Complete Visibility for Robots with Lights in $O(1)$ Time

Gokarna Sharma[1(✉)], Ramachandran Vaidyanathan[2], Jerry L.Trahan[2], Costas Busch[2], and Suresh Rai[2]

[1] Department of Computer Science, Kent State University, Kent, OH 44242, USA
sharma@cs.kent.edu
[2] School of Electrical Engineering and Computer Science, Louisiana State University, Baton Rouge, LA 70803, USA
{vaidy,jtrahan,srai}@lsu.edu, busch@csc.lsu.edu

Abstract. We consider the problem of repositioning N autonomous robots on a plane so that each robot is visible to all others (the Complete Visibility problem); a robot cannot see another robot if its visibility is obstructed by a third robot positioned between them on a straight line. This problem is important since it provides a basis to solve many other problems under obstructed visibility. Robots operate following *Look-Compute-Move* (LCM) cycles and communicate with other robots using colored lights as in the recently proposed *robots with lights* model. The challenge posed by this model is that each robot has only a constant number of colors for its lights (symbols for communication) and no memory (except for the persistence of lights) between LCM cycles. Our goal is to minimize the number of rounds needed to solve Complete Visibility, where a round is measured as the time duration for all robots to execute at least one complete LCM cycle since the end of the previous round. The best previously known algorithm for Complete Visibility on this robot model has runtime of $O(\log N)$ rounds. That algorithm has the assumptions of full synchronicity, chirality, and robot paths may collide. In this paper we present the first algorithm for Complete Visibility with $O(1)$ runtime that runs on the semi-synchronous (and also the fully synchronous) model. The proposed algorithm is deterministic, does not have the chirality assumption, and is collision free.

1 Introduction

In the classical model of distributed computing by mobile robots, each robot is modeled as a point in the plane that is equipped with a local coordinate system and sensory capabilities to determine the positions of other robots [10]. The robots are *autonomous* (no external control), *anonymous* (no unique identifiers), *indistinguishable* (no external identifiers), and possibly *disoriented* (no agreement on local coordinate systems and units of distance measures). They execute the same algorithm. Each robot proceeds in *Look-Compute-Move* (LCM) cycles; that is, when a robot becomes active, it uses its sensory capabilities to get a

B. Bonakdarpour and F. Petit (Eds.): SSS 2016, LNCS 10083, pp. 327–345, 2016.
DOI: 10.1007/978-3-319-49259-9_26

snapshot of its surroundings (*Look*), then computes a destination point based on the snapshot (*Compute*), and finally moves towards the destination point (*Move*). Furthermore, the robots are assumed to be *oblivious* in the sense that in each cycle, each robot has no memory of its past LCM cycles [10].

Although the robots in the classical model have vision and mobility, they are *silent* because they do not communicate directly, and only vision and mobility enable the robots to coordinate their actions. While silence has advantages, for example in hostile environments, in many other situations direct communication is assumed. A model that incorporates direct communication is called *robots with lights* [7,10,15]. In this model, each robot is provided with a local externally visible light which can assume colors from a fixed constant size set; robots explicitly communicate with each other using these colors. The colors are persistent; i.e., the color is not erased at the end of a cycle. Except for the lights, the robots are oblivious as in the classical model.

Much of the work on both the classical model and the model of robots with lights assumes that visibility is *unobstructed*; that is, three collinear robots are assumed to be visible to each other. The notion of *obstructed visibility* is captured in the so-called *fat robot* model where robots are non-transparent unit discs [1,6]. However, the fat robot model does not assume the availability of lights.

Di Luna et al. [13] gave the first algorithm for robots with lights under obstructed visibility to solve the fundamental Complete Visibility problem: Given an arbitrary initial configuration of robots located in distinct points on a plane, reach a configuration in which each robot is in a distinct position from which it can see all other robots. This problem is important since it provides a basis to solve many other problems requiring complete visibility among robots under obstructed visibility. Moreover, robots cannot share positions during the execution of the algorithm to reach a complete visibility configuration, that is, sharing the same position by two or more robots constitutes a robot collision. Initially some robots may be obstructed from the view of other robots and the total number of robots, N, is not known to robots. The solution of Di Luna et al. [13] arranges robots on corners of a convex polygon, which naturally solves the Complete Visilibity problem. They proved the correctness of their algorithm but gave no runtime analysis except a proof of its termination in finite time. Runtime is measured in terms of rounds. A *round* ends as soon as all robots have executed at least one complete LCM cycle since the end of the previous round [5].

Recently, Vaidyanathan et al. [17] presented an algorithm for this problem which has a running time of $O(\log N)$ rounds for any initial configuration of $N \geq 4$ robots in the fully synchronous setting (where all robots are active in all rounds). However, their solution allows the paths of robots to cross. Moreover, their solution assumes *chirality* [10] – robots agree on the orientation of the axes of their local coordinate system. The focus of other recent work is only on solvability, minimizing the number of colors and does not provide runtime. The goal of this work is to develop an optimal algorithm with constant runtime and constant number of colors for Complete Visibility on the robots with lights model.

Contributions. We consider the same robot model as in the work of Di Luna et al. [13], namely, robots are oblivious except for a persistent light that can assume a constant number of colors. Visibility could be obstructed by other robots in the line of sight. We assume that N is not known and the robots may be disoriented. We consider the fully synchronous and semi-synchronous models of computation (Sect. 2). Moreover, we assume that a robot in motion cannot be stopped (by an adversary). As in the model of Di Luna et al. [13], we assume that two robots cannot head to the same destination point (this would constitute a collision). In this paper, we present, to our knowledge, the first algorithm for Complete Visibility which has the running time of constant rounds and uses a constant number of colors. In particular, we prove the following theorem.

Theorem 1. *For any initial configuration of $N \geq 1$ robots with lights, there is a deterministic algorithm that solves* Complete Visilibity *in $O(1)$ rounds with $O(1)$ colors and without collisions on the semi-synchronous model.*

Our algorithm is deterministic and has three phases: Phase 0 (*initialization*), that breaks up any initial linear arrangement of robots and places all robots within or on a convex polygon P (convex hull of points); Phase 1 (*interior depletion*), which places all robots on the corners or sides of a convex polygon P'; and Phase 2 (*edge depletion*), which moves each robot on a side of P' to a corner of a new convex polygon P''. Key to Phase 1 is a *corner moving* procedure that permits all interior robots to see all corners of the hull. Key to Phase 2 is a *corner insertion* procedure that moves robots to corners while retaining convexity. Both the corner moving and corner insertion procedures may have independent interest.

Previous Work. The problem of uniformly spreading robots in a line, studied by Cohen and Peleg [4], considers the case of obstructed visibility, but these robots have no lights. The work of Pagli et al. [14] considers a problem where collisions must be avoided between robots. However, they do not provide runtime analysis. Similarly, much work on the classical robot model (with no lights) [2,4,16,18] showed that Gathering (robots come together to be in a not predefined point) is achieved in finite time without a full runtime analysis, except in a few cases [8,9,12]. Furthermore, Izumi et al. [11] considered the robot scattering problem (opposite of the gathering problem) in the semi-synchronous setting and provided a solution with an expected runtime of $O(\min\{N, D^2 + \log N\})$; here D is the diameter of the initial configuration.

Paper Organization. In Sect. 2 we provide details of our model and some preliminaries. For clarity, the Complete Visilibity algorithm is first presented for the fully synchronous model in Sect. 3. The conversion of the algorithm to the semi-synchronous model is discussed in Sect. 4. We then conclude in Sect. 5. Many proofs, details, and pseudocodes are omitted due to space constraints.

2 Model and Preliminaries

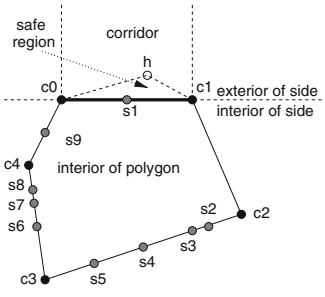
This paper uses a distributed system of N robots (agents) with index from set $\mathcal{R} = \{0, 1, \cdots, N-1\}$. We will then use a variable, for example r, to indicate a robot on or the point on the plane it is positioned at. Each robot is a (dimensionless) point that can move in an infinite 2-dimensional real space $\mathbb{R}^2$. A robot i can see, and be visible to, another robot j iff there is no third robot k in the line segment joining i and j. Each robot has a light that can assume one of a constant number of colors.

Look-Compute-Move. Each robot i is either active or inactive. When a robot i becomes active, it performs the "Look-Compute-Move" cycle described below.

- *Look:* For each robot j that is visible to it, robot i can observe the position of j on the plane and the color of the light of j. Robot i can also observe its own color and position; that is, i is visible to itself. Each robot observes position on its own frame of reference. That is, two different robots observing the position of the same point may produce different coordinates. However a robot observes the positions of points accurately within its own reference frame.
- *Compute:* In any cycle, robot i may perform an arbitrary computation using only the colors and positions observed during the "look" portion of that cycle. This computation also includes determination of a (possibly) new position and light color for i for the start of next cycle. Robot i maintains this new color from the current cycle to the next cycle.
- *Move:* At the end of the cycle, robot i moves to its new position and changes its light to the new color.

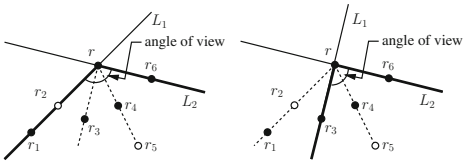
In the fully synchronous model, every robot is active in every LCM cycle. In the semi-synchronous model, a subset of robots (zero to all) in $\mathcal{R}$ are active, and over an infinite number of LCM cycles, every robot is active infinitely often. In the fully synchronous model, one round is always one LCM cycle. In the semi-synchronous model, a round can take an arbitrary number of LCM cycles. Depending on the activation schedule, some robots may be active for many cycles in a round before every robot has been active at least once. Note that our time bounds for the semi-synchronous model hold regardless of the activation schedule.

Convex Polygon. For $N \geq 3$, a *convex polygon* can be represented as a sequence $\mathbb{P} = (c_0, c_1, \cdots, c_{N-1})$ of *corner points* in a plane that enumerates the polygon vertices in clockwise order. A point s on the plane is a *side point* of $\mathbb{P}$ iff there exists $0 \leq i < N$ such that $c_i, s, c_{(i+1)(\mathrm{mod}\ N)}$ are collinear; for the rest of the paper we will implicitly assume the above modulo operation and write $c_{(i+1)(\mathrm{mod}\ N)} =$



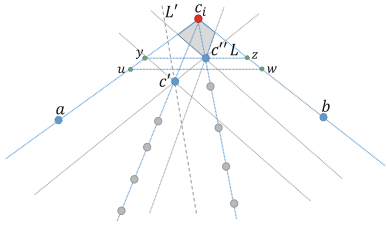
c_{i+1}. A side $S = (c_i, s_1, s_2, \cdots, s_m, c_{i+1})$ is a sequence of collinear points whose beginning and end are adjacent corner points and whose remaining points are side points. We write $\overline{pq}$ for a line segment connecting two points p, q and denote by $\mathsf{length}(\overline{pq})$ its length. For a given polygon $\mathbb{P}$, the plane can be divided into the interior and exterior parts. For a given side S of $\mathbb{P}$, the infinite line obtained by extending side S divides the plane into the interior and exterior parts of the side. The interior part of S contains the interior of the polygon. The *corridor* of S is the infinite subregion on its exterior that is bounded by S and perpendicular lines through points c_i, c_{i+1} of S. The corridors of the sides of $\mathbb{P}$ are disjoint. The figure above illustrates these concepts.

Lines and Angles of View. For any robot r, let V_r be the set of robots (other than itself) visible to r. Let L_1, L_2 be lines through robot r, such that θ_1 is the smallest induced angle at r whose region accommodates all robots of V_r; call this the *region of view* of r. The line segments of L_1, L_2 that border the region of view of r are called the *lines of view* of r and the associated angle is called the *angle of view* of r. These ideas can also be used with a subset of the robots visible to r. The figure below illustrates these ideas: In the left part all elements of $V_r = \{r_2, r_3, r_4, r_6\}$ are considered; in the right part only r_3, r_4, r_6 are considered.



Configuration and Local Convex Polygon. A *configuration* $\mathbb{C} = \{(p_0, col_0), \ldots, (p_{N-1}, col_{N-1})\}$ defines the positions of the robots in $\mathcal{R}$ and their colors; here $p_i = (x_i, y_i)$ is the position of robot i and col_i is the color of its light. A configuration for a robot $i \in \mathcal{R}$, $\mathbb{C}(i)$, is a configuration that defines the positions of the robots in $\mathcal{R}$ that are visible to i (including i) and their colors, i.e., $\mathbb{C}(i) \subseteq \mathbb{C}$. The convex hull of points in $\mathbb{C}(i)$ is denoted by $\mathbb{P}(i)$. $\mathbb{P}(i)$ is *local* to i since $\mathbb{P}(i)$ depends only on the points that are visible to i. We sometime write $\mathbb{C}_t, \mathbb{P}_t, \mathbb{C}_t(i), \mathbb{P}_t(i)$ to denote $\mathbb{C}, \mathbb{P}, \mathbb{C}(i), \mathbb{P}(i)$, respectively, for any round $t \geq 0$. Moreover, we sometime write $\mathbb{C}(r_i), \mathbb{P}(r_i)$ instead of $\mathbb{C}(i), \mathbb{P}(i)$.

Eligible Area. Let $\mathcal{A}$ be a set of points and $\mathbb{P}$ be the convex polygon of the points in $\mathcal{A}$. Let R_c, R_s, R_i be the set of points at corners, sides, and the interior of $\mathbb{P}$. Moreover, let c_i be a corner point of $\mathbb{P}$ and a, b be the counterclockwise and clockwise neighbors of c_i in the perimeter of $\mathbb{P}$. The *eligible area* for c_i, denoted as $EA(c_i)$, is a polygonal subregion inside $\mathbb{P}$ within the triangle $c_i uw$, where u, w are the midpoints of edges $\overline{c_i a}, \overline{c_i b}$, respectively. It is easy to show that the eligible areas for any two corner points of $\mathbb{P}$ are disjoint. Due to obstructed visibility, $EA(c_i)$ is computed based on $\mathbb{C}(c_i)$ and the corresponding polygon $\mathbb{P}(c_i)$. One prominent property

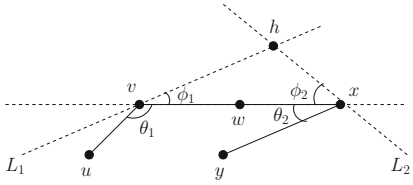


of the eligible area is that c_i remains a corner of $\mathbb{P}$ even after it moves to any point inside $EA(c_i)$ (except the points on the lines going through $EA(c_i)$). The other prominent property is that all the points in R_s, R_i are visible to c_i (and vice-versa), when c_i moves to any point inside $EA(c_i)$. This computation is used in Phase 1 of our algorithm.

We outline here how $EA(c_i)$ is computed for any corner point c_i of $\mathbb{P}$. The pseudocode is omitted due to space constraints. Initially, c_i sets the triangle c_iuw as its $EA(c_i)$. However, if c_i sees some point of $\mathcal{A}$ inside c_iuw, then it sets as $EA(c_i)$ the triangle c_iyz such that there is no point inside c_iyz. Note that $\overline{yz}$ is parallel to $\overline{ab}$. Let c' be a point in $\mathbb{C}(c_i)$. For every other point $c'' \in \mathbb{C}(c_i), c'' \neq c', c'' \neq c_i$, c_i computes a line, L', parallel to $\overline{c_ic''}$ passing through c'. Let HP be the half-plane divided by L' such that c_i is in HP. Corner c_i then updates its $EA(c_i)$ by keeping only the portion of $EA(c_i)$ that is in the half-plane HP. This process is repeated for all $c' \in \mathbb{C}(c_i)\backslash\{c_i\}$ and $EA(c_i)$ is updated in every iteration. Now from the area $EA(c_i)$ that remains, c_i removes the points that are in the perimeter of $EA(c_i)$ and also the points that are part of the lines $\overleftrightarrow{c_ix}, x \in \mathbb{C}(c_i)\backslash\{a, b, c_i\}$, passing from inside of $EA(c_i)$. This removal of points is crucial to guarantee that when c_i moves to a point in $EA(c_i)$, it does not become collinear with any robot in R_s, R_i. The figure above illustrates the computation of $EA(c_i)$; the shaded area is $EA(c_i)$ except the points on the lines inside it (e.g., the point of lines $\overline{c_ic'}$ and $\overline{c_ic''}$ inside $EA(c_i)$).

Lemma 1. *The eligible area $EA(c_i)$ for each corner robot c_i in $\mathbb{P}$ is non-empty. Moreover, when c_i moves to a point inside $EA(c_i)$, then c_i remains as a corner of $\mathbb{P}$ and all internal and side robots in $\mathbb{P}$ are visible to c_i (and vice-versa).*

Safe Angle and Apex. Let u, v, w, x, y be points such that (a) v, w, x are collinear with w between v and x, (b) u, y are not collinear with line segment v, w, x, and (c) u, y lie on the same side of line v, w, x such that line segments $\overline{uv}$ and $\overline{xy}$ do not intersect. The figure below illustrates safe angles and apex. Define (non-reflex) angles $\theta_1, \theta_2 < 180°$ as $\theta_1 = \angle(u, v, w)$ and $\theta_2 = \angle(w, x, y)$. Let $\phi_1 = 45° - \frac{\theta_1}{4}$ and $\phi_2 = 45° - \frac{\theta_2}{4}$ be the "safe angles" for v and x, respectively. Let L_1 (resp., L_2) be the line traversing point v (resp., x) such that it forms an angle ϕ_1 (resp., ϕ_2) with line segment v, w, x as shown in the figure in the right. Since$\phi_1, \phi_2 < 45°$, lines L_1, L_2 will intersect on the side of line v, w, x, opposite to that of points u, y. Call this point of intersection h as the *safe apex* of w with respect to (or wrt)u, v, x, y.



Observe that if $\overline{vx}$ is a side S of $\mathbb{P}$ with v, x as corner points and w as a side point, and if u, y are adjacent corner points for side S, then define triangle v, x, h as the *safe area* for side S. The pseudocode outlining the technique of computing safe apex for a side robot s_i of $\mathbb{P}$ is omitted due to space constraints. This is used in Phase 2 of our algorithm.

3 Algorithm in the Fully Synchronous Model

We outline an $O(1)$ round algorithm for Complete Visibility in the fully synchronous model; we will then convert this algorithm for the semi-synchronous model in Sect. 4. Our algorithm consists of three phases converging toward a configuration where all the robots are in a convex hull (see Fig. 1). The goal of Phase 0 is to reposition robots (if needed) so that they are inside or on the corners and sides of a convex polygon $\mathbb{P}$. Phase 0 (initialization) is performed if a robot i sees only at most two other robots and the robots seen by i are in a line (this case happens only if $N \geq 2$ in $\mathcal{R}$ and all N robots are collinear). If i sees two other robots, it moves small distance δ directly perpendicular to the line joining $j, l \in \mathbb{C}(i)$. If $N \geq 3$, this action ensures that in the resulting configuration not all robots are collinear. Figure 1a depicts a worst case scenario where all robots are initially collinear.

Phase 1 (interior depletion) starts as soon as the robots in $\mathbb{C}_0$ reach a non-collinear configuration (robots on or in convex polygon $\mathbb{P}$). In Phase 1 the algorithm first identifies corner and side robots as follows (Fig. 1b). For robot i, if all other visible robots are within an angle of view of $< 180°$ (respectively, $= 180°$), then i is a corner (resp., side) robot of a convex polygon $\mathbb{P}$. The remaining robots (that lie in the interior of $\mathbb{P}$) are called "interior robots". Phase 1 moves all interior points of $\mathbb{P}$ to the sides of a slightly smaller convex polygon $\mathbb{P}'$ (Fig. 1e). It accomplishes this by first moving the corner robots of $\mathbb{P}$ to some point inside the eligible area in the interior of $\mathbb{P}$, where now all the corner robots are visible to the interior robots (Fig. 1b). The interior robots then move toward the closest corners of $\mathbb{P}'$ (Fig. 1c), and finally to the sides of $\mathbb{P}'$ (Fig. 1d). We show later that this phase runs in $O(1)$ rounds in any configuration of the robots.

Phase 2 (edge depletion) relocates side robots of $\mathbb{P}'$ to the corners of a slightly larger convex polygon $\mathbb{P}''$. It accomplishes this by moving only the side robots of $\mathbb{P}'$ into the safe area of the side they belong to. This proceeds by first moving two side robots that are neighbors of the corner robots of that side to the safe area (Fig. 1f), after that forming a circle segment using the information provided by three of (at most) four robots (two endpoints of the side, and one of the two robots that moved to the safe area, as shown in Fig. 1g), and then moving all other remaining side robots of that side perpendicularly to the points in the formed circle segment for the side. Figure 1h shows the resulting convex hull. This phase also runs in $O(1)$ rounds irrespective of the number of side robots in any side of $\mathbb{P}'$.

The pseudocode of the algorithm is omitted due to space constraints. Each robot i works autonomously having only the information about $\mathbb{C}(i)$. If $\mathbb{P}(i)$ is not a line segment for each $i \in \mathcal{R}$, then Phase 1 starts immediately. If $\mathbb{P}(i)$ is a line segment, then in one round, the robots in $\mathbb{C}_0$ result into a non-collinear configuration $\mathbb{C}_0$ and Phase 1 starts in the second round. Phase 1 proceeds autonomously until all (visible) robots are colored either `corner` or `side`. This acts as the starting configuration for Phase 2, which proceeds autonomously until all (visible) robots have color `corner`. The algorithm then terminates. The total number of colors used through Phases 0–2 is 9 and the algorithm runs for

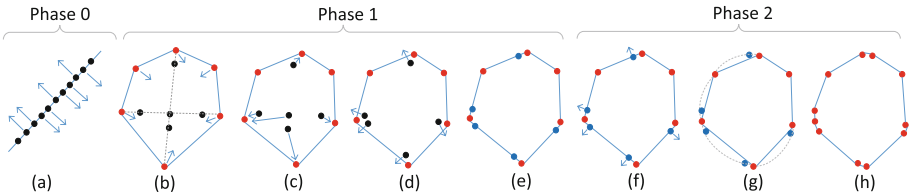


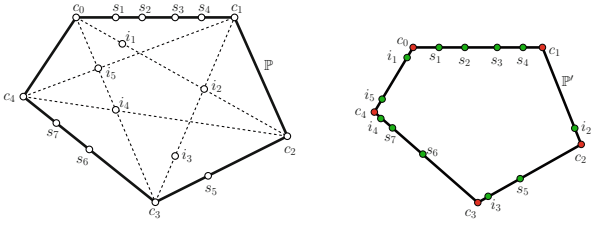
Fig. 1. The three phases of the algorithm.

(at most) 9 rounds. We provide details of Phases 0–2 separately below. The moves of each robot in Phases 0–2 can also be described as condition/action pairs (which we omit due to space constraints).

3.1 Phase 0 - Initialization

Initially, all $N \geq 1$ robots in $\mathcal{R}$ have color `start`. Assume that $\mathbb{C}_0$ is collinear (otherwise, this phase is not required). We have that $\mathbb{P}$ is a line segment. Let $r_j, \ldots, r_l$ be the robots in $\mathbb{P}$ (a line segment $\overline{r_j r_l}$) with r_j, r_l be the endpoints. Let x be a robot in $\overline{r_j r_l}$ between r_j, r_l and let y, z be two other robots it sees. Robot x moves perpendicular to line $\overline{yz}$ for a short distance δ keeping its color `start` in the first round. Robots r_j, r_l change their color to `ready` without moving, since they see only one other robot. At the end of the first round, it is impossible for robots in $\mathcal{R}$ to be in a straight line, if $N \geq 3$. Consequently, there exists polygonal $\mathbb{P}$ on or in which all robots lie. If $N = 2$, one robot sees the light of one other robot with color `ready` and figures out that there are only 2 robots in $\mathcal{R}$ and terminates. This happens at the second (and final) round. If a robot x sees no other robot, it can simply terminate.

3.2 Phase 1 - Interior Depletion



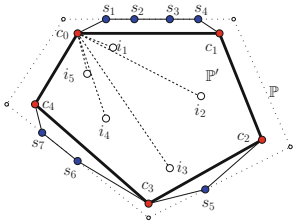
(a) Initial positions and colors (b) Final positions and colors

At the start of Phase 1, each robot is colored `start` or `ready`. All the robots in $\mathcal{R}$ are colored `start` if Phase 0 was not executed. A robot with color `ready` is located at a corner of $\mathbb{P}$ and a robot with color `start` is located at a corner or side or in the interior of $\mathbb{P}$. Let R_c, R_s, R_i be the sets of robots at corners, sides, and the interior of $\mathbb{P}$. Let $\mathbb{P}'$ be a convex hull formed by the robots in R_c after they moved to a point in their eligible areas, $EA(*)$, and have assumed color `corner`. Note that $\mathbb{P}'$ is completely contained inside $\mathbb{P}$. The goal of Phase 1 is to move the robots in sets R_s, R_i to make them side robots of $\mathbb{P}'$ with color `side`. Therefore, at the

end of Phase 1, all the robots are in the corners and edges of $\mathbb{P}'$ with corner robots colored `corner` and side robots colored `side`. The figure above illustrates Phase 1.

Phase 1 has four rounds. In Round 1.1, all corners of $\mathbb{P}$ become corners of $\mathbb{P}'$ with color `corner` and the side robots of $\mathbb{P}$ change their color to `side1` without moving. In Round 1.2, all interior robots of $\mathbb{P}$ (also interior in $\mathbb{P}'$) assume color `transit` moving closer to their closest corners in $\mathbb{P}'$ and the robots with color `side1` move to the closest sides of $\mathbb{P}'$ assuming color `side`. In Round 1.3, some `transit` colored robots become side robots of $\mathbb{P}'$ and, by the end of Round 1.4, all `transit` colored robots become side robots of $\mathbb{P}'$.

We give details on each round separately below.



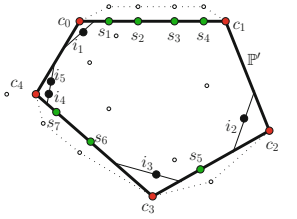
Round 1.1: Each corner r_c of $\mathbb{P}$ (the robot is in R_c) computes its eligible area $EA(r_c)$, moves to a point x in $EA(r_c)$, and assumes color `corner`. Since all the robots in R_c move simultaneously, they all become corners of $\mathbb{P}'$ by the end of Round 1.1 and do not move in any future rounds (Lemma 1). The side robots R_s of $\mathbb{P}$ also change their color to `side1` from `start` at Round 1.1 without moving. The interior robots of R_i do nothing. The figure on the right illustrates this round. We have the following results by the end of Round 1.1.

Lemma 2. *The set of robots R_i in the interior of $\mathbb{P}$ remain as interior robots of $\mathbb{P}'$.*

Observation 1. *Let r_c be a corner robot in $\mathbb{P}'$. Let r_{ccw}, r_{cw} be the neighbor corners of r_c in $\mathbb{P}'$ in the counterclockwise and clockwise directions of r_c, respectively, in its local coordinate system. Robot r_c sees both r_{ccw}, r_{cw}. (c_1 and c_4 for corner c_0 in the figure above.)*

Round 1.2: Since each $r_c \in R_c$ moved to $EA(r_c)$ and become a corner of $\mathbb{P}'$ in Round 1.1, all the robots in R_i, R_s see all corner robots of $\mathbb{P}'$ (Lemma 1) and each internal robot $r_i \in R_i$ can determine the closest corner robot r_c in $\mathbb{P}'$. Robot r_i can also see r_c's neighbors r_{ccw} and r_{cw} in $\mathbb{P}'$ (Lemma 1). Moreover, all robots in R_i are in the interior of $\mathbb{P}'$ (Lemma 2). We need the following definition.

Definition 1. Let r_c, r_{ccw}, r_{cw} be the robots defined in Observation 1. The line segment $\overline{xy}$ connects points x, y, where $x = \mathsf{length}(\overline{r_c r_{ccw}})/8$ from r_c in line $\overline{r_c r_{ccw}}$ and $y = \mathsf{length}(\overline{r_c r_{cw}})/8$ from r_c in line $\overline{r_c r_{cw}}$.



Interior robot r_i determines the line $\overline{xy}$, moves to the the intersection point z of $\overline{xy}$ and $\overleftrightarrow{r_i r_c}$, and assumes color `transit`. The robots in R_s (which were colored `side1` in Round 1.1) also see all the corner robots of $\mathbb{P}'$. Let S be a side of $\mathbb{P}'$ such that a robot $r_s \in R_s$ is in its corridor. Let $\hat{S} = \overline{x'y'}$ be the line segment connecting point x', y', where x' is

the point at S at distance $\mathsf{length}(S)/4$ from its one endpoint and y' is the point at distance $\mathsf{length}(S)/4$ from its other endpoint. Let p be the midpoint of $\hat{S}$ and $\alpha = \angle x' p r_s$. Robot r_s computes point $q = \frac{\alpha}{180^\circ} \cdot \mathsf{length}(\hat{S})$ from x' on $\hat{S}$, moves to q, and assumes color `side`. This computation of q guarantees that each angle α is mapped to a different position q on $\hat{S}$ and q does not coincide with either x' or y' [17]. The figure above illustrates this round. We have the following observations at the end of Round 1.2.

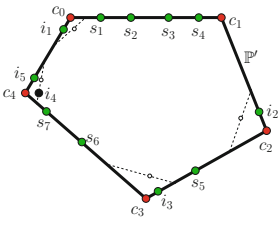
Observation 2. *The internal robots R_i are in the lines $\overline{xy}$ of the corner robots of $\mathbb{P}'$ and the side robots R_s are in the sides of $\mathbb{P}'$.*

Observation 3. *Let S be a side in $\mathbb{P}'$. If there are robots on S, they are in the positions of S between points $x' = \mathsf{length}(S)/4$ from one endpoint of S and $y' = \mathsf{length}(S)/4$ from the other endpoint of S.*

Round 1.3: The robots in R_s do not move in the remaining rounds of Phase 1 since they already have become side robots of $\mathbb{P}'$ in Round 1.2. Therefore, we only deal with the internal robots in R_i (all are colored `transit`) in this and the next round. Using the same notation as in Round 1.2, each $r_i \in R_i$ sees r_c (the closest corner of $\mathbb{P}'$) even after it has moved to point z in $\overline{xy}$. If r_i sees both r_{ccw} and r_{cw} (as defined in Observation 1), it can move to become a side robot of $\mathbb{P}'$ as follows: r_i draws two lines L and L' parallel to line segments $\overline{r_c r_{ccw}}, \overline{r_c r_{cw}}$, respectively, passing through r_i, then moves to the intersection point of $L, \overline{r_c r_{cw}}$ or $L', \overline{r_c r_{ccw}}$ whichever is closest to r_i (with respect to the distance from r_i to the intersection points) and assumes color `side`.

There are situations where r_i may not see r_{ccw} and/or r_{cw}, for example, the moves of other internal robots in R_i with the closest corners r_{ccw}, r_{cw} may block the visibility of r_i to see r_{ccw}, r_{cw}. In this case, r_i tries to find whether it sees two robots, s_a, s_b, with color `side` as the adjacent robots of r_c in $\mathbb{P}'$, instead of r_{ccw}, r_{cw}.

Lemma 3. *Let r_c be the closest corner of $r_i \in R_i$. If $r_i \in \overline{xy}$ sees two robots s_a, s_b with color* `side` *adjacent to r_c in the counterclockwise and clockwise direction of r_c, respectively, then all the robots in $\overline{xy}$ see both s_a, s_b.*



Robot r_i draws two lines L and L' parallel to line segments $\overline{r_c s_a}, \overline{r_c s_b}$, respectively (instead of $\overline{r_c r_{ccw}}, \overline{r_c r_{cw}}$), passing through r_i, and then moves to the intersection point of $L, \overline{r_c s_b}$ or $L', \overline{r_c s_a}$ whichever is closest to r_i (with respect to the distance from r_i to the intersection points) and assumes color `side`. The figure to the right illustrates this round. If each $r_i \in R_i$ sees either both s_a, s_b with color `side` or both r_{ccw}, r_{cw}, all the robots in R_i become side robots of $\mathbb{P}'$ in this round and Phase 1 finishes.

However, there are situations where all the robots in R_i may not even see both s_a, s_b. In this case, we have the following lemma.

Lemma 4. *Let $r_i \in R_i$ be the robot in line $\overline{xy}$ of corner r_c and $r_j \in R_i$ be the robot in line $\overline{xy}$ of corner r_{cw}. Suppose r_i, r_j are closest to the side $\overline{r_c r_{cw}}$ of $\mathbb{P}'$ among the robots in their lines $\overline{xy}$. At least one of r_i, r_j sees both r_c, r_{cw}.*

Proof. Robots r_i, r_c, r_{cw} form a triangle which is non-empty. Robot r_i sees r_c since there is no robot inside triangle $r_c xy$ and all robots closest to r_c are in line segment $\overline{xy}$. If r_i sees r_{cw}, we are done. Otherwise, r_j or some other robot in the line $\overline{xy}$ of r_{cw} must be collinear with side $\overline{r_i r_{cw}}$. Since r_j is the closest robot to side $\overline{r_c r_{cw}}$ and there is no robot in side $\overline{r_i r_c}$, r_j must see r_c. Moreover, r_j sees r_{cw} since r_{cw} is the closest corner to it. □

Therefore, either of r_i, r_j that sees both r_c, r_{cw} moves to the point at $\mathsf{length}(\overline{r_c r_{cw}})/4$ from its closest corner in $\mathbb{P}'$ in $\overline{r_c r_{cw}}$ and assumes color `side`.

Lemma 5. *Let $S = \overline{c_1 c_2}$ be a side of $\mathbb{P}'$. When a robot r' with light* `transit` *in a line $\overline{xy}$ of c_1 (or c_2) moves to a point at $\mathsf{length}(S)/4$ in S from c_1 (or c_2), then all the robots in lines xy of both c_1 and c_2 see r'.*

Proof. Let S' denote the other side incident on c_1 in $\mathbb{P}'$. Similar to Lemma 3, since the line $\overline{xy}$ of c_1 connects points $x = \mathsf{length}(S')/8$ from c_1 in S' and $x = \mathsf{length}(S)/8$ from c_1 in S and r' is in position $\mathsf{length}(S)/8$ from c_1 in S, the robots in $\overline{xy}$ can not be collinear with r'. Similarly, it holds for the robots in $\overline{xy}$ closest to c_2. □

Round 1.4: If Phase 1 did not finish in Round 1.3, each $r_i \in R_i$ sees r_c (the closest corner in $\mathbb{P}'$) and a robot each with light `side` as neighbors of r_c in both directions of r_c at the end of Round 1.3 (Lemma 5). The part b of the figure in the beginning of this section illustrates this round as i_4 moves to a side of $\mathbb{P}'$ and assumes color `side`. The move technique is similar to Round 1.3. We prove the following results for Phase 1.

Lemma 6. *At every round of Phase 1, each robot sees at least one robot with color from* {`start`, `ready`, `side1`, `transit`}.

Proof. Initially, all robots have color `start`. Some robot assume color `ready` if robots execute Phase 0. Therefore, at Round 1.1, each robot sees only `ready` or `start` colored robots. At Rounds 1.2 and 1.3 robots must see some robot with color in {`start`, `ready`, `side1`, `transit`}, otherwise there will be only `corner` and `side` colored robots in $\mathbb{P}'$ and Phase 1 execution is finished. □

Theorem 2. *Given a set of N robots placed on corners, sides, and interior of a convex polygon $\mathbb{P}$ such that all robots have color either* `start` *or* `ready`*, Phase 1 executes in at most four (fully synchronous) rounds avoiding collisions and uses* 6 *colors.*

Proof. We first prove that the rounds of Phase 1 follow in the order indicated by their names and if any round is skipped then all the robots are already colored either `corner` or `side` and they are in the perimeter of $\mathbb{P}'$.

Phase 1 begins when the set of colors visible to each robot is {`start`, `ready`}. This causes Round 1.1 to be executed by corner and side robots of $\mathbb{P}$. Since internal and side robots do not move until they see robots with light `corner`, Round 1.2 follows Round 1.1. All side robots of $\mathbb{P}$ become side robots of $\mathbb{P}'$ in Round 1.2 and do not move in future rounds. Similarly, Round 1.3 follows Round 1.2 since this is the first time internal robots have light `transit`. Similarly, Round 1.4 follows Round 1.3 since this is the first time the internal robots that did not see side robots as neighbors of their closest corner robot of $\mathbb{P}'$ will see such side robots.

Initially, all robots have color `start`. In Phase 0, only color `ready` is introduced. At Rounds 1.1, 1.2, and 1.3, colors {`side1`, `transit`, `corner`, `side`} are introduced. Therefore, there are total 6 colors.

We now show that the execution of Phase 1 is collision free. In Round 1.1, a corner robot r_c does not collide with any side or internal robot of $\mathbb{P}$ while moving to a point inside $EA(r_c)$ since there is no robot inside $EA(r_c)$. Robot r_c does not collide with any other corner robot r_d since eligible areas for any two corner robots of $\mathbb{P}$ do not overlap.

In Round 1.2, the robots with color `side1` of $\mathbb{P}$ moving to become side robots of $\mathbb{P}'$ with color `side` do not collide, since the technique they use to find a point to move to in $\mathbb{P}$ guarantees that all robots moving to a side do not collide. The interior robots with color `start` moving to the positions of lines $\overline{xy}$ of their closest corners of $\mathbb{P}'$ also do not collide.

The argument is as follows. For another internal robot r_j moving to the same line segment $\overline{xy}$ as r_i, r_c is visible to both r_i and r_j (Lemma 1), so the path from r_i to r_c is unobstructed and so is the path from r_j to r_c. Robot r_i moves along line $\overleftrightarrow{r_i r_c}$ to the intersection with line segment $\overline{xy}$, while r_j moves along line $\overleftrightarrow{r_j r_c}$ to its intersection with line segment $\overline{xy}$, so the paths of r_i and r_j do not cross and they do not collide.

For another internal robot r_k closest to a corner robot r_d different from r_c, r_k moves along line $\overleftrightarrow{r_k r_d}$ to a point z'. Every point on the path from r_k to z' is closer to r_d than to any other corner. Likewise, every point on the path from r_i to z is closer to r_c than to any other corner. Therefore, the paths of r_i and r_k do not cross and hence r_i and r_k do not collide.

In Round 1.3, either all robots in line $\overline{xy}$ move to become side robots of $\mathbb{P}'$ or the two endpoint robots among the robots in $\overline{xy}$ move to become side robots. In the first case, the robots fall in the positions of sides $\overline{r_c x}$ and $\overline{r_c y}$ where there are no robots in those sides and there are no robots inside triangle $r_c xy$. Their paths to the positions in $\overline{r_c x}$ and $\overline{r_c y}$ do not cross since they go to the closest side between $\overline{r_c x}$ and $\overline{r_c y}$. The moves of endpoint robots also do not collide since all the other internal robots are in lines $\overline{xy}$ of the corner robots of $\mathbb{P}'$ and its destination on the side it is moving to is not occupied by any other robot and there is no robot in its path.

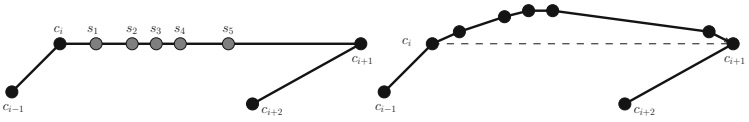
In Round 1.4, there is no collision using the similar argument as of Round 1.3. The theorem follows. □

3.3 Phase 2 - Edge Depletion

At the start of Phase 2, each robot is colored `corner` or `side` and is located at a corner or side of $\mathbb{P}'$. Let R_c, R_s be the set of robots at corners and sides, respectively of $\mathbb{P}'$. Phase 2 moves the robots of R_s to corners of an N-sided convex polygon $\mathbb{P}''$ that also has the robots of R_c as its corners. At the end of Phase 2, all robots have color `corner`.

Consider any side $S = \langle c_i, s_1, \cdots, s_m, c_{i+1} \rangle$, where c_i, c_{i+1} are corner points and $s_1, \cdots, s_m$ are side points. The other corner points of the side adjacent to S are c_{i-1} and c_{i+2}. For each side $S = \langle c_i, s_1, \cdots, s_m, c_{i+1} \rangle$, Phase 2 places all side points s_i on a circle segment traversing c_i and c_{i+1} and which has entirely within the safe area of S. To determine this circle, each robot needs to see three point on the circle. The figure to the right illustrates Phase 2 for a side with 5 side points (colored gray).

We will use this as a running example to illustrate this phase.



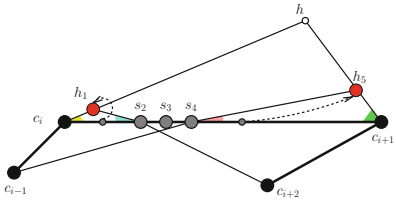
(a) Initial positions and colors (b) Final positions and colors

Phase 2 has four rounds. In Round 2.1, the side robots of $\mathbb{P}'$ that are neighbors of at least one corner of $\mathbb{P}'$ move to their safe apexes and assume color `scout1`. In Round 2.2, each scout for a particular side S of $\mathbb{P}'$ computes two circles (based on the two corners, itself, and the possibly another scout it sees), places itself on the circle with larger radius, and changes its color to `scout2`. If there is only one scout, the scout robot is already on a circle and can simply change its color to `scout`. In Round 2.3, two side robots in S that are now neighbors of their corners move to place themselves on the circle. This is simple since each of them can see a corner and two scouts on the circle. In Round 2.4, all remaining side robots of S move to the circle assuming color `corner` and the robots already in the circle change their color to `corner`.

We give details on each round separately below. For this discussion, we assume that $m \geq 5$. The case of $m < 5$ is explained later. Before we proceed, we develop a few results that will be useful later.

Round 2.1: Consider a robot s on side S that can see at least one of the two corners of S. Moreover, it cannot see any robot with color from {`start`, `ready`, `side1`, `transit`}.

The last sentence ensures that this condition is not met during Phase 1 (see Lemma 6). We assume this additional condition is added to each of the rounds of Phase 2 (although it is not needed, it will make the overall proof of correctness easier).



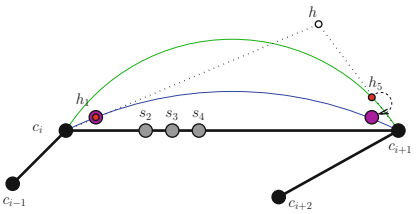
For our example, the two side points s_1, s_m bordering corner points, c_i, c_{i+1}, of S participate during this round; we will call these side points "extremal side points" of side S. Robot s_1 moves to its safe apex, h_1, wrt points $c_{i-1}, c_i, s_2, c_{i+2}$. Similarly, s_m moves to its safe apex, h_m, wrt $c_{i-1}, s_{m-1}, c_{i+1}, c_{i+2}$. Robots s_1, s_m assume color `scout1`.

The figure above illustrates Round 2.1. Color `scout1` is shown in red. Triangle c_i, c_{i+1}, h is the safe area for the entire side. For clarity, the safe angles for c_i, s_2 (due to point s_1) are shown in yellow and light blue, respectively. Similarly, the safe angles of s_4 and c_{i+1} are shown in pink and light green. Observe that s_1 easily determines its side S and that all of the points $c_{i-1}, c_i, s_2, c_{i+2}$ are visible to it.

Lemma 7. *At the end of Round 2.1, each scout of a side S can see the other scout (if any) of the side, as well as the corners c_i, c_{i+1} of side S.*

Round 2.2: It is well known (for example, [3, Sect. 7.2.3]) that for any three non-collinear points a, b, c, there is a unique circle that traverses a, b, c. Let $Circle(a, b, c)$ denote this circle. In this round, each scout for side S determines $Circle(c_i, h_1, c_{i+1})$ and $Circle(c_i, h_m, c_{i+1})$ and selects the one with the larger radius (flatter circle), denoted by $Circle(*)$ and called the *safe circle* of side S. From Lemma 7 each scout can see all the robots (including itself) to determine the circles $Circle(c_i, h_1, c_{i+1})$, $Circle(c_i, h_m, c_{i+1})$ and, hence, $Circle(*)$. The scout then moves (if needed) to position itself on the safe circle; this movement could, for example, be perpendicular to side S. It then changes its color to `scout2`.

The figure on the right illustrates these ideas. Color `scout2` is shown in purple; $Circle(c_i, h_1, c_{i+1}) = Circle(*)$ and $Circle(c_i, h_m, c_{i+1})$ are shown in blue and green, respectively. Observe that the arc of $Circle(*)$ between corners c_i, c_{i+1} lies entirely within the safe area of side S. Also note that placing corner points on $Circle(*)$ (over all sides S) will keep the polygon convex.



Round 2.3: Here the current extremal side points of S (s_2, s_4 in our example) move to the safe circle $Circle(*)$. This is straightforward as each extremal point can see a corner and two scouts (totally three non-collinear points) that are on $Circle(*)$.

We now explain this and the next round as condition/action pair (C,A). That is, each robot that satisfies a condition C performs a corresponding action A. The appendix gives all steps of the algorithm as condition/action pairs.

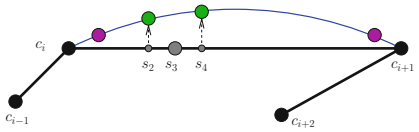
Condition 2.3.1: Robot s colored `side` is on side S and it can see at least one of the two corners of S and two points colored `scout2` in the exterior half-plane

of S. Moreover, it cannot see any robot with color from {`anchor`, `start`, `ready`, `side1`, `transit`}.

Action 2.3.1: Robot s moves to the safe circle, $Circle(*)$, of S and colors itself `anchor`.

Figure on the right illustrates Round 2.3. Color `anchor` is shown in green.

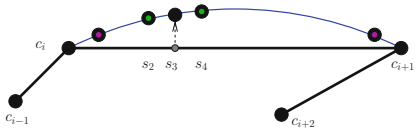
This round has another move needed for the case $m < 5$; this is explained later.



Condition 2.3.2: Robot s colored `scout2` corresponding to side S can see no points colored `side` in S. Moreover, it cannot see any robot with color from {`start`, `ready`, `side1`, `transit`}.

Action 2.3.2: Robot s colors itself `corner`.

Round 2.4: At this stage, every side robot on S can see at least four points on the safe circle $Circle(*)$. This allows the side point to determine $Circle(*)$ and position itself on the circle. All points are now placed on corners of a new convex polygon $\mathbb{P}''$.



Condition 2.4.1: Robot s colored `side` is on side S and it can see a point colored `anchor` in the exterior half-plane of S. Moreover, it cannot see any robot with color from {`start`, `ready`, `side1`, `transit`}.

Action 2.4.1: Robot s moves perpendicular to S and places itself on $Circle(*)$. It changes its color to `corner`.

Robot s determines $Circle(*)$ as in Round 2.2. Observe that while our example shows only one robot, s_4, performing the above action, the above condition would be satisfied by all remaining side points on S and the action executed in parallel by all of them.

The following condition/action pair is to change the colors of robots already in $Circle(*)$ to `corner`.

Condition 2.4.2: Robot x has color from {`scout2`, `anchor`} and it can see a robot of color `anchor` (including possibly itself). Moreover, it cannot see any robot with color from {`start`, `ready`, `side1`, `transmit`}.

Action 2.4.2: Robot x changes its color to `corner`.

Here all that matters is that every anchor or scout will be able to see an anchor; whether the anchor is in its own corridor or not is not important. Figure on the right illustrates Round 2.4.

We have assumed that the number of side points m on side S is at least 5. If $m < 5$, then no robot satisfies Condition 2.4.1. For $m = 4$, four robots satisfy Condition 2.4.2 and the phase ends with these robots at corners of convex poly-

gon $\mathbb{P}'$. If $m = 3$, then only one robot satisfies Condition 2.3.1 and three robots satisfy Condition 2.4.2. If $m = 2$, then the two scouts were the only side points on S. They change to color `corner` at Round 2.3 and terminate Phase 2 for the side. If $m = 1$, then there is only one scout and, again, Phase 2 terminates for side S after Round 2.3. If $m = 0$ (no side points), then the side does not execute Phase 2; notice that none of the conditions in the four rounds are satisfied by robots of color `corner`.

We have following theorem for the correctness of Phase 2 (a more systematic proof of overall correctness of the algorithm is omitted due to space constraints).

Theorem 3. *Given a set of N robots placed on corners and sides of a convex polygon $\mathbb{P}'$ with all corner robots colored* `corner` *and all side robots colored* `side`*, Phase 2 executes in at most* 4 *(fully synchronous) rounds avoiding collisions and uses* 5 *colors (out of which 2 colors are common with Phase 1).*

Proof. We first prove that the rounds of Phase 2 follow in the order indicated by their names and that if any side skips a round, then it has completed its side depletion with all robots, originally on that side, now colored `corner`. This does not affect the progress of other sides as corner robots of a side S are needed only by non-corner robots of side S to determine S itself and its exterior.

Initially, let us assume that the number of side points on a side S is $m \geq 5$. Phase 2 begins when the set of colors visible to each robot is {`corner`, `side`}. This causes Round 2.1 to be executed by robots satisfying Condition 2.1. Since S has a side point, all non-extremal elements of this side see a robot of color `scout1`. So Round 2.2 follows Round 2.1 as the color `scout1` appears only at the end of Round 2.1. Similarly, Rounds 2.3 follows Round 2.2 as this is the only time the color `scout2` is visible without color `anchor`. Now Round 2.4 follows Round 2.3 as this the only time `anchor` is visible.

It is easy to verify that when m (the number of side points in side S) is 4 or 3, then all four rounds are executed in the above order. If $m = 1, 2$, then after rounds 2.1, 2.2, and 2.3 the robots of side S are all colored `corner` and Round 2.4 is skipped. This does not impact other sides executing Round 2.4 as the color `corner` does not affect the conditions of this round.

We now show that the execution of Phase 2 is collision free. The robots in two different sides never collide since they do not go outside the corridor of the side they belong to at the end of Phase 1. In Round 2.1, according to the safe apex computation, the at most 2 robots moving to the safe apex for each do not collide. From Round 2.2 until Round 2.4 robots move perpendicularly to the side of $\mathbb{P}'$. □

We have the following theorem combining the results of Theorems 2 and 3.

Theorem 4. *For any initial configuration of $N \geq 1$ robots with lights, there is an algorithm that solves* Complete Visibility *avoiding collisions and has runtime of 9 rounds and uses 9 colors in the fully synchronous model of computation.*

4 Conversion to the Semi-synchronous Model

We now discuss how to convert the fully synchronous algorithm (Sect. 3) to the semi-synchronous model. The technique for Phase 0 converts similarly to Phase 0 for the semi-synchronous model and needs at most one (semi-synchronous) round. For Phases 1 and 2, we describe the difficulty in converting it to work in the semi-synchronous model, and then how we handle the difficulty. (The detailed description of each round of Phases 0–2 is omitted due to space constraints.)

Since not all corner robots are able to move to $EA(*)$ in the same cycle in the semi-synchronous model, an internal robot r_i may not see three corner robots (one closest corner and its two neighbor corners) necessary to make a move to become `transit` colored robot. Therefore, to be able to handle this situation for internal robots, we need at most 5 (semi-synchronous) rounds and two new colors `corner1`,`corner2` while converting Round 1.1 of the fully synchronous model to the semi-synchronous model.

In Round 1.1, instead of directly changing the color of corner and side robots of $\mathbb{P}$ from {`ready`, `start`} to `corner`, each corner robot r_c of $\mathbb{P}$ that moves to some point in $EA(r_c)$ changes its color to `corner1` and each side robot r_s of $\mathbb{P}$ changes its color to `side1` without moving. Therefore, by the end of Round 1.1, at least all corner robots of $\mathbb{P}$ have color `corner1` (and do not move in future rounds) and all side robots of $\mathbb{P}$ have color `side1`. In Round 1.2, side robots that are now corners and neighbor of r_c change their color to `corner1` after moving to $EA(*)$ if their color $\notin$ {`corner1`, `corner2`, `corner`}. After both the neighbors of any corner robot r_c have color $\in$ {`corner1`, `corner2`, `corner`}, then r_i sees three corner robots needed to become `transit` colored robot. However, if there are robots inside triangle r_cxy (points x, y are defined similarly as in Sect. 3), r_i's view of neighbor corner robots r_{ccw} and r_{cw} of r_c (Observation 1) may be blocked by other internal robots that have already moved to become `transit` colored robots, and therefore r_i might perceive a wrong view of r_{ccw} and r_{cw}. To avoid this situation, we use the technique in which if a corner r_c has color `corner1`, both of its neighbors in $\mathbb{P}$ have color $\in$ {`corner1`, `corner2`, `corner`}, and there are robots inside triangle r_cxy, then robot r_c changes its color to `corner2`. That means an internal robot r_i waits until r_c is colored `corner`. Using this technique, all corners r_c of $\mathbb{P}$ with robots inside triangle r_cxy will be colored `corner2` by the end of Round 1.3. By the end of Round 1.4, all the robots inside triangle r_cxy can move to line $\overline{xy}$ and assume color `transit`. Then, by the end of Round 1.5, all the corner robots of $\mathbb{P}$ assume color `corner`. This finishes the conversion of Round 1.1 of the fully synchronous model to the semi-synchronous model and the conversion makes sure that when an internal robot r_i sees r_c colored `corner`, it also sees its two neighbors r_{ccw} and r_{cw} that are in fact corners of $\mathbb{P}$.

Note also that by the end of Round 1.5, there is no internal robot in r_cxy of any corner r_c of $\mathbb{P}$. Therefore, similar to Round 1.2 of the fully synchronous model, by the end of Round 1.6, all internal robots of $\mathbb{P}$ move to lines $\overline{xy}$ and assume color `transit`. Furthermore, by the end of Round 1.8, all `transit` colored robots assume color `side`, all robots with color `side1` (if any) assume color `side`, and the robots with color `corner1`, `corner2` assume color `corner`. In

other words, Rounds 1.2, 1.3, and 1.4 of the fully synchronous model convert to (semi-synchronous) Rounds 1.6, 1.7, and 1.8.

The conversion for Phase 2 is relatively simple and works for the semi-synchronous model with no change in the number of rounds. However, robots may face ambiguity about identifying the exterior direction of the side of $\mathbb{P}$ in Round 2.4 which is handled introducing one additional color `corner3`. We have the following theorem.

Theorem 5. *For any initial configuration of $N \geq 1$ robots with lights, there is an algorithm that solves* Complete Visibility *avoiding collisions and has runtime of 13 rounds and uses 12 colors in the semi-synchronous model of computation.*

We obtain Theorem 1 combining the results of Theorems 4 and 5.

5 Concluding Remarks

We have presented, to our knowledge, the first algorithm for Complete Visibility in the robots with lights model that has runtime of $O(1)$ rounds and uses $O(1)$ colors in the semi-synchronous (and also in the fully synchronous) model. This problem is fundamental with application in solving other problems, e.g., on the fully synchronous model, gathering robots to a point requires only one round beyondComplete Visibility.

Several questions remain open. Our solution assumes no intervention by an adversary.Can this be relaxed? Our solution assumes semi-synchrony.Is a similar algorithm possible for asynchronous robots?

References

1. Agathangelou, C., Georgiou, C., Mavronicolas, M.: A distributed algorithm for gathering many fat mobile robots in the plane. In: PODC, pp. 250–259 (2013)
2. Ando, H., Suzuki, I., Yamashita, M.: Formation and agreement problems for synchronous mobile robots with limited visibility. In: ISIC, pp. 453–460 (1995)
3. Barry, P.D.: Geometry with Trigonometry. Horwood Publishing Limited, Chichester (2001)
4. Cohen, R., Peleg, D.: Local spreading algorithms for autonomous robot systems. Theor. Comput. Sci. **399**(1–2), 71–82 (2008)
5. Cord-Landwehr, A., et al.: A new approach for analyzing convergence algorithms for mobile robots. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 650–661. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22012-8_52
6. Czyzowicz, J., Gasieniec, L., Pelc, A.: Gathering few fat mobile robots in the plane. Theor. Comput. Sci. **410**(6–7), 481–499 (2009)
7. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: Autonomous mobile robots with lights. Theor. Comput. Sci. **609**, 171–184 (2016)
8. Degener, B., Kempkes, B., Langner, T., Meyer auf der Heide, F., Pietrzyk, P., Wattenhofer, R.: A tight runtime bound for synchronous gathering of autonomous robots with limited visibility. In: SPAA, pp. 139–148 (2011)

9. Degener, B., Kempkes, B., Meyer auf der Heide, F.: A localo(n^2) gathering algorithm. In: SPAA, pp. 217–223 (2010)
10. Flocchini, P., Prencipe, G., Santoro, N.: Distributed computing by oblivious mobile robots. Synth. Lect. Distrib. Comput. Theor. **3**(2), 1–185 (2012)
11. Izumi, T., Potop-Butucaru, M.G., Tixeuil, S.: Connectivity-preserving scattering of mobile robots with limited visibility. In: Dolev, S., Cobb, J., Fischer, M., Yung, M. (eds.) SSS 2010. LNCS, vol. 6366, pp. 319–331. Springer, Heidelberg (2010). doi:10.1007/978-3-642-16023-3_27
12. Kempkes, B., Kling, P., Meyer auf der Heide, F.: Optimal and competitive runtime bounds for continuous, local gathering of mobile robots. In: SPAA, pp. 18–26 (2012)
13. Luna, G.A., Flocchini, P., Gan Chaudhuri, S., Santoro, N., Viglietta, G.: Robots with lights: overcoming obstructed visibility without colliding. In: Felber, P., Garg, V. (eds.) SSS 2014. LNCS, vol. 8756, pp. 150–164. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11764-5_11
14. Pagli, L., Prencipe, G., Viglietta, G.: Getting close without touching. In: Even, G., Halldórsson, M.M. (eds.) SIROCCO 2012. LNCS, vol. 7355, pp. 315–326. Springer, Heidelberg (2012). doi:10.1007/978-3-642-31104-8_27
15. Peleg, D.: Distributed coordination algorithms for mobile robot swarms: new directions and challenges. In: Pal, A., Kshemkalyani, A.D., Kumar, R., Gupta, A. (eds.) IWDC 2005. LNCS, vol. 3741, pp. 1–12. Springer, Heidelberg (2005). doi:10.1007/11603771_1
16. Prencipe, G.: Autonomous mobile robots: a distributed computing perspective. In: Flocchini, P., Gao, J., Kranakis, E., Meyer auf der Heide, F. (eds.) ALGOSENSORS 2013. LNCS, vol. 8243, pp. 6–21. Springer, Heidelberg (2014). doi:10.1007/978-3-642-45346-5_2
17. Vaidyanathan, R., Busch, C., Trahan, J.L., Sharma, G., Rai, S.: Logarithmic-time complete visibility for robots with lights. In: IPDPS, pp. 375–384 (2015)
18. Yamashita, M., Suzuki, I.: Characterizing geometric patterns formable by oblivious anonymous mobile robots. Theor. Comput. Sci. **411**(26–28), 2433–2453 (2010)