Fault-Tolerant Complete Visibility for Asynchronous Robots with Lights Under One-Axis Agreement

Aisha Aljohani, Pavan Poudel, and Gokarna Sharma(✉)

Department of Computer Science, Kent State University, Kent, OH 44242, USA
aaljoha6@kent.edu, {ppoudel,sharma}@cs.kent.edu

Abstract. We consider the distributed setting of N autonomous mobile robots that operate in *Look-Compute-Move* (LCM) cycles and communicate with other robots using colored lights under the *robots with lights* model. We study the fundamental Complete Visibility problem of repositioning N robots on a plane so that each robot is visible to all others. We assume *obstructed visibility* under which a robot cannot see another robot if a third robot is positioned between them on the straight line connecting them. We are interested in *fault-tolerant* algorithms. We study fault-tolerance with respect to failures on the mobility of robots. Therefore, any algorithm for Complete Visibility is required to provide visibility between all non-faulty robots, independently of the behavior of the faulty ones. We model mobility failures as *crash faults* in which each faulty robot is allowed to stop its movement at any time and, once the faulty robot stopped moving, that robot will remain stationary indefinitely thereafter. There exists an algorithm for this problem that tolerates a single faulty robot in the semi-synchronous setting under both-axis agreement. In this paper, we provide the first algorithm for this problem that tolerates $f \leq N$ faulty robots in the asynchronous setting under one-axis agreement. The proposed algorithm is *collision-free* – robots do not share positions and their paths do not cross, *energy efficient* – each robot performs at most one move, and handles *non-rigidity* of the robot movements.

1 Introduction

In the well-celebrated classical model of distributed computing by mobile robots, each robot is modeled as a point in the plane [11]. The robots are assumed to be *autonomous* (no external control), *anonymous* (no unique identifiers), *indistinguishable* (no external identifiers), and *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: When a robot becomes active, it first gets a snapshot of its surroundings (*Look*), then computes a destination point based on the snapshot (*Compute*), and finally moves towards the destination point (*Move*). Moreover, the robots are *oblivious*, i.e., in each

M. S. Rahman et al. (Eds.): WALCOM 2018, LNCS 10755, pp. 169–182, 2018.
https://doi.org/10.1007/978-3-319-75172-6_15

LCM cycle, each robot has no memory of its past LCM cycles [11]. Furthermore, the robots are *silent* because they do not communicate directly, and only vision and mobility enable them to coordinate their actions.

While silence has advantages, direct communication is preferred in many other situations, for example, hostile environments, which makes coordination efficient and relatively viable. One model that incorporates direct communication is the *robots with lights* model [9,11,15], where each robot has an externally visible light that can assume colors from a constant sized set, and hence robots can 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 lights, the robots are oblivious as in the classical model.

Di Luna *et al.* [13] gave the first algorithm for robots with lights to solve the fundamental Complete Visibility problem defined as follows: Given an arbitrary initial configuration of N autonomous mobile robots located in distinct points on a plane, they reach a configuration in which each robot is in a distinct position from which it can see all other robots. 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 importance of this problem is that it makes it possible to solve many other robotic problems, including gathering, shape formation, and leader election, under obstructed visibility [12,16]. Most importantly, it recovers unobstructed visibility configuration starting from an obstructed visibility configuration. Subsequently, several papers [12,16] solved this problem minimizing number of colors. Recently, faster runtime algorithms [18–20] were studied for this problem in the lights model (details in **Related Work**). This problem is also called Mutual Visibility in some papers [12,16].

In this paper, we are interested in the fault-tolerant algorithms for Complete Visibility in the robots with lights model. We study fault-tolerance with respect to failures on the mobility of robots. Therefore, any algorithm for Complete Visibility is required to provide visibility between all non-faulty robots, independently of the behavior of the faulty ones and the locations of the faulty robots. We model mobility failures as *crash faults* where each faulty robot is allowed to stop its movement at any moment of time and remains stationary indefinitely thereafter [2]. The only previous work that studied faults for this problem is [3] in which the authors solved the problem for a single faulty robot in the semi-synchronous setting under both-axis agreement. In this paper, we focus on solving this problem tolerating $f > 1$ faulty robots in the weakest fully asynchronous setting and under weaker one-axis agreement.

Contributions. We consider the same robot model as in [12,13], 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 and N is not known. We assume that the setting is *asynchronous* where there is no notion of common time and robots perform their LCM cycles at arbitrary time. We also assume *non-rigid* moves – a robot in motion can be stopped (by an adversary) before it reaches its destination point with the only constraint that the robot moves at least distance $\Delta > 0$, otherwise Complete Visibility

cannot be solved [12]. As in [13], we assume that two robots cannot head to the same destination and their paths when they move cannot cross. This would constitute a *collision*. Furthermore, we assume *one-axis agreement* – all robots agree on either x-axis or y-axis [11]. In this paper, we prove the following result.

Theorem 1. *For any input configuration of $N \geq 3$ robots (with lights) in distinct positions in a plane,* Complete Visibility *can be solved tolerating (up to) N crash-faulty robots using 4 colors in $\mathcal{O}(N)$ time without collisions in the asynchronous setting.*

To the best of our knowledge, Theorem 1 is the first result for Complete Visibility that tolerates (up to) N faulty robots in the asynchronous setting. In the semi-synchronous (and also fully synchronous) setting, Theorem 1 only needs 2 colors, which is optimal with respect to the number of colors used [12]. One prominent feature of our algorithm is that each robot moves at most once during the execution and it has implications on energy efficiency of robots on solving Complete Visibility.

When the robots are fault-free, the idea used in the existing algorithms [12,13,16,18–20] is to reposition the robots so that they all become corners of a N-corner convex hull. After that, a property of the convex hull guarantees that there is a line connecting each corner with all others of the hull without any third robot being collinear on those lines, i.e., a convex hull naturally solves Complete Visibility. However, when robots are faulty, the faulty robots may be in the interior of the convex hull and it is challenging to guarantee that all non-faulty robots see each other (that is, faulty robots do not block the view for the non-faulty robots to see each other). Since robots are oblivious and non-faulty robots do not know which robots are faulty, this task becomes quite challenging. Aljohani and Sharma [3] managed to address this challenge only when at most one robot in the interior of the hull experiences fault. In this paper, we develop a technique in which non-faulty robots do not need to be positioned on the corners of a hull to see other non-faulty robots and this gives the scalability on number of faults that can be tolerated. Our idea is to move the robots in a sequence starting from the Southmost robot and ending at the Northmost robot, and guarantee that, when a robot makes a move, it moves to a position in such a way that it sees from that position all robots that are positioned South of it (both faulty and non-faulty).

Related Work. Di Luna *et al.* [13] gave the first algorithm for Complete Visibility in the robots with lights model. They solved the problem using 6 colors in the semi-synchronous setting and 10 colors in the asynchronous setting under both rigid and non-rigid movements. Di Luna *et al.* [12] solved the problem using 2 colors in the semi-synchronous setting under rigid movements. They solved the problem using 3 colors in the semi-synchronous setting under non-rigid movements and in the asynchronous setting under rigid movements. They also provided a solution using 3 colors in the asynchronous setting under non-rigid movements under one-axis agreement. Sharma *et al.* [16] improved the number of colors in the solution of Di Luna *et al.* [12] from 3 to 2. In the classical

oblivious model (with no lights), Bhagat *et al.* [5] solved Complete Visibility under one-axis agreement without the need of robots to be positioned on the corners of a convex hull. However, all these results provided no runtime analysis. Moreover, none of these results tolerate faults.

Vaidyanathan *et al.* [20] considered runtime for the very first time for Complete Visibility giving an algorithm that runs in $\mathcal{O}(\log N)$ time using $\mathcal{O}(1)$ colors in the fully synchronous setting under rigid movements. Later, Sharma *et al.* [18] provided an $\mathcal{O}(1)$ time algorithm using $\mathcal{O}(1)$ colors in the semi-synchronous setting under rigid movements. Recently, Sharma *et al.* [17,19] provided an $\mathcal{O}(1)$ time algorithm using $\mathcal{O}(1)$ colors in the asynchronous setting under rigid movements. However, all these algorithms are not fault-tolerant. Aljohani and Sharma [3] provided an algorithm that tolerates one faulty robot when robots have both axis agreement in the semi-synchronous setting under rigid movements. The algorithm we present in this paper assumes one-axis agreement, handles non-rigid movements, and works in the fully asynchronous setting.

The computational power of the robots with lights model is studied in [9] while the robots are working on the Euclidean plane and in [10] while the robots are working on graphs.

The obstructed visibility, in general, is considered in the problem of uniformly spreading robots operating on a line [6] and also in the near-gathering problem [14] where collisions must be avoided among robots. It is also considered in the so-called *fat robots* model [1,8] in which robots are not points, but non-transparent unit discs. However, these works do not consider faulty robots. The faults are considered for the gathering problem in the classical oblivious robots model [2,4]. Our definition of crash faults is borrowed from [2].

Paper Organization. The rest of the paper is organized as follows. We present the robot model and preliminaries in Sect. 2. We then present and analyze our fault-tolerant Complete Visibility algorithm in Sect. 3 and conclude in Sect. 4. Some proofs and pseudocodes are omitted due to space constraints.

2 Model and Preliminaries

Robots. We consider a distributed system of N autonomous robots from a set $\mathcal{Q} = \{r_1, \ldots, r_N\}$. Each robot $r_i \in \mathcal{Q}$ is a (dimensionless) point that can move in the two-dimensional Euclidean plane $\mathbb{R}^2$. Throughout the paper, we denote by r_i the robot r_i as well as its position p_i in $\mathbb{R}^2$. We assume that each robot $r_i \in \mathcal{Q}$ shares one coordinate axis with other robots in $\mathcal{Q}$, i.e., they agree on either x-axis or y-axis (we use y-axis).

A robot r_i can see, and be visible to, another robot r_j if and only if there is no third robot r_k in the line segment $\overline{r_i r_j}$ connecting r_i and r_j. Each robot $r_i \in \mathcal{Q}$ has a light that can assume a color at a time from a set of constant number of different colors. We denote the color of a robot $r_i \in \mathcal{Q}$ at any time by variable $r_i.light$. If $r_i.light =$ `Red`, then it means that r_i has color `Red`. Moreover, the color `Red` of r_i is seen by all robots that can see r_i at that time (r_i also can see its current color). The execution starts at time $t = 0$ and at time $t = 0$ all robots in $\mathcal{Q}$ are stationary with each of them colored `Off`.

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

- *Look:* For each robot r_j that is visible to it, r_i can observe the position of r_j on the plane and the color of the light of r_j. Robot r_i can also observe its own color and position; that is, r_i is visible to itself. Each robot observes positions 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 LCM cycle, r_i may perform an arbitrary computation using only the colors and positions observed during the "look" portion of that LCM cycle. This includes determination of a (possibly) new position and color for r_i for the start of next LCM cycle. Robot r_i maintains this new color from that cycle to the next.
- *Move:* At the end of the LCM cycle, r_i changes its light to the new color and moves to its new position.

Robot Activation. In the fully synchronous setting ($\mathcal{FSYNC}$), every robot is active in every LCM cycle. In the semi-synchronous setting ($\mathcal{SSYNC}$), at least one robot is active, and over an infinite number of LCM cycles, every robot is active infinitely often. In the asynchronous setting ($\mathcal{ASYNC}$), there is no common notion of time and no assumption is made on the number and frequency of LCM cycles in which a robot can be active. The only guarantee is that every robot is active infinitely often. The moves of the robots may be *non-rigid* – during the *Move* phase the robots move in a straight line but they may stop their movement before they reach to the destination point computed in the *Compute* phase, with the only exception that they move at least some distance $\Delta > 0$. We assume that the faulty robot can crash at any moment of time. After the robot crashes, it does not move again (i.e., stays stationary indefinitely). However, even after the robot crashes, we assume that it does not have impact on the operations of its light. That is, the robot can correctly change its color to any color in the color set according to the algorithm. We will argue in Sect. 4 that it seems necessary to guarantee termination tolerating $f > 1$ robot faults even under one-axis agreement.

Runtime. For the $\mathcal{FSYNC}$ model, we measure time in rounds, where one round is one LCM cycle. As a robot in the $\mathcal{SSYNC}$ (and $\mathcal{ASYNC}$) model could stay inactive for an indeterminate number of cycles and (time), we use the notion of an epoch to measure runtime [7]. Let t_0 denote the start time of the computation. Epoch i is time period from t_{i-1} to t_i where t_i is the earliest time after t_{i-1} when all robots have executed a complete LCM cycle at least once. In the $\mathcal{FSYNC}$ model, an epoch is one round (one LCM cycle). We will use the term "time" generically to mean rounds for the $\mathcal{FSYNC}$ model and epochs for the $\mathcal{SSYNC}$ and $\mathcal{ASYNC}$ models.

Configuration. A *configuration* $\mathbf{C}_t = \{(r_1^t, col_1^t), \ldots, (r_N^t, col_N^t)\}$ defines the positions of the robots in $\mathcal{Q}$ and their colors for any time $t \geq 0$. A configuration for a robot $r_i \in \mathcal{Q}$, $\mathbf{C}_t(r_i)$, defines the positions of the robots in $\mathcal{Q}$ that are visible to r_i (including r_i) and their colors, i.e., $\mathbf{C}_t(r_i) \subseteq \mathbf{C}_t$, at time t. For simplicity, we sometime write $\mathbf{C}, \mathbf{C}(r_i)$ to denote $\mathbf{C}_t, \mathbf{C}_t(r_i)$, respectively.

Visible Area. Let $\mathcal{A}$ be a set of points (which are the current positions of the robots in $\mathbb{R}^2$) and $\mathbf{P}$ be the convex hull of the points in $\mathcal{A}$. $\mathbf{P}$ has the property that all the points of $\mathcal{A}$ are either in the perimeter or in its interior. The points in the perimeter of $\mathbf{P}$ are either on corners of $\mathbf{P}$ or on the edges of $\mathbf{P}$, which we call corner and side points of $\mathbf{P}$, respectively. Let $\mathcal{Q}_c, \mathcal{Q}_s, \mathcal{Q}_i$ be the set of points at corners, sides, and the interior of $\mathbf{P}$. Moreover, let c_i be a corner point of $\mathbf{P}$ and a, b be the counterclockwise and clockwise neighbors of c_i in the perimeter of $\mathbf{P}$. The *visible area* for c_i, denoted as $Visible\_Area(c_i)$, is a polygonal subregion inside $\mathbf{P}$ within the triangle c_iuw, where u, w are the midpoints of edges $\overline{c_ia}, \overline{c_ib}$, respectively. According to this computation, the visible areas for any two corner points of $\mathbf{P}$ are disjoint. Due to obstructed visibility, $Visible\_Area(c_i)$ is computed based on $\mathbf{C}(c_i)$ and the corresponding hull $\mathbf{P}(c_i)$. This computation is used in Sect. 3.

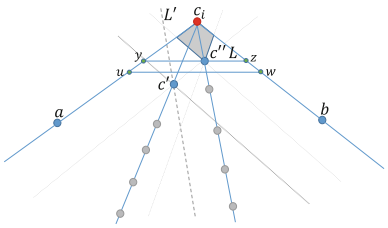

Fig. 1. Visible area

We now outline how $Visible\_Area(c_i)$ is computed for any corner c_i of $\mathbf{P}$. The pseudocode is omitted due to space constraints. Initially, c_i sets the triangle c_iuw as its $Visible\_Area(c_i)$. However, if c_i sees some point of $\mathcal{A}$ inside c_iuw, then it sets as $Visible\_Area(c_i)$ 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 $\mathbf{C}(c_i)$. For every other point $c'' \in \mathbf{C}(c_i), c'' \neq c', c'' \neq c_i$, 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 $Visible\_Area(c_i)$ by keeping only the portion of $Visible\_Area(c_i)$ that is in the half-plane HP. This process is repeated for all $c' \in \mathbf{C}(c_i)\backslash\{c_i\}$ and $Visible\_Area(c_i)$ is updated in every iteration. Now from the area $Visible\_Area(c_i)$ that remains, c_i removes the points that are in the perimeter of $Visible\_Area(c_i)$ and also the points that are part of the lines $\overleftrightarrow{c_ix}, x \in \mathbf{C}(c_i)\backslash\{a, b, c_i\}$, passing inside of $Visible\_Area(c_i)$. This removal of points is crucial to guarantee that when c_i moves to a point in $Visible\_Area(c_i)$, it does not become collinear with any robot in $\mathcal{Q}_s, \mathcal{Q}_i$. Figure 1 illustrates the computation of $Visible\_Area(c_i)$; the shaded area is $Visible\_Area(c_i)$ for corner c_i of $\mathbf{P}$ except the points on the lines inside it (e.g., the point of lines $\overline{c_ic'}$ and $\overline{c_ic''}$ inside $Visible\_Area(c_i)$. We have the following lemma from [18].

Lemma 1. *$Visible\_Area(c_i)$ for each corner robot c_i in $\mathbf{P}$ is non-empty. Moreover, when c_i moves to a point inside $Visible\_Area(c_i)$ and no other robot in $\mathbf{P}$ is moving simultaneously with c_i, then c_i remains as a corner of $\mathbf{P}$ and all the other robots in $\mathbf{P}$ are visible to c_i (and vice-versa).*

3 Algorithm

In this section, we present our COMPLETE VISIBILITY algorithm for $N \geq 3$ robots with lights tolerating $f \leq N$ faulty robots, starting from any arbitrary initial configuration with robots being in the distinct positions in a plane. The algorithm works in the $\mathcal{ASYNC}$ setting handling non-rigid moves, under the assumption that robots have one-axis agreement. We first provide a high level overview and then give its details.

Algorithm 1. COMPLETE VISIBILITY for a robot $r_i \in \mathcal{Q}$ in the $\mathcal{ASYNC}$ model

1 *// Look-Compute-Move cycle for each robot* $r_i \in \mathcal{Q}$
2 $Hor(r_i) \leftarrow$ horizontal line passing through r_i;
3 $\mathbf{C}(r_i) \leftarrow$ configuration $\mathbf{C}$ for robot r_i (including r_i);
4 $\mathbf{C}_{Hor}(r_i) \leftarrow$ configuration $\mathbf{C}(r_i)$ of robots South of $Hor(r_i)$;
5 **if** $|\mathbf{C}_{Hor}(r_i)| = \emptyset$ **then**
6 **if** $r_i.light =$ `Off` $\wedge$ *there is no other robot on* $Hor(r_i)$ **then** $r_i.light =$ `Final`;
7 **if** $r_i.light =$ `Off` $\wedge$ *there are robots on* $Hor(r_i) \wedge r_i$ *is the endpoint robot on* $Hor(r_i)$ **then** $r_i.light =$ `Intermediate`;
8 **if** $r_i.light =$ `Intermediate` **then**
9 Set $r_i.light =$ `Transit` and move vertically South distance 1;
10 **if** $r_i.light =$ `Transit` $\wedge$ r_i *sees no* `Intermediate` *colored robot* **then** $r_i.light =$ `Final`;
11 **if** $|\mathbf{C}_{Hor}(r_i)| \neq \emptyset$ **then**
12 **if** $r_i.light =$ `Off` $\wedge$ *there is no other robot on* $Hor(r_i) \wedge r_i$ *sees no robot colored* `Off`, `Intermediate`, *or* `Transit` *South of* $Hor(r_i)$ **then**
13 $V_i \leftarrow Visible\_Area(r_i, \mathbf{C}_{Hor}(r_i) \cup \{r_i\})$;
14 $Hor(r_j) \leftarrow$ horizontal line passing through robot r_j South of $Hor(r_i)$ closest to $Hor(r_i)$;
15 $V_i \leftarrow V_i$ after removing the area South of $Hor(r_j)$;
16 Set $r_i.light =$ `Transit` and move to a point in V_i;
17 **if** $r_i.light =$ `Off` $\wedge$ *there are robots on* $Hor(r_i) \wedge r_i$ *is the endpoint robot on* $Hor(r_i) \wedge$ *there is no robot colored* `Off`, `Intermediate`, *or* `Transit` *South of* $Hor(r_i)$ **then** $r_i.light =$ `Intermediate`;
18 **if** $r_i.light =$ `Intermediate` **then**
19 $V_i \leftarrow Visible\_Area(r_i, \mathbf{C}_{Hor}(r_i) \cup \{r_i\})$;
20 **if** *there is another robot* r_k *on* $Hor(r_i)$ **then**
21 $L' \leftarrow$ line connecting r_k with a robot r South of $Hor(r_i)$ such that there is no robot in the cone area formed by lines $Hor(r_i)$ and $\overleftrightarrow{r_k r}$;
22 $Hor(r_j) \leftarrow$ horizontal line passing through robot r_j South of $Hor(r_i)$ closest to $Hor(r_i)$;
23 $V_i \leftarrow V_i$ after removing the area beyond line L' and South of $Hor(r_j)$;
24 **else** $V_i \leftarrow V_i$ after removing the area South of $Hor(r_j)$;
25 Set $r_i.light =$ `Transit` and move to a point in V_i;
26 **if** $r_i.light =$ `Transit` $\wedge$ r_i *sees no* `Intermediate` *colored robot* **then** $r_i.light =$ `Final`;

High Level Overview of the Algorithm. The goal is to make robots progress toward a configuration where no three non-faulty robots are collinear and no faulty robot is in a line connecting two non-faulty robots. When all (non-faulty) robots in $\mathcal{Q}$ satisfy this property, this solves Complete Visibility. All previous algorithms for Complete Visibility [12,13,16,18–20] arrange robots on the corners of a convex hull. Although convex hull is not the required condition for Complete Visibility (i.e., it is a sufficient condition), the correctness analysis becomes easier. However, when faulty robots are in hull's interior, it is difficult to arrange robots on the corners of a hull.

Our idea is to develop a technique which does not require robots to be positioned on the corners of a convex hull, and hence scales on the number of faults it can tolerate. Let $\mathbf{C}_0$ be any initial configuration of the robots in $\mathcal{Q}$ with robots being in the distinct positions on the plane. Let L be a vertical line (robots agree on y-axis). The robots in $\mathcal{Q}$ can be projected to L so that all the robots are between positions b_L and t_L on L, where b_L is bottommost position on L that the robots in $\mathcal{Q}$ are projected to and t_L is the topmost position on L that the robots on L are projected to. There can be at most N different points on L that the robots in $\mathcal{Q}$ can be projected to. The idea in our algorithm is to ask the robots whose positions were projected on b_L to move first. Those robots then terminate. Until this time, the robots that are not projected to b_L do nothing. After that, the robots that are projected to a point b_L^1 (the neighboring point of b_L on L) move and terminate; the robots that are not projected to b_L^1 do nothing. This process then repeats until the robots that are projected to t_L move and terminate. The algorithm then finishes. We show that this process guarantees that Complete Visibility is achieved for the non-faulty robots in $\mathcal{Q}$ even when (up to) $f \leq N$ robots experience faults.

At any time which robots of $\mathcal{Q}$ move and which robots of $\mathcal{Q}$ do not move is determined through the colors displayed on the lights. We need to be careful how the robots move when two or more robots are projected to the same position on L. We handle this issue by asking robots that are at the two extremal points on the horizontal line they are positioned on to move first and then their neighbors can move subsequently.

In $\mathbf{C}_0$, all robots in $\mathcal{Q}$ have color `Off` and are stationary. But, in the Complete Visibility configuration, all robots in $\mathcal{Q}$ have color `Final`. The algorithm uses four colors `Final`, `Transit`, `Intermediate`, and `Off`. The colors `Intermediate` and `Transit` are to synchronize the simultaneous moves of the (at most) two robots at any moment of time in the $\mathcal{ASYNC}$ setting to make sure that Complete Visibility is achieved satisfying Theorem 1. These two colors are not required in the $\mathcal{SSYNC}$ (and $\mathcal{FSYNC}$) setting (details in Sect. 4). Moreover, robots do not know N and their termination decision is solely based on the color they assume. The robots work autonomously only having the information about the robots they see.

Details of the Algorithm. The pseudocode of the algorithm is given in Algorithm 1. Initially at $\mathbf{C}_0$, the lights of all robots are set to color `Off` and the robots are stationary. Let r_i be a robot in $\mathcal{Q}$. Let $Hor(r_i)$ be a horizontal line passing

through the position of r_i. We first discuss how r_i moves if it sees no robot South of $Hor(r_i)$, i.e., r_i is the Southmost robot in the configuration. Robot r_i can determine whether it is a Southmost robot or not as it knows y-axis. Robot r_i simply changes its color to `Final` without moving if it sees no other robot on $Hor(r_i)$. If r_i sees some other robot on $Hor(r_i)$, it changes its color to `Intermediate` (without moving) if it is positioned on $Hor(r_i)$ such that it sees robots on only one side on $Hor(r_i)$. We call r_i the *endpoint* robot on $Hor(r_i)$ if the above condition is satisfied. Otherwise, r_i is on $Hor(r_i)$ with at least a robot on $Hor(r_i)$ on its both sides and r_i does nothing until it either becomes the endpoint robot on $Hor(r_i)$ (fault-free case) or the robot on at least one side of $Hor(r_i)$ is colored `Final` (faulty case). After r_i is colored `Intermediate`, it assumes color `Transit` and moves distance 1 vertically South. If r_i is colored `Transit`, it assumes color `Final` if it sees no `Intermediate` colored robot. If r_i sees an `Intermediate` colored robot, they both were on $Hor(r_i)$ before r_i moved, and the waiting makes sure that the `Intermediate` colored robot also moves before r_i terminates. This makes synchronization easier in the $\mathcal{ASYNC}$ setting. We will discuss in Sect. 4 this is not needed in the $\mathcal{SSYNC}$ (and $\mathcal{FSYNC}$) setting.

We now discuss how r_i moves if it sees at least a robot South of $Hor(r_i)$. Robot r_i does not move until it sees at least a `Off`, `Intermediate`, or `Transit` colored robot South of $Hor(r_i)$. The idea here is for r_i to compute $Visible\_Area(r_i, \mathbf{C}_{Hor}(r_i) \cup \{r_i\})$ considering the robots South of $Hor(r_i)$ that it sees and move to a point in $Visible\_Area(r_i, \mathbf{C}_{Hor}(r_i) \cup \{r_i\})$. If r_i is the only robot on $Hor(r_i)$, it assumes color `Transit` and moves to a point in $Visible\_Area(r_i, \mathbf{C}_{Hor}(r_i) \cup \{r_i\})$. If r_i is not the only robot on $Hor(r_i)$ but an endpoint robot on $Hor(r_i)$, then it first assumes color `Intermediate` from `Off`. After r_i colored `Intermediate`, it moves as follows: r_i computes $Visible\_Area(r_i, \mathbf{C}_{Hor}(r_i) \cup \{r_i\})$ and moves to a point in $Visible\_Area(r_i, \mathbf{C}_{Hor}(r_i) \cup \{r_i\})$ assuming color `Transit`. After colored `Transit`, it sets its light to `Final` if it does not see any `Intermediate` colored robot. If it sees an `Intermediate` colored robot r_j, r_j must be North of $Hor(r_i)$ and it waits until r_j assumes color `Transit`. After colored `Final`, r_i terminates its computation when it becomes active next time.

We restrict how a point in $Visible\_Area(r_i, \mathbf{C}_{Hor}(r_i) \cup \{r_i\})$ is selected to avoid robot collisions. Suppose $Hor(r_j)$ is the horizontal line passing through a robot r_j South of $Hor(r_i)$ such that there is no robot between lines $Hor(r_i)$ and $Hor(r_j)$. We restrict that r_i can not move to positions of $Hor(r_j)$ or South of it. This will avoid collisions between robots of $Hor(r_i)$ and $Hor(r_j)$. To avoid collisions between two robots of $Hor(r_i)$ (that can move simultaneously) and also to make sure that the moves of those robots do not block the visibility of each other to see the robots South of $Hor(r_i)$, we restrict r_i not to move on or beyond $\overline{r_k r}$ (in addition to not moving beyond $Hor(r_j)$), where r_k is the neighboring robot of r_i on $Hor(r_i)$ and r is the robot South of $Hor(r_i)$ such that there is no robot in the *cone area* formed by lines $Hor(r_i)$ and $\overline{r_k r}$. Figure 2 illustrates these ideas.

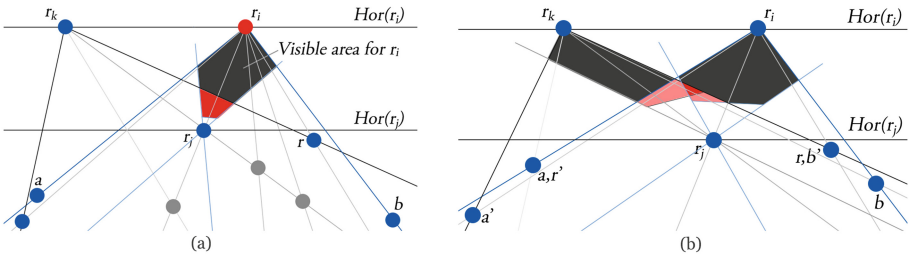

Fig. 2. (a) $Visible\_area(r_i)$ for r_i (black region) is computed by removing the part of it beyond $\overline{r_k r}$ (red region) and (b) disjoint $Visible\_Area(r_i)$ and $Visible\_Area(r_k)$ for two robots r_i, r_k on $Hor(r_i)$ (black regions) using the technique of (a). (Color figure online)

Analysis of the Algorithm. We now analyze the correctness of the algorithm. Particularly, we show that the algorithm solves Complete Visibility starting from any initial configuration $\mathbf{C}_0$ with all robots in $\mathcal{Q}$ being in the distinct positions in the plane and (up to) N robots become faulty. We further show that the algorithm terminates in $\mathcal{O}(N)$ time and the execution is collision- and deadlock-free. We start with the following lemma.

Lemma 2. *Let $Hor(r_i)$ and $Hor(r_j)$ be horizontal lines passing through robots r_i, r_j such that there is no robot in the area between lines $Hor(r_i)$ and $Hor(r_j)$ and $Hor(r_j)$ is in South of $Hor(r_i)$. No robot on $Hor(r_i)$ is colored* `Intermediate`*,* `Transit`*, or* `Final` *until all the robots on $Hor(r_j)$ are colored* `Final`*.*

Lemma 3. *When a robot r_i on $Hor(r_i)$ computes $Visible\_Area(r_i, \mathbf{C}_{Hor}(r_i) \cup \{r_i\})$, it is a corner of the convex hull $\mathbf{P}$ of the robots of $\mathbf{C}_{Hor}(r_i) \cup \{r_i\}$.*

Proof. We have that $\mathbf{C}_{Hor}(r_i)$ consists of the robots South of $Hor(r_i)$ that r_i sees. Therefore, when r_i computes a convex hull $\mathbf{P}(r_i)$ of the robots in $\mathbf{C}_{Hor}(r_i) \cup \{r_i\}$, it makes an angle of $<180°$ with its two neighboring corners of $\mathbf{P}(r_i)$ since all the robots on $\mathbf{C}_{Hor}(r_i)$ are in one side of $Hor(r_i)$. □

Lemma 4. *When a (non-faulty) robot r_i moves once, it sees all robots (both faulty and non-faulty) South of $Hor(r_i)$.*

Proof. We have from Lemma 1 that when a corner r_i of a convex hull $\mathbf{P}$ moves to a point in $Visible\_Area(r_i, \mathbf{C}_{Hor}(r_i) \cup \{r_i\})$ and no other robot is moving simultaneously with r_i, r_i sees all other robots of $\mathbf{P}$ (corners, side, and interior). We have from Lemma 3 that r_i is a corner of a convex hull $\mathbf{P}$ formed by the robots in $\mathbf{C}_{Hor}(r_i) \cup \{r_i\}$. When r_i moves in Algorithm 1, no other robot of $\mathbf{P}$ formed from $\mathbf{C}_{Hor}(r_i) \cup \{r_i\}$ is moving, therefore r_i sees all the robots that are South of $Hor(r_i)$. It only remains to show that, at most one other robot r on $Hor(r_i)$ that is moving simultaneously with r_i is also visible to r_i and vice-versa. This is immediate from the visible areas $Visible\_Area(r_i, \mathbf{C}_{Hor}(r_i) \cup \{r_i\})$ and $Visible\_Area(r, \mathbf{C}_{Hor}(r) \cup \{r\})$ computed by r_i and r, respectively. Let a, b be

the left and right neighbors of r_i in $\mathbf{P}(r_i)$ among the robots in $\mathbf{C}_{Hor}(r_i) \cup \{r_i\}$. Moreover, let a', b' be the left and right neighbors of r in $\mathbf{P}(r)$ among the robots in $\mathbf{C}_{Hor}(r) \cup \{r\}$. We have that $Visible\_Area(r_i)$ does not contain the area beyond line $\overline{rb'}$ and $Visible\_Area(r)$ does not contain the area beyond line $\overline{r_i a}$ (Fig. 2). Therefore, even if r_i, r move simultaneously, r_i does not become collinear with r in line $\overline{rx}$ connecting r with any robot $x \in \mathbf{C}_{Hor}(r)$ and r does not become collinear with r_i in line $\overline{r_i x}$ connecting r_i with any robot $x \in \mathbf{C}_{Hor}(r_i)$. Moreover, even after r_i and r move simultaneously, r_i sees r and vice-versa follows immediately since they move in the area between $Hor(r_i)$ and $Hor(r_j)$ with r_j the same robot in the view of both r_i, r and there is no third robot in that area. □

Lemma 5. *Each (non-faulty) robot does at most one move during entire execution.*

Proof. Pick any robot r_i. If it is a Southmost robot and there is no robot on $Hor(r_i)$, it terminates without moving. If it picks color `Intermediate`, then it does so without moving. If it picks color `Transit`, then it moves. If r_i is already colored `Transit`, it changes its color to `Final` without moving. If r_i is colored `Final`, it terminates. Therefore, a robot r_i moves only once when it picks color `Transit` either from `Off` or from `Intermediate`. Therefore, each non-faulty robot moves at most once. □

Lemma 6. *Algorithm 1 is collision-free.*

Proof. Let $Hor(r_i)$ and $Hor(r_j)$ be two horizontal lines such that there is no robot in the area between $Hor(r_i)$ and $Hor(r_j)$. Let $Hor(r_j)$ be South of $Hor(r_i)$. The robots on lines $Hor(r_i)$ and $Hor(r_j)$ do not collide since the robots on $Hor(r_i)$ never reach to positions of $Hor(r_j)$ and the robots on $Hor(r_j)$ never move North of $Hor(r_j)$. Therefore, it only remains to show that the robots on $Hor(r_i)$ do not collide with each other. We have that at most 2 endpoint robots r_i, r_k on $Hor(r_i)$ move simultaneously. The robots move in such a way that r_i does not cross line $\overline{r_k r}$ and r_k does not cross line $\overline{r_i a}$ (as defined in Fig. 2b) and hence this avoids collisions between them. □

Lemma 7. *Algorithm 1 is deadlock-free.*

Lemma 8. *Algorithm 1 runs for $\mathcal{O}(N)$ epochs.*

Lemma 9. *The non-rigid movements of robots do not impact the guarantees of the algorithm.*

Proof. Let d_i be the point in $Visible\_Area(r_i)$ that r_i moves under rigid movements. Let $\overline{r_i d_i}$ be the line segment connecting r_i with d_i before r_i moves to d_i. Under non-rigid movements, r_i may stop anywhere between r_i and d_i (we know that it does not stop at r_i since it moves at least $\Delta > 0$). We have from $Visible\_Area(r_i)$ that r_i is visible to all other non-moving robots if it moves to any point in $Visible\_Area(r_i)$. According to the visible area construction,

all points in line $\overline{r_i d_i}$ contain inside the visible area $Visible\_Area(r_i)$. Therefore, even under non-rigid movements, the algorithm provides all guarantees we obtained under rigid movements. □

Proof of Theorem 1. We have Theorem 1 combining the results of Lemmas 4–9.

4 Discussion and Concluding Remarks

Improved Color Algorithm for the $\mathcal{SSYNC}$ (and $\mathcal{FSYNC}$) Model. In the $\mathcal{SSYNC}$ setting (and the $\mathcal{FSYNC}$ setting), we need only two colors in Algorithm 1, which is optimal with respect to the number of colors when N is not known [12]. In particular, we do not need colors `Intermediate` and `Transit`. The colors `Intermediate` and `Transit` in Algorithm 1 are to synchronize the moves of the robots when there are two or more robots on a horizontal line $Hor(r_i)$ in the $\mathcal{ASYNC}$ setting. However, in the $\mathcal{SSYNC}$ (and also in the $\mathcal{FSYNC}$) setting, this can be achieved without these colors since: (i) if only one robot r_i of $Hor(r_i)$ moves at round k, at round $k+1$, the other robot r_j already sees r_i South of $Hor(r_i)$ and it can move in such a way that all the robots South of $Hor(r_i)$ see it; (ii) if both robots r_i, r_j on $Hor(r_i)$ move in round k, then at round $k+1$ they will be on their final positions, all the robots South of $Hor(r_i)$ see both of them, and r_i, r_j see each other. All these results can be proved extending the analysis of Sect. 3 and runtime is still $\mathcal{O}(N)$.

Impact of Correctness of Lights after Faults. The tolerance to faults in our algorithm depends on the correctness of the colors of the lights even after robots experience (mobility) faults. I.e., even after robot becomes faulty, lights can be correctly set from `Off` to `Final`, possibly going through the changes to `Intermediate` and `Transit` (without moving). If the robot color stays as the color it has at the time of fault, then we cannot guarantee termination and also whether Complete Visibility is solved. This is because, it is difficult to determine for a robot r_i whether the (non-faulty) robots that are South of $Hor(r_i)$ already moved once or not. Therefore, it is an open problem to solve Complete Visibility in this setting tolerating multiple faults. The algorithm in [3] handles a single fault even when the light stays at the color at the time of fault.

Concluding Remarks. We have presented, to our best knowledge, the first fault-tolerant algorithm for the Complete Visibility problem using 4 colors for robots with lights in the $\mathcal{ASYNC}$ setting under non-rigid movements and one-axis agreement, tolerating (up to) N faulty robots, not known a priori. The algorithm terminates in $\mathcal{O}(N)$ time avoiding collisions. The previous work [3] was only able to handle one faulty robot in the $\mathcal{SSYNC}$ setting under rigid movements and both-axis agreement using 3 colors. We then showed that the number of colors can be improved from 4 to 2 in the $\mathcal{SSYNC}$ setting (and also in the $\mathcal{FSYNC}$) setting.

Many questions remain for future work. It will be interesting to minimize the number of colors from 4 to 2 in our algorithm in the $\mathcal{ASYNC}$ setting. Most importantly, it will be interesting to remove the one-axis agreement assumption and solve this problem tolerating multiple faults when lights can be faulty in addition to mobility faults.

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. Agmon, N., Peleg, D.: Fault-tolerant gathering algorithms for autonomous mobile robots. SIAM J. Comput. **36**(1), 56–82 (2006)
3. Aljohani, A., Sharma, G.: Complete visibility for mobile agents with lights tolerating a faulty agent. In: APDCM, pp. 834–843 (2017)
4. Bhagat, S., Chaudhuri, S.G., Mukhopadhyaya, K.: Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. J. Discrete Algorithm. **36**, 50–62 (2016)
5. Bhagat, S., Chaudhuri, S.G., Mukhopadhyaya, K.: Formation of general position by asynchronous mobile robots under one-axis agreement. In: Kaykobad, M., Petreschi, R. (eds.) WALCOM 2016. LNCS, vol. 9627, pp. 80–91. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30139-6_7
6. Cohen, R., Peleg, D.: Local spreading algorithms for autonomous robot systems. Theor. Comput. Sci. **399**(1–2), 71–82 (2008)
7. Cord-Landwehr, A., Degener, B., Fischer, M., Hüllmann, M., Kempkes, B., Klaas, A., Kling, P., Kurras, S., Märtens, M., Meyer auf der Heide, F., Raupach, C., Swierkot, K., Warner, D., Weddemann, C., Wonisch, D.: 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). https://doi.org/10.1007/978-3-642-22012-8_52
8. Czyzowicz, J., Gasieniec, L., Pelc, A.: Gathering few fat mobile robots in the plane. Theor. Comput. Sci. **410**(6–7), 481–499 (2009)
9. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: Autonomous mobile robots with lights. Theor. Comput. Sci. **609**, 171–184 (2016)
10. D'Emidio, M., Frigioni, D., Navarra, A.: Characterizing the computational power of anonymous mobile robots. In: ICDCS, pp. 293–302 (2016)
11. Flocchini, P., Prencipe, G., Santoro, N.: Distributed computing by oblivious mobile robots. Synth. Lectur. Distrib. Comput. Theor. **3**(2), 1–185 (2012)
12. Luna, G.A.D., Flocchini, P., Chaudhuri, S.G., Poloni, F., Santoro, N., Viglietta, G.: Mutual visibility by luminous robots without collisions. Inf. Comput. **254**, 392–418 (2017)
13. Di 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, Cham (2014). https://doi.org/10.1007/978-3-319-11764-5_11
14. Pagli, L., Prencipe, G., Viglietta, G.: Getting close without touching: Near-gathering for autonomous mobile robots. Distrib. Comput. **28**(5), 333–349 (2015)
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). https://doi.org/10.1007/11603771_1

16. Sharma, G., Busch, C., Mukhopadhyay, S.: Mutual visibility with an optimal number of colors. In: Bose, P., Gąsieniec, L.A., Römer, K., Wattenhofer, R. (eds.) ALGOSENSORS 2015. LNCS, vol. 9536, pp. 196–210. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28472-9_15
17. Sharma, G., Vaidyanathan, R., Trahan, J.L.: Constant-time complete visibility for asynchronous robots with lights. In: Spirakis, P., Tsigas, P. (eds.) SSS 2017. LNCS, vol. 10616, pp. 265–281. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69084-1_18
18. Sharma, G., Vaidyanathan, R., Trahan, J.L., Busch, C., Rai, S.: Complete visibility for robots with lights in O(1) time. In: Bonakdarpour, B., Petit, F. (eds.) SSS 2016. LNCS, vol. 10083, pp. 327–345. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49259-9_26
19. Sharma, G., Vaidyanathan, R., Trahan, J.L., Busch, C., Rai, S.: Logarithmic-time complete visibility for asynchronous robots with lights. In: IPDPS, pp. 513–522 (2017)
20. 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)