Conflict-Free Chromatic Guarding of Orthogonal Polygons with Sliding Cameras

Yeganeh Bahoo* Onur Çağırıcı* Kody Manastyrski* Rahnuma Islam Nishat† Christopher Kolios* Roni Sherman*

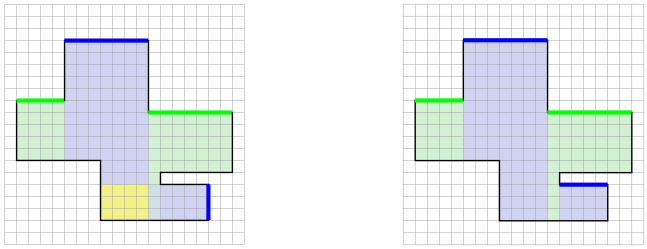
Abstract

In *conflict-free chromatic guarding of a polygon*, every guard is assigned a color such that every point in the polygon, including the points on the boundaries, must see at least one unique color. The goal of this problem is to minimize the number of colors needed. In this paper, we study the conflict-free chromatic guarding of *simple orthogonal* polygons with *sliding cameras*, where cameras are allowed to slide along the length of the corresponding edge. We investigate two versions of the Conflict-free Sliding Camera problem: for orthogonal polygons without holes (CFSC), and for orthogonal polygons with holes (CFSC-H), we show that two colors are always sufficient and sometimes necessary for a CFSC, and give an $O(n \log n)$ time algorithm to compute a CFSC using two colors, where n is the number of vertices of the polygon. We give an $O(n \log n)$ time algorithm to obtain a CFSC-H using three colors. We also show that for a special case of CFSC-H two colors suffice.

1 Introduction

The polygon guarding problem is a well-studied problem in the field of computational geometry, which is also known as the *art gallery problem* [7, 18]. Given a polygon P, the goal of the polygon guarding problem is to find the minimum number of guards needed so that any point in P is visible to at least one guard. Two points p and q are *visible* to each other when the line segment pq, also known as *line-of-sight visibility*, does not intersect any edges of P. This problem has been studied in many different settings, such as for general polygons [10, 11, 17], for weak-visibility polygons [2, 3], and for orthogonal polygons [15, 16].

The very first version of this problem, introduced by Victor Klee [18] considered line-of-sight visibility. Later, variations of the problem were studied while assuming different visibility models, such as α-visibility [14], and π-visibility [19].



(a) (b)

Figure 1: (a) An assignment with conflicts: the yellow region is covered by two blue guards and no green guards; the blue-green region has a blue and a green guard and causes no conflict. (b) A conflict-free assignment.

Although Fisk applied graph coloring in the proof of bounds in art gallery problems [13], chromatic guarding of polygons is a more recent topic. In chromatic guarding problem, we look for a set of guards such that each point of the polygon is visible to a subset of the guards and assign a color to each guard from a set of available colors. The set of guards along with the color assignment is conflict-free if for each point of the polygon, there is at least one guard with a unique color [1]. Conflict-free chromatic guarding has applications in the assignment of radio frequencies to sensors placed on the vertices of the polygon to guide mobile robots in triangulating their positions in the polygon [1, 5].

Conflict-free chromatic guarding has been studied in the context of orthogonal polygons, and bounds have been given on *chromatic numbers* χ_P of an orthogonal polygon P, i.e., the minimum number of colors required to guard P without conflict. Given a polygon P, Bärtschi and Suri [1] showed that $\chi_P \in O(\log n)$, where n refers to the number of vertices of the polygon, by subdividing P into "weak-visibility suppolygons". Erickson and LaValle [12] showed that for orthogonal staircase polygons, the bound is $\chi_P \leq 3$.

In this work, we use *sliding cameras* as guards [4, 8, 9, 16], where a guard or camera is *directional* (i.e. it has directional view oriented towards the interior of the given polygon) and can travel along a boundary edge

*Department of Computer Sciece, Toronto Metropolitan University, Toronto, ON, Canada, {bahoo, cagirici, kody.a.manastyrski, ckolios, roni.sherman}@torontomu.ca

†Department of Computer Science, Mathematics, Physics and Statistics, University of British Columbia, Kelowna, BC, Canada, rahnuma.nishat@ubc.ca

of the polygon. See Figure 1(b) for an example, where sliding cameras or guards are assigned (only to horizontal) edges of the polygon. Note that two consecutive horizontal edges on the polygon have not been assigned cameras of the same color as that would create conflict at the boundary of the two guarded regions. Throughout the paper, the terms *sliding camera* and *guard* are used interchangeably.

Our contributions. We give upper and lower bounds on χ_P for both CFSC and CFSC-H. In Section 3, we prove that two colors are sometimes necessary and always sufficient for a conflict-free chromatic guarding of an orthogonal polygon without holes. Our bound on the chromatic number is tight. We also give an $O(n \log n)$ time algorithm that solves CSFO with two colors, where n refers to the number of vertices of P. In Section 4.1, we propose an $O(n \log n)$ time algorithm to solve a CSFO-H, using three colors ($\chi_P \leq 3$). Finally, in Section 4.2, we study a special case of CFSC-H, where each hole has a rectangular boundary and the order of the holes inside the polygon is X-*monotone*. In this case, we show that two colors are sufficient ($\chi_P = 2$). In all our algorithms, the outer boundary of the polygon and the boundaries of the holes are axis-parallel.

2 Preliminaries

We define the terminology used throughout the paper.

A simple *polygon* P is an enclosed area in the Euclidean plane bounded by a finite number of straight line segments that form a polygonal chain; each such straight line segment is called an *edge* of P, and a pair of edges meet in a *vertex*. In this paper, first we consider simple polygons; i.e. no pair of edges intersect except at their common endpoints (vertices). Starting from Section 4, we suppose that a polygon with holes is given. A polygon with holes is a polygon enclosing several other polygons; the inner polygons are known as holes.

The *boundary* of P is the closed polygonal chain formed by the edges of P. We assume that the boundary is directed clockwise, i.e., while walking along the boundary of the polygon the interior would always be on the right. The vertices of a polygon are denoted by $v_1, v_2, \ldots, v_n$ in clockwise order. The edge that connects the pair of vertices v_i and v_{i+1} is denoted by e_i, where $1 \leq i \leq n$, and the edge from v_n to v_1 is denoted by e_n.

The polygon P is called an *orthogonal polygon*, if every pair of consecutive edges are perpendicular to each other. Throughout this manuscript, we assume that a given orthogonal polygon is oriented in a way such that each edge is parallel to either x-axis (horizontal), or y-axis (vertical). We classify the edges as north-facing, south-facing, east-facing and west-facing depending on the orientation of the perpendicular ray towards the interior of the polygon.

An X-*monotone* polygon is an orthogonal polygon that intersects any vertical line ℓ at most twice, where an intersection is either a point on a horizontal edge of the polygon, or an entire vertical edge.

Definition 1 *A* maximal monotonous south-facing chain *is a maximal set of south-facing edges $e_1, \ldots, e_k$ such that the x-coordinates of the starting points of the edges are in increasing order, and e_i and e_{i+1}, for each $i < k$, are connected by a vertical edge.*

Let $\mathbb{S}$ be the set of all maximal monotonous south-facing chains $S_1, S_2, \ldots, S_q$ in P. For any chain $S_i \in \mathbb{S}$, we denote the edges of S_i by $e^i_1, e^i_2, \ldots, e^i_{k_i}$ from left to right. The *visibility region* of a chain S of $\mathbb{S}$ is the region of P, including the boundary of P, that is visible to the guards assigned to the edges of S. We observe the following property of the set $\mathbb{S}$.

Lemma 1 *A sliding camera at each edge of $\mathbb{S}$ collectively guards the entire polygon P.*

Proof. From any point $p \in P$, if we draw a vertical line upward, the first edge e that the line intersects must be south-facing; hence, e must belong to a chain in $\mathbb{S}$. □

Lemma 1 forms the basis of our algorithms in this paper, where we find efficient ways to put a camera on each of the edges of $\mathbb{S}$ such that the number of colors needed for a conflict-free guarding of P is minimized.

We use blue, red and green for the three colors assigned to guards or cameras. When a guard is assigned a color, say red, we write *red guard* or *red camera*.

3 Orthogonal polygons without holes

In this section, we discuss the CFSC problem (for orthogonal polygons without holes) and give tight upper and lower bounds on the conflict-free chromatic number χ_P for them. We show that two colors are sometimes necessary and always sufficient to guard an orthogonal polygon with sliding cameras, thus giving a lower and an upper bound on the number of colors required to guard any orthogonal polygons. The lower bound holds for polygons with holes as well.

We first prove the lower bound on χ_P.

Theorem 2 *There exists an orthogonal polygon that requires sliding cameras of at least two colors for CFSC.*

Proof. Let P be the orthogonal polygon in Figure 2. By exhaustive search, we conclude that a sliding camera on any of the 8 edges of P cannot guard the whole polygon. If there exists only one color of gaurds, any combination of guards in this figure will result in a conflict as no consecutive edges of P can be assigned the same color. Therefore, we need at least two cameras. The

visibility regions of any two cameras guarding P will intersect at least at the boundaries. Therefore, we must need at least two different colors for the cameras. □

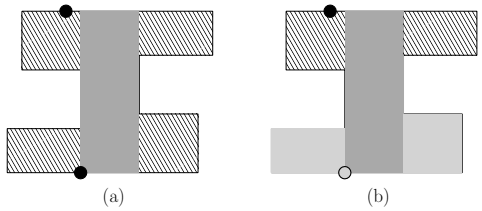


Figure 2: (a) Two black guards guarding the whole polygon; the patterned regions are visible to a single guard, and the solid region to both guards. (b) A conflict-free guarding by a black (patterned regions) and a gray guards. The dark solid region is covered by both guards.

We now show that two colors are always sufficient for a CFSC of any given orthogonal polygon. First, we describe an algorithm to obtain a CFSC for an X-monotone polygon, and then we generalize the idea to non-monotone polygons.

By definition, an X-monotone polygon P has exactly one maximal monotonous south-facing chain S. We place a blue camera on all the edges e_i of S, where i is odd, and a red camera on the edges e_i, where i is even; see Figure 3 for an example. We name this

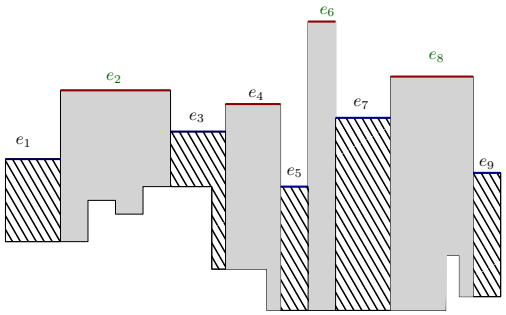


Figure 3: Two colors are sufficient to guard a monotone orthogonal polygon.

algorithm CFSC-MONOTONE. The following theorem proves the correctness and running time of the above algorithm. The proof is in the appendix.

Theorem 3 *Algorithm* CFSC-MONOTONE *computes a CFSC of an X-monotone polygon P without holes in $O(n)$ time using only two colors, which is optimal.*

Now consider the case where the input polygon P is not X-monotone. Then $\mathbb{S}$ has more than one chain. We then need some special data structure to assign colors to the edges in $\mathbb{S}$ efficiently.

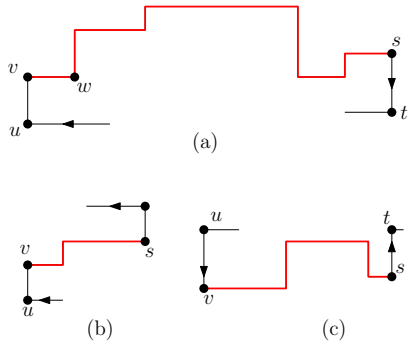


Figure 4: Maximal monotonous south-facing chain with (a) no upward connection, (b) one upward connection, and (c) two upward connections.

Before describing the data structures for our algorithm, we need to define some terminology. Let the vertical edges preceding and following a chain $S_i \in \mathbb{S}$ be (u, v) and (s, t), respectively, where v and s are the endpoints of the first edge e_1^i and last edge e_k^i of S_i, respectively. If u has a *higher* y-coordinate than v then we say that e_1^i is a *upward connection* S_i. Similarly, e_k^i is an upward connection of S_i if t has a higher y-coordinate than s. Clearly, S_i can have upward connections 0, 1, or 2 as shown in Figure 4.

We now describe the data structures we need. In the preprocessing step, we populate a list called $\mathcal{L}$ that stores pointers for each $S \in \mathbb{S}$ of the chains that are influenced by or influence the coloring of S. For this we build a trapezoidal map $\mathcal{T}$ and the associated search structure $\mathcal{D}$ with the south-facing edges of P using the algorithm proposed in [7]. Therefore, with each edge, the chain associated with $\mathbb{S}$ is given. Now for each *upward connection* v of each $S \in \mathbb{S}$, we traverse the search structure $\mathcal{D}$ to find the edge e and the associated chain S_i just above v. Let v be incident to edge e' in S. We then put (e', e, S_i) in the $\mathcal{L}$ entry of S, and put (e, e', S) in the corresponding entry for S_i. Building $\mathcal{T}$ and $\mathcal{D}$ takes expected $O(n \log n)$ time using the incremental randomized algorithm in [7], and finding the edge above each endpoint takes $O(\log n)$ time. Therefore, $\mathcal{L}$ can be populated from $\mathcal{D}$ in $O(n \log n)$ time.

Figure 5 shows an orthogonal polygon without holes, and Table 1 shows the corresponding data structure $\mathcal{L}$.

We now briefly describe the algorithm for assigning two-color sliding guards to the edges of $\mathbb{S}$ to obtain a conflict-free guarding of P. We call the algorithm CFSC-TWOCOLORS.

We pick any chain S from $\mathbb{S}$ and assign guards to the chain by putting a red camera on the edges e_i, where i is odd, and a blue camera on the edges e_i, where i is even. For each entry (e, e', S') in the influence list $\mathcal{L}$ of S, where e is an edge of S and $c(e)$ denotes the color

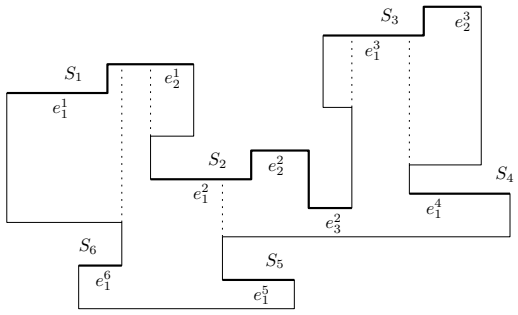


Figure 5: The chains in $\mathbb{S}$ are shown, with the edges, and the upward connections with dotted lines.

S_i	influence list
S_1	(e_1^6, e_2^1, S_6), (e_1^2, e_2^1, S_2)
S_2	(e_1^2, e_2^1, S_1), (e_1^5, e_1^2, S_5), (e_1^3, e_3^2, S_3)
S_3	(e_3^2, e_1^3, S_2), (e_1^3, e_1^4, S_4)
S_4	(e_1^4, e_1^3, S_3)
S_5	(e_1^5, e_1^2, S_2)
S_6	(e_1^6, e_2^1, S_1)

Table 1: Data structure $\mathcal{L}$ for the polygon in Figure 5.

assigned to edge e, we put $(c(e), e', S')$ in a queue Q. In the next step, when we remove that entry $(c(e), e', S')$ from Q, we assign the opposite color of $c(e)$ to e' of S', and continue to assign alternating colors to the edges on each side of S'. After that, we put all the chains in the influence list of S' that have not been colored into Q. We stop when the queue is empty.

Figure 6 shows how the algorithm works by showing the first step. The following theorem proves the correctness and running time of the algorithm. See the appendix for the complete proof.

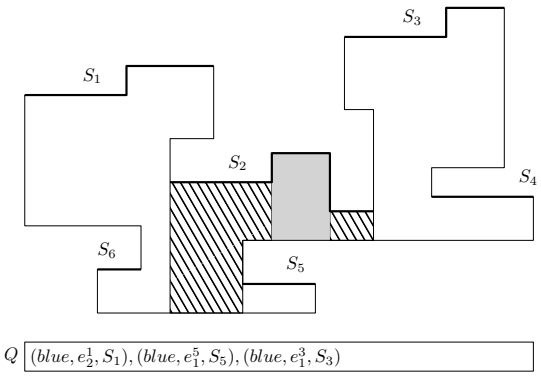


Figure 6: A CFSC with two colors, after assigning colors to the edges of $S_2 \in \mathbb{S}$; and the queue Q after coloring S_2.

Theorem 4 *Algorithm* CFSC-TwoColors *computes a CFSC of an orthogonal polygon P without holes in $O(n \log n)$ time using two colors.*

Proof Sketch. After coloring a chain S, if we remove the visibility region of S from P, we get disjoint sub-polygons for each of which only one endpoint of one south-facing edge has been assigned a color. Since the corresponding chain S' is placed in Q before any other chains in that sub-polygon, S' also gets a conflict-free coloring. By inductively applying the above logic, we can prove the correctness. Assigning colors to all the south-facing edges takes $O(n)$ time. Building $\mathcal{T}$, $\mathcal{D}$, and then populating $\mathcal{L}$ from $\mathcal{D}$ takes $O(n \log n)$ time. Therefore, the total time is $O(n \log n)$.

4 Orthogonal polygons with holes

In this section, we study the problem *conflict-free chromatic guarding of orthogonal polygons with holes* or CFSC-H for short. We first give an algorithm that we call CFSC-H-ThreeColors that uses three colors, thus proving an upper bound on the chromatic number for orthogonal polygons with holes.We also define a special class of polygons for which two colors are sufficient for CFSC-H, and give an algorithm to achieve such an assignment of colors.

4.1 Three colors are sufficient

Let P be an orthogonal polygon with holes (see Figure 7 for an example), and let $\mathbb{S}$ be the set of all maximal monotonous south-facing chains in P. Note that the chains in $\mathbb{S}$ belong to the outer boundary of P as well as the holes of P. By Lemma 1, a guard assigned to each of the south-facing edges, including the south-facing edges on the holes, covers the entire polygon P. Therefore, our goal is to assign a guard to each of the south-facing edges in a conflict-free manner.

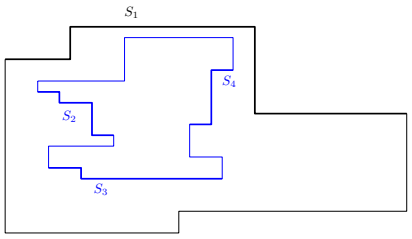


Figure 7: An orthogonal polygon with an orthogonal hole shown in blue; outer boundary has one and the hole has three maximal monotonous south-facing chains.

Since we are allowing one more color to be used than CFSC, a straightforward idea could be to follow a similar techniques to Algorithm CFSC-TwoColors from the previous section. However, that may not be possible. Take the polygon P in Figure 7 as an example.

Suppose that we first assign guards to S_3, which puts S_2 and S_4 in Q. After assigning guards to S_2, the queue Q contains S_4 and S_1. Now when we assign guards to S_4, we either have to put a duplicate entry for S_1 in the Q, or retrieve S_1 from Q to update the color constraints and then put it back again. In either case, searching the queue Q for an entry for a specific chain increases the time complexity of the algorithm. Therefore, we need some tool to obtain an efficient algorithm for CFSC-H. In this regard, we introduce the notion of *relationship graph*.

Definition 2 (Relationship graph) *The relationship graph $G_{\mathbb{S},P}$ of P is a directed graph whose nodes are the chains of $\mathbb{S}$ such that for each pair of chains $S, S' \in \mathbb{S}$, G has a directed edge from S to S' for each upward connection of S that is in the visibility region of S'. See Figure 8 for an example.*

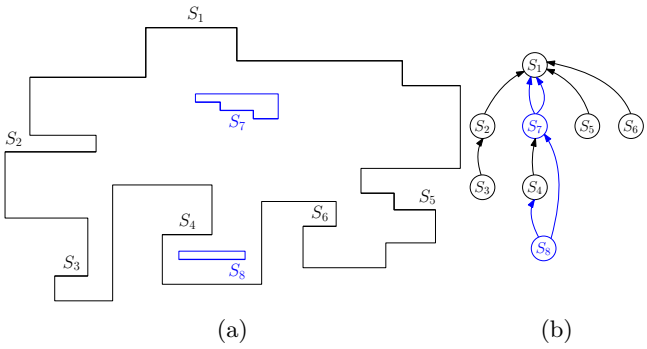


Figure 8: (a) Set $\mathbb{S}$ of maximal monotonous south-facing chains of orthogonal polygon P with holes, the holes are blue. (b) The relationship graph $G_{\mathbb{S},P}$.

We observe some properties of $G_{\mathbb{S},P}$.

Lemma 5 *$G_{\mathbb{S},P}$ is a directed acyclic graph (DAG) where each node of G has at most two outgoing edges.*

Proof. Since P is connected and simple, by Definition 2, there cannot be a directed cycle. Since a chain can have at most two upward connections, the number of outgoing edges of a node of G must be at most two. □

We can build the relationship graph $G_{\mathbb{S},P}$ of P using the trapezoidal map $\mathcal{T}$ of the chains of $\mathbb{S}$ as in the previous section. For each *upward connection* e of each $S \in \mathbb{S}$, we traverse the search structure $\mathcal{D}$ associated with $\mathcal{T}$ to find the edge e' and the associated chain S' just above e. Let v be incident to edge e in S. We then add the edge (S, S') with the label (e, e') in $G_{\mathbb{S},P}$. Building $\mathcal{T}$ and $\mathcal{D}$ takes expected $O(n \log n)$ time using the incremental randomized algorithm in [7], and finding the edge above each endpoint takes $O(\log n)$ time. Therefore, the relationship graph G can be built from $\mathcal{D}$ in $O(n \log n)$ time.

We now describe Algorithm CFSC-H-ThreeColors. Since $G_{\mathbb{S},P}$ is a DAG by Lemma 5, we obtain a topological ordering of the nodes of the graph [6] in $O(n)$ time. We then assign guards to the chains of $\mathbb{S}$ according to topological ordering. For any chain $S \in \mathbb{S}$, one of these cases holds. **Case 1.** *S* **does not have any upward connections.** We place a blue camera at all edges e_i, where i is odd, and a red camera when i is even. **Case 2.** *S* **has only one upward connection.** Let the label of the upward connection edge be (e, e') where e belongs to S. Without loss of generality, assume that $c(e')$ is red. We then color e blue, and on either side of e, we assign red and blue cameras alternatingly to the rest of the edges of S. **Case 3.** *S* **has two upward connections.** Let the labels of the upward connections be (e_1, e'_1) and (e_q, e'_q) where e_1 and e_q are edges of S. Without loss of generality, assume that e'_1 is colored red. If e'_q is also red, we color $e_1, \ldots, e_q$ with colors blue and green alternatingly. If e'_q has a different color than e'_1, say blue, we assign the colors green and red to $e_1, \ldots, e_q$ starting with green. Then, we will not have conflict whether q is odd or even.

We now prove the correctness and time complexity of the algorithm. See the appendix for the complete proof.

Theorem 6 CFSC-H-ThreeColors *computes a conflict-free chromatic guarding of an orthogonal polygon P in $O(n \log n)$ time using three colors.*

Proof Sketch. The correctness follows from the fact that the topological ordering of the chains would ensure that any chain S with an upward connection to another chain S' would be colored after S. The calculation of running time is similar to Theorem 4.

4.2 Special case: monotonous rectangular holes

We describe a restricted class of orthogonal polygons, where the boundary of each hole is a rectangle and the order of the holes inside the polygon is monotone with respect to either x-axis or y-axis; see Figure 9(a). We call this restricted class of polygons *orthogonal polygons with monotonous rectangular holes*. Without loss of generality, we assume that the order of the holes is X-monotone. We give an algorithm that we call CFSC-H-TwoColors to obtain a CFSC-H using two colors.

Let $\mathbb{S}$ and $\mathbb{H}$ be the sets of maximal monotonous south-facing chains from the outer boundary of P and from the holes of P. As in Algorithm CFSC-TwoColors, we build the trapezoidal map $\mathcal{T}$ with $\mathbb{S}$ and $\mathbb{H}$, and from it we build the data structure $\mathcal{L}$. However, when checking the search structure of $\mathcal{T}$ for the edges above a chain $H \in \mathbb{H}$, we add all the chains that are above H.

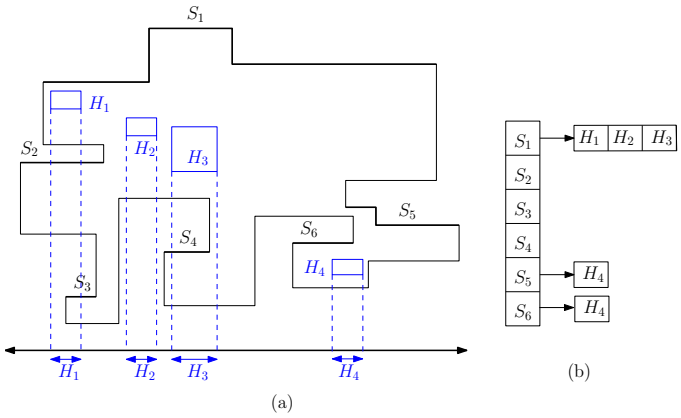


Figure 9: (a) An orthogonal polygon with rectangular X-monotone holes; the projection of the hole boundaries on the X-axis are nonoverlapping. (b) List of holes under each chain ordered from left to right according to the holes' projections on the X-axis.

We need another data structure to keep track of holes that are *directly under* a chain $S \in \mathbb{S}$. We consider the list $\mathcal{A}$ in Figure 9(b), where the holes directly under a chain S are listed from left to right. We can populate $\mathcal{A}$ at the same time as $\mathcal{L}$; and later sort each individual list of holes for each chain in $\mathbb{S}$ in ascending order of the x-coordinates of the holes.

Algorithm CFSC-H-TwoColors uses the same idea as Algorithm CFSC-TwoColors. For the first chain S chosen, we start from the leftmost edge e_1 of S. We assign blue to e_1, red to e_2, and continue this way until we reach a hole H_1 under S. Let $e', e'' \in S$, where $e' = e''$ may hold, be the two edges intersected by the two vertical edges of hole H when extended in the direction of positive y-axis. We assign the same color to all the edges of S from e' to e'', and assign the opposite color to e' to the top edge and the bottom edge of H_1. We then continue as before until we reach the next hole H_2 and follow the same procedure.

After coloring S and all the holes under it, we put the entries for all the uncolored chains in $\mathbb{S}$ that are influenced by the coloring of S in the queue Q. Since all the holes under S have already been colored, no holes will be added to Q at this step. Then for each hole H_i under S, we check if a chain S' influenced by it is already in Q. If S' is in Q, that means that it is also influenced by S. Then both S and S' must be above H_i and influenced by the guard of the north-facing edge e_h of H_i. Then we can assign the same color to all edges of S and S' that are in the visibility region of e_h. So we remove the entry for S' from Q and add a new entry that applies the opposite color of e_h to the visible edges of S'. Since the holes are X-monotone, no holes would be added to Q at this stage also.

We then remove the chain at the front of Q and apply the same procedure. We keep dequeuing chains from Q and assigning guards to them until Q is empty. Note that since the holes under a chain are colored at the time of coloring the chain, no hole would be added to Q at any point. Figure 10 describes some scenarios that may occur. Note that, in the case where e'' belong to a different chain $S_j \neq S_i$ (as in S_5 and S_6 for the hole H_4 in Figure 9), the guards on the top edge of H will see some edges of S_j. When coloring S_j, we propagate the opposite color of H on S_j from where the visibility of the top edge of H ends.

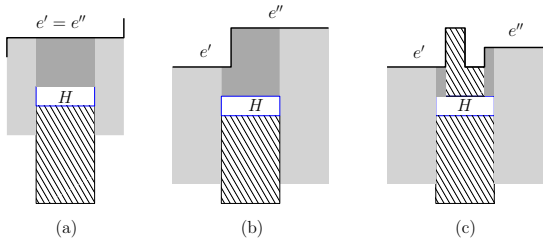


Figure 10: Different scenarios when assigning colors to the hole; the dark gray regions are covered by one blue (black patterned) and one red (solid light gray) guard: (a) $e' = e''$, (b) e', e'' are adjacent, and (c) there are edges between e' and e''.

The following theorem summarizes the time complexity and correctness of the above algorithm.

Theorem 7 *Algorithm* CFSC-H-TwoColors *computes a CFSC-H of an orthogonal polygon P with monotonous rectangular holes in $O(n^2)$ time using two colors.*

Proof. Building the data structure $\mathcal{L}$ and $\mathcal{A}$ takes $O(n \log n)$ time each by Theorem 4. Sorting all lists in $\mathcal{A}$ takes $O(n \log n)$ time in total. Since the holes are colored at the time of coloring the chains, the coloring is done in $O(n)$ time. However, when adding chains influenced by a hole, we have to look for them in Q. This may take $O(n)$ time. Therefore, the total running time of the algorithm is $O(n^2)$. □

5 Conclusion

We have given an $O(n \log n)$ time algorithm for CFSC with two colors for orthogonal polygons without holes and showed that the bound is tight. We have given an $O(n \log n)$ time algorithm for CFSC-H with three colors for polygons with holes, while a special case requires two colors. The question of whether two colors are sufficient for the general case of CFSC-H remains open. In our algorithms, the guards are placed only on horizontal edges. It would be interesting to investigate whether less guards are needed when placed on vertical and horizontal edges.

References

[1] A. Bärtschi and S. Suri. Conflict-free chromatic art gallery coverage. *Algorithmica*, 68:265–283, 2014.

[2] P. Bhattacharya, S. K. Ghosh, and B. Roy. Vertex Guarding in Weak Visibility Polygons. In *CALDAM*, pages 45–57, 2015.

[3] P. Bhattacharya, S. K. Ghosh, and B. Roy. Approximability of guarding weak visibility polygons. *Discrete Applied Mathematics*, 228:109–129, 2017.

[4] T. Biedl, T. M. Chan, S. Lee, S. Mehrabi, F. Montecchiani, and H. Vosoughpour. On guarding orthogonal polygons with sliding cameras. In *WALCOM: Algorithms and Computation: 11th International Conference and Workshops, WALCOM 2017, Hsinchu, Taiwan, March 29–31, 2017, Proceedings*, pages 54–65. Springer, 2017.

[5] O. Çağırıcı, S. K. Ghosh, P. Hliněný, and B. Roy. On conflict-free chromatic guarding of simple polygons. In *COCOA*, 2019.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

[7] M. de Berg, O. Cheong, M. Kreveld, and M. Overmars. *Computational Geometry, Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.

[8] M. de Berg, S. Durocher, and S. Mehrabi. Guarding monotone art galleries with sliding cameras in linear time. *Journal of Discrete Algorithms*, 44:39–47, 2017.

[9] S. Durocher, O. Filtser, R. Fraser, A. D. Mehrabi, and S. Mehrabi. Guarding orthogonal art galleries with sliding cameras. *Comput. Geom.*, 65:12–26, 2017.

[10] A. Efrat and S. Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100:238–245, 2006.

[11] S. Eidenbenz, C. Stamm, and P. Widmayer. Inapproximability Results for Guarding Polygons and Terrains. *Algorithmica*, 31:79–113, 2001.

[12] L. H. Erickson and S. M. LaValle. An art gallery approach to ensuring that landmarks are distinguishable. In *Robotics: science and systems*, volume 7, pages 81–88, 2012.

[13] S. Fisk. A short proof of Chvátal's watchman theorem. *J. Comb. Theory, Ser. B*, 24:374, 1978.

[14] M. Ghodsi, A. Maheshwari, M. N. Baygi, J.-R. Sack, and H. Zarrabi-Zadeh. α-visibility. *Comput. Geom. Theory Appl.*, pages 435–446, 2014.

[15] C. Iwamoto and T. Ibusuki. Computational Complexity of the Chromatic Art Gallery Problem for Orthogonal Polygons. In *WALCOM*, pages 146–157, 2020.

[16] M. J. Katz and G. Morgenstern. Guarding Orthogonal Art Galleries with Sliding Cameras. *Int. J. Comput. Geometry Appl.*, 21:241–250, 2011.

[17] J. King and D. G. Kirkpatrick. Improved Approximation for Guarding Simple Galleries from the Perimeter. *Discrete & Computational Geometry*, 46:252–269, 2011.

[18] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[19] J. Urrutia. Chapter 22 - Art Gallery and Illumination Problems. In *Handbook of Computational Geometry*, pages 973–1027. Elsevier, 2000.

Appendix

Lemma 1. *A sliding camera on each south-facing edge of $\mathbb{S}$ collectively guards the entire polygon P.*

Proof. From any point $p \in P$ if we draw a vertical line upward, the first edge e that the line intersects must be south-facing. Then e must belong to one of the chains in $\mathbb{S}$. Therefore, each point of P is covered by an edge of $\mathbb{S}$, and thus putting a sliding camera on each of these edges will cover the whole polygon. □

Theorem 3. *Algorithm* CFSC-MONOTONE *computes a CFSC of an X-monotone polygon P without holes in $O(n)$ time using only two colors, which is optimal.*

Proof. By Lemma 1, a sliding camera on each edge of S will guard the whole polygon. Since we are assigning alternating colored guards for adjacent edges of S, each point of P will have at least one unique colored guard. The guards can be assigned by a walk along the boundary of P, which takes $O(n)$ time. By Theorem 2, the lower bound for the number of colors needed for the guards is also two. Therefore, the number of colors used in this algorithm is optimal. □

Theorem 4. *Algorithm* CFSC-TWOCOLORS *computes a CFSC of an orthogonal polygon P without holes in $O(n \log n)$ time using two colors.*

Proof. We first prove that the algorithm assigns guards to all the south-facing edges of P using only two colors. It is easy to see that the first chain S considered by the algorithm gets a conflict-free coloring with two colors. Now, if we remove the visibility region of S from P, we get disjoint sub-polygons $P_1, P_2, \ldots, P_q$ for some $q < n$. For each P_i, $1 \leq i \leq q$, only one endpoint of one south-facing edge has been assigned a color. Since the corresponding chain S' is placed in Q before any other chains in that sub-polygon, S" gets a conflict-free coloring with two colors. We then remove the visibility region of S' and prove the claim inductively for all the $S_i \in \mathbb{S}$.

We now prove the running time of the algorithm. We assume that P is input as the sequence of vertices $v_1, \ldots, v_n$ in clockwise order. Then we can find the chains in $\mathbb{S}$ by walking around the polygon. We represent each chain $S \in \mathbb{S}$ by a tuple (v_i, v_j), where v_i is the first vertex and v_j is the last vertex of S on the walk. The south-facing edges in S can easily be calculated from the indices of the end vertices of S as $e_i, e_{i+2}, \ldots, v_{j-1}$. Therefore, assigning colors to all the south-facing edges takes $O(n)$ time. Building the trapezoidal map $\mathcal{T}$, searching the structure $\mathcal{D}$, and then populating the data structure $\mathcal{L}$ from $\mathcal{D}$ takes $O(n \log n)$ time. Therefore, the total time required is $O(n \log n)$. □

Theorem 6. *Algorithm* CFSC-H-THREECOLORS *computes a conflict-free chromatic guarding of an orthogonal polygon P in $O(n \log n)$ time using three colors.*

Proof. The correctness follows from the fact that the topological ordering of the chains would ensure that any chain S with an upward connection to another chain S' would be colored after S.

We assume that the outer boundary of P is given as a clockwise sequence of vertices on it, and the hole boundaries are given as counterclockwise sequence of vertices on them. Building the trapezoidal map $\mathcal{T}$ takes expected $O(n \log n)$ time, and building the graph G from the search structure takes $O(n \log n)$ time. We obtain a topological ordering of the chains of $\mathbb{S}$ from the DAG G in $O(n)$ time. We then color the chains in $O(n)$ time in total. Therefore, the total time complexity of the algorithm is $O(n \log n)$. □

Figure 11 shows how Algorithm CFSC-H-THREECOLORS works by showing the first two steps.

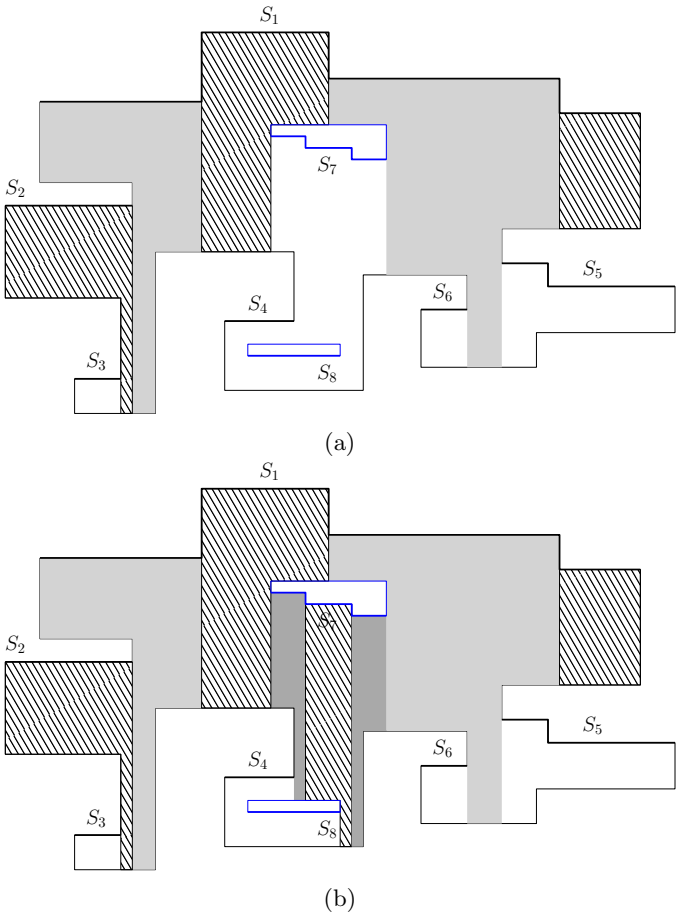


Figure 11: A conflict-free chromatic guarding of the polygon in Figure 8 with three colors. (a) After coloring S_1 and S_2. (b) After coloring S_7.