Guarding Monotone Art Galleries with Sliding Cameras in Linear Time

Mark de Berg[a,1], Stephane Durocher[b,2], Saeed Mehrabi[c,*]

[a] *Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands.*
[b] *Department of Computer Science, University of Manitoba, Canada.*
[c] *Cheriton School of Computer Science, University of Waterloo, Canada.*

Abstract

A sliding camera in an orthogonal polygon P—that is, a polygon all of whose edges are axis-parallel—is a point guard g that travels back and forth along an axis-parallel line segment s inside P. A point p in P is guarded by g if and only if there exists a point q on s such that line segment pq is normal to s and contained in P. In the minimum sliding cameras (MSC) problem, the objective is to guard P with the minimum number of sliding cameras.

We give a dynamic programming algorithm that solves the MSC problem exactly on monotone orthogonal polygons in $O(n)$ time, improving the 2-approximation algorithm of Katz and Morgenstern (2011). More generally, our algorithm can be used to solve the MSC problem in $O(n)$ time on simple orthogonal polygons P for which the dual graph induced by the vertical decomposition of P is a path. Our results provide the first polynomial-time exact algorithms for the MSC problem on a non-trivial subclass of orthogonal polygons.

Keywords: Art gallery problem, Orthogonal polygons, Sliding cameras,

[☆] A preliminary version of this paper appeared in proceedings of the 8th International Conference on Combinatorial Optimization and Applications (COCOA 2014) [1].

[*] Corresponding author

Email addresses: mdberg@win.tue.nl (Mark de Berg), durocher@cs.umanitoba.ca (Stephane Durocher), smehrabi@uwaterloo.ca (Saeed Mehrabi)

[1] Work of the author is supported by the Netherlands Organisation for Scientific Research (NWO) under project 024.002.003.

[2] Work of the author is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

Dynamic programming

2010 MSC: 00-01, 99-00

---

1. Introduction

Art gallery problems, introduced by Klee in 1973 [2], are some of the most widely studied problems in computational geometry. In the original formulation of the problem, an input polygon P is given, for which a set of point guards must be assigned, using as few guards as possible. Thus, the objective is to find a set of point guards such that every point in P is seen by at least one of the guards, where a guard g sees a point p if and only if the segment gp is contained in P. Chvátal [3] proved that $\lfloor n/3 \rfloor$ point guards are always sufficient and sometimes necessary to guard a simple polygon with n vertices. Lee and Lin [4] showed that finding the minimum number of point guards needed to guard an arbitrary polygon is NP-hard for arbitrary polygons. The art gallery problem is also NP-hard for orthogonal polygons [5] and it even remains NP-hard for monotone polygons [6]. Moreover, Eidenbenz [7] proved that the problem is APX-hard on simple polygons.

Ghosh [8] gave an $O(\log n)$-approximation algorithm that runs in $O(n^4)$ time on simple polygons. King and Kirkpatrick [9] gave an $O(\log \log \mathrm{OPT})$-approximation algorithm for the vertex guards (and in fact when the guards can be anywhere on the boundary of the polygon), where OPT is the size of an optimal solution. Their algorithm is based on the fact that there is an ϵ-net of size $O(\frac{1}{\epsilon} \log \log \frac{1}{\epsilon})$ for the corresponding hitting set problem. Notice that the existence of such an ϵ-net along with the technique of Bronnimann and Goodrich [10] provides the desired approximation factor. King [11] improved the running time of this algorithm to $O(n^3)$ for simple polygons. Krohn and Nilsson [6] gave a constant-factor approximation algorithm on monotone polygons. They also gave a polynomial-time algorithm for the orthogonal art gallery problem that computes a solution of size $O(\mathrm{OPT}^2)$, where OPT is the cardinality of an optimal solution. In terms of parameterized complexity of the

art gallery problem, Bonnet and Miltzow [12] recently showed that the problem is W[1]-hard parameterized by the number of guards. See the surveys by O'Rourke [2] or Urrutia [13] for a history of the art gallery problem.

Many variants of the art gallery problem have been studied [14, 15, 16, 17, 18]. The version in which we are interested was introduced recently by Katz and Morgenstern [19], and it concerns *sliding cameras* in orthogonal polygons. A sliding camera in an orthogonal polygon P is a point guard that travels back and forth along an axis-parallel line segment $s \subset P$. The point guard can see a point $p \in P$ if and only if there is a point $q \in s$ such that the line segment pq is normal to s and contained in P. The *minimum sliding cameras (MSC) problem* is to guard P with the minimum number of sliding cameras.

Related Work. Katz and Morgenstern [19] first considered a restricted version of the MSC problem in which only *vertically sliding cameras* are allowed; that is, point guards that travel back and forth along a segment s that is parallel to the y-axis. They solved this restricted version in polynomial time for simple orthogonal polygons. For the unrestricted version, where both vertically and horizontal sliding cameras are allowed, they gave a 2-approximation algorithm for x-monotone orthogonal polygons. An orthogonal polygon P is x-monotone if the intersection of P with any every vertical line is connected. Durocher and Mehrabi [20] showed that the MSC problem is NP-hard when the polygon P is allowed to have holes. They also considered a variant of the problem, called the MLSC problem, in which the objective is to minimize the sum of the lengths of line segments along which cameras travel, and proved that the MLSC problem is polynomial-time solvable even on orthogonal polygons with holes (see also [21]). For orthogonal polygons with holes, Biedl et al. [22] showed that the problem is still NP-hard if only horizontal sliding cameras are allowed. They also gave an $O(1)$-approximation algorithm for the MSC problem based on ϵ-nets, and showed that the problem becomes polynomial-time solvable if the dual graph of a so-called pixelation of the polygon has bounded treewidth.

Biedl et al. [23] studied the MSC problem under k-visibility; that is, the line

of sight of a camera can intersect the boundary of the polygon at most k times (note that when $k = 0$, we have the standard MSC problem studied in this paper). The k-visibility has been already studied under the classical art gallery problem [18, 17, 24]. Biedl et al. [23] showed that the MSC problem under k-visibility is NP-hard on simple orthogonal polygons for any $k > 0$, even if the polygon is monotone. They also gave an $O(1)$-approximation algorithm for any fixed $k > 0$. Seddighin [25] proved that the MLSC problem (i.e., minimizing the total length of cameras) is NP-hard under k-visibility for any fixed $k > 0$.

Our main interest is in the standard MSC problem, where the objective is to minimize the number of cameras. As discussed above, the complexity of the MSC problem on simple orthogonal polygons remains unknown. Indeed, even for x-monotone orthogonal polygons there is only an approximation algorithm for the problem. Recall that the classical art gallery problem is NP-hard on simple orthogonal polygons [5], simple monotone polygons [6] and even on terrains [26].

Our Results. In this paper, we give a linear-time dynamic programming algorithm for the MSC problem on orthogonal x-monotone polygons P. This not only improves the 2-approximation algorithm of Katz and Morgenstern [19], but also provides, to the best of our knowledge, the first polynomial-time algorithm for the MSC problem on a non-trivial subclass of orthogonal polygons. We also show how to extend this result to so-called *orthogonal path polygons*. These are orthogonal polygons for which the dual graph induced by the vertical decomposition of P is a path. (The vertical decomposition of an orthogonal polygon P is the decomposition of P into rectangles obtained by extending the vertical edge incident to every reflex vertex of P inward until it hits the boundary of P. The dual graph of the vertical decomposition is the graph that has a node for each rectangle in the decomposition and an edge between two nodes if and only if their corresponding rectangles are adjacent.) Observe that the class of orthogonal monotone polygons is a subclass of orthogonal path polygons.

2. Preliminaries

A polygon is orthogonal if all of its edges are axis-parallel. A simple orthogonal polygon P is *x-monotone* if the intersection of P with any vertical line is at most one single line segment. For a simple orthogonal and x-monotone polygon P, the leftmost and rightmost vertical edges of P are unique and we denote them by leftEdge(P) and rightEdge(P), respectively. For a sliding camera s in P, we define the *visibility polygon* of s as the maximal subpolygon $P(s)$ of P such that every point in $P(s)$ is guarded by s.

Let $V_P = \{e_1 = \text{leftEdge}(P), e_2, \ldots, e_m = \text{rightEdge}(P)\}$, for some $m > 0$, be the set of vertical edges of P ordered from left to right. For simplicity we assume that every two vertical edges in V_P have distinct x-coordinates, but it is easy to adapt the algorithm to handle degenerate cases. Let P_i^+ (resp., P_i^-), for some $1 \leq i \leq m$, denote the subpolygon of P that lies to the right (resp., to the left) of the vertical line through e_i.

For an axis-parallel line segment s in P, we denote the left endpoint and the right endpoint of s by left(s) and right(s), respectively. If s is vertical, we define its left and right endpoints to be its upper and lower endpoints, respectively. We denote the x-coordinate of a point p by $x(p)$. Let s_i and s_j be two horizontal line segments in P. We define the *overlap region of s_i and s_j* as the set of points in P that are visible to both s_i and s_j; if $P(s_i) \cap P(s_j)$ is a line or a point (i.e., it has measure zero), then we consider the overlap region of s_i and s_j to be empty. We first show that we can restrict our attention to solutions that are in some suitable canonical form.

Canonical Form.

A *feasible solution* to the MSC problem is a set M of sliding cameras that guards the entire polygon P. We say that a feasible solution M is in *canonical form* if and only if the following properties hold:

(i) Every vertically sliding camera in M is *vertically maximal*, meaning that it extends as far upwards and downwards as possible.

(ii) No vertically sliding camera in M intersects the interior or passes through the right endpoint of any horizontal sliding camera in M.

(iii) The overlap region of s_i and s_j is empty, for every two horizontal sliding cameras $s_i, s_j \in M$ such that $s_i \neq s_j$.

(iv) Every horizontal sliding camera $s \in M$ is *rightward maximal*, meaning that s extends at least as far to the right as any horizontal sliding camera $s' \subset P$ starting at the same x-coordinate; that is, with $x(\text{left}(s)) = x(\text{left}(s'))$.

(v) Let $s_1, \ldots, s_k$ be the sequence of line segments (corresponding to sliding cameras) in M ordered from left to right according to their left endpoint, where in case of ties vertical line segments come before horizontal line segments, and let $M_i := \{s_1, \ldots, s_i\}$. Then, M_i guards every point of P that is to the left of the vertical line $x = x(right(s_i))$.

Lemma 1. *For any x-monotone orthogonal polygon P, there exists an optimal solution M for the MSC problem on P that is in canonical form.*

PROOF. Consider the sequence $s_1, \ldots, s_k$ of line segments in M ordered from left to right according to their left endpoint, where in case of ties vertical line segments come before horizontal line segments. This ordering is well defined, because an optimal solution will never have two vertical line segments with the same x-coordinates or two horizontal line segments whose left endpoints have the same x-coordinates. We now show how to modify the line segments in M to get an optimal solution in canonical form. Without loss of generality, we assume that all vertical line segments in M are already vertically maximal.

We first modify M so that if s_1 is horizontal, then $\text{left}(s_1)$ lies on $\text{leftEdge}(P)$, the leftmost vertical edge of P. Assume this is not the case. Then, $\text{leftEdge}(P)$ is seen by a vertical line segment s_j, for some $j > 1$. We now replace s_1 and s_j by two horizontal line segments, as follows. The first line segment is a rightward maximal line segment s starting on $\text{leftEdge}(P)$—note that s must intersect s_j—and the second horizontal line segment is a rightward maximal line segment s'

with $x(\text{left}(s')) = x(\text{right}(s))$. Clearly replacing s_1, s_j by s, s' gives another optimal solution. With a slight abuse of notation we let M denote this new optimal solution, and we let $s_1, \ldots, s_k$ denote the ordered set of line segments in the new solution. Note that we now have that if s_1 is horizontal, then it starts at leftEdge(P) and it is rightward maximal.

Next we turn M into an optimal solution in canonical form. To this end we go over the line segments in order. When we handle line segment s_i we will replace s_i by a line segment s'_i, but we will not modify any other line segment. Let $M_i := \{s'_1, \ldots, s'_i\}$. We maintain the following invariant:

> *Invariant:* After handling s_i, the modified set M is still an optimal solution. Moreover, M_i has all the required properties: (i) all vertical line segments in M_i are vertically maximal, (ii) no vertical line segment in M_i intersects the interior or passes through the right endpoint of any horizontal line segment in M_i, (iii) the overlap region of any two horizontal line segments in M_i is empty, (iv) every horizontal line segment in M_i is rightward maximal, and (v) M_i guards everything to the left of the vertical line $x = x(\text{right}(s_i))$.

Handling s_1 is trivial: we simply set $s'_1 := s_1$. If s_1 is vertical then this clearly establishes the invariant—note that no line segment s_j with $j > 1$ can see anything to the left of s_1 that is not also seen by s_1 (since s_1 is vertically maximal), which implies that s_1 must see everything to its left. If s_1 is horizontal then the invariant holds as well, since we already made sure that s_1 is rightward maximal if it is horizontal. Now suppose the invariant holds after we have handled s_{i-1}, and consider s_i. There are two cases.

- If s_i is vertical, then we proceed as follows. Observe that M_i must guard everything to the left of s_i, since M is feasible and no line segment s_j with $j > i$ can see a point to the left of s_i that is not seen by s_i. Since s_i is vertical, the fact that M_{i-1} satisfies the invariant immediately implies that M_i has properties (iii) and (iv). So the only problem is that s_i may intersect the interior or may pass through the right endpoint of some

line segment s'_j with $j < i$. If this is not the case we simply set $s'_i := s_i$, otherwise we replace s_i by a rightward maximal line segment s'_i with $x(\text{left}(s'_i)) = x(\text{right}(s'_j))$.

After the replacement, M is still a feasible (and, hence, optimal) solution. Indeed, everything to the left of the vertical line $x = x(\text{right}(s'_j))$ is guarded by M_j and to the right of the vertical line $x = x(\text{right}(s'_j))$, the new line segment s'_i sees at least as much as s_i. By the same argument, M_i guards everything to the left of the vertical line $x = x(\text{right}(s'_i))$ and, therefore, the property (v) holds. Finally, M_i has properties (i) and (ii) because M_{i-1} had those properties and the new line segment s'_i is horizontal, and M_i has properties (iii) and (iv) by construction.

There is one subtlety that we must address. Namely, we have to show that after replacing s_i by s'_i the order of the line segments does not change. In other words, we must show that s'_i is still the i-th line segment in the order. (Otherwise we would have to argue about a different set M_i.) Obviously $\text{left}(s'_i)$ lies to the right of $\text{left}(s'_j)$ for all $j < i$. Moreover, there cannot be any line segment s_k with $k > i$ such that $\text{left}(s_k)$ lies in between s_i and $\text{left}(s'_i)$. Indeed, such a line segment could be omitted due to monotonicity of P, contradicting the optimality of M.

- If s_i is horizontal, we proceed as follows. Obviously, the only properties that may be violated are properties (iii) and (iv). It might be the case that a vertical line segment $s_j \in M$ intersects the interior or passes through the right endpoint of s_i (thus violating the property (ii)), but this may happen only if $j > i$ and, therefore, the invariant is still maintained for M_i; if such line segment s_j exists, then the set M will be modified when we later handle s_j. If M_i only violates property (iv), then we replace s_i by a rightward maximal line segment s'_i with $x(\text{left}(s'_i)) = x(\text{left}(s_i))$. If s_i violates property (iii), then let $s'_j \in M_{i-1}$ be the horizontal line segment that has an overlap with s_i. We now replace s_i by a rightward maximal line segment s'_i with $x(\text{left}(s'_i)) = x(\text{right}(s'_j))$.

Since s'_i sees at least as much as s_i (except possibly for points that were already seen by s'_j), the new solution is still feasible. Moreover, M_i sees everything to the left of the vertical line $x = x(\text{right}(s'_i))$. Therefore, since M_{i-1} satisfies the invariant and because of the way s'_i is constructed, M_i has all the properties (i)–(v). Finally, the new line segment s'_i is still the i-th line segment in the order, as can be verified in the same way as before.

After handling the last line segment s_k in M, the set M_k is an optimal solution in canonical form, thus proving the lemma. □

3. A Dynamic Programming Algorithm

In this section, we present the linear-time exact algorithm for the MSC problem on orthogonal x-monotone polygons. Our algorithm is based on a dynamic programming approach [27].

3.1. The Recursive Structure

Let P be an orthogonal x-monotone polygon with n vertices. Below we discuss the recursive structure of the MSC problem on P and we define the subproblems we use in our dynamic programming algorithm.

Let $M_{\text{OPT}} = \{s_1, \ldots, s_k\}$ be an optimal solution for the MSC problem on P that is in canonical form, where the segments are numbered from left to right. Consider a segment $s_j \in M_{\text{OPT}}$. By property (v) of the canonical form, no segment $s_{j'} \in M_{\text{OPT}}$ with $j' > j$ is needed to guard anything to the left of $\text{right}(s_j)$. Hence, after having selected $s_1, \ldots, s_j$, the subproblem we are left with is to guard P_i^+, where i is such that $\text{right}(s_j)$ lies on the line containing the vertical edge e_i. Note that when s_j is vertical, we already guarded a part of P_i^+, and we have to take this into account in our subproblem. Hence, we define two types of subproblems.

Type A. Given $1 \leq i \leq k$, guard P_i^+ with the minimum number of sliding cameras.

Type B. Given $1 \leq i \leq k$, guard P_i^+ with the minimum number of sliding cameras, under the assumption that the subregion of P_i^+ that is visible from leftEdge(P_i^+) has already been guarded.

We denote the number of guards needed in an optimal solution of Type A on the polygon P_i^+ by $A[i]$ and the number of guards needed in an optimal solution of Type B on the polygon P_i^+ by $B[i]$. Note that the minimum number of cameras needed to guard the entire polygon P is $A[1]$. In the sequel we show how to compute the values $A[i]$ and $B[i]$; computing the actual solution can then be done in a standard manner.

3.2. Solving the Subproblems

We now give the recursive formulas for our dynamic programming algorithm. Recall that the vertical edges of P are number $e_1, \ldots, e_k$ from left to right. We denote the vertical line containing e_i by $\ell(e_i)$. The following lemma gives the recursive formula for solving the subproblem of Type A on P_i^+.

Lemma 2. *Let s be a rightward maximal line segment whose left endpoint lies on $\ell(e_i)$, and let e_{i_1} be the vertical edge of P on which* right(s) *lies. Furthermore, let e_{i_2} be the rightmost vertical edge of P such that s', the vertically maximal segment aligned with e_{i_2}, guards everything of P_i^+ lying to the left of e_{i_2}. See Figure 1 for an illustration. Then,*

$$A[i] = \begin{cases} 0 & \text{if } i = k \\ \min\left(A[i_1], B[i_2]\right) + 1 & \text{if } i < k \end{cases}$$

Proof. Trivially $A[i] = 0$ for $i = k$, so assume $i < k$.

Consider the first segment s^* of an optimal solution for P_i^+ that is in canonical form. By property (v), we know that s^* must guard leftEdge(P_i^+). Hence, if it is horizontal, it must start at leftEdge(P_i^+). By property (iv), segment s^* is rightward maximal. Hence, if the first segment is horizontal then the segment s is the correct choice. After choosing s, we have to guard everything to the right of s. Note that properties (ii) and (iii) imply that the next segment to be

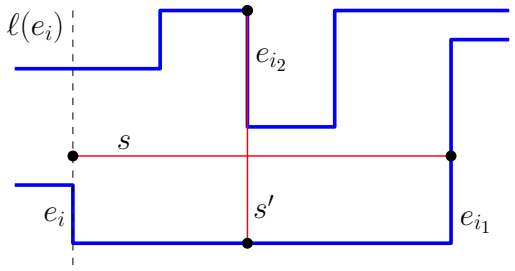


Figure 1: An illustration of the two cases in solving subproblem of Type A on P_i^+.

chosen lies in $P_{i_1}^+$ (see Figure 1). Hence, if we decide to pick segment s then we are indeed left with solving the subproblem of Type A on $P_{i_1}^+$. Thus in this case $A[i] = A[i_1] + 1$.

The other option is that the first segment s^* is vertical. Again, by property (v) we know that s^* must guard everything between leftEdge(P_i^+) and s^*. But then it is obviously best to choose s^* as far to the right as possible. Hence, s' is the correct choice. Now the subproblem we are left with is of Type B and on $P_{i_2}^+$ (see Figure 1), so we have $A[i] = B[i_2] + 1$.

The best way to solve subproblem of Type A on P_i^+ is the best of these two options, which proves the lemma. □

For the subproblems of Type B we have a similar lemma.

Lemma 3. *Let $e_{i'}$ be the leftmost vertical edge in P_i^+ that is not seen by* leftEdge(P_i^+)*, let s be a rightward maximal line segment whose left endpoint lies on $\ell(e_{i'})$, and let e_{i_1} be the vertical edge of P on which* right(s) *lies. Furthermore, let e_{i_2} be the rightmost vertical edge of P such that s', the vertically maximal segment aligned with e_{i_2}, together with* leftEdge(P_i^+) *guards everything of P_i^+ lying to the left of e_{i_2}. See Figure 2 for an illustration. Then,*

$$B[i] = \begin{cases} 0 & \text{if } i = k \\ \min\left(A[i_1], B[i_2]\right) + 1 & \text{if } i < k \end{cases}$$

Proof. Trivially $B[i] = 0$ for $i = k$, so assume $i < k$.

Consider the first segment s^* of an optimal solution for P_i^+ that is in canonical form. First, suppose that s^* is horizontal. Obviously it is best to make s^*

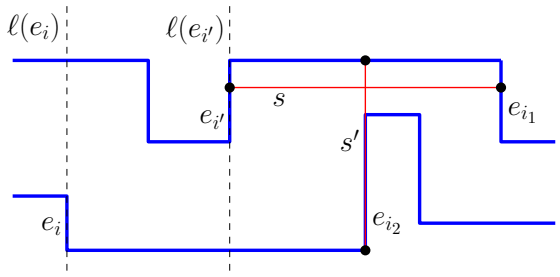


Figure 2: An illustration of the two cases in solving subproblem of Type B on P_i^+.

extend to the right as much as possible, which means left(s^*) should be to the right as far as possible. However, left(s^*) cannot go beyond $e_{i'}$ by property (v). By property (iv), segment s^* is rightward maximal. Hence, if the first segment is horizontal then the segment s is the correct choice. After choosing s, we have to guard everything to the right of s. Note that properties (ii) and (iii) imply that the next segment to be chosen lies in $P_{i_1}^+$. Moreover, since s is rightward maximal and starts to the right of leftEdge(P_i^+), the edge leftEdge(P_i^+) cannot see anything to the right of right(s). Hence, if we decide to pick segment s then we are indeed left with solving the subproblem of Type A on $P_{i_1}^+$ (see Figure 2). Thus, in this case $A[i] = A[i_1] + 1$.

The other option is that the first segment s^* is vertical. Again, by property (v) we know that s^*, together with leftEdge(P_i^+), must guard everything between leftEdge(P_i^+) and s^*. But then it is best to choose s^* as far to the right as possible. Hence, s' is the correct choice. Now the subproblem we are left with is of Type B and on $P_{i_2}^+$ (see Figure 2), so we have $A[i] = B[i_2] + 1$.

The best way to solve subproblem of Type A on P_i^+ is the best of these two options, which proves the lemma. □

3.3. Algorithmic Details

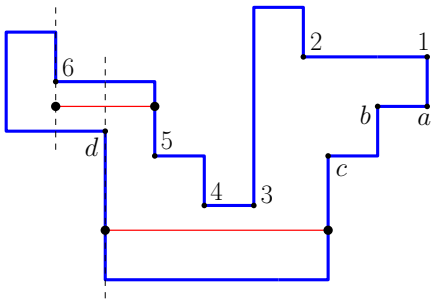
In this section, we analyze the algorithm and describe how it can be implemented in linear time. To compute the optimal solution for guarding P_i^+, we need to solve two subproblems; that is, we need to solve a subproblem of Type A and a subproblem of Type B for P_i^+. To solve the subproblem of Type A for

P_i^+, we need to solve two subproblems: one is of Type A for which we need to find the vertical edge e_{i_1} described in Lemma 2, and the other one is of Type B for which we need to find the vertical edge e_{i_2} described in Lemma 2. Similarly, to solve the subproblem of Type B for P_i^+, we need to solve two subproblems: one is of Type A for which we need to find the vertical edge e_{i_1} described in Lemma 3, and the other one is of Type B for which we need to find the vertical edge e_{i_2} described in Lemma 3. Therefore, each vertical edge $e_i \in V_P$ is associated with at most four other vertical edges of P; we call these four edges the *associated edges of* e_i. In the following, we show how the associated edges can be computed in $O(n)$ time for all the vertical edges in V_P.

We first give some definitions. A reflex vertex v of P is called *right* reflex (resp. *left* reflex) if the interior of P lies to the right (resp., to the left) of the vertical edge incident to v. Moreover, for a reflex vertex v_i of P, we denote the vertical edge incident to v_i by e_i and the maximal vertical line segment in P aligned with e_i by L_i.

Lemma 4. *The associated edges of all the vertical edges in V_P can be computed in $O(n)$ time.*

Proof. We show that each of the four types of associated edges can be computed in linear time for all the vertical edges in V_P. In the following, we assume that the sequence of the reflex vertices of P ordered from right to left is given. (1) First, we compute the associated edge e_{i_1} described in Lemma 2. To this end, we use a vertical line sweeping P from right to left; the sweep line halts at each reflex vertex of P. Let `UQ` and `LQ` be two double-ended queues that store the reflex vertices, respectively, on the upper chain and lower chain of P. By one exception, we assume that `UQ` (resp., `LQ`) contains initially the upper vertex (resp., the lower vertex) of rightEdge(P). Reflex vertices are added to the end of the queues, but they might be removed from either the front or the end of the queues. The vertices are removed from a queue depending on whether the vertex v_i at which the sweep line is currently halted lies on the upper chain or on the lower chain of P and also depending on where v_i lies on the chain relative



(a)

(b)

	front	end
UQ	2, 3, 4, 5	
LQ	c	

(c)

	front	end
UQ	5	
LQ	d	

Figure 3: An illustration of the sweep line algorithm. (a) An orthogonal x-monotone polygon P with its reflex vertices on the upper and lower chains labeled from right to left. (b) The status of queues UQ and LQ when the sweep line halts at vertex d and the invariant is maintained: the associated edge e_{i_1} for the vertical edge incident to d is set to the vertical edge incident to vertex c and vertex d is then added to LQ. (c) The status of queues UQ and LQ when the sweep line halts at vertex 6 and the invariant is maintained: the associated edge e_{i_1} for the vertical edge incident to 6 is set to the vertical edge incident to vertex 5 and vertex 6 is then added to UQ.

to the previously visited vertices. We maintain the following invariant:

> *Invariant:* When the sweep line halts at the reflex vertex v_i, (i) the queue UQ stores a reflex vertex v_j of the upper chain if and only if v_j lies to the right of L_i and L_i can see at least one point on L_j; the part of L_j that is visible to L_i is also stored. The vertices in UQ are sorted from right to left by their x-coordinate, and (ii) the queue LQ stores a reflex vertex $v_{j'}$ of the lower chain if and only if $v_{j'}$ lies to the right of L_i and L_i can see at least one point on $L_{j'}$; the part of L_j that is visible to L_i is also stored. The vertices in LQ are sorted from right to left by their x-coordinate.

Consider v_i, the vertex at which the sweep line is currently halted, and suppose that v_x and v_y are the vertices at the front of the two queues. First, we maintain the invariant. To this end, if the part of L_j that is visible to L_i is empty, then we remove v_j from UQ for all v_j in UQ. Similarly, if the part of $L_{j'}$ that is visible to L_i is empty, then we remove $v_{j'}$ from LQ for all $v_{j'}$ in LQ. Next, we set the

associated edge for e_i to e_x or e_y whichever is further to the right from e_i. The vertex v_i is then added to the appropriate queue. See Figure 3 for an example. Since every reflex vertex of P is added to a queue at most once, this step can be completed in $O(n)$ time.

(2) To find the associated edge e_{i_2} described in Lemma 2 for a vertical edge e_i, we note that e_{i_2} is in fact the edge in V_P for which the reflex vertex v incident to it is the leftmost left reflex vertex of P such that $x(v) > x(v_i)$; such vertex v and, therefore, its incident vertical edge e_{i_2} can be computed in linear time for all the vertical edges in V_P.

(3) To compute the associated edge e_{i_1} described in Lemma 3 for an edge $e_i \in V_P$, we first need to compute the vertical edge $e_{i'}$ of P. The edge $e_{i'}$ for e_i is the edge $e \in V_P$ such that the reflex vertex v incident to e is the leftmost right reflex vertex of P such that $x(v) > x(v_i)$. Then, the edge e_{i_1} for e_i is exactly the associated edge that we have already computed for $e_{i'}$ in (1).

(4) Finally, to find the associated edge e_{i_2} described in Lemma 3 for a vertical edge e_i, we first find the leftmost right reflex vertex v_j such that $x(v_j) > x(v_i)$; observe that every point of P that lies between L_i and L_j (i.e., the maximal vertical line segments in P aligned with e_i and e_j, respectively) is guarded by leftEdge(P_i^+). Therefore, the associated edge e_{i_2} for e_i is in fact the vertical edge that is furthest to the right from L_j such that every point between L_j and L_{i_2} is guarded by L_{i_2}. But, L_{i_2} is aligned with exactly the associated edge that we have already computed for e_j in (2). Therefore, to compute the associated edge e_{i_2} for e_i, we first find the leftmost right reflex vertex v_j to the right of e_i and then return the associated edge computed in Step 2 for e_j.

Therefore, we can compute all the four associated edges in $O(n)$ time for all the vertical edges in V_P. This completes the proof of the lemma. □

By Lemma 4, we first compute the associated edges of all the vertical edges of P in $O(n)$ time. Then, we consider the vertical edges of P in order from right to left and compute the optimal solution for guarding P_i^+ in $O(1)$ time by computing $A[i]$ and $B[i]$ as described, respectively, in Lemma 2 and Lemma 3.

Finally, $A[1]$ is returned as the optimal solution for the MSC problem on P. Therefore, we have the main result of this section:

Theorem 5. *There exists an algorithm that solves the MSC problem on any simple orthogonal and x-monotone polygon with n vertices in $O(n)$ time.*

4. Orthogonal Path Polygons

In this section, we show that the dynamic programming algorithm given in Section 3 can be used to solve the MSC problem on any orthogonal path polygon P with n vertices in $O(n)$ time; that is, we show that the MSC problem can be solved in $O(n)$ time on any simple orthogonal polygon P for which the dual graph $G(P)$ is a path. To this end, we first describe the structure of P and then will show that P can be converted into an x-monotone polygon by *unfolding*.

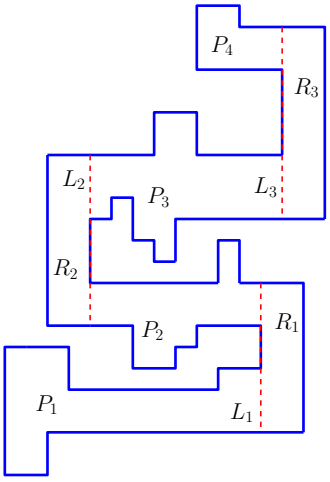


Figure 4: An example of an orthogonal path polygon P that is not x-monotone along with an illustration of partitioning P into x-monotone subpolygons.

Let P be an orthogonal path polygon with n vertices. If P is x-monotone, then we solve the MSC problem on P in linear time by Theorem 5. If polygon P is not x-monotone, then we first partition P into x-monotone subpolygons as follows. Since polygon P is not x-monotone, it must have a vertical edge e whose both endpoints are reflex vertices of P. Partition P into three subregions by the maximal vertical line segment L that is aligned with e. The subregions induced by L are a rectangle R and two subregions P_L and P_U that are connected to lower and upper parts of one of the sides of R, respectively. Partition P_L and P_R recursively until the subregions induced by the partitions become x-monotone; see Figure 4 for an illustration. Let $P_1, P_2, \ldots, P_k$ be the set of x-monotone subpolygons of P from bottom to top. Moreover, let L_i, for all $1 \leq i < k$, be the maximal line segment by which we perform the partition

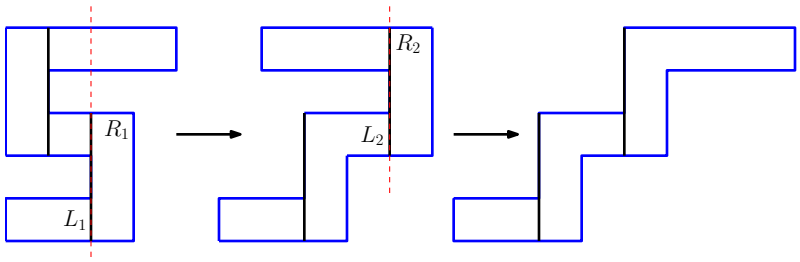


Figure 5: An illustration of transforming a non-x-monotone polygon into an x-monotone polygon by *unfolding* the polygon.

and let R_i, for all $1 \leq i < k$, be the corresponding rectangle. Now, for each rectangle R_i in order, we unfold P by flipping the subregion $P_{i+1} \cup P_{i+2} \cup \cdots \cup P_k$ across the line through L_i such that R_{i+1} lies to the same side of L_i as R_i lies. The ith flip ensures that the subregion $P_1 \cup P_2 \cup \cdots \cup P_{i+1}$ of P is an x-monotone polygon. Therefore, polygon P is converted to an x-monotone polygon after the last flip. See Figure 5 for an illustration.

To summarize, we first convert P into an x-monotone polygon using at most $k < n$ flip operations as described above and then solve the MSC problem on the resulting x-monotone polygon using the dynamic programming algorithm given in Section 3. We can compute the set of line segments L_i, for all $1 \leq i < k$, in $O(n)$ time by detecting each vertical edge of P whose both endpoints are reflex vertices of P. Next, by keeping track of the lower and upper chains of P starting from L_1, we can compute the flipped polygon in $O(n)$ time. Therefore, we have the following theorem:

Theorem 6. *There exists an algorithm that solves the MSC problem on any orthogonal path polygon with n vertices in $O(n)$ time.*

5. Conclusion

In this paper, we gave a linear-time exact dynamic programming algorithm for the problem of guarding a simple orthogonal and x-monotone polygon with the minimum number of sliding cameras (i.e., the MSC problem). This im-

proves the 2-approximation algorithm of Katz and Morgenstern [19]. Moreover, we showed that our dynamic program can be used to solve the MSC problem on orthogonal polygons for which the dual graph induced by the vertical decomposition of P is a path (i.e., orthogonal path polygons).

As the main open problem, the complexity of the MSC problem on simple orthogonal polygons remains open. Biedl et al. [22] gave an $O(1)$-approximation algorithm for the MSC problem, but the constant is likely large due to the use of ϵ-nets [10]. Can we get an improved approximation algorithm? In particular, does the problem admit a PTAS or it is APX-hard?

Another direction for future work is to consider the MSC problem under k-visibility; i.e., the sliding k-transmitters problem [23]. Biedl et al. [23] proved that the problem is NP-hard (even on simple and monotone polygons) for any $k > 0$, and gave an $O(1)$-approximation algorithm for this problem on any orthogonal polygon; could our algorithm be used or adapted to improve the approximation factor on orthogonal and monotone polygons? Finally, what are the other nontrivial classes of orthogonal polygons for which our algorithm could be used to solve the MSC problem in polynomial or better in linear time?

References

[1] M. de Berg, S. Durocher, S. Mehrabi, Guarding monotone art galleries with sliding cameras in linear time, in: Proceedings of 8th International Conference on Combinatorial Optimization and Applications (COCOA 2014), Maui, HI, USA, 2014, pp. 113–125.

[2] J. O'Rourke, Art gallery theorems and algorithms, Oxford University Press, Inc., New York, NY, USA, 1987.

[3] V. Chvátal, A combinatorial theorem in plane geometry, J. Comb. Theory, Ser. B 18 (1975) 39–41.

[4] D. T. Lee, A. K. Lin, Computational complexity of art gallery problems, IEEE Trans. Inf. Theory 32 (2) (1986) 276–282.

[5] D. Schuchardt, H.-D. Hecker, Two NP-hard art-gallery problems for ortho-
polygons, Math. Logic Quart. 41 (2) (1995) 261–267.

[6] E. Krohn, B. J. Nilsson, Approximate guarding of monotone and rectilinear polygons, Algorithmica 66 (3) (2013) 564–594.

[7] S. Eidenbenz, Inapproximability results for guarding polygons without holes, in: Proc. ISAAC, Vol. 1533 of LNCS, 1998, pp. 427–436.

[8] S. K. Ghosh, Approximation algorithms for art gallery problems in polygons, Disc. App. Math. 158 (6) (2010) 718–722.

[9] J. King, D. G. Kirkpatrick, Improved approximation for guarding simple galleries from the perimeter, Discrete & Computational Geometry 46 (2) (2011) 252–269.

[10] H. Brönnimann, M. T. Goodrich, Almost optimal set covers in finite VC-dimension, Discrete & Computational Geometry 14 (4) (1995) 463–479.

[11] J. King, Fast vertex guarding for polygons with and without holes, Comput. Geom. 46 (3) (2013) 219–231.

[12] É. Bonnet, T. Miltzow, Parameterized hardness of art gallery problems, in: 24th Annual European Symposium on Algorithms (ESA 2016), Aarhus, Denmark, 2016, pp. 19:1–19:17.

[13] J. Urrutia, Art gallery and illumination problems, in: Handbook Comp. Geom., North-Holland, 2000, pp. 973–1027.

[14] R. Motwani, A. Raghunathan, H. Saran, Covering orthogonal polygons with star polygons: the perfect graph approach, in: Proc. ACM SoCG, 1988, pp. 211–223.

[15] C. Worman, J. M. Keil, Polygon decomposition and the orthogonal art gallery problem, Int. J. Comp. Geom. & App. 17 (2) (2007) 105–138.

[16] T. C. Biedl, M. T. Irfan, J. Iwerks, J. Kim, J. S. B. Mitchell, The art gallery theorem for polyominoes, Discrete & Computational Geometry 48 (3) (2012) 711–720.

[17] B. Ballinger, N. Benbernou, P. Bose, M. Damian, E. D. Demaine, V. Dujmovic, R. Y. Flatland, F. Hurtado, J. Iacono, A. Lubiw, P. Morin, V. S. Adinolfi, D. L. Souvaine, R. Uehara, Coverage with k-transmitters in the presence of obstacles, J. Comb. Optim. 25 (2) (2013) 208–233.

[18] O. Aichholzer, R. F. Monroy, D. Flores-Peñaloza, T. Hackl, J. Urrutia, B. Vogtenhuber, Modem illumination of monotone polygons, in: European Workshop on Computational Geometry, also available at `https://arxiv.org/abs/1503.05062`, 2009, pp. 167–170.

[19] M. J. Katz, G. Morgenstern, Guarding orthogonal art galleries with sliding cameras, Int. J. Comp. Geom. & App. 21 (2) (2011) 241–250.

[20] S. Durocher, S. Mehrabi, Guarding orthogonal art galleries using sliding cameras: algorithmic and hardness results, in: Proc. MFCS, Vol. 8087 of LNCS, 2013, pp. 314–324.

[21] S. Mehrabi, Geometric optimization problems on orthogonal polygons: hardness results and approximation algorithms, Ph.D. thesis, University of Manitoba, Winnipeg, Canada (August 2015).

[22] T. C. Biedl, T. M. Chan, S. Lee, S. Mehrabi, F. Montecchiani, H. Vosoughpour, On guarding orthogonal polygons with sliding cameras, Proceedings of the 11th International Conference and Workshops on Algorithms and Computation (WALCOM 2017), Hsinchu, Taiwan.

[23] T. C. Biedl, S. Mehrabi, Z. Yu, Sliding k-transmitters: Hardness and approximation, in: Proceedings of 28th Canadian Conference on Computational Geometry (CCCG 2016), Vancouver, BC. Canada, 2016.

[24] S. Cannon, T. G. Fai, J. Iwerks, U. Leopold, C. Schmidt, Combinatorics and complexity of guarding polygons with edge and point 2-transmitters, arXiv: https://arxiv.org/abs/1503.05681.

[25] S. Seddighin, Guarding polygons with sliding cameras, Master's thesis, Sharif University of Technology (2014).

[26] J. King, E. Krohn, Terrain guarding is NP-hard, SIAM J. Comput. 40 (5) (2011) 1316–1339.

[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms (3. ed.), MIT Press, 2009.