst case. Finally, we report the main result of this paper: we show that even though the s-kernel of a polygon with holes may have $\Omega(n^2)$ components it is possible to recognize such polygons in $O(n \log n)$ time.

**Key Words:** *Polygon Recognition, Visibility, Computational Geometry.*

## 1 Introduction

Problems dealing with the visibility properties of polygons are central in computational geometry and robotics. Visibility graphs induced by a collection of polygonal obstacles have been used for planning the motion of robots(1). Similarly, visibility properties of polygons have been useful for decomposing/partitioning polygons into simpler components(2,3,4,5). Applications of visibility in robotics, pattern recognition, and in manufacturing have also been reported(4,6,7). An important problem is the computation of visibility polygon from a given point in the presence of polygonal obstacles (visibility polygon from a given point is the set of points visible from it). Under the standard definition of visibility, two points inside a polygon are **visible** if they can be connected by a line segment without intersecting the exterior of the polygon. $O(n)$ time algorithms for computing the visibility polygon from a given point inside a simple polygon have been developed(8,9), where $n$ is the number of edges of the underlying polygon. When the polygon contains holes, the visibility polygon can be computed in $O(n \log n)$ time(10). Visibility properties of polygons using other notions of visibility have also been considered[2,3,4,5,11,12,13,23]. Covering properties of orthogonal polygons under s-visibility and r-visibility have also been reported [2,11,13]. Note that two points inside a polygon are said to be r-visible if they can be contained in a rectangle $R$ (with edges parallel to the coordinate axes) such that $R$ does not intersect with the exterior of the polygon.

In this paper we consider the problem of computing visibility polygon, the problem of computing kernel, and the problem of recognizing star polygons under stair-case visibility (**s-visibility**). One motivation for considering these problems is the usefulness of s-visibility for developing polygon covering algorithms. For example, the minimum star cover of a simple orthogonal polygon under s-visibility can be computed in $O(n^8)$ time[2], although the same problem is NP-hard under standard visibility[14]. Other motivation comes from the applicability of monotone paths in robotics[6,15], and s-visibility is based on monotonicity in two perpendicular directions.

Perhaps the main contribution of this paper is the demonstration that an s-star polygon, possibly containing holes, can be recognized in $O(n \log n)$ time even though its s-kernel can have $\Omega(n^2)$ components. Our results can be summerized as follows: (a) We present an $O(n)$ time algorithm for computing the s-visibility polygon from a point inside a simple orthogonal polygon. (b) For orthogonal polygons with holes, we show that s-visibility polygon from an interior point can be computed in $O(n \log n)$ time. We prove this time complexity to be optimal by linear time reduction from the sorting problem. (c) We give a simple construction for converting an instance of computing s-visibility from an exterior point to an instance of computing s-visibility from an interior point. (d) We present algorithms which compute s-kernels of simple orthogonal polygons and orthogonal polygons with holes in $O(n)$ and $O(n \log n)$ time, respectively. Both complexities are optimal in the worst case. (e) Finally, we show that s-star polygons with holes can be recognized in $O(n \log n)$ time.

## 2 Preliminaries

The input orthogonal polygon $Q$, whose edges are parallel to coordinate axes, is given as an ordered sequence of vertices $v_0, v_1, v_2, v_3, ..., v_{n-1}$, in the order in which they occur along the clockwise traversal of its boundary. Note that if the boundary of the polygon is traversed in this order then the interior of the polygon is always to the right. A **stair-case path** is a path consisting of line segments parallel to the $x$ or $y$ coordinate axes such that any horizontal or vertical line intersect with at most one line segment of the path. Stair-case paths can be categorized into two types: (a) a **type I stair-case path** extends from south-west to north-east, and (b) a **type II stair-case path** extends from south-east to north-west (Figure 1a). Two points in the interior of a polygon are said to be **s-visible** if they can be connected by a stair-case path without intersecting the exterior of the polygon. A polygonal path is **monotone** along a given direction $d$ if any line perpendicular to $d$ intersects with at most one line segment of the path. An orthogonal path having monotonicity along the $x$-axis direction is called an **x-monotone path**. Similarly, **y-monotone path** is defined. In term of monotonicity, a stair-case path can be viewed as an orthogonal path having monotonicity in both x-axis and y-axis directions. A maximal boundary chain of an orthogonal polygon which is also a type I stair-case path is referred to as a **type I boundary s-chain**. Similarly, a **type II boundary s-chain** is defined. In Figure 1e, chain $(v_8, v_9, ..., v_{14})$ is a type I boundary s-chain and chain $(v_{17}, v_{18}, v_{19})$ is a type II boundary s-chain. The boundary of an orthogonal polygon can be partitioned into the above two types of s-chains. A horizontal or vertical line segment inside the polygon with end points at its boundary is a **chord** of the polygon.

The visibility polygon from a point $q$ can be defined in terms of x-monotone path, y-monotone path, or stair-case path. The **x-monotone visibility polygon** ( respectively, the **y-monotone visibility polygon**) from an interior point $q$ is the set of points that can be connected to $q$ by a x-monotone path (respectively, y-monotone path). Similarly, the **s-visibility polygon** from a point is the set of points that can be connected to $q$ by a stair-case path. Figure 1 illustrates the above three types of visibility polygons. Let $V_q^s(Q)$, $V_q^x(Q)$, and $V_q^y(Q)$ denote the s-visibility polygon, x-monotone visibility polygon, and y-monotone visibility polygon, respectively, from a point $q$ inside polygon $Q$.

**Lemma 1**: For a simple orthogonal polygon $Q$, the s-visibility polygon $V_q^s(Q)$ is given by the intersection of the x-monotone visibility polygon $V_q^x(Q)$ and the y-monotone visibility polygon $V_q^y(Q)$.

**Proof**: Let $t$ be an arbitrary point in the intersection of $V_q^x(Q)$ and $V_q^y(Q)$. Then $t$ is both x-monotone visible and y-monotone visible from $q$. Consider the x-monotone path of shortest length $R$ connecting $q$ and $t$. We claim that $R$ must be a stair-case path. If $R$ is not a stair-case path then the boundary of the polygon must have two crossed U-turns $A$ and $B$ as shown in Figure 1b (if U-turns $A$ and $B$ are not as shown in the figure then $R$ can be shortened.) Now the only way to connect $q$ and $t$ by a y-monotone path is to go above $B$ or below $A$ implying that the polygon has holes, a contradiction. Thus we get a contradiction if $R$ is not a stair case path. $\square$

**Lemma 2**: $V_q^s(Q)$, the s-visibility polygon from a point $q$ inside an orthogonal polygon $Q$, is an orthogonal polygon.

**Proof**: Assume the contrary that $V_q^s(Q)$ is not orthogonal. Then some edge $(a, b)$ on the boundary of $V_q^s(Q)$ is neither vertical nor horizontal. Consider an s-visibility path connecting $q$ to an interior point $x$ in edge $(a, b)$. Since edge $(a, b)$ is not parallel to the $x$

or $y$ axis and is inside the polygon $Q$, the segment of the s-visibility path incident on $x$ can be extended beyond $x$ to the outside of $V_q^s(Q)$. This means that a point lying outside of $V_q^s(Q)$ is s-visible from $q$, implying that $V_q^s(Q)$ is not the entire s-visibility polygon from point $q$ - a contradiction. □

## 3 Algorithms for Computing s-Visibility Polygon

In this section we develop an algorithm for computing the s-visibility polygon from a point inside an orthogonal polygon. An overview of the algorithm can be stated as follows: (i) We first compute $R = V_q^x(Q)$, the x-monotone visibility polygon from point $q$ for polygon $Q$. (ii) We then compute the y-monotone visibility polygon from point $q$ for polygon $R$, which is precisely the required s-visibility polygon $V_q^s(Q)$ (Lemma 1). To compute the x-monotone visibility polygon efficiently, we proceed by partitioning the polygon into four parts, induced by horizontal and vertical chords through $q$ (see Figure 2). We refer to these parts as **NE-part, NW-part, SE-part,** and **SW-part**, with obvious meaning. In what follows, we describe the computation of the x-monotone visibility polygon for the NE-part. Computation for other parts are similar. We partition the NE-part into trapezoids by extending vertical edges of the polygon to its interior. Since the polygon is orthogonal, all trapezoids are rectangles. We call two rectangles **neighbors** if they share a vertical chord. A neighbor can be either an east neighbor (**E-neighbor**) or a west neighbor (**W-neighbor**).

We start with the unique rectangle of the NE-part having $q$ as its vertex as the partially constructed x-monotone visibility polygon. As the algorithm proceeds, other rectangles are added, one at a time, by examining the neighbors of the rectangles lying on the boundary of the partially constructed s-visibility polygon. The partially constructed solution is expanded by adding rectangles lying to the east of its boundary. The process of expansion stops when no more rectangles lying to the east of the partial solution are found. We can implement this mechanism by using a simple queue data structure. We first enqueue the unique rectangle with $q$ as one of its vertices. We dequeue a rectangle $r$ from the queue and this becomes the currently selected candidate to be added to the partially constructed solution. Neighbors of $r$ lying to the east are added to the queue and the currently examined rectangle is added to the partial solution. (In Figure 2, the shaded area is a partial solution; rectangles $r_1$ and $r_2$ are adjacent to the partial solution and only $r_2$ is to the east of the boundary of the partial solution and hence it is used to include and expand the partial solution.) This process is continued until the queue is empty. The resulting solution is the required x-monotone visibility polygon for the NE-part. Similarly, the x-monotone visibility polygon for the remaining three parts are computed and $R = V_q^x(Q)$ is determined by putting these four sub-solutions together. We next compute the y-monotone visibility polygon from point $q$ for polygon $R$, which precisely gives the s-visibility polygon $V_q^s(Q)$. A formal description of the algorithm is listed as Algorithm 1.

**Algorithm 1**
**Input:** A simple orthogonal polygon $Q$ and a point $q$ inside it.
**Output:** s-Visibility polygon from $q$.
**Step 1:** Partition the polygon $Q$ into four parts $Q_1, Q_2, Q_3, Q_4$ by extending horizontal and vertical lines through point $q$. Let $Q_1$ be the NE-part.
**Step 2:** Partition $Q_1$ into rectangles by extending vertical edges of $Q_1$.
**Step 3:** $S_1 = \phi$; $Queue = \phi$; Add to Queue the rectangle of $Q_1$ with point $q$ as its vertex;
**Step 4:** While the Queue is not empty do
- **i.** Delete a rectangle $curr$ from the Queue;
- **ii.** Add to Queue all E-neighbors of $curr$;
- **iii.** $S_1 = S_1 \cup curr$;

**Step 5:** By following the method similar to Steps 2-4, compute x-monotone visibility polygons $S_2$, $S_3$, and $S_4$ from point $q$ for polygons $Q_2$, $Q_3$, and $Q_4$, respectively.
**Step 6:** Let $Q_x = S_1 \cup S_2 \cup S_3 \cup S_4$;
**Step 7:** Compute y-monotone visibility polygon for $Q_x$ from point $q$ by performing corresponding computations as in Steps 2-6, which gives the required solution.

**Lemma 3**: Algorithm 1 correctly computes the s-visibility polygon from point $q$.

**Proof**: The fact that Steps 2-4 correctly compute the x-monotone visibility polygon for polygon $Q_1$ from point $q$ can be argued as follows. The partial solution $S_1$ remains as an $x$-monotone visibility polygon throughout the iteration of the while loop. This follows from the fact that only rectangles attached to the east of the boundary of a partial solution are added to update it. Thus rectangles used to construct the solution $S_1$ are those rectangles that can be connected by an $x$-monotone path from $q$. To complete the argument we need to establish that all rectangles that can be connected by an $x$-monotone path from $q$ are present on $S_1$ when the loop terminates. Assume the contrary, that there exists a rectangle $r'$ that can be connected from $q$ by an $x$-monotone path and is not present in $S_1$ when the while loop terminates. Then consider the $x$-monotone path connecting $q$ to $r'$. The path stabs a sequence of vertical rectangles $r_1, r_{i_1}, r_{i_2}, ..., r_{i_p}, r'$. Since $q$ is a south west vertex of $Q_1$, each rectangle in the above sequence lies to the east of its predecessor. This implies that at some point during the iteration of the while loop the E-neighbor of a rectangle lying on the boundary of the partial solution was not added to the solution, which is a contradiction to the condition of the while loop. Symmetrical arguments establish the correctness of Steps 5-7. □

**Theorem 1:** The s-visibility polygon from a point inside a simple orthogonal polygon can be computed in $O(n)$ time, where $n$ is the number of vertices of the given polygon.

**Proof**: Step 1 can be done in $O(n)$ time by simply checking the intersection of vertical and horizontal lines through $q$ with the edges of the polygon. Step 2 can be done by trapezoidization of the polygon in $O(n)$ time[(16)]. Each rectangle enters the queue at most once and during each iteration at least one rectangle is dequeued and hence Steps 3 and 4 take $O(n)$ time. Analogously, the time complexity of steps 5-7 is $O(n)$ and the total time adds up to $O(n)$. $\square$.

**Remark 1:** It is not trivial to generalize Theorem 1 to include orthogonal polygons with holes. It is interesting to note that Lemma 1 does not hold for polygon with holes. This is shown in Figure 3a, where point $a$ and $q$ are both $x$-monotone visible and $y$-monotone visible but not $s$-visible. The problem is that $x$-monotone visibility path and $y$-monotone visibility path go around "opposite" sides of the hole[1] .

In order to develop an efficient algorithm for computing $s$-visibility polygon inside a polygon with holes we have to use a different approach than the one outlined in Algorithm 1. The idea is to trapezoid the polygon and check for the $s$-visibility of rectangles by using coloring arguments and plane sweep as outlined in the proof of Theorem 2.

**Theorem 2**: The s-visibility polygon from a point inside a polygon with holes can be computed in $O(n \log n)$ time.

**Proof:** First, we divide the polygon into 4 quadrants, as in Algorithm 1. Consider the quadrant with $q$ in the lower-left corner; the other three quadrants are similar. We vertically trapezoidize this polygon. We assume that the trapezoidization is kept in a nice topological data structure, like a winged-edge or quad-edge or doubly connected edge list (DCEL)[22]. We sweep a line from left to right in this trapezoidization. We stop whenever we reach the end (right end) of a trapezoid. When we stop, we will either: (1) color the trapezoid black, meaning that it is not seen from $q$, (2) color the trapezoid white, meaning that it is seen from $q$, or (3) divide the trapezoid by a horizontal line, coloring the bottom part black and the top part white.

We start by coloring black all of the regions corresponding to holes and polygon exterior. We sweep from left to right. When we stop, we are at the end of some trapezoid $T$. (If

[1] We are indebted to Tom Shermer and Patrice Belleville for suggesting the example of Figure 3a and for the proof of Theorem 2

there are several trapezoids that end where we stop, we process them from bottom to top). We examine the lower-left corner $r$ of $T$. If this point $r$ is reachable by a staircase path from $q$, then all of $T$ is reachable in the obvious fashion (add one vertical and one horizontal segment to the staircase from $q$ to $r$).

Note that $r$ is the intersection of a horizontal and a vertical segment in the subdivision– and since the trapezoidization introduces only vertical segments, the horizontal segment bounding $T$ from below must be part of an edge of the polygon or a hole. Thus, a staircase path that enters $T$ from below must pass through $r$ (and if such a path exists, then $T$ should be colored white).

So, we examine the region or two regions to the left of $T$ that contain $r$. These regions will have already been processed (and correctly colored) due to our left-to-right scan. If this region (or either of these regions) is colored white (or in the special case $r = q$), then $r$ is reachable by a staircase path from $q$, and we color $T$ white. If this region is (or both regions are) colored black, then the only way for a staircase path from $q$ to enter $T$ is from the left (the bottom is ruled out by the paragraph above). So, we walk up the left edge of $T$, looking at all of the regions to the left in bottom-to-top order. When we first find a region $R$ on the left that is colored white, then we divide $T$ by a horizontal line at the level of the bottom of $R$, coloring the bottom part black and the top part white. This is correct, because the lower-left corner of the upper part is in a region that is colored white (namely $R$), so the upper part should be colored white, and the bottom part should be black because no staircase path can enter it through the bottom or through the left side. If we reach the top of $T$ without encountering any white regions to the left, then we color $T$ black, as no staircase path can enter the bottom or left of $T$.

Finally, we must go back through the trapezoidization, merging adjacent black and white regions (for each edge of the trapezoidization, if both regions bounding the edge are the same color, then remove the edge).

The time complexity can be analyzed as follows. Trapezoidization and a plane sweep take $O(n \log n)$ time[17,22]. Adding up the work at each stop of the plane sweep, we (1) find the lower-left corner of each trapezoid once, and possibly (2) walk the entire left side of each trapezoid once, so the total work here is linear. Hence the total time is $O(n \log n)$. □.

The time complexity stated in Theorem 2 is optimal, within a constant factor. This fact can be established by demonstrating that the sorting problem can be reduced in linear time to the problem of computing s-visibility polygon from a point inside a polygon with holes.

**Theorem 3:** Sorting is linear time transformable to the problem of computing the s-visibility polygon from a point inside a polygon with holes. Therefore, finding the s-visibility polygon from a point inside a polygon of $n$ vertices with holes requires $\Omega(n \log n)$ time.

**Proof:** Given $n$ real positive numbers $x_1, x_2, x_3, ..., x_n$, we show how an algorithm $A$ for computing the s-visibility polygon can be used to sort them with only linear overhead. We construct a polygon with holes based on $x_i$'s as follows. Corresponding to each number $x_i$ we construct a thin vertical rectangle lying between points $(x_i, x_i)$ to $(x_i, 0)$. We choose point $q$ at $(0, \frac{x_s}{2})$, where $x_s$ is the smallest number among $x_i$'s. Finally we construct an enclosing square that contains point $q$ and all vertical rectangles as illustrated in Figure 3. The time needed for this construction is $O(n)$. The north west vertices of all vertical rectangles lie on the line passing through the origin and at $45^0$ with the $x$-axis. Also, these vertices appear as alternate vertices of the s-visibility polygon from point $q$. In Figure 3, the s-visibility polygon is shown shaded. By using algorithm 1, we can obtain the s-visibility polygon $V_q$ from point $q$. One scan of the boundary of the s-visibility polygon will enable us to read off $x_i$'s in the sorted order. □.

## 4 Computing s-Kernel and Recognizing s-Star Polygons

Under standard visibility, the **kernel** of a polygon $Q$ is the set of points from which all points inside $Q$ are visible. The kernel of a polygon may be empty or non-empty and a polygon with non-empty kernel is called a **star polygon**. A linear time algorithm for computing kernel of a polygon under standard visibility has been reported[(19)]. We consider the star-shape property under s-visibility. An orthogonal polygon is a **s-star** polygon (or simply s-star) if it contains a point from which all other points inside the polygon can be connected by a stair-case path, and the set of such points is its **s-kernel**. Figure 4 shows examples of s-star polygons. It is interesting to note that whereas star polygons are always simple, s-star polygons may contain holes (Figure 4b). Also, although the kernel of a polygon is always connected and convex, the s-kernel need not be connected.

In this section we develop algorithms for computing the s-kernel and for recognizing s-star polygons. From here to Theorem 5, all polygons are considered to be simple (i.e., without holes). Imagine partitioning polygon $Q$ into rectangles by extending its edges into its interior. Observe that the s-visibility polygons from points on the same rectangle are identical. Hence a straight forward way to recognize an s-star polygon is to compute the s-visibility polygon from a point on each rectangle and accept the polygon as s-star if the whole polygon is visible from at least one rectangle. Since there can be potentially $O(n^2)$ rectangles, this approach requires $O(n^3)$ time, which is rather expensive. To develop an efficient algorithm we start with few definitions. If we traverse the boundary of a simple orthogonal polygon in the clockwise direction, keeping the interior to the right, then at the vertex of the polygon we either turn $90^0$ right (outside corner) or $90^0$ left (inside corner). An edge of an orthogonal polygon whose both ends are inside corners is referred to as a **dent**[(20)]. The direction of dent traversal gives its orientation which we indicate as **N, S, E,** and **W** dents. In Figure 4a, edges (a,b), (c,d), (e,f), and (g,h) are $N, E, S$, and $W$-dents, respectively. The s-kernel can be captured in terms of critical chords. An **east critical chord** is the chord passing through the left most E-dent; if there is no E-dent then the unique right most vertical edge is taken as the east critical chord. Similarly, **west critical chord, south critical chord**, and **north critical chord** are defined. Critical chords are drawn dashed in Figure 4a. The following lemma is similar in spirit to Lemma 3.3 given in ref[(23)].

**Lemma 4**: A simple orthogonal polygon is an s-star polygon if and only if the following two conditions are satisfied: (a) The x-coordinate of the east critical chord is greater than the x-coordinate of the west critical chord, and (b) The y-coordinate of the north critical chord is greater than the y-coordinate of the south critical chord.

**Proof**:

($\Leftarrow$) We need to prove that if conditions (a) and (b) are satisfied then the polygon is s-star. Four critical chords partition the polygon into at most nine regions which we refer to as **N, S, E, W, NE, NW, SE, SW**, and **central regions** ( Figure 4a). Note that some of the regions would be empty if all dent types are not present. The central region is orthogonally convex (an orthogonal polygon is orthogonally convex is its intersection with any vertical or horizontal line is either empty or a line segment) and hence if a point is s-visible from some point in the central region then it is also s-visible from all points in the central region. We establish the if part of the lemma by showing that any point on a region can be connected to the central region by a stair case path. Since $E$ and $W$ regions lie between the south critical chord and the north critical chord, no $N$-dent or $S$-dent can lie in these regions. This implies that any point in the $E$ or $W$ region can be connected to the central region by a stair case path consisting of one horizontal segment. (If any point in the $E$-region or

$S$-region can not be connected to the central region by a horizontal line segment then $E$ region or $S$-region would contain N-dent or or S-dent, a contradiction.) Similarly, any point in the $N$ or $S$ region can be connected to the central region by a stair case path consisting of a single vertical segment. The existence of stair case path connecting points in corner regions (NE, NW, SE, and SW regions) to the central region can be argued as follows. We present the argument for $NE$ region. Arguments for other corner regions are symmetrical. Let $p$ be a point in the $NE$ region. We shoot a ray from $p$ towards the west direction. If the ray hits the east critical chord $c$ then $p$ is connected by a stair case path to the central region. If it does not hit $c$, then let $x$ be the first boundary point hit by the ray. Since the ray did not hit chord $c$, the point $x$ must be in the $NE$ region. Since the $NE$ region does not contain $S$- or $W$- dents the point $x$ must lie on a *type I boundary s-chain* (recall the definition of *type I boundary s-chain* from Section 2). We follow this s-chain until we reach its end or meet chord $c$. If chord $c$ is found then we have constructed the stair case path. If end of the s-chain is reached then we shoot a horizontal ray from the corresponding N-dent in the west direction (the south-west end of a type I s-chain is a N-dent). This ray will either meet chord $c$ or hit another *type I boundary s-chain* in the $NE$ region. We continue this process until chord $c$ is encountered. Since each *type I boundary s-chain* traversal makes an advance in the south west direction, we will eventually hit the chord $c$ and the resulting trail is a stair case path connecting $p$ to the central region.

($\Rightarrow$) We need to prove that if a polygon is s-star then conditions (a) and (b) are satisfied. Assume to the contrary that condition (a) is not satisfied. Then we must have the situation as shown in Figure 5 (i.e., the x-coordinate of the east critical chord is less than the x-coordinate of the west critical chord). Consider now the partitioning of the polygon into five parts $R_1, R_2, R_3, R_4$, and $R_5$ by these critical chords. Let us now examine which region can possibly contain the s-kernel. No point in region $R_1$ is s-visible from any point of $R_5$ and hence $R_1$ can not contain the s-kernel. Similarly, other regions cannot contain s-kernel. Hence the polygon is not s-star - a contradiction. Similar arguments establish the validity of condition (b). $\square$.

**Theorem 4**: The s-kernel of a simple orthogonal polygon can be computed in $O(n)$ time.

**Proof**: In one scan of the boundary of the polygon we can determine four critical chords by simply checking angles between consecutive edges and their coordinates, which takes $O(n)$ time. We can check for the validity of conditions (a) and (b) in constant time once critical chords are available. The boundary of the s-kernel can be determined in $O(n)$ time once critical chords are known. Thus the total time adds up to $O(n)$. The correctness of the theorem follows from Lemma 4. The optimality (in the worst case) of the time complexity follows from the fact that the s-kernel can have $O(n)$ edges. $\square$.

We next consider stair-case visibility in the exterior of the polygon. External visibility properties of polygons under standard visibility have been considered(14,21). A simple orthogonal polygon is an **external s-star** if there exists a point from which all points in the exterior of the polygon are s-visible. The **bounding box** of a simple orthogonal polygon is the smallest isothetic rectangle that encloses the polygon as shown in Figure 6 (dashed boundary). Note that a rectangle is called isothetic if its edges are parallel to coordinate axes. If a vertex of the bounding box does not coincide with a vertex of the polygon, then that vertex is called a **caved corner**. In Figure 6, the bounding box has three caved corners.

**Observation 1**: A simple orthogonal polygon is an external s-star if and only if it is externally s-visible from one of the vertices of its bounding box.

**Observation 2**: If vertex $v'$ of a bounding box of a simple orthogonal polygonal is caved, then the polygon is not externally s-visible from the vertex diagonally opposite to $v'$.

Observation 1 implies that we can recognize a simple external s-star by checking its s-visibility from its bounding box vertices. Visibility from a bounding box vertex $v$ can be checked as follows. Assume without loss of generality that $v$ is the south west vertex. Let $v'$ be the vertex of the bounding box diagonally opposite to $v$. If $v'$ is caved then the polygon is not s-visible from $v$ (Observation 2), and this can be checked by one scan of the boundary in $O(n)$ time. If $v'$ is non-caved then we construct a modified simple orthogonal polygon from the original polygon. We enclose the polygon by a rectangle slightly larger than the bounding box. We split $v'$ and the north east vertex of the rectangle and connect them by two type I stair case paths as shown in Figure 7. The resulting structure $Q'$ is the modified simple orthogonal polygon. By construction, it is clear that if the modified polygon is internally s-visible from $v$ then the original polygon is externally s-visible from $v$. Thus, if the s-visibility polygon from $v$ in $Q'$ is the same as $Q'$, then the original polygon is externally s-visible from $v$, which can be done in $O(n)$ time using Algorithm 1. We repeat the process for the remaining three vertices. We therefore have the following theorem.

**Theorem 5**: An externally s-star simple orthogonal polygon can be recognized in $O(n)$ time. Within the same time its external $s-$kernel can be computed.

We now consider the problem of computing the s-kernel of orthogonal polygons that may contain holes. For the purpose of simplifying the exposition, we make the following assumption. We will explain how to modify the algorithm when the outer boundary in not necessarily a rectangle.

**Assumption 1**: The outer boundary of the polygon (the boundary that encloses all holes) is the boundary of a rectangle.

For a polygon with holes we use the term "chord" to indicate only those horizontal or vertical line segments inside the polygon whose both end points are on the outer boundary of the polygon. This means that horizontal or vertical line segments inside the polygon with end points on the boundary of holes are not taken as chords. We need a few more definitions. A **prime chord** is a chord that contains an edge of the polygon. Prime chords partition the polygon into rows and columns of rectangular blocks and each such block is referred to as an **elementary block** (Figure 8). The partitioning itself is referred to as **prime chord partitioning**. A row of elementary blocks is called a **horizontal corridor** if no block in the row encloses holes; if some elementary block of the row encloses a hole then that row is called a **horizontal strip**. Similarly, **vertical corridor** and **vertical strip** are defined. We number rows from top to bottom. The $i^{th}$ horizontal corridor and the $i^{th}$ horizontal strip are denoted by $C_i$ and $H_i$, respectively. The $j^{th}$ elementary block in $C_i$ and $H_i$ are denoted by $C_{i,j}$ and $H_{i,j}$, respectively. A **corridor intersection** is an elementary block formed by the intersection of a vertical and horizontal corridor. Note that corridors and strips occur alternately. Corridor intersections are feasible regions to be a part of the s-kernel. This is stated in the following lemma.

**Lemma 5**: An orthogonal polygon containing holes is s-star if and only if it is s-visible from one of the corridor intersections.

**Proof**: If the polygon is s-visible from a corridor intersection, then it is trivially s-star. Conversely, suppose a polygon is s-star. Then it should be s-visible from some point $p$. The fact that $p$ must lie on a corridor intersection can be argued as follows. If $p$ does not lie on a corridor intersection then at least one of the four rays (two vertical and two horizontal) going away from $p$ must hit the boundary of a hole at point $h$. If the ray incident at $h$ is extended further, it will enter the interior of the hole and emerges out of the hole at a point $h'$. Observe that no stair-case path can connect points $p$ and $h'$, implying that point $h'$ is not s-visible from $p$ - a contradiction. $\square$

We may use Lemma 5 to check for the s-star shape property of polygon $Q$ by computing the s-visibility polygon from a point on each corridor intersection and we report the polygon to be s-star if the s-visibility polygon from some corridor intersection is the entire polygon. The time complexity of the algorithm designed by employing this strategy is rather high. Since there can be $O(n^2)$ corridor intersections and the s-visibility polygon computation from a point takes $O(n \log n)$ time, the total time becomes $O(n^3 \log n)$.

To develop a faster algorithm we identify corridor intersections satisfying certain visibility properties. A corridor intersection $C_{i,j}$ is said to **cover** $H_k$ if all points in $H_k$ are s-visible from $C_{i,j}$. Let $S(i,j) = \{C_{i,j} \mid C_{i,j}$ is a cover for $H_j$ $\}$. A corridor intersection is a **top covering rectangle** if all points in the polygon lying above it are s-visible from it. Similarly, a **bottom covering rectangle** is defined. (Note that a point $p_1$ is said to be **above** point $p_2$ if the y-coordinate of $p_1$ is greater than the y-coordinate of $p_2$; and, a region $r_1$ is said to be above region $r_2$ if all points in $r_1$ are above any point in $r_2$. The terms **below, right**, and **left** are defined analogously.) We use $T_i$ and $B_i$ to denote the set of top covering rectangles and the set of bottom covering rectangles, respectively, in corridor $C_i$. Let *BT(Q)* and *TB(Q)* denote the corridor rows of $Q$ containing the bottom most top covering rectangles and the top most bottom covering rectangles, respectively.

**Example 1**: In Figure 8, $C_{2,7}$ is a cover for $H_1$; $C_{2,3}$ is not a cover for $H_1$ because $H_{1,4}$ is not s-visible from $C_{2,3}$. $C_{10,5}$ is a bottom covering rectangle (all points of $Q$ below $C_{10,5}$ are s-visible from $C_{10,5}$). $C_{10,3}$ is not a bottom cover because $H_{10,4}$ is not s-visible from $C_{10,3}$.

**Lemma 6:** An orthogonal polygon $Q$ containing holes is s-star if and only if *BT(Q)* is not above *TB(Q)*.

**Proof**: If the condition is satisfied then some corridor intersection $C_{i,j}$ is both a top covering rectangle and a bottom covering rectangle, and hence all interior points are s-visible from $C_{i,j}$. Conversely, if the polygon is s-star then the corridor intersections on the bottom most row and the corridor intersections on the top most row of its s-kernel satisfy the condition of the lemma. □

**Example 2:** In Figure 8, *BT(Q)* is $C_6$ and *TB(Q)* is $C_4$. The s-kernel consist of nine corridor intersections: $C_{4,11}, C_{4,13}, C_{4,15}, C_{5,11}, C_{5,13}, C_{5,15}, C_{6,11}, C_{6,13}$, and $C_{6,15}$.

The corridor intersections in $T_i$, $B_i$, and $S(i,j)$ occur in alternate order in $C_i$. This is stated in the following lemma.

**Lemma 7:** If $C_{i,j} \in T_i$ and $C_{i,j+k} \in T_i$ then all corridor intersections in $C_i$ lying between $C_{i,j}$ and $C_{i,j+k}$ must be in $T_i$, i.e., $C_{i,j+r} \in T_i$, where $r$ is even and $r < k$. Identical properties hold for corridor intersections in $B_i$ and $S(i,j)$.

**Proof**: Consider a corridor intersection $C_{i,j+r}$ lying between $C_{i,j}$ and $C_{i,j+k}$. Consider any elementary block $H_{p,q}$ lying above $C_{i,j+r}$ (Figure 9). Assume without loss of generality that $H_{p,q}$ lies in the north-east region from $C_{i,j+r}$. Since $C_{i,j}$ is in $T_i$, any point in $C_{i,j}$ and any point in $H_{p,q}$ must be connectable by a stair-case path $R$. The path $R$ must intersect the vertical corridor through $C_{i,j+r}$. This means that any point in $C_{i,j+r}$ can be connected to any point in $H_{p,q}$ by a stair case path (shown by thick stair-case path in Figure 9). □

**Lemma 8**: Let $T_i = \{ C_{i,j}, C_{i,j+2}, C_{i,j+4}, ..., C_{i,j+k} \}$ be the set of top covering rectangles in $C_i$. Let $A = \{ C_{i+1,j}, C_{i+1,j+2}, C_{i+1,j+4}, ..., C_{i+1,j+k} \}$. Then, $T_{i+1}$, the set of top covering rectangles in $C_{i+1}$ is given by $T_{i+1} = A \bigcap S(i+1, i)$.

**Proof:** We observe that the same set of vertical corridors pass through corridor intersections in $A$ and $T_i$, and hence each of the elements in $A$ covers the region of $Q$ above $C_i$. No corridor intersection in $C_{i+1}$ which is not in $A$ can cover the region above $C_i$; if some corridor intersection $C_{i+1,q}$ which is not in $A$ covers the region above $C_i$ then the corridor intersection $C_{i,q}$ which is not in $T_i$ would cover the region above $C_i$, a contradiction. Thus, $A$ is the set of corridor intersections in $C_{i+1}$ each of which covers the region above $C_i$. By definition, $S(i+1,i)$ is the set of corridor intersections in $C_{i+1}$ each of which covers the region between $C_i$ and $C_{i+1}$. Hence common elements in $A$ and $S(i+1,i)$ cover the region above $C_{i+1}$. □.

**Lemma 9:** Let $H_{i,r}$ be the right most elementary block in $H_i$ that is not s-visible from its south-west vertex. Let $H_{i,l}$ be the left most elementary block in $H_i$ that is not s-visible from its south-east vertex. Then $S(i+1,i)$ exactly contains corridor intersections in $C_{i+1}$ that lie to the left of $H_{i,l}$ and to the right of $H_{i,r}$.

**Proof:** Any corridor intersection in $C_{i+1}$ that lies to the left of $H_{i,r}$ can not cover $H_{i,r}$. Any corridor intersection in $C_{i+1}$ that lies to the right of $H_{i,l}$ can not cover $H_{i,l}$. Finally, any corridor intersection in $C_{i+1}$ that is to the left of $H_{i,l}$ and to the right of $H_{i,r}$ is a cover of $H_i$. □

We are now ready to describe the algorithm. Prime chords of the polygon are determined by using the plane sweep method[22]. Prime chords partition the polygon into a planar graph consisting of rows and columns of elementary blocks. This planar graph is represented by a doubly connected edge list data structure (DCEL)[22], which facilitates the traversal of the graph easily from face to face. Vertices of elementary blocks are marked as follows. A vertex $v$ of elementary block $r$ is marked '1' if the interior of $r$ is s-visible from $v$; otherwise, it is marked '0'. The algorithm then proceeds to find *BT(Q)* and *TB(Q)* by computing $T_i$'s and $B_i$'s. $T_i$'s are computed starting from the top most corridor and processing corridors downwards, one by one. $T_1$ is given by all corridor intersections in $C_1$ (there are no holes above $C_1$). $S(i+1,i)$ is determined by checking the marking on the vertices of elementary blocks in $H_i$ (Lemma 9). $T_{i+1}$ is computed by making use of

$S(i+1,i)$ and $T_i$ (Lemma 8). The process of computing $T_i$'s stops whenever a corridor not containing any top covering rectangle is found or the bottom most corridor is encountered. *TB(Q)* is determined similarly by computing $B_i$'s starting from the bottom and proceeding upward. A formal description of the algorithm is listed as Algorithm 2.

**Algorithm 2**

**Input:** Orthogonal polygon $Q$ possibly containing holes.
**Output:** s-kernel of $Q$.
**Step 1:** Partition the polygon $Q$ into strips and corridors by determining prime chords and represent the resulting planar graph by doubly connected edge list (DCEL) data structure[22].
**Step 2:** Mark vertices of elementary blocks as follows. If the interior of an elementary block is s-visible from its vertex $v$ then $v$ is marked '1'; otherwise, it is marked '0'.
**Step 3:** {Determine *BT(Q)* }.
(a) Set $T_1$ to the set of corridor intersections in $C_1$; i=i+1;
(b) While ($C_{i-1}$ is not the bottom most corridor) and ($T_{i-1}$ is not empty) do
(i) Find $S(i,i-1)$ by examining markings on the vertices of elementary blocks in $H_{i-1}$ (application of Lemma 9);
(ii) Compute $T_i$ by using $T_{i-1}$ and $S(i,i-1)$ (Lemma 8);
(iii) $i=i+1$;
end {while}
(c) *BT(Q)* is given by $T_{i-1}$.
**Step 4:** Determine *TB(Q)* by computing $B_i$'s (starting from the bottom most corridor) as in Step 3.
**Step 5:** Top covering rectangles lying between (and including) corridors *TB(Q)* and *BT(Q)* give the s-kernel of $Q$ (Lemma 6).

**Theorem 6**: The s-Kernel of an orthogonal polygon with holes can be computed in $O(n^2)$ time.

**Proof**: Prime chords can be determined by using plane the sweep technique[22] in $O(n \log n)$ time. The size of the planar graph formed by the intersection of prime chords and polygon $Q$ is $O(n^2)$ and this graph can be maintained in a DCEL data structure in time proportional to the size of the graph[22]. Hence Step 1 takes $O(n^2)$ time. Whether the interior of an elementary block in $H_i$ is s-visible from its four vertices can be determined in $O(n_i \log n_i)$ time using the result of Theorem 2, where $n_i$ is the number of vertices of $Q$ inside the elementary block $H_i$. Hence the s-visibility of all elementary blocks (from respective four vertices) can be computed in $O(n_1 \log n_1 + n_2 \log n_2 + ... + n_k \log n_k) = O(n \log n)$ time. Thus, markings in Step 2 take $O(n \log n)$ time. One iteration of Step 3bi can be done in $O(n)$ time by checking markings on vertices of elementary blocks in $H_i$. Corridor intersections in $T_i$ and $S(i+1, i)$ can be found in $O(n)$ time by examining $C_i$ and $C_{i+1}$ in parallel. The while loop executes in $O(n)$ time and hence Step 3 takes $O(n^2)$ time. Step 4 and Step 5 take $O(n^2)$ time, and hence the total time adds up to $O(n^2)$. The correctness of the algorithm follows from Lemma 6. □.

The s-kernel of an orthogonal polygon with holes may contain $\Omega(n^2)$ connected components (Figure 10) and hence $O(n^2)$ time is the lower bound for computing the s-kernel of an orthogonal polygon with holes.

**Removal of Assumption 1**: Consider the situation when the outer boundary of polygon $Q$ is not rectangular. Let $Q'$ be the simple orthogonal polygon obtained by removing all holes of polygon $Q$. Let $Q''$ be the polygon obtained by enclosing holes of $Q$ by a rectangular boundary. We first compute s-kernels $R'$ and $R''$ of $Q'$ and $Q''$, respectively. We observe that the s-kernel $K$ of $Q$ is given by $K = R' \bigcap R''$, which can be computed in $O(n^2)$ time.

The $O(n^2)$ algorithm for computing s-kernel could be used to recognize s-star polygons with holes. It is interesting to develop sub-quadratic time complexity algorithms to recognize s-star polygons with holes, without having to output its s-kernel. The main bottle neck in this direction is the size of the planar graph formed by the polygon and the intersection of prime chords. The approach used in Algorithm 2 is the computation of top covering rectangles (and bottom covering rectangles) for each horizontal corridor. Since there can be $O(n^2)$ top covering (or bottom covering) rectangles in total, it is not possible to come up with a sub-quadratic time algorithm by explicitly maintaining covering rectangles.

We next consider how to adapt Algorithm 2 to develop a sub-quadratic time complexity algorithm to recognize s-star polygons with holes. We define the **upper x-extent** (respectively, **lower x-extent**) of a polygon to be the upper horizontal edge (respectively, lower horizontal edge) of its bounding box (recall the definition of bounding box, Figure 7a). Our algorithm is based on computing the lower x-extents of top covering rectangles (and upper x-extents of bottom covering rectangles) without constructing an $O(n^2)$ planar graph. Let $Q_i$ denote the polygon obtained by enclosing $C_i, H_i$, and $C_{i+1}$ by a rectangular boundary (see $Q_{11}$ in Figure 8).

**Key Idea:** We note the following two facts. (i) The total number of elementary blocks in the prime partitioning of $Q$ is $O(n^2)$. (ii) The total number of elementary blocks in the prime partitioning of all $Q_i$'s is $O(n)$. The correctness of (i) is obvious: $n$ vertical prime chords and $n$ horizontal prime chords can intersect to form at most $O(n^2)$ regions, and Figure 10 shows an example that this bound can be achieved. The correctness of (ii) can be argued as follows. There are two horizontal corridors, one horizontal strip, and $O(n_i)$ vertical prime chords in the prime partitioning of polygon $Q_i$, where $n_i$ is the number of vertices of polygon $Q_i$. This implies that the number of elementary blocks in the prime partitioning of $Q_i$ is $O(n_i)$, which in turn implies that the total number of elementary blocks in all $Q_i$'s is $O(n_1) + O(n_2) + O(n_3) + ... + O(n_k) = O(n)$. (In Figure 10, the total number of elementary blocks in the prime partitioning of $Q$ is 23*23 = 529. The total number of elementary blocks in the prime partitioning of $Q_1$, $Q_2$ and $Q_{11}$ are 11, 5, and 5 respectively.) The above facts lead us to conclude that a faster s-star recognition algorithm could be developed if x-extents of *BT(Q)* and *TB(Q)* can be computed by examining the s-visibility properties of elementary blocks of $Q_i$'s, rather than elementary blocks of $Q$.

**Observation 3**: Let $X_i$ be the x-extent of *BT($Q_i$).* Let $W_i$ be the set of prime chords of

$Q$ that intersect with $X_i$. The smallest sub-segment of $X_i$ that intersect with all chords in $W_i$ gives the x-extent of $S(i, i-1)$.

Observation 3 is illustrated in Figure 11. Dashed vertical lines are vertical prime chords of $Q$. Solid vertical lines are vertical prime chords of $Q_i$. The algorithm for recognizing an s-star polygon $Q$ by computing x-extents of $BT(Q_i)$ and $TB(Q_i)$ is listed as Algorithm 3. Algorithm 3 mimics Algorithm 2 in many respects. The main (and crucial) differences are: (i) Algorithm 3 makes use of s-visibility properties of elementary blocks of $Q_i$'s, rather than those of $Q$, and (ii) it maintains the x-extents of $T_i$'s, rather than $T_i$'s themselves.

**Algorithm 3**

**Input:** Orthogonal polygon $Q$, possibly having holes.
**Output:** 'Yes' if $Q$ is s-star and 'No' otherwise.
**Step 1:** Partition the polygon $Q$ into horizontal strips and horizontal corridors by determining horizontal prime chords; find $Q_i$'s.
**Step 2:** Determine vertical prime chords $W$ and represent them in a list sorted by x-coordinates.
**Step 3:** Partition $Q_i$'s into elementary blocks by determining their vertical prime chords. Mark the vertices of elementary blocks of $Q_i$'s as follows. If the interior of an elementary block of $Q_i$ is s-visible from its vertex $v$ then $v$ is marked '1'; otherwise, it is marked '0'.
**Step 4:** {Determine the lower x-extent of *BT(Q)* }.
{$R_i$ is the lower x-extent of the top covering rectangles in $T_i$.}
(a) $i = 1$; Set $R_1$ to the lower x-extent of $C_1$; $i = i + 1$;
(b) While (all $R_{i-1}$'s are not determined) and ($R_{i-1}$ is not empty) do
(i) Find $X_i$ by by checking markings on the vertices of elementary blocks of $Q_i$ (Apply Lemma 9).
(ii) Determine the lower x-extent of $S(i, i-1)$ from $W$ and $X_i$ by using Observation 3.
(iii) Determine $R_i$ by using $R_{i-1}$ and x-extent of $S(i, i-1)$ (Lemma 8).
(iv) $i = i + 1$;
end {while}
(c) The x-extent of *BT(Q)* is given by $R_{i-1}$.
**Step 5:** Determine the upper x-extent of *BT(Q)* by following the method as in Step 3.
**Step 6:** Compare the lower x-extent of *BT(Q)* and the upper x-extent of *TB(Q)* to recognize whether polygon $Q$ is s-star or not (Lemma 6).

**Theorem 7:** S-Star polygon, possibly containing holes, can be recognized in $O(n \log n)$ time.

**Proof:** Step 1 and Step 2 can be done in $O(n \log n)$ time by using plane sweep methods[22]. Vertical prime chords of all $Q_i$'s can be computed in $O(n \log n)$ time; the s-visibility polygons (from its four vertices) of an elementary block $r$ of $Q_i$ can be determined in $O(m_j \log m_j)$, where $m_j$ is the number of vertices of $Q$ inside $r$; hence the total time to mark the vertices of elementary blocks of all $Q_i$'s is $O(m_1 \log m_1 + m_2 \log m_2 + ... + m_k \log m_k) = O(n \log n)$; thus Step 3 takes $O(n \log n)$ time. One iteration of Step 4bi takes $O(n_i)$ time, where $n_i$ is the number of elementary blocks of $Q_i$; hence the total time for all iteration of Step 4bi is $O(n_1) + O(n_2) + O(n_3)... = O(n)$. The list $W$ is sorted and hence the x-extent of $S(i, i-1)$ can be found by performing binary search of the x-coordinates of end points of $X_i$ in $W$, and this takes $O(\log n)$ time. Thus all iterations of Step 4bii take $O(n \log n)$ time. All iterations of Step 4biii take $O(n)$ time. Therefore Step 4 and Step 5 take $O(n \log n)$ time. Step 6 takes $O(1)$ time and the overall time complexity becomes $O(n \log n)$. The correctness of the algorithm follows from Lemma 6. $\square$

## 5 Conclusion

We presented algorithms to compute the s-visibility polygon from a point inside an orthogonal polygon. The algorithm for simple orthogonal polygons runs in $O(n)$ time and that for orthogonal polygons with holes runs in $O(n \log n)$ time. Both algorithms are optimal within a constant factor. We presented a simple construction with linear overhead to convert an instance of the problem of computing the s-visibility polygon from a point in the exterior of the polygon to an instance of the problem of computing the s-visibility polygon from an interior point.

We presented an optimum $O(n)$ time algorithm for computing the s-kernel of simple orthogonal polygons. We showed that the s-kernel of an orthogonal polygon with holes can be computed in $O(n^2)$ time. Even though the s-kernel of a polygon with holes may have $\Omega(n^2)$ components we showed that it is possible to recognize s-star polygons in $O(n \log n)$ time.

Schuierer and Wood have used a different approach[18] to prove that s-kernels of polygons with holes can be computed in $O(n^2)$ time. Their approach is to compute an external s-star for each hole and compute their intersection. Although the approach of Schuierer and Wood does not lead to a sub-quadratic time complexity recognition algorithm, it has its own merit to include *O-Staircase* visibility[18]. Our algorithm to compute s-kernel leads to an $O(n \log n)$ recognition algorithm. We do not compute the intersection of external s-kernels of holes. In fact we compute s-kernels of cluster of holes (which we call elementary blocks) and avoid computing their intersection. We compute the intersection only one time: to find the common area between the s-kernel of polygon $Q'$ and the s-kernel of polygon $Q''$ (see Removal of Assumption 1 in Section 4). Our algorithm can be generalized to include non-orthogonal polygons without increasing complexity: reflex vertices play the role of dent lines and the algorithm generalizes naturally.

It is not clear if the $O(n \log n)$ time algorithm to recognize s-star polygons with holes is optimal or not. It would be therefore interesting to obtain a faster recognition algorithm or to prove that the complexity is optimal.

**Acknowledgements:** We thank Patrice Belleville, Simeon Ntafos, and Tom Shermer for many helpful suggestions. Tom Shermer and Patrice Belleville suggested the proof of Theorem 2. We also thank a referee for many helpful suggestions and detailed criticism of the paper.

## References


[1] Y. K. Hwang and N. Ahuja, "Gross Motion Planning - A Survey", *ACM Computing Survey*, Vol. 24, No. 3, 1992, pp. 219-291.

[2] R. A. Motwani, A. Raghunathan and H. Saran, "Covering Orthogonal Polygons With Star Polygons: The Perfect Graph Approach", *Journal of Computer and Systems Sciences,* Vol. 40, 1990, pp. 19-48.

[3] L. Gewali, and S. Ntafos, "Covering Grids and Orthogonal Polygons With Periscope Guards", To Appear in *Computational Geometry: Theory and Applications*, 1993.

[4] G. Toussaint, (Editor), Computational Morphology, North Holland, 1988.

[5] D. Wood and P. Yamamoto, "Dent and StairCase Visibility", *Proc. of 5th Canadian Conference on Computational Geometry*, Waterloo, 1993.

[6] P. Bose and G. Toussaint, "Geometric and Computational Aspects of Injection Molding", Technical Report No. SOCS 92.16, School of Computer Science, McGill University, Canada, 1992.

[7] G. Toussaint, "Pattern Recognition and Geometric Complexity", *Proc. 5th International Conference on Pattern Recognition*, 1980.

[8] H. ElGindy and D. Avis, "A Linear Time Algorithm for Computing the Visibility Polygon From a Point", Journal of Algorithms, Vol. 2, pp. 186-197, 1981.

[9] D. T. Lee, "Visibility of a Simple Polygon", *Computer Vision, Graphics, And Image Processing*, Vol. 22, pp. 207-221, 1983.

[10] T. Asano, "An Efficient Algorithm for Finding the Visibility Polygon for a Polygonal Region With Holes", *Transactions of IECE of Japan*, E 68(9), pp. 557-559, 1985.

[11] L. Gewali, M. Keil, and S. Ntafos, "On Covering Orthogonal Polygons With Star-Shaped Polygons", *Information Sciences*, Vol. 65, pp. 45-63, 1992.

[12] L. Gewali, "On Computing Staircase Visibility Polygon", *Proceedings of SPIE (Vision Geometry)*, 1993.

[13] T. Shermer, "Recent Results in Art Galleries", *Proceedings of The IEEE*, Vol. 80, No. 9, September 1992.

[14] J. O,Rourke, "Art Gallery Theorem and Applications", Oxford University Press, 1987.

[15] G. T. Toussaint, "Movable Separability of Sets", *Computational Geometry*, Edited by G. T. Toussaint, North Holland Publishing Company, 1985.

[16] B. Chazelle, "Triangulating a Simple Polygon in Linear Time", *Proc. 31st Annual Symp. on Foundation of Computer Science*, pp.220-230, 1990.

[17] A. Fournier and D. Y. Moutuno, "Triangulating Simple Polygons and Equivalent Polygons", *ACM Transactions on Graphics*, Vol. 3, No. 2, pp. 153-174, 1984.

[18] S. Schuierer and D. Wood, "Generalized Kernels of Polygons With Holes", *Proc. of The Fifth Canadian Conference on Computational Geometry*, 1993.

[19] D. T. Lee and F. P. Preparata, "An optimal algorithm for finding the kernel of a simple polygon", *Journal of the ACM*, Vol. 26, 1979, pp. 415-421.

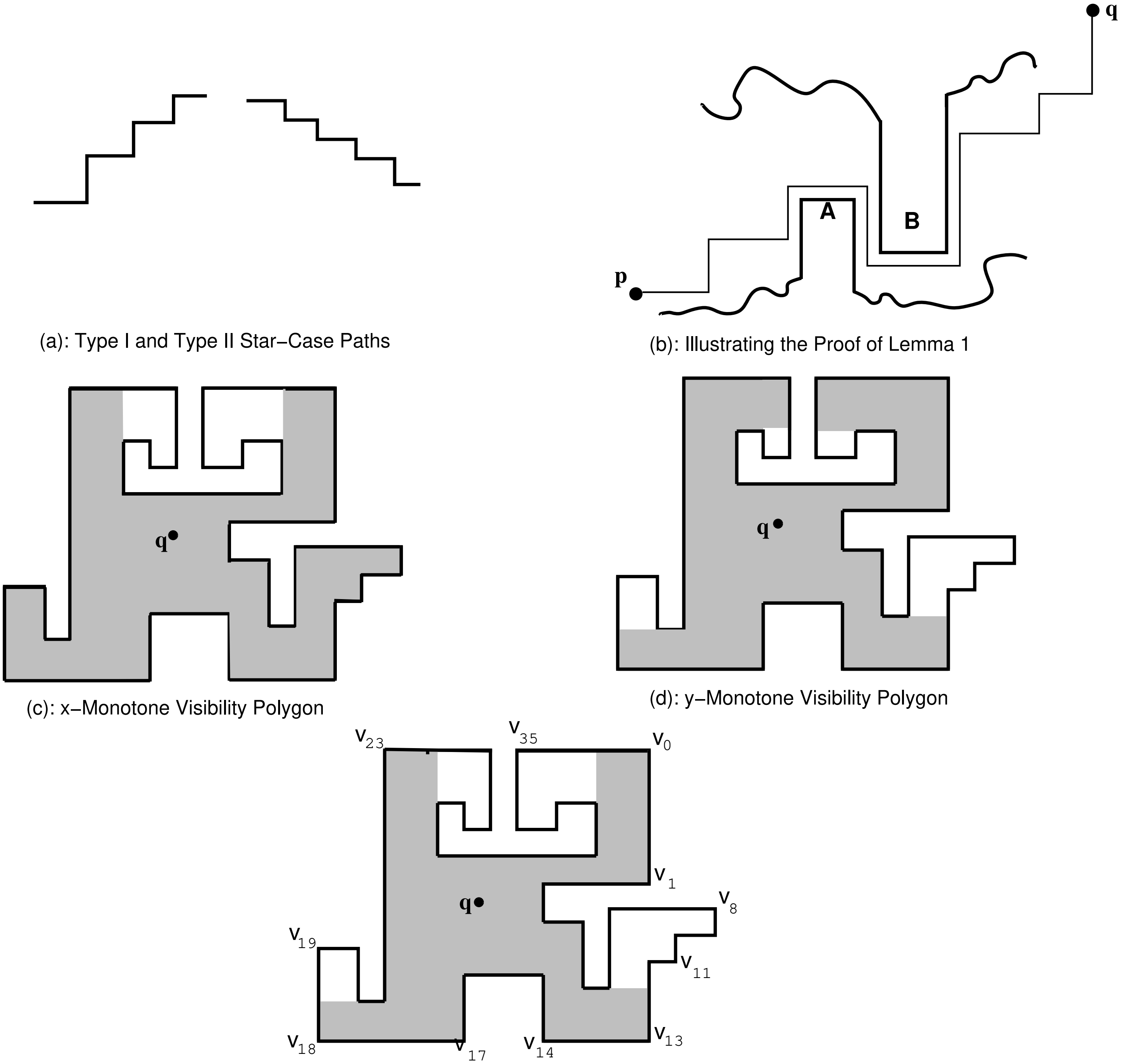
[20] R. Reckhow and J. Culberson, "Covering Simple Orthogonal Polygons With a Minimum Number of Orthogonally Convex Polygons", *Proceedings of the Third Annual Symposium on Computational Geometry* (1987), pp. 268-277.

[21] B. K. Bhattacharya, A. Mukhopadhyaya, and G. T. Toussaint, "A Linear Time Algorithm for Computing the Shortest Line Segment from Which a Polygon is Weakly Externally Visible", *Proceedings of the WADS* (Lecture Notes in Computer Science, No. 519), 1991, pp. 412-424.

[22] F. P. Preparata, and M. I. Shamos, "Computational Geometry - An Introduction", Springer Verlag, 1985.

[23] S. Schuierer, G. Rawlins, and D. Wood, "A Generalization of Stair-Case Visibility", *Proceedings of the Third Canadian Conference on Computational Geometry*, 1991.

[24] Thomas Shermer and Patrice Belleville, Personal communication, 1994.




(a): Type I and Type II Star−Case Paths

(b): Illustrating the Proof of Lemma 1

(c): x−Monotone Visibility Polygon

(d): y−Monotone Visibility Polygon

(e): s−Visibility Polygon

Figure 1: Illustrating Stair−Case Path and s−Visibility Polygon

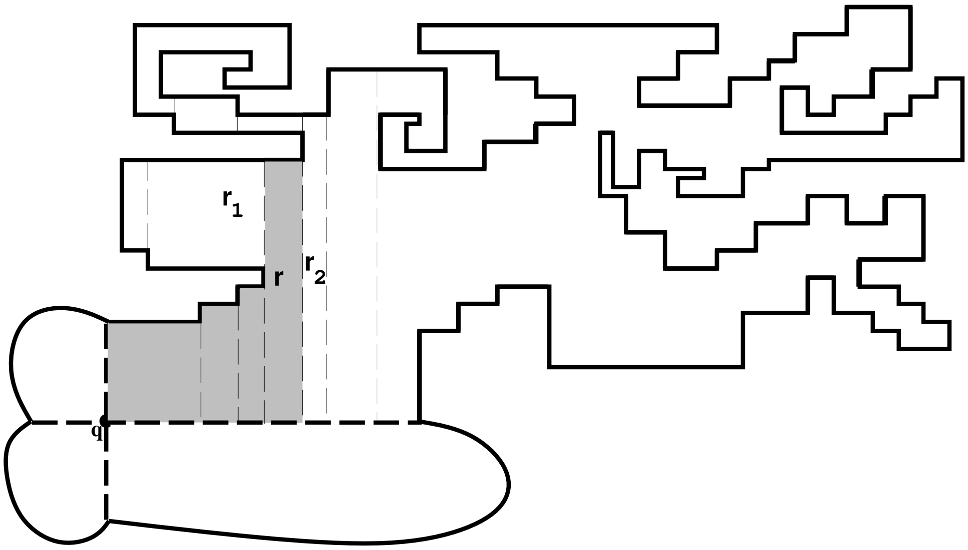



Figure 2: Illustrating the Construction of x−Monotone Visibility Polygon in NE−Part (Shaded)

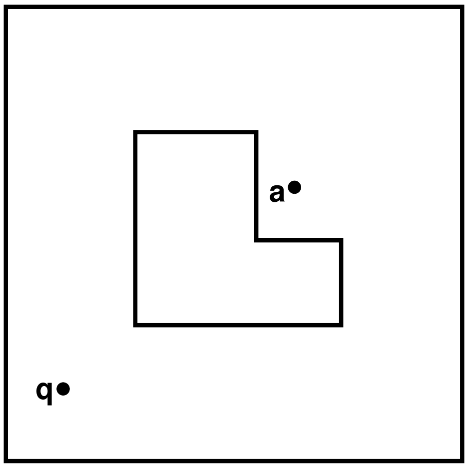



Figure 3a: Construction for Remark 1

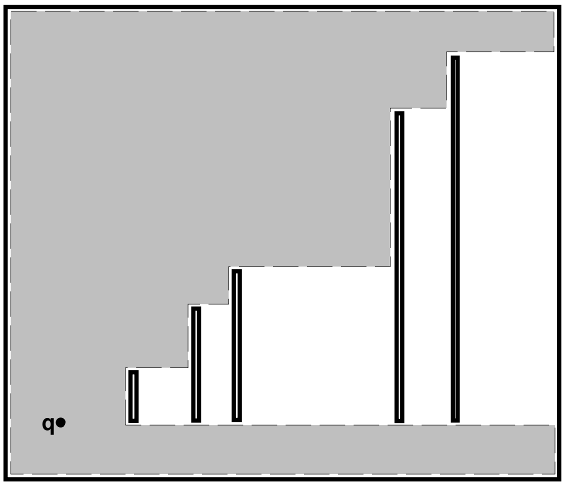



Figure 3b: Construction for Reducing Sorting Problem to the Problem of Computing s−Visibility Polygon

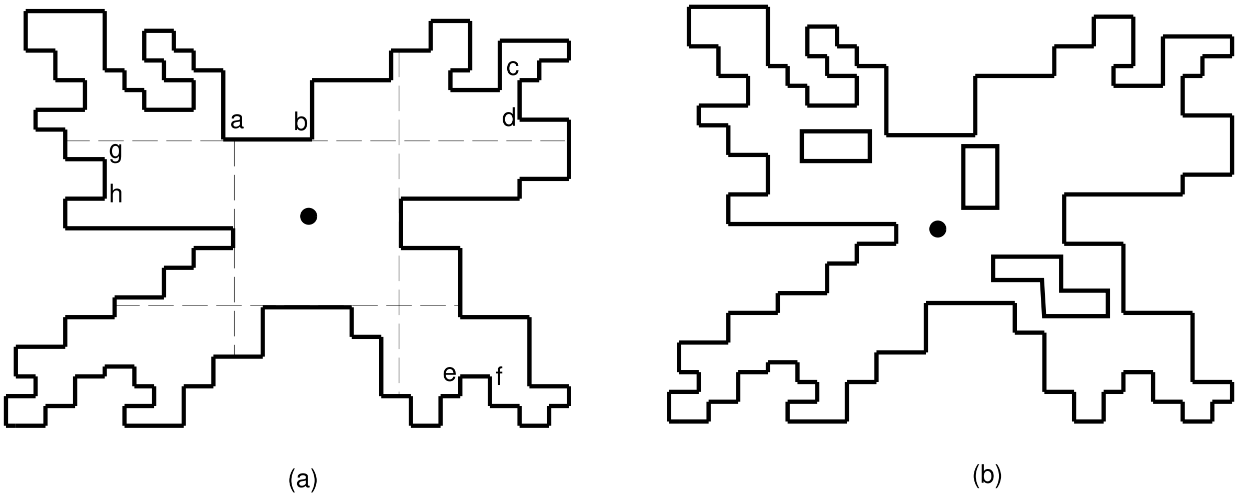



Figure 4: Examples of s−Star Polygons

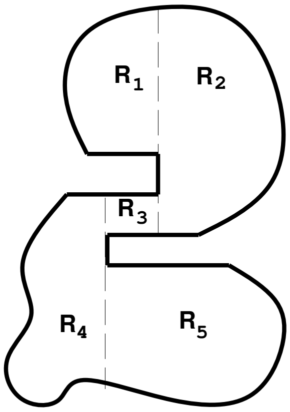



Figure 5: Illustrating the Proof of Lemma 4

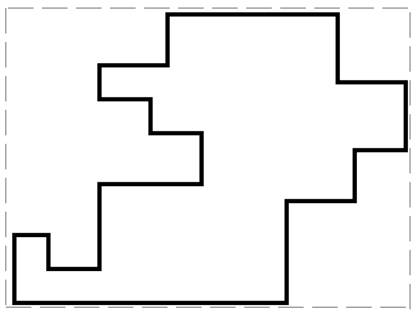
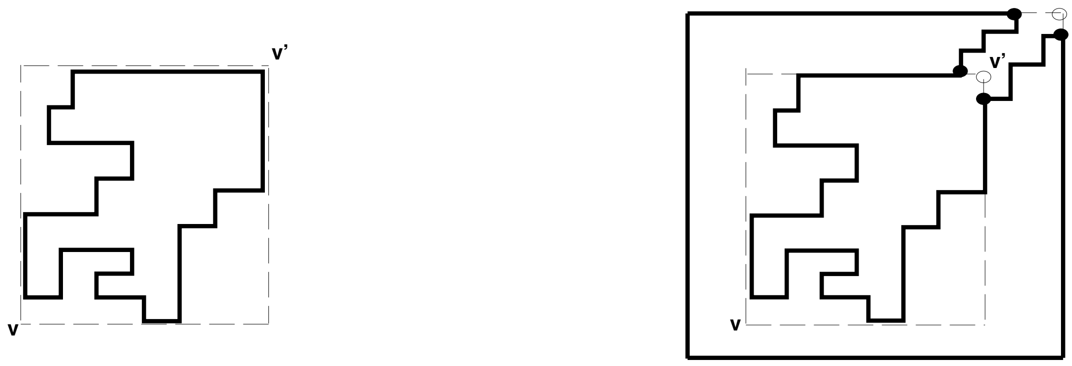


Figure 6: Illustrating Bounding Box and Caved Corner




(a): Original Polygon (b): Modified Polygon

Figure 7: Reducing External Visibility Problem to Internal Visibility Problem

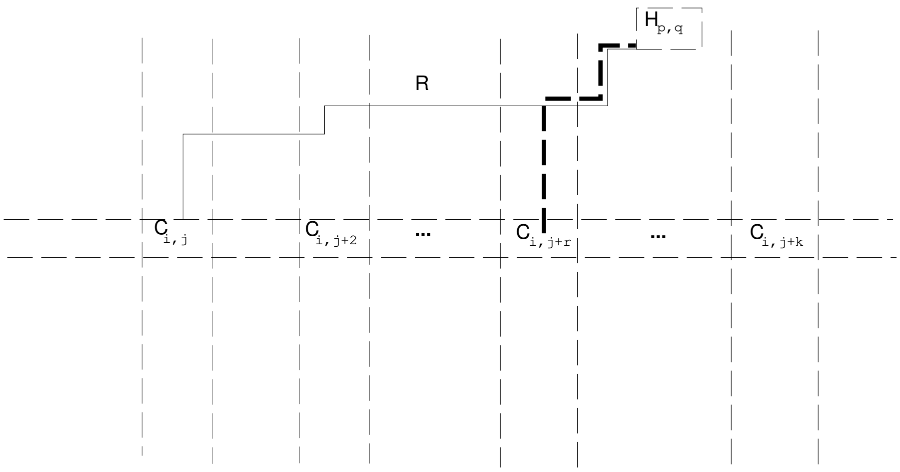



Figure 9: Illustrating the Proof of Lemma 7

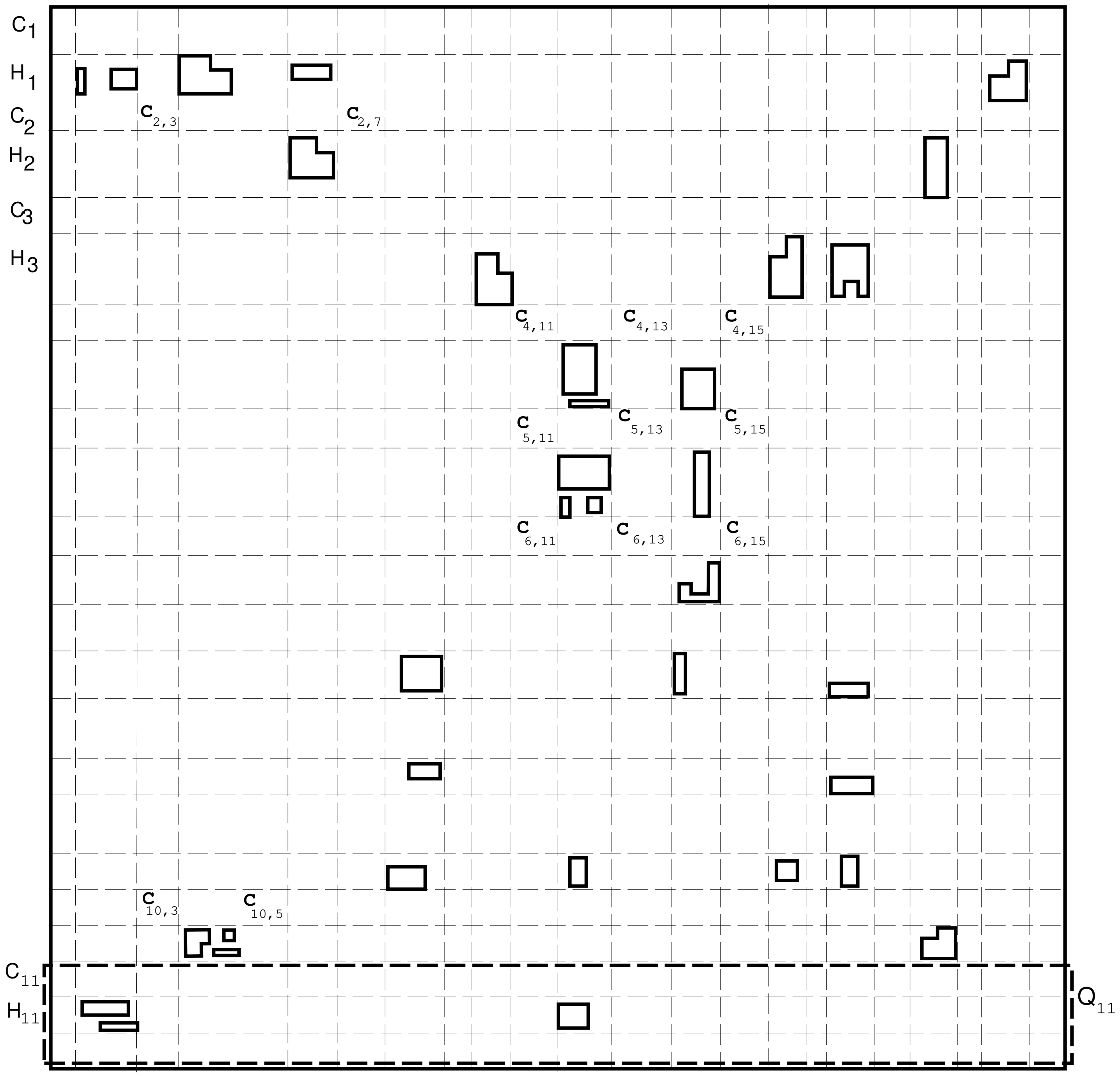



Figure 8: Partitioning the Polygon into Corridors and Strips

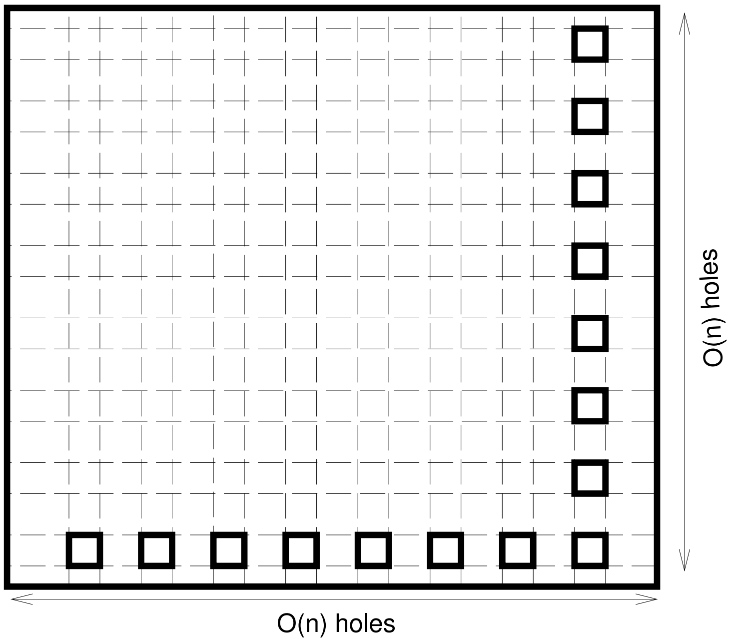



Figure 10: An Example Showing That s−Kernel Of a Polygon With Holes Can Have O(n**2) Components

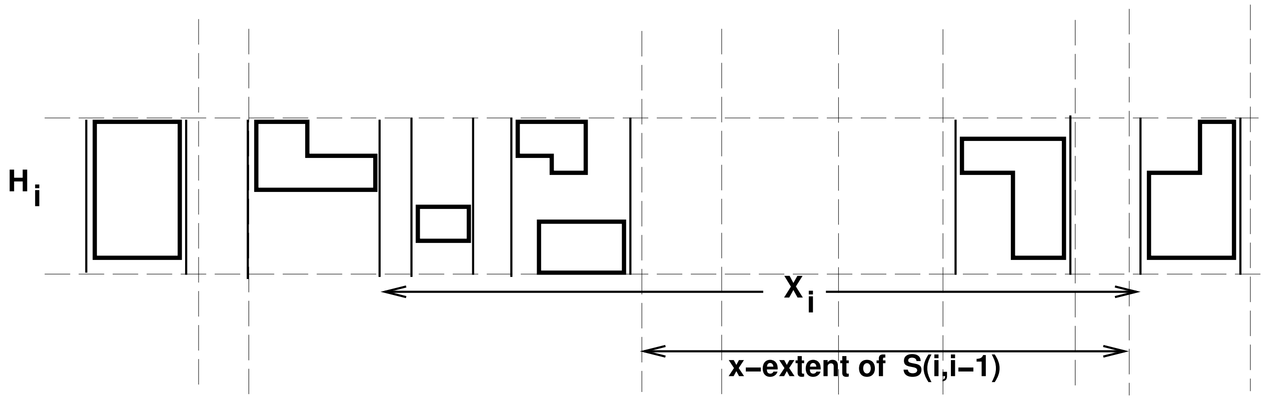



Figure 11: Illustrating Observation 3