International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 8, Number 1, pages 32–52, January 2018

Complete Visibility for Mobile Robots with Lights Tolerating Faults

Aisha Aljohani
Kent State University
Kent, OH, USA
Email: aaljoha6@kent.edu

Gokarna Sharma
Kent State University
Kent, OH, USA
Email: sharma@cs.kent.edu

Received: August 1, 2017
Revised: October 31, 2017
Accepted: November 29, 2017
Communicated by Susumu Matsumae

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 (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; all existing algorithms for this problem are not fault-tolerant (except handling some special cases). 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. In this paper, we present and analyze an algorithm that solves Complete Visibility tolerating one crash-faulty robot in a system of $N \geq 3$ robots, starting from any arbitrary initial configuration. We also provide an impossibility result on solving Complete Visibility if a single robot is Byzantine-faulty in a system of $N = 3$ robots; in the Byzantine fault model, a faulty robot might behave in an unpredictable, arbitrary, and unforeseeable ways. Furthermore, we discuss how to solve Complete Visibility for some initial configurations of robots (which we call feasible initial configurations) in the crash fault model, where two robots are (crash) faulty.

1 Introduction

In the classical model of distributed computing by mobile robots, each robot is modeled as a point in the plane [10]. 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 an 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 LCM cycle, each robot has no memory of its past LCM cycles [10]. 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 make coordination efficient and relatively viable. One model that incorporates direct communication is the *robots with lights* model [8, 10, 15], where each robot has an externally visible light that can assume colors from a constant sized set, and hence robot 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 the lights, the robots are oblivious as in the classical model.

Di Luna *et al.* [12] 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 solving COMPLETE VISIBILITY is that it makes it possible to solve many other robotic problems, including gathering, shape formation, and leader election, under obstructed visibility in the robots with lights model. That is, in the lights model, instead of robots terminating their execution, they assume a special color, say "Done", and then a node with color "Done" can start executing the algorithm for some other robotic problem after all robots that are visible to it also have color "Done". It can be shown that a robot colored "Done" finds all robots it sees also have color "Done" only after COMPLETE VISIBILITY is solved. We refer to Di Luna *et al.* [11] for an example where they solve circle formation problem after complete visibility is achieved by robots in the lights model. Most importantly, COMPLETE VISIBILITY recovers unobstructed visibility configuration starting from obstructed visibility configuration. Subsequently, several papers, e.g. [11, 17], focused on solving this problem minimizing the number of colors. Recently, faster runtime algorithms for COMPLETE VISIBILITY [19–22] were studied in the lights model (details in Section 2).

In this paper, we are interested in the fault-tolerant algorithms for COMPLETE VISIBILITY in the robots with lights model. All existing algorithms, except the work of Di Luna *et al.* [11], do not consider faults and hence may fail to solve this problem when robots are faulty. However, Di Luna *et al.* [11] only handles the special case of a faulty robot being in the perimeter of the hull, i.e., Di Luna *et al.* [11] cannot handle if the faulty robot is in the interior of the hull.

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, and *Byzantine faults* where each faulty robot behaves in unpredictable, arbitrary, and unforeseeable ways [2].

Contributions We consider the same robot model as in Di Luna *et al.* [11, 12], 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. Moreover, we assume that the setting is *semi-synchronous* where there is a notion of common time, at least one robot is active in each LCM cycle, and the robots that are active perform their cycles simultaneously. We also assume that a robot in motion cannot be stopped (by an adversary) before it reaches its destination point, that is, the moves of the robots are *rigid*. As in Di Luna *et al.* [12], two robots cannot head to the same destination and their paths when they move cannot cross. The path crossing would constitute a *collision*. Furthermore, we assume that the robots agree on common x- and y-axes (directions and orientations) [10]. In this paper, we prove the following result which, to our knowledge, is the first algorithm for COMPLETE VISIBILITY that tolerates a single faulty robot in the semi-synchronous setting.

Theorem 1.1 *For any initial configuration of $N \geq 3$ robots (with lights) being in the distinct positions in a plane, there is an algorithm that solves the* Complete Visibility *problem tolerating a crash-faulty robot using 3 colors and without collisions in the semi-synchronous setting.*

When the robots are non-faulty, the idea used in the existing algorithms [11, 12, 17, 20–22] is to reposition the robots so that they all become corners of a N-corner convex hull. When all N robots are positioned in the corners of a convex hull, 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. Therefore, this naturally solves Complete Visibility.

Consider the situation where one robot is crash faulty. If the faulty robot is in the corner (or side) of the hull, then it does not block the visibility to other non-faulty robots and some of the previous algorithms, especially of Di Luna *et al.* [11, 12], naturally handle this case. However, the faulty robot may be in the interior of the convex hull. Therefore, the question is how to figure out this situation and guarantee that all non-faulty robots see each other. Since robots are oblivious and non-faulty robots do not know which robots are faulty, this task becomes quite challenging. In this paper, we develop a technique that guarantees that Complete Visibility is achieved even when the (crash) faulty robot is in the interior of the hull.

We also show that it is impossible to solve Complete Visibility even if a single robot is Byzantine faulty. For this impossibility proof, we consider a system of $N = 3$ robots, out of which one is Byzantine faulty. This result shows that we cannot hope for a Complete Visibility solution tolerating even a single fault in the Byzantine fault model.

Given the above impossibility result, it is quite natural to look at whether two or more faulty robots can be tolerated in the crash-fault model. We found that this is quite challenging for arbitrary initial configurations. Therefore, we consider a subset of arbitrary initial configurations (which we call feasible initial configurations) and outline an algorithm for Complete Visibility that tolerates two (crash) faulty robots in a system of $N \geq 3$ robots using 3 colors in the semi-synchronous setting.

Remarks We do not know whether the semi-synchronous model is necessary to solve the problem. But, we could not manage to proof the correctness of our algorithm (that robots achieve Complete Visibility configuration and terminate their computation) when the model is fully asynchronous. Nevertheless, we conjecture that it may be possible by increasing the number of colors.

When robots are fault-free, it is known from [11] that any 2-color algorithm for Complete Visibility is optimal w.r.t. the number of colors in the robots with lights model, when N is not known to robots. When N is known to robots and no faults, Complete Visibility can be solved in the semi-synchronous model without colors [13]. However, with just 2 colors in our algorithm, since N is not known, robots have difficulty deciding whether Complete Visibility configuration is achieved or not. Therefore, the third color allows them to break that ambiguity. Therefore, it will be interesting to have a 2-color algorithm for Complete Visibility when N is not known in the fault model or prove that any 3-color solution is optimal. Note that in the fault-free model, a 2-color algorithm is known for Complete Visibility [17] even when N is not known to robots.

Paper Organization The rest of the paper is organized as follows. We discuss the detailed related work in Section 2. We present the robot model and some preliminaries in Section 3. We then present our Complete Visibility algorithm tolerating a single crash-faulty robot in Section 4 and analyze it in Section 5. We then present an impossibility result on solving Complete Visibility in the Byzantine fault model in Section 6. After that, we present a Complete Visibility algorithm that tolerates 2 crash faulty robots for certain initial configurations in Section 7. Finally, we conclude in Section 8 with a short discussion.

2 Detailed Related Work

Di Luna *et al.* [12] 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). After that, a series of papers

[11, 17] provided solutions to COMPLETE VISIBILITY minimizing the number of colors. Di Luna *et al.* [11] solved the problem using 2 colors in the semi-synchronous setting under rigid movements and using 3 colors in the semi-synchronous setting under non-rigid movements and in the asynchronous setting under rigid movements. The also provided a solution using 3 colors in the asynchronous setting under non-rigid movements under one-axis agreement. Sharma *et al.* [17] improved the number of colors in the solution of Di Luna *et al.* [11] from 3 to 2 (in the semi-synchronous setting under non-rigid movements and in the asynchronous setting under both rigid and non-rigid movements). All these results proved the correctness of their algorithms but provided no runtime analysis (except the finite time termination of their algorithms).

Vaidyanathan *et al.* [22] considered runtime for the very first time for COMPLETE VISIBILITY in the robots with lights model. They provided 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.* [20] provided an $\mathcal{O}(1)$ time algorithm using $\mathcal{O}(1)$ colors in the semi-synchronous setting under rigid movements. Recently, Sharma *et al.* [21] provided an $\mathcal{O}(\log N)$ time algorithm using $\mathcal{O}(1)$ colors in the asynchronous setting under rigid movements, which they improved to $\mathcal{O}(1)$ in [19]. In the classic oblivious robots model (with no lights), Di Luna *et al.* [13] solved COMPLETE VISIBILITY assuming N is known to robots in the semi-synchronous setting. However, they did not provide the runtime analysis except the proof that their algorithm terminates in finite time. Recently, Sharma *et al.* [18] showed that the algorithm of Di Luna *et al.* [13] has a runtime lower bound of $\Omega(N^2)$ rounds in the fully synchronous setting and provided an algorithm that runs in $\mathcal{O}(N)$ rounds in the fully synchronous setting. However, all these previous algorithms [11–13, 16, 18, 20–22] are not fault-tolerant.

The obstructed visibility, in general, is considered in the problem of uniformly spreading robots operating in a line, studied by Cohen and Peleg [5]. The work of Pagli *et al.* [14] considers the near-gathering problem where collisions must be avoided among robots. The obstructed visibility is also considered in the so-called *fat robots* model [1, 4, 6, 7, 9] in which robots are not points, but non-transparent unit discs, and hence they can obstruct visibility of collinear robots. However, all these work do not consider faulty robots. The faults (both crash and Byzantine) are considered for the gathering problem in Agmon and Peleg [2] in the classical oblivious robots model. We borrow the definitions of crash and Byzantine faults from Agmon and Peleg [2].

3 Model and Preliminaries

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 directions and orientations with other robots in $\mathcal{Q}$, i.e., they agree on both x- and y-axes. Due to this agreement assumption on x- and y-axes, we can denote the position p_i of the robot r_i by its x and y coordinates, i.e., $p_i = (r_i.x, r_i.y)$. We can then denote counterclockwise and clockwise directions as usual which is the same for all the robots in $\mathcal{Q}$.

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 $r_i.light = \texttt{Red}$, 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 assumption is that r_i can read the color that is assigned to variable $r_i.light$). 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.

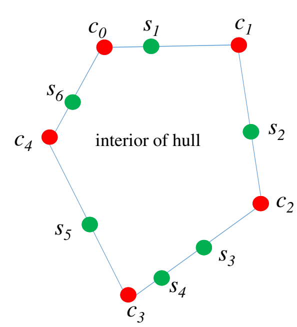
- *Look:* For each robot r_j that is visible to it, r_i can observe the position of r_j on the plane and the color of the light of r_j. Robot r_i can also observe its own color and position; that is, r_i is visible to itself.

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, an 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 LCM cycle to the next LCM cycle.

- *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 and Synchronization 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. The activations are decided by an (indeterministic) adversary which applies also to the asynchronous model. 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. We assume that the moves of the robots are *rigid* – during the *Move* phase the robots move in a straight line and they stop their movement only after they reach to the destination point computed in the *Compute* phase. We assume that the faulty robot can crash (behave in a Byzantine manner) at any moment of time. That means, the robot may crash (become Byzantine) at any time during the LCM cycle. After the robot crashes, it does not become active again (i.e., stays stationary indefinitely). Moreover, after the robot crashes, its color remains as the color that it had at the time of crashing. However, after the robot becomes Byzantine, it might behave in arbitrary and unforeseeable way, which includes assuming any color it wants from the color set and move (or not move) wherever it wants.

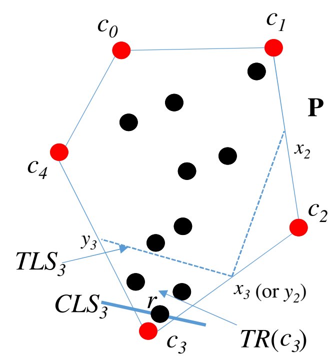


Convex Hull For any set of $N \geq 3$ robots in $\mathcal{Q}$, a *convex hull* (or polygon) may be visualized as the shape enclosed by a rubber band stretched around Q so that all the robots of Q are either in the perimeter of the shape or in the interior of it. It can be represented as a sequence $\mathbf{P} = (c_0, c_1, \cdots, c_{m-1}, c_0)$ of *corner points* (or robots) in a plane that enumerates the polygon corners in clockwise order starting and ending at the same corner, where m is the number of corners in $\mathbf{P}$. Figure on the right shows a 5-corner convex hull $(c_0, c_1, c_2, c_3, c_4, c_0)$. A point s on the plane is a *side point* of $\mathbf{P}$ if and only if there exists $0 \leq i < m$ such that $c_i, s, c_{(i+1)(\text{mod } m)}$ are collinear. Figure on the right shows six side points s_1–s_6. 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 say that the area enclosed by $\mathbf{P}$ the interior of the hull (except the boundary points), the rest is exterior. For any pair of points a, b, we denote the line segment connecting them by $\overline{ab}$ and the length of $\overline{ab}$ by $\mathsf{length}(\overline{ab})$.

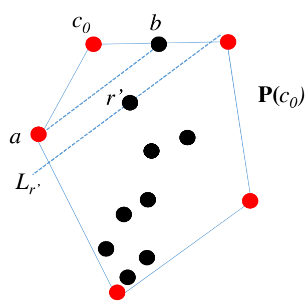
Configuration and Local Convex Hull A *configuration* $\mathbf{C}_t = \{(r_0^t, col_0^t), \ldots, (r_{N-1}^t, col_{N-1}^t)\}$ defines the positions of the robots in $\mathcal{Q}$ and their colors for any time $t \geq 0$. A configuration for an robot $r_i \in \mathcal{Q}$, $\mathbf{C}_t(r_i)$, defines 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. The convex hull formed by $\mathbf{C}_t(r_i)$, $\mathbf{P}_t(r_i)$, is *local* to r_i since $\mathbf{P}_t(r_i)$ depends only on the points that are visible to r_i at time t. For simplicity, we sometime write $\mathbf{C}, \mathbf{P}, \mathbf{C}(r_i), \mathbf{P}(r_i)$ to denote $\mathbf{C}_t, \mathbf{P}_t, \mathbf{C}_t(r_i), \mathbf{P}_t(r_i)$, respectively.

Algorithm 1: Complete Visibility algorithm for any robot $r_i \in \mathcal{Q}$

// Look-Compute-Move cycle for each robot $r_i \in \mathcal{Q}$
$\mathbf{C}(r_i) \leftarrow$ configuration $\mathbf{C}$ for robot r_i (including r_i);
$\mathbf{P}(r_i) \leftarrow$ convex hull of the robots in $\mathbf{C}(r_i)$;
if $|\mathbf{C}(r_i)| = 2$ **then**
  $r_j \leftarrow$ the robot in $\mathbf{C}(r_i) \backslash \{r_i\}$;
  if $r_i.x < r_j.x$ **then**
    move perpendicular to (the line segment) $\mathbf{P}(r_i)$ in clockwise direction by distance $\delta > 0$;
  else if $r_i.x > r_j.x$ **then**
    move perpendicular to (the line segment) $\mathbf{P}(r_i)$ in counterclockwise direction by distance $\delta > 0$;
  else if $r_i.y < r_j.y$ **then**
    move perpendicular to (the line segment) $\mathbf{P}(r_i)$ in clockwise direction by distance $\delta > 0$;
  else if $r_i.y > r_j.y$ **then**
    move perpendicular to (the line segment) $\mathbf{P}(r_i)$ in counterclockwise direction by distance $\delta > 0$;
else
  if r_i *is a corner of* $\mathbf{P}$ **then**
    $Corner(r_i, \mathbf{C}(r_i), \mathbf{P}(r_i))$;



Corner Triangle, Corner Line Segment, and Triangle Line Segment Let c_i be a corner of $\mathbf{P}$. Let n_{i-1} and n_{i+1} be the neighbors of c_i in $\mathbf{P}$ (either corners or sides). Indeed, the neighbors n_{i-1} and/or n_{i+1} may not necessarily be the corners c_{i-1} and/or c_{i+1} of $\mathbf{P}$, respectively, when there are side robots on $\overline{c_i c_{i-1}}$ and/or $\overline{c_i c_{i+1}}$. If there are no side robots on $\overline{c_i c_{i-1}}$ and $\overline{c_i c_{i+1}}$, then n_{i-1} is c_{i-1} and n_{i+1} is c_{i+1}. In the side robots case, we take the neighboring robots of c_i that are closest to c_i in the boundary of $\mathbf{P}$ as n_{i-1} and n_{i+1}. Let x_i, y_i be the points in lines $\overline{c_i n_{i-1}}$ and $\overline{c_i n_{i+1}}$ at distance $\mathsf{length}(\overline{c_i n_{i-1}})/2$ and $\mathsf{length}(\overline{c_i n_{i+1}})/2$, respectively, from c_i. We say that line segment $\overline{x_i y_i}$ is the *triangle line segment* for c_i, denoted as TLS_i. We say that the triangular area of $\mathbf{P}$ divided by TLS_i towards c_i is the *corner triangle* for c_i, denoted as $TR(c_i)$. Let r be any robot inside $TR(c_i)$ and CLS_i be the line segment parallel to TLS_i passing through r. We say that CLS_i is the *corner line segment* for c_i if there is no other robot inside $TR(c_i)$ divided by CLS_i towards c_i. Figure on the right shows $TR(c_3)$, TLS_3, CLS_3 for corner c_3 of $\mathbf{P}$.



Closest robot to a corner Let c_i be a corner robot of $\mathbf{P}(c_i)$ and a, b be its left and right neighbors in the boundary of $\mathbf{P}(c_i)$. Let r' be an robot of $\mathbf{P}(c_i) \backslash \{a, b, c_i\}$ and $L_{r'}$ be a line parallel to line segment $\overline{ab}$ passing through r'. Robot r' is said to be *closest robot* to c_i if there is no other robot in $\mathbf{P}(c_i)$ divided by line $L_{r'}$ towards c_i. If there are other robots on $L_{r'}$, then we take as closest robot to c_i the robot on $L_{r'}$ that is closer to b (note that b is the right neighbor of c_i in $\mathbf{P}(c_i)$). Figure on the right shows the closest robot r' to the corner c_0 in $\mathbf{P}(c_0)$.

4 Algorithm Tolerating a Single Fault

In this section, we present our Complete Visibility algorithm for $N \geq 3$ robots with lights tolerating a crash-faulty robot, starting from any arbitrary initial configuration with robots being in the distinct positions in a plane. We first provide a high level overview and then give its details.

4.1 High Level Overview of the Algorithm

The idea is to make robots progress toward converging to a configuration where all the robots (except at most one) are in the corners of a convex hull $\mathbf{P}$. When all the robots in $\mathcal{Q}$ are on the corners of $\mathbf{P}$, the property of a convex hull naturally solve the Complete Visibility problem. All previous algorithms for Complete Visibility [11–13, 16, 18, 20–22] also arrange robots on the corners of a convex hull. Although convex hull is not the required condition to solve Complete Visibility (i.e., it is a sufficient condition), the correctness analysis becomes easier for convex hull [11, 16]. It is largely an open question to deterministically solve Complete Visibility without arranging robots on the corners of a convex hull and recently an attempt has been made in [3].

We differentiate initial configurations $\mathbf{C}_0$ into two categories as follows:

(i) all robots in $\mathcal{Q}$ are collinear (in a line) and

(ii) not all robots in $\mathcal{Q}$ are collinear.

If the category (i) is satisfied for $\mathbf{C}_0$, we ask the endpoint robots in the line configuration to move small distance perpendicular to the line, which ensures that the resulting configuration will be of category (ii). When the robots of $\mathcal{Q}$ satisfy category (ii), there is a convex polygon $\mathbf{P}$ (with at least three corners) so that all the robots in $\mathcal{Q}$ are on the corners, sides, and in the interior of $\mathbf{P}$. We ask the robots in the corners of $\mathbf{P}$ to move inward to shrink the hull (the side and interior robots of $\mathbf{P}$ do nothing until they become corners of $\mathbf{P}$). Due to the shrinking, the robots that were in sides and in the interior of $\mathbf{P}$ start becoming new corners of $\mathbf{P}$. This process repeats until either there is no robot in the interior of $\mathbf{P}$ or there is exactly one robot in the interior of $\mathbf{P}$.

In the former case, we are done. In the latter case, we ask the corner robots and the only one interior robot to detect the situation and terminate their computation making an appropriate move so that the Complete Visibility problem is solved. The guarantee we provide is that when the robots terminate, the Complete Visibility problem is indeed solved for all non-faulty robots (even when the faulty robot is in the interior of $\mathbf{P}$). The synchronization between robots to reach such configuration is provided by the colors they can assume during the execution of the algorithm. In particular, in the initial configuration $\mathbf{C}_0$ (at time $t = 0$), all robots in $\mathcal{Q}$ have color `Off` and are stationary. But, in the Complete Visibility configuration, say $\mathbf{C}_{mv}$, all non-faulty robots have color `Green` and the faulty robot has color $\in \{$`Green`, `Red`, `Off`$\}$. Since there is a single faulty robot, at most one robot in $\mathbf{P}$ can have color `Red` or `Off` when all the robots in $\mathcal{Q}$ terminate. The color `Red` is assumed by robots during the computation until Complete Visibility is reached. Therefore, the algorithm uses three colors `Green`, `Red`, and `Off`. Moreover, note that the robots do not know N and their termination decision is solely based on the colors of the other robots that they see in their view $\mathbf{C}(*)$.

4.2 Details of the Algorithm

The pseudocode of the algorithm is given in Algorithms 1–3. Initially in $\mathbf{C}_0$, the lights of all robots are set to color `Off` and the robots are stationary. We first discuss how any initial collinear configuration $\mathbf{C}_0$ is transformed to a non-collinear (polygonal) configuration with at least three corners. We will then discuss how robots reach to a Complete Visibility configuration $\mathbf{C}_{mv}$ and then how they terminate their computation.

4.2.1 Transforming a collinear $\mathbf{C}_0$ to a Non-collinear $\mathbf{C}_0$

When $\mathbf{C}_0$ is collinear, any robot $r_i \in \mathcal{Q}$ sees at least one (if an endpoint robot) and at most two other robots (if not an endpoint robot) in $\mathbf{C}(r_i)$. Let $c_1, \ldots, c_N$ be the robots in the line segment convex hull $\mathbf{P}$ formed from $\mathbf{C}_0$ with c_1, c_N be its endpoints. Robots c_1 and c_N see one other robot in $\mathbf{C}(c_1)$ and $\mathbf{C}(c_N)$ and they move to make $\mathbf{C}_0$ non-collinear (we discuss later our selection of moving the endpoint robots c_1, c_N, not the non-endpoint robots $c_2, \ldots, c_{N-1}$). The remaining

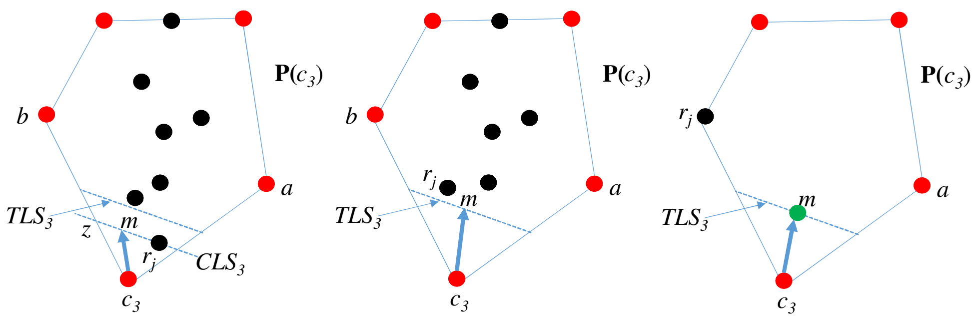
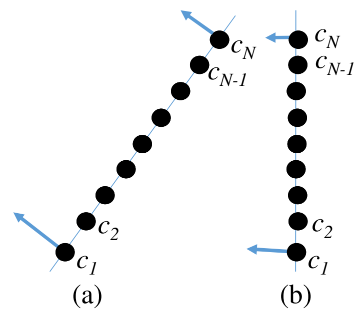


Figure 1: An illustration of how a corner of **P** moves when it (left) sees (at least) two robots with color "Off" and there are robots inside its $TR(*)$, (middle) sees (at least) two "Off" robots but no robot is inside $TR(*)$, and (right) sees one "Off" robot which is its neighbor in **P**.

robots $c_2, \ldots, c_{N-1}$ (that see two other robots in their $\mathbf{C}(*)$) do nothing. Let c_2, c_{N-1} be the only other robot in $\mathbf{C}(c_1), \mathbf{C}(c_N)$, respectively.

We now describe how c_1 moves (the case for c_N is analogous). If $c_1.x < c_2.x$ (the x-coordinates), then c_1 moves perpendicular to $\overline{c_1c_2}$ in clockwise direction by a small distance $\delta > 0$ (keeping its color `Off`). In this case, c_1 is the leftmost endpoint in the line segment hull **P**. If $c_1.x > c_2.x$, then c_1 moves perpendicular to $\overline{c_1c_2}$ in counterclockwise direction by a small distance $\delta > 0$ (keeping its color `Off`). In this case, c_1 is the rightmost endpoint in the line segment hull **P**. If $c_1.x = c_2.x$, then c_1 recognizes that the line segment **P** is vertical. In this case, c_1 compares its y-coordinate with the y-coordinate of c_2. If $c_1.y < c_2.y$, c_1 is the lowermost endpoint of the line segment **P**. Robot c_1 moves perpendicular to $\overline{c_1c_2}$ in clockwise direction by a small distance $\delta > 0$ (keeping its color `Off`). Otherwise, c_1 moves perpendicular to $\overline{c_1c_2}$ in counterclockwise direction by a small distance $\delta > 0$ (keeping its color `Off`). Figure on the right illustrates these ideas. This transformation finishes when at least one of c_1, c_N moves one time. We will prove in Lemma 5.1 that this technique indeed transforms any collinear $\mathbf{C}_0$ to non-collinear $\mathbf{C}_0$ in our $\mathcal{SSYNC}$ setting.



We now argue our selection of moving the endpoint robots (not the non-endpoint robots) to transform collinear $\mathbf{C}_0$ to a non-collinear $\mathbf{C}_0$. Consider the case of $N = 3$ with all three robots c_1, c_2, c_3 collinear in a line L with c_2 between c_1 and c_3. If c_2 becomes crash faulty when it becomes active for the first time, it may never move away from L and Complete Visibility is never achieved.

4.2.2 Reaching Complete Visibility Configuration from a Non-Collinear $\mathbf{C}_0$

We now describe in detail how to reach a Complete Visibility configuration $\mathbf{C}_{mv}$ starting from a non-collinear initial configuration $\mathbf{C}_0$.

Let $\mathcal{Q}_c, \mathcal{Q}_s, \mathcal{Q}_i$ be the set of corners, sides, and interior robots of **P**. Note that each of these sets are disjoint from each other, i.e., if a robot $r_i \in \mathcal{Q}_i$, then $r_i \notin \mathcal{Q}_s$ and $r_i \notin \mathcal{Q}_c$. A robot $r_c \in \mathcal{Q}_c$, after its first activation, assumes color `Red` (without moving). The colors of the robots in $\mathcal{Q}_s$ and $\mathcal{Q}_i$ (the side and interior robots of **P**) remain `Off` until they become corners of **P**. Since there is a faulty robot, the color of at most one robot in $\mathcal{Q}_c$ (the corners of **P**) may also be `Off`.

Our idea is to shrink the convex hull **P** by moving the corners in $\mathcal{Q}_c$ to the interior of **P** in such a way that they remain as corners of **P** and within finite time at least one side or one interior robot of **P** becomes a new corner of **P**. Notice that interior and side robots in $\mathcal{Q}_i$ and $\mathcal{Q}_s$ do not move (until they become corners of **P**).

We now describe how the corner robots in $\mathcal{Q}_c$ move. Let $r_c \in \mathcal{Q}_c$ be a corner of **P**. Let a, b be its counterclockwise and clockwise neighbors in the boundary of **P**. Robot r_c, after colored `Red`, considers two different scenarios and moves accordingly as described below.

Algorithm 2: $Corner(r_i, \mathbf{C}(r_i), \mathbf{P}(r_i))$

if $r_i.light$ = `Off` **then** $r_i.light \leftarrow$ `Red`;
else if $r_i.light$ = `Red` *and* $\forall r_j \in \mathbf{C}(r_i) \backslash \{r_i\}, r_j.light \in \{$`Red`, `Green`$\}$ **then** $r_i.light$ = `Green`;
else if $\forall r_j \in \mathbf{C}(r_i), r_j.light$ = `Green` **then** Terminate;
else if $\forall r_j \in \mathbf{C}(r_i) \backslash \{r_i\}, r_j.light$ = `Green` *and* r_i *is in the interior of* $\mathbf{P}(r_i)$ *with light* `Off` **then** Terminate;
else if $\forall r_j \in \mathbf{C}(r_i) \backslash \{r_i, r_k\}, r_j.light$ =`Green` *and* $r_k.light$ = `Red` *and* r_i *is in the interior of* $\mathbf{P}(r_i)$ *with light* `Off` **then** Terminate;
else if $\forall r_j \in \mathbf{C}(r_i) \backslash \{r_i, r_k\}, r_j.light$ =`Green` *and* $r_i.light \in \{$`Red`, `Green`$\}$ *and* r_k *is in the interior of* $\mathbf{P}(r_i)$ *with light* `Off` **then** Terminate;
else if $\forall r_j \in \mathbf{C}(r_i) \backslash \{r_i\}, r_j.light$ = `Green` *and* r_i *is a corner of* $\mathbf{P}(r_i)$ *with light* $\in \{$`Red`, `Off`$\}$ **then** Terminate;
else if $\forall r_j \in \mathbf{C}(r_i) \backslash \{r_i, r_k\}, r_j.light$ = `Green` *and* $r_i.light$ = `Green` *and* r_k *is a corner of* $\mathbf{P}(r_i)$ *with light* $\in \{$`Red`, `Off`$\}$ **then** Terminate;
else
  $a \leftarrow$ counterclockwise neighbor on the boundary of $\mathbf{P}(r_i)$;
  $b \leftarrow$ clockwise neighbor on the boundary of $\mathbf{P}(r_i)$;
  $x \leftarrow$ midpoint of the line segment $\overline{r_i a}$;
  $y \leftarrow$ midpoint of the line segment $\overline{r_i b}$;
  if *there exists more than one robot in* $\mathbf{C}(r_i) \backslash \{r_i\}$ *with light* `Off` **then**
    if $r_i.light$ = `Green` **then** $r_i.light \leftarrow$ `Red`;
    $r_j \leftarrow$ robot in $\mathbf{C}(r_i)$ with light `Off` that is the closest to r_i (if more than one on CLS_i, pick one closer to b);
    if r_j *is not inside triangle* $\Delta r_i x y$ **then**
      move to the midpoint of TLS_i;
    else
      $z \leftarrow$ intersection point of CLS_i and $\overline{r_i b}$;
      move to the midpoint of the line segment $\overline{r_j z}$;
  else if *there exists only one robot* $r_j \in \mathbf{C}(r_i) \backslash \{r_i\}$ *with light* `Off` *and* r_j *is not* a *and not* b **then**
    $r_{in} \leftarrow r_j$;
    $r_{max} \leftarrow$ the robot in $\mathbf{C}(r_i)$ with maximum x-coordinate; (if more than one satisfies this criteria, choose as r_{max} the robot with maximum y-coordinate)
    if $r_i == r_{max}$ **then** $r_{max} \leftarrow$ the closest robot to r_i in $\mathbf{C}(r_i)$ w.r.t. to x-coordinate; (if more than one satisfies this criteria, choose as r_{max} the robot with maximum y-coordinate)
    if $\overline{r_{in} r_i}$ *is in counterclockwise direction from* $\overline{r_{in} r_{max}}$ **then**
      $r_k \leftarrow$ the first robot in the counterclockwise direction of $\overline{r_i r_{in}}$ in the opposite side of r_i ;
      $W \leftarrow$ the intersection of the line $\overline{r_k r_{in}}$ with $\mathbf{P}(r_i)$;
      if $\angle r_i r_{in} W \leq \angle r_i r_{in} a$ **then** $Destination(r_i, r_{in}, r_{max}, W)$;
      else $Destination(r_i, r_{in}, r_{max}, a)$;
    if $\overline{r_{in} r_i}$ *is in clockwise direction from* $\overline{r_{in} r_{max}}$ **then**
      $r_k \leftarrow$ the first robot in the clockwise direction of $\overline{r_i r_{in}}$ in the opposite side of r_i;
      $W \leftarrow$ the intersection of the line $\overline{r_k r_{in}}$ with $\mathbf{P}(r_i)$;
      if $\angle r_i r_{in} W \leq \angle r_i r_{in} b$ **then** $Destination(r_i, r_{in}, r_{max}, W)$;
      else $Destination(r_i, r_{in}, r_{max}, b)$;
  else if *there exists only one robot* $r_j \in \mathbf{C}(r_i) \backslash \{r_i\}$ *with light* `Off` *and* r_j *is either* a *or* b **then**
    move to the midpoint of TLS_i and set $r_i.light \leftarrow$ `Green`;

- **Robot r_c sees at least two robots with color `Off`:** Robot r_c finds the closest robot among the `Off` colored robots it sees in $\mathbf{C}(r_i)$. Let r_j be that robot. If r_j is inside the corner triangle $TR(r_c)$, it finds the intersection point z of the corner line segment CLS_c and $\overline{r_c b}$ and moves to the midpoint m of the line segment $\overline{r_j z}$ that connects r_j with point z (point z is on segment $\overline{r_c b}$). The left of Fig. 1 illustrates this move for a corner c_3 of $\mathbf{P}$. If r_j is not inside $TR(r_c)$, it moves to the midpoint m of the triangle line segment TLS_c. The middle of Fig. 1 illustrates this move for a corner c_3 of $\mathbf{P}$. In both the moves, r_c keeps its color `Red`.

- **Robot r_c sees one robot with color `Off`:** Let r_j be the robot with color `Off` that r_c sees in $\mathbf{C}(r_c)$. Robot r_c considers the following two sub-cases.

  - **Robot r_j is either a or b:** Robot r_c moves to the midpoint m of the triangle line segment TLS_c and assumes color `Green`. The right of Fig. 1 illustrates this move for a corner c_3 of $\mathbf{P}$.

  - **Robot r_j is not a and not b:** This is the most involved case. In this case, r_j may be in the interior of $\mathbf{P}$ or on a side of $\mathbf{P}$ (which is different than the sides $\overline{r_c a}$ and $\overline{r_c b}$ of $\mathbf{P}$). If r_j is a side robot then the neighbor corners of r_j will do the move as in the previous sub-case to make r_j a corner. The other remaining corner robots of $\mathbf{P}$ (including r_c) will treat r_j as an interior robot of $\mathbf{P}$ and make the move as described below.

    For clarity of discussion, let r_j be denoted as r_{in} (to indicate the it is an interior robot of $\mathbf{P}$ in view of r_c). We describe the move for corner r_c (the move for other corners of $\mathbf{P}$ is analogous). Let r_{max} be the robot in $\mathbf{C}(r_c)$ with maximum x-coordinate. If r_c itself is r_{max}, then r_c takes as r_{max} the robot in $\mathbf{C}(r_c)\backslash\{r_c\}$ with maximum x-coordinate (denote this robot as r'_{max}). If more than one robot satisfies this criteria, then r_c takes as r_{max} the robot with maximum y-coordinate. Notice that r'_{max} is a neighbor of r_{max} in $\mathbf{P}$ because $r_{\max}$ is the maximum x-coordinate robot and r'_{max} must be at least the second largest x-coordinate robot. Let $\overline{r_c r_{in}}$ and $\overline{r_{in} r_{max}}$ be the line segments that connect robots r_c and r_{max} with robot r_{in}.

    We first define one notion that we will heavily use. Extend the line segment $\overline{r_c r_{in}}$ from the endpoint r_{in} so that it intersects the perimeter of $\mathbf{P}$. Denote the point of intersection by H (see the left of Fig. 2 taking c_0 as r_c). We say that point H is in the *opposite side* of r_c. Therefore, when we say counterclockwise (or clockwise) direction of $\overline{r_c r_{in}}$ in the opposite side of r_c, then we mean the neighbor robot of H in the perimeter of $\mathbf{P}$ in counterclockwise (or clockwise) direction from H.

    If r_c is in the counterclockwise direction from r_{max} (the angle $\angle r_c r_{in} r_{max} < 180°$), let r_k be the first robot in the counterclockwise direction of $\overline{r_c r_{in}}$ in the opposite side of r_c. Connect r_k with r_{in} and extend towards r_c. Let W be the intersection point of line $\overrightarrow{r_k r_{in}}$ in the boundary of $\mathbf{P}$. W may be the point on $\overline{r_c a}$ or on some other edge of $\mathbf{P}$. If W is not on $\overline{r_c a}$, r_c takes point a as W, and moves to a point r' in $\overline{r_c W}$ assuming color `Green`. The left of Fig. 2 illustrates this move for a corner c_0 of $\mathbf{P}$. Note that point r' where c_0 moves is in the counterclockwise direction of c_0 (and r_{max}). We will describe later how the point r' is computed (the pseudocode of this technique is in Algorithm 3).

    If r_c is in the clockwise direction from r_{max}, then r_k is the first robot in the clockwise direction of $\overline{r_c r_{in}}$ in the opposite side of r_c. Then, W is either point b (if the line $\overrightarrow{r_k r_{in}}$ intersects the boundary of $\mathbf{P}$ not at side $\overline{r_c b}$) or the point at $\overline{r_c b}$ where the line $\overrightarrow{r_k r_{in}}$ intersects $\overline{r_c b}$. Robot r_c then moves to a point r' in $\overline{r_c b}$ assuming color `Green`. The right of Fig. 2 illustrates this move for a corner c_3 of $\mathbf{P}$. Note that the point r' where c_3 moves is in the clockwise direction from c_3 (and r_{max}).

We now describe how the point r' is computed. The pseudocode is in Algorithm 3. Let γ be the angle r_c forms with robot r_{in} and point W. If r_c is in the counterclockwise direction of r_{max}, then W is either the counterclockwise neighbor corner a of r_c in the boundary of $\mathbf{P}$ or some point in the side $\overline{r_c a}$. However, if r_c is in the clockwise direction of r_{max}, then W is either the clockwise neighbor corner b of r_c in the boundary of $\mathbf{P}$ or some point in the side $\overline{r_c b}$. Let L be the line that passes

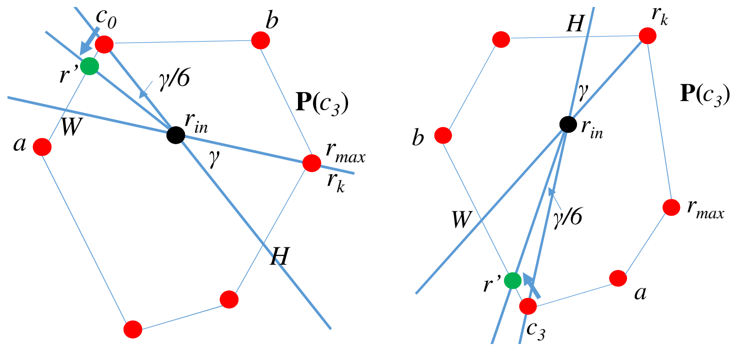


Figure 2: An illustration of how a corner moves when its sees only one “Off” robot that is not its neighbor in $\mathbf{P}$: (left) when the corner is in the counterclockwise direction of r_{max} and (right) when the corner is in the clockwise direction of r_{max}.

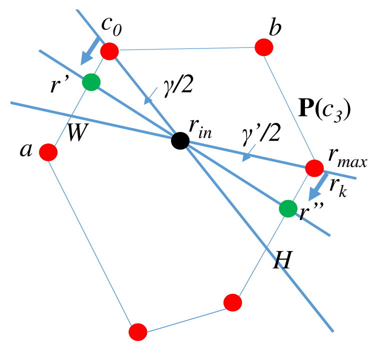


Figure 3: An illustration of a possible collinear configuration when choosing angle $\gamma/2$.

through r_{in} and intersects line $\overline{r_cW}$ making an angle $\gamma/6$ with $\overline{r_cr_{in}}$. The point r' is the intersection point of L and $\overline{r_cW}$. Fig. 2 illustrates these ideas.

The main idea behind choosing $\gamma/6$ (say not $\gamma/2$) is the following. The idea is illustrated in Fig. 3. Let r_c (c_0 in Fig. 3) be in the counterclockwise direction of r_{max} and γ be the angle r_c forms with the interior robot r_{in} and point W. Let γ' be the angle in the opposite side of r_c that the first robot r_k in the counterclockwise direction of $\overline{Hr_{in}}$ makes with lines $\overline{r_cr_{in}}$ and $\overrightarrow{r_kr_{in}}$. In Fig. 3, r_k also happened to be r_{max}. Suppose γ and γ' are equal and so does the sides $\overline{r_cW}$ and $\overline{r_kH}$. In a symmetric configuration, if we choose angles $\gamma/2$ and $\gamma'/2$ (instead of $\gamma/6$ and $\gamma'/6$ as discussed in the previous paragraph), then r_c and r_k might move to point r' on $\overline{r_cW}$ and r'' on $\overline{r_kH}$, respectively, so that they would be collinear again with the interior robot r_{in}. Our selection of angles $\gamma/6$ and $\gamma'/6$ avoids this collinear situation.

There might be scenarios where after r_c assumes color `Green`, it sees two or more robots with color `Off`. This happens in scenarios where all the interior robots are collinear in a line and only one robot on that line is visible to r_c (before r_c moves). In this case, r_c changes its color back to `Red` and continue its convex hull shrinking process by moving inward in $\mathbf{P}$ keeping its color `Red` until it again sees exactly one robot with light `Off`. We then provide the guarantee that if r_c does not see two or more robots with color `Off` after changing its color to `Green` from `Red`, then there must not be more than one robot with color `Off` in the system. This plays a crucial role in the termination guarantee of the algorithm described in the next subsection.

4.2.3 Termination of the Algorithm

We now describe when robots terminate their computation solving Complete Visibility. Each robot $r_i \in \mathcal{Q}$ terminates as soon as one of the following conditions holds for it.

Algorithm 3: *Destination*$(r_i, r_{in}, r_{max}, q)$

```
1 α ← angle ∠r_i r_in r_max;
2 β ← angle ∠q r_in r_max;
3 γ = (β − α) with sides r_i r_in and q r_in;
4 L ← line that makes angle γ/6 with side r_i r_in towards q;
5 r′ ← the intersection point of L and r_i q;
6 move to point r′;
7 r_i.light ← Green;
```

- All the robots in $\mathbf{C}(r_i)$ are colored `Green`. That is, when r_i sees all the robots in $\mathbf{C}(r_i)$ have color `Green`, then all the robots in $\mathcal{Q}$ must be in the corners of a hull $\mathbf{P}$. This is because, the robots in the sides and interior of $\mathbf{P}$ never assume color `Red` or `Green` until they become corners of $\mathbf{P}$.

- Robot r_i is in the interior of $\mathbf{P}(r_i)$ (with light `Off`) and all the other robots in $\mathbf{C}(r_i)$ are colored `Green`. This condition guarantees that r_i is the only robot in the interior of $\mathbf{P}$. This is because, otherwise r_i would have seen at least one other robot in $\mathbf{C}(r_i)$ colored $\in$ {`Red`, `Off`}.

- Robot r_i is in the interior of $\mathbf{P}(r_i)$ (with light `Off`) and all the other robots in $\mathbf{C}(r_i)$ are colored $\in$ {`Green`, `Red`}. This condition guarantees that r_i is the only robot in the interior of $\mathbf{P}$. This is because, otherwise r_i would have seen at least one other robot in $\mathbf{C}(r_i)$ colored `Off`. This also extends to the case where r_i has light $\in$ {`Green`, `Red`} and some other robot is in the interior of $\mathbf{P}(r_i)$ with color `Off`.

- Robot r_i is on a corner of $\mathbf{P}(r_i)$ (with light $\in$ {`Red`, `Off`}) and all the other robots in $\mathbf{C}(r_i)$ are colored `Green`. This condition guarantees that there is no robot in the interior of $\mathbf{P}$. This is because, otherwise, there must be at least one other robot in $\mathbf{C}(r_i)$ colored `Off`. This also extends to the case where r_i has light `Green` but some other robot in a corner of $\mathbf{P}(r_i)$ has light $\in$ {`Red`, `Off`}.

Observe that throughout the algorithm we used three colors `Off`, `Red`, and `Green` for the robots in $\mathcal{Q}$.

5 Analysis of the Single Fault Algorithm

In this section, we 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 at most one robot is crash-faulty). We further show that the algorithm terminates in finite time and the execution is collision-free. We start with the following lemma which shows that if the initial configuration $\mathbf{C}_0$ is a line, it correctly transforms to a non-collinear configuration $\mathbf{C}_0$.

Lemma 5.1 *When at least one robot in the endpoint of the collinear $\mathbf{C}_0$ moves once and $N \geq 3$, there exists a hull $\mathbf{P}$ (with at least three corners) such that all robots in $\mathcal{Q}$ are on the corners and sides of $\mathbf{P}$ with color* `Off`.

Proof. Let $c_1, c_2, \ldots, c_{N-1}, c_N$ be the collinear robots of $\mathbf{C}_0$ in a line $\overline{c_1 c_N}$ with c_1, c_N be its endpoints and c_2, c_{N-1} be the neighbors of c_1, c_N in $\overline{c_1 c_N}$, respectively. Let c_1 be the leftmost (or the bottommost robot if $\overline{c_1 c_N}$ is a vertical line) on $\overline{c_1 c_N}$. Since there is a faulty robot, at least one of the endpoint robots c_1, c_N in the collinear $\mathbf{C}_0$ can move even if the faulty robot is either c_1 or c_N. Since c_1 (and/or c_N) moves perpendicular to the collinear $\mathbf{C}_0$, there must be at least three corners since $\angle c_1 c_2 c_N < 180°$ after c_1 moves once. If both c_1, c_N move then $\mathbf{P}$ has 4 corners when $N \geq 4$ as they move in the same side of the collinear $\mathbf{C}_0$ (c_1 moves in the clockwise direction and c_N moves in the counterclockwise direction). For $N = 3$, $\mathbf{P}$ is still a triangle since again both c_1, c_N move in the same side of collinear $\mathbf{C}_0$. It is easy to see that when c_1 moves once, c_2 becomes a corner and the other robots $c_3, \ldots, c_{N-1}$ remain as side robots in $\overline{c_2 c_N}$ if c_N has not moved. If c_N has moved, c_{N-1} also becomes a corner and all other robots $c_3, \ldots, c_{N-2}$ remain as side robots in $\overline{c_2 c_{N-1}}$. All the robots have light `Off` since they do not change color when they move in the collinear case. □

We now prove the following lemma which shows that the execution of the algorithm is collision-free. Particularly, we show that the paths the robots follow do not intersect and no two robots move to the same position during the execution of the algorithm.

Lemma 5.2 *The execution of the algorithm is collision-free.*

Proof. We first show that the paths of robots do not cross each other throughout the execution of the algorithm. When all the robots are collinear, only at most two endpoint robots move perpendicular to the line segment hull, which immediately shows that the paths of two robots do not cross. Starting from any non-collinear configuration $\mathbf{C}_0$, observe that only the corner robots in the set $\mathcal{Q}_c$ move during the execution of the algorithm (and the robots in $\mathcal{Q}_s$ and $\mathcal{Q}_i$ do not move). When a corner $c_i \in \mathcal{Q}_c$ moves, it makes two kind of moves:

(i) moving to the positions of TLS_i or CLS_i, or

(ii) moving to the point r' in the side $\overline{c_i a}$ or $\overline{c_i b}$ of $\mathbf{P}$ (Fig. 2), where a, b are the counterclockwise and clockwise neighbors of c_i in the boundary of $\mathbf{P}$.

We first show that the paths of corner robots of $\mathbf{P}$ do not cross when they move to the positions of TLS_* or CLS_* (Case (i)). When a corner $c_i \in \mathcal{Q}_c$ moves, it moves somewhere in the line TLS_i (or CLS_i) in the triangular area $TR(c_i)$ and $TR(c_i)$ does not overlap with the triangular area $TR(c_j)$ of any other corner $c_j \in \mathcal{Q}_c$. Therefore, paths of any two robots do not cross.

We now show that the path of c_i does not cross with the path of any other corner c_j when c_i moves to the point r' in $\overline{c_i a}$ or $\overline{c_i b}$ of $\mathbf{P}$ (Case (ii)). Note that c_i makes its move to r' when it sees (exactly) one robot r with color `Off` and r is not a and not b. At this case, the neighbor corners of c_i either move to their TLS_* (or CLS_*) or to their point r'. In this former case, r' is not inside the corner triangle $TR(*)$ of the neighbors of c_i. In the latter case, the position r' that c_i and r'' that c_i's neighbor c_j move are on $\overline{c_i c_j}$ (even if they move toward each other) such that r' is closer to c_i than c_j and r'' is closer to c_j than c_i (so that it never be the case that point c_i is also the point c_j and vice-versa).

We now show that the robots do not share positions throughout the execution of the algorithm. In the initial configuration $\mathbf{C}_0$, they do not share positions since they are already in distinct positions due to our assumption on any initial configuration $\mathbf{C}_0$. After that, if a corner robot c_i moves on TLS_i, it does not share position with any other robot since there is no robot on TLS_i and inside the corner triangle $TR(r_i)$. If a corner robot c_i moves on CLS_i, it does not share position with any other robot on CLS_i (in this case there is at least one robot on CLS_i) because c_i takes the closer robot r to b on CLS_i and moves to the midpoint of $\overline{rz}$, where z is the intersection point of CLS_i on side $\overline{c_i b}$ connecting c_i with its clockwise neighbor b in $\mathbf{P}$. The robots c_i, c_j moving on the sides of $\mathbf{P}$ do not share positions since their destination points are never the same point and those destination points are closer to them than the other robots. □

We now show that corner robots of $\mathbf{P}$ remain as corners throughout the execution of the algorithm and the corners of $\mathbf{P}$ monotonically increase which is essential to guarantee progress towards a Complete Visibility configuration $\mathbf{C}_{mv}$.

Lemma 5.3 *No corner robot becomes internal or side robot of* $\mathbf{P}$ *throughout the execution of the algorithm. Moreover, the corner robots of* $\mathbf{P}$ *monotonically increase.*

Proof. We first show that no corner of $\mathbf{P}$ becomes internal or side robot of $\mathbf{P}$ throughout the execution of the algorithm. Let c_i be a corner robot of $\mathbf{P}$ and a, b be its counterclockwise and clockwise neighbors in the boundary of $\mathbf{P}$. Robot c_i either (i) moves toward the interior of $\mathbf{P}$ to a position itself in either the triangle line segment TLS_i (or the corner line segment CLS_i) or (ii) moves to a position in edge $\overline{c_i a}$ or $\overline{c_i b}$ in the boundary of **P**. Notice that both TLS_i and CLS_i are inside triangular area Δac_ib. Furthermore, both TLS_i and CLS_i are parallel to $\overline{ab}$ (and also parallel to each other).

We first show that c_i does not become internal or side robot in the former case (Case (i)). We have that before c_i moves to TLS_i or CLS_i, $\angle ac_ib < 180°$. In other words, all the robots in $\mathcal{Q}$ are in the region divided by lines $\overline{c_i a}$ and $\overline{c_i b}$ making angle $< 180°$ at c_i. Let x_i be the position of r_i on TLS_i or CLS_i after it moves. We will show that either angle $\angle cr_id < 180°$ from the new position x_i of r_i, where c, d are either a, b, respectively, or some other internal robots in Δac_ib that become corners of **P** due to the move of c_i. Suppose first that a, b are still the neighbors of c_i. Since TLS_i (and CLS_i) is inside Δac_ib, the angle $\angle ac_ib < 180°$ as robot c_i becomes a side robot of **P** if and only if it moves to a position on $\overline{ab}$. If a and/or b is not c_i's neighbor after it moves to x_i, then either there is another robot on CLS_i or there is an robot inside Δac_ib between lines TLS_i and $\overline{ab}$. In the case of some another robot r' on CLS_i, since c_i moves to a position x_i on CLS_i such that x_i is closer to b (the right neighbor of c_i) than r' (and any other robot in CLS_i), c_i still has $\angle r'c_ib < 180°$. This is because CLS_i is parallel to $\overline{ab}$ and hence b, c_i, r' can not be collinear. In the case of some robot r'' inside Δac_ib between lines TLS_i and $\overline{ab}$, if a and/or b is not the neighbor of c_i after it moved to x_i, then r'' becomes a neighbor of c_i and since r'' is between TLS_i and $\overline{ab}$, $\angle ac_ir'' < 180°$ (if r'' is the new neighbor of c_i instead of b) and $\angle r'c_ib < 180°$ (if r'' is the new neighbor of c_i instead of a).

We now show that c_i does not become internal or side robot in the latter case (Case (ii)). In Case (ii), c_i moves to a position on either $\overline{c_i a}$ or $\overline{c_i b}$. Note that when c_i moves on $\overline{c_i a}$ or $\overline{c_i b}$ at r', there is no robot inside triangle Δac_ib. Therefore, $\angle ac_ib < 180°$ from its new position r since r is not the point on $\overline{ab}$.

We now show that the corner robots of **P** monotonically increase. This follows analogously to the proof of monotonic increase of the corners of **P** provided by Di Luna *et al.* [13] for their algorithm. This is because, similar to the algorithm of Di Luna *et al.* [13], the corners of **P** always move toward the interior of **P** to shrink the hull in our algorithm until there is at most one robot in the interior of **P**. Due to being a faulty robot, there is a situation in which a corner r_i may see only one robot in the interior of **P** even when there are many robots in the interior of **P**. This is the case of all interior robots in the line $\overline{r_i r_{in}}$, where r_{in} is the interior robot of **P** that is visible to r_i. In this case, the move of r_i in the perimeter of **P** assuming color `Green` breaks the collinearity so that r_i sees all the robots that were blocked by r_{in} previously. The lemma follows. □

We will prove in Theorem 1.1 that, after the execution of the algorithm, either all the robots in $\mathcal{Q}$ are positioned on the corners of a hull **P** or there is a single robot r_{in} in the interior of **P**. We start with a proof that the interior robot r_{in} is not collinear with any two corners of **P**. This is needed to guarantee that Complete Visibility is solved for all (at least $N-1$) non-faulty robots.

Lemma 5.4 *If there exists a robot r_{in} in the interior of* **P** *after the robots in the boundary of* **P** *terminate, then r_{in} is not collinear in all the lines joining any two corners of* **P**.

Proof. Let r_i be a corner in **P** and let r_{in} be the only robot in the interior of **P**. Robot r_{in} can determine if it is the only interior robot in **P** when all other robots it sees in $\mathbf{C}(r_i)$ have color $\in$ {`Red`, `Green`}. robot r_i can determine r_{in} is the only interior robot in **P** if it sees all other robots have color `Green` or `Red`. In some cases, this might not be true due to many collinear interior robots in **P**. We will show in the proof of Theorem 1.1 that this does not hamper the algorithm.

Extend the line $\overline{r_i r_{in}}$ to the opposite of r_i. Line $\overline{r_i r_{in}}$ intersects the boundary of **P** at point H. H can be a point on a side of **P** joining two consecutive corners or a corner point in **P**. If H is a point on a side, it is immediate that r_{in} does not block any robot from the view of r_i (since r_{in} is the only robot in the interior of **P**). However, since r_i does not see H, r_i cannot decide whether there is an robot on point H or not. Therefore, r_i always assumes that H is a corner point of **P** with a corner robot positioned on it. We have that $\angle r_ir_{in}H = 180°$ (Fig. 2 illustrates this idea).

Let $r_{max} \neq r_i$ be the robot in $\mathbf{C}(r_i)$ with maximum x-axis. We will consider the case of r_i itself as $r_{\max}$ later. Suppose r_i is in the counterclockwise direction from the line segment $\overline{r_{in}r_{max}}$. (We have the notion of counterclockwise and clockwise directions for r_i based on the angle $\angle r_ir_{in}r_{max}$; if $\angle r_ir_{in}r_{max} < 180°$ in the counterclockwise direction of $\overline{r_{in}r_{max}}$ then we will say that r_i is in the counterclockwise direction of r_{max}. Otherwise, we will say that r_i is in the clockwise direction of r_{max}.)

Let $\alpha = \angle r_i r_{in} r_{max}$ and $\theta = \angle H r_{in} r_{max}$. We have that both $\alpha, \theta < 180°$ and $\alpha + \theta = 180°$. We will show that after r_i and/or H (we assume that there is a robot positioned on point H) moves, the new angles they make with r_{max} (assuming that r_{max} is stationary) is such that $\alpha' \geq \alpha, \theta' \geq \theta$, and $\alpha' + \theta' > 180°$. This guarantees that both H and r_i are not collinear with r_{in} anymore.

Let a be the counterclockwise neighbor of r_i and r_k be the counterclockwise neighbor of (the hidden robot) H in the boundary of **P**. Observe that a is a corner of **P** with color $\in$ {`Red`, `Green`}. Let W be the intersection point of $\overrightarrow{r_k r_{in}}$ in the boundary of **P** towards r_i. If W is a point on edge $\overline{r_i a}$, then $\gamma = \angle r_i r_{in} W$, otherwise $\gamma = \angle r_i r_{in} a$. Fig. 2 illustrates this construction.

We have the following three scenarios due to the $\mathcal{SSYNC}$ setting:

(i) only r_i (or H) moves in a round,

(ii) both r_i, H move in a round, and

(iii) r_{max} moves after r_i (or H) moves and then H (or r_i) moves.

We first consider Case (i). Let r' be the point on $\overline{r_i a}$ so that $\angle r_i r_{in} r' = \frac{\gamma}{6}$. Since r_i moves to r' and H, r_{max}, r_{in} are stationary $\angle r' r_{in} r_{max} = \alpha + \frac{\gamma}{6} > \alpha$. Since $\theta' = \theta$ (as H does not move), $\alpha' + \theta' > 180°$ and hence r_{in} is not collinear with r_i and H anymore.

We now consider Case (ii). We have from Case (i) that $\alpha' > \alpha$. Let b' be the clockwise neighbor of H. The point W' be the intersection point of $\overrightarrow{r'_k r_{in}}$ in the boundary of **P** towards H, where r'_k is the clockwise neighbor of r_i in the boundary of **P**. Since H moves to point r'' on $\overline{Hb'}$, $\angle r'' r_{in} r_{max} = \theta + \frac{\gamma}{6} > \theta$, where $\gamma = \angle H r_{in} r''$. Therefore, $\alpha' + \theta' > 180°$ and hence r_i and H are not collinear with r_{in} anymore.

Consider now Case (iii). Let r_i moved at round t, r_{max} moved at round t', and H moved at round t''. We consider the case $t < t' < t''$; if any two robots among r_i, r_{max}, and H move at some round $t = t'$, we can argue the correctness using Case (ii). The case of H moving first, r_{max} second, and then r_i is analogous. Since r_i moved first, we have that $\alpha' > \alpha$. It remains to show that $\alpha' + \beta' > 180°$. Let r'_{max} be the new position of r_{max} after it is moved at round t'. Since H is not r_{max}, H is on the same side of line $\overline{r_i r_{max}}$ joining r_{max} with r_i. Therefore, H again moves in the clockwise direction. Since H and b' are not the same point, we have that $\theta' > \theta$. Therefore, r_i and H are not collinear with r_{in} anymore.

We now consider the case of r_i itself as r_{max}. In this situation, r_i would consider as r_{max} the closest robot to it w.r.t. the x-axis. Denote that robot as r'_{max}. Note that the robot H that is hidden in $\overline{r_{in} r_i}$ would also choose as r_{max} the robot r'_{max}. This is because r_i (which is also r_{max} is hidden from its view). This would ensure that r_i moves in the counterclockwise direction from r'_{max} and H would move in the clockwise direction from r'_{max} and $\alpha' + \theta' > 180°$.

Finally, we consider the situation where some robots of **P** think one corner of **P** as r_{max} and the remaining robots of **P** think another corner of **P** as r_{max}. Fig. 4 illustrates a situation in which corners a (blocked by r_{in} to see r_i) and r_i of **P** think corner d of **P** as r_{max} and the remaining corners of **P** think r_i as r_{max}. Observe that besides r_i and a, no other corner of **P** thinks d as r_{max}; that is, there are at most 2 robots of **P** that think d as r_{max} which is different from r_i, the actual r_{max}. Note that d is the robot in **P** that is closer to r_i w.r.t. the x-axis. According to the algorithm, robot r_i would move to position r'' in line $\overline{r_i H}$ and a would move to position r' in line $\overline{a r_j}$. There may be the situation that r_j (a neighbor of a in the counterclockwise of r_i) move towards a to point r'_j in line $\overline{a r_j}$. Even in this case, a and r_j do not cross each other in $\overline{a r_j}$ and, therefore, $\alpha' + \theta' > 180°$ for any two robots collinear with r_{in} in $\overline{r_i H}$. □

We are now ready to prove the main result from the analysis of our algorithm, Theorem 1.1.

Proof of Theorem 1.1. We have from Lemma 5.1 that if the initial configuration $\mathbf{C}_0$ is a line, then it correctly transforms to a non-collinear configuration $\mathbf{C}_0$, when at least an endpoint robot of that line moves once. Moreover, all the robots in the non-collinear $\mathbf{C}_0$ have color `Off`.

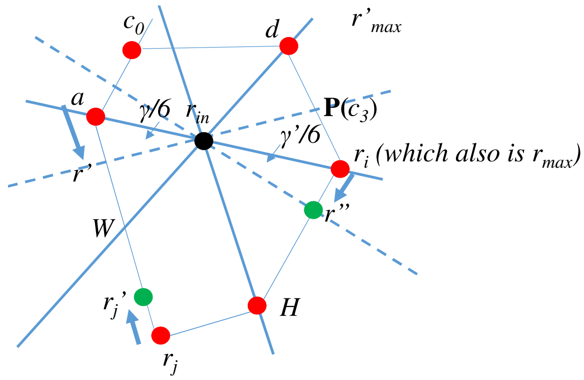


Figure 4: An illustration of a configuration in which r_{max} for some robots (corners a, r_i) is different than the actual r_{max} (corner r_i) in $\mathbf{P}$.

Therefore, suppose $\mathbf{P}$ is a convex hull of the robots in $\mathcal{Q}$ with at least three corners and colors of all the robots `Off`. Let $\mathcal{Q}_c, \mathcal{Q}_s, \mathcal{Q}_i$ be the robots on the corners, sides, and in the interior of $\mathbf{P}$, respectively. We will show that, when all the robots in $\mathcal{Q}$ terminate, then either all the robots in the sets $\mathcal{Q}_s, \mathcal{Q}_i$ also become corners of $\mathbf{P}$ (i.e., $|\mathcal{Q}_c| = N$) or there is exactly one robot r_{in} in the interior of $\mathbf{P}$ which is not collinear with any two corners of $\mathbf{P}$ (i.e., $|\mathcal{Q}_c| = N-1$ and $|\mathcal{Q}_i| = 1$). Finally, we will show that the robots terminate in finite time avoiding collisions.

Initially, all robots in $\mathcal{Q}$ have color `Off`. The corners in the set $\mathcal{Q}_c$ assume color `Red` (without moving) when they become active for the first time. Let r_c be a corner of $\mathbf{P}$. After colored `Red`, until r_c sees at least 2 robots with light `Off` in $\mathbf{C}(r_c)$, it moves toward the interior of $\mathbf{P}$ to shrink $\mathbf{P}$, keeping its color `Red`. Suppose r_c sees, in some round, only one robot r with color `Off`. Robot r might be on the side, corner, or in the interior of $\mathbf{P}$. For all these cases, we prove by contradiction that r will not be collinear with any two corners of $\mathbf{P}$.

Consider first that r is a side robot of $\mathbf{P}$. We have that $r.light$ = `Off`. Suppose for the contradiction that when all the robots of $\mathbf{P}$ terminate, r still remains as a side robot. Let a, b be the counterclockwise and clockwise neighbors of r_c on the boundary of $\mathbf{P}$. If r is a or b for r_c then r_c assumes color `Green` and moves toward the interior of $\mathbf{P}$ to shrink $\mathbf{P}$. After r_c moves once inward, there are two cases: Either r_c still sees only one robot with light `Off` or more than one robot with light `Off`. If r_c sees only one robot with light `Off` when it has color `Green`, it terminates. This is because, due to the move of r_c inward and r does not move, r must become the corner of $\mathbf{P}$ and r is not collinear with any two robots of $\mathbf{P}$ anymore, a contradiction.

If r_c sees more than one robot with color `Off`, it changes its color back to `Red` and continue until it sees again only one robot with color `Off`. Robot r_c then eventually sees only one robot with light `Off` after it assumes color `Green` and the above argument guarantees that r becomes a corner of $\mathbf{P}$ (and it is not collinear with any two corners of $\mathbf{P}$). The corners of $\mathbf{P}$ except r_c consider r as an interior robot in $\mathbf{P}$. We will show below when they move, r does not become collinear with any two corners of $\mathbf{P}$.

Consider now that r is a corner robot of $\mathbf{P}$. If it is a or b, then arguing similarly as above, r_c knows that r is a corner of $\mathbf{P}$. If r is not a and not b, r_c considers r as an interior robot in $\mathbf{P}$ and r_c moves to a point r' in $\overline{r_c a}$ or $\overline{r_c b}$ and assumes color `Green`. We will show that after this r is not collinear with any two corners of $\mathbf{P}$.

We now consider r in the interior of $\mathbf{P}$. We have from Lemma 5.4 that when r_c and/or H (the robot that is hidden from the view of r_c since it is collinear with r_c) moves, r is not collinear with r_c and H.

We now show that after r becomes non-collinear with r_c and H, it does not become collinear again. The argument is as follows. Note that before r_c and/or H move, they have color `Red`. After they move as in Lemma 5.4, they assume color `Green`. After assuming color `Green`, r_c and H terminate if they see again only one robot with color `Off`. Therefore, r does not become collinear again.

We now show that after the robots terminate, the Complete Visibility problem is in fact solved. If r_c is a corner and sees all the robots with color `Green`, then all robots in $\mathcal{Q}$ are in the corners of $\mathbf{P}$. Since otherwise, there must be at least one robot with color `Off`. If r_c is a corner with light `Off` or `Red`, it sees all other robots with light `Green`. Robot r_c can then terminate since it is not collinear any other corner of $\mathbf{P}$. If r_c is in the interior of $\mathbf{P}$ and sees all other robots with color `Green`, then there is no other robot in the interior of $\mathbf{P}$ and r_c can simply terminate and Lemma 5.4 provides the guarantee that the Complete Visibility is solved. If r_c sees all other robots with light `Green` except one robot with light `Red`, then all the robots of $\mathcal{Q}$ (except r_c) are in the corners of $\mathbf{P}$ and Lemma 5.4 guarantees Complete Visibility. Due to the single faulty robot, only one robot can have light other than `Green` throughout the execution of the algorithm.

We now show that the algorithm terminates in finite time. It is immediate that all the robots in $\mathcal{Q}$ become active within finite time. The corners of $\mathbf{P}$ make a move inward or on the edge of $\mathbf{P}$ every time they become active. We have from Lemma 5.3 that each corner r_c of $\mathbf{P}$ remains as a corner of $\mathbf{P}$ and there is no collision between any two robots of $\mathcal{Q}$ (Lemma 5.2). Moreover, we have from Lemma 5.3 that the number of corners of $\mathbf{P}$ monotonically increase during the execution of the algorithm. Furthermore, the execution of the algorithm guarantees that a corner r_i of $\mathbf{P}$ can not terminate until there are more than one robot in the interior of $\mathbf{P}$. This is because, if r_i sees exactly one robot r_{in} in the interior, it assumes color `Green` moving appropriately, which makes sure that r_i sees all the robots blocked by r_{in}. Robot r_i then moves inward changing color to `Red` until it again sees at most one robot in the interior. Note that an interior robot can hide other robots only one the line $\overline{r_i rin}$ and after r_i moves it sees all of them. Therefore, since we have finite number of robots in $\mathcal{Q}$, the algorithm terminates in finite time. Moreover, only three colors `Off`, `Red`, and `Green` are used by robots throughout the execution of the algorithm. □

6 Impossibility of Complete Visibility under Byzantine Faults

We studied so far the crash-fault model and provided an algorithm that solves Complete Visibility tolerating a faulty robot in a system of $N \geq 3$ robots using 3 colors in the semi-synchronous setting. Note that in the crash-fault model, the faulty robot is allowed to stop its movement at any time but after it becomes faulty it remains stationary thereafter. A natural question is to see whether Complete Visibility can be solved if a robot is Byzantine faulty. Therefore, we consider here the Byzantine fault model. Note that in the Byzantine fault model, after a robot becomes faulty, it might behave in an unpredictable and unforeseeable way. That is, it may exhibit arbitrary behavior and movements. We now prove that in the semi-synchronous setting, it is impossible for any algorithm to achieve Complete Visibility in a system of $N = 3$ robots in the Byzantine fault model, when one robot is Byzantine faulty (this extends also to the fully synchronous model which we describe later). In particular, we prove the following theorem.

Theorem 6.1 *Given $N = 3$ robots (with lights) being in the distinct positions in a plane, there is no algorithm that solves* Complete Visibility *tolerating a Byzantine-faulty robot in the semi-synchronous setting.*

Proof. Suppose that there is an initial configuration $Conf_1$ consisting of three robots r_1, r_2, and r_3 which are collinear and r_1 and r_3 are blocked from each other by r_2. Consider the scenario where both r_1 and r_3 are active at round t (r_2 is not active at round t) and the Complete Visibility algorithm instructs them to move in such away that they become visible by each other, resulting in a configuration $Conf_2$. Since there are only three robots, $Conf_2$ must be a triangle with r_1, r_2, r_3 being its corners. Suppose r_2 is Byzantine faulty. At round $t+1$, assume that r_1 and r_3 are inactive and r_2 becomes active (r_2 was inactive at round t). Since r_2 is Byzantine faulty, suppose it moves to a point on edge $\overline{r_1 r_3}$ of the triangle so that it again becomes collinear with r_1 and r_3, resulting in a configuration equivalent to $Conf_1$. Assume at round $t+2$, both r_1 and r_3 become active and the algorithm instructs them to move in such away that they again become visible to each other, resulting in a configuration equivalent to $Conf_2$. Since r_2 is Byzantine faulty, the execution can then alternate between configurations $\{Conf_1 \rightarrow Conf_2 \rightarrow Conf_1 \rightarrow Conf_2 \rightarrow \ldots\}$. □

Remarks It is easy to see that Theorem 6.1 extends also to the fully synchronous setting. Starting from $Conf_1$, we can simulate the actions of r_1, r_2, r_3 to act like the proof of Theorem 6.1 as follows. At round t, even if r_2 is active, it does not move giving configuration $Conf_2$. At round $t+1$, r_1, r_3 do not move (although they are active) and r_2 moves similarly as in Theorem 6.1 to provide configuration $Conf_1$. And, this can alternative forever.

7 Algorithm Tolerating Two Crash Faults

We now discuss how Complete Visibility can be solved in a system of $N \geq 3$ robots when two robots are crash-faulty, extending the techniques developed in Section 4. Note that our algorithm for one crash-faulty robot (Section 4) might not be able to solve Complete Visibility, starting from any arbitrary initial configuration $\mathbf{C}_0$, when two robots are crash-faulty. For an illustration, consider a configuration where there are two crash-faulty robots u, v with some other robots on the line segment $\overline{uv}$ (between u and v). Since u, v do not move, the robots that are not on line $\overline{uv}$ (including u, v) eventually converge to $\overline{uv}$ as the robots between u and v on $\overline{uv}$ also can not move in addition to u, v in our algorithm. Therefore, we consider a subset of arbitrary initial configurations $\mathbf{C}_0$, which we call feasible initial configurations, $\mathbf{C}_{feasible}$, defined as follows.

Definition 1 *Given a set of $N \geq 3$ robots (with lights) being in the distinct positions in a plane, the feasible initial configurations $\mathbf{C}_{feasible}$ are all arbitrary initial configurations $\mathbf{C}_0$ where, for any two robots u, v that become crash-faulty (at any time $t \geq 0$), there is no third robot between u and v in the line segment $\overline{uv}$ connecting u and v.*

Before describing the algorithm, we provide a definition that we need in the algorithm. Let r_c be a corner of $\mathbf{P}$. The *eligible area* for r_c, denoted as $EA(r_c)$, is a polygonal subregion inside corner triangle $TR(r_c)$ for r_c. We have from Sharma *et al.* [20] that $EA(r_c)$ can be computed for each corner r_c of $\mathbf{P}$ such that it satisfies the following lemma. Furthermore, the eligible areas of two different corners of $\mathbf{P}$ do not overlap [20].

Lemma 7.1 ([20]) *The eligible area $EA(r_c)$ for each corner r_c of $\mathbf{P}$ is a non-empty convex polygon. When r_c moves to any point in $EA(r_c)$, r_c remains as a corner of $\mathbf{P}$ and all the internal and side robots of $\mathbf{P}$ are visible to r_c (and vice-versa).*

Algorithm We now discuss how Complete Visibility can be solved starting from all feasible initial configurations $\mathbf{C}_{feasible}$ (Definition 1). The collinear $\mathbf{C}_{feasible}$ can be transformed to a non-collinear $\mathbf{C}_{feasible}$ using the technique of Section 4.2.1. It is easy to see that Definition 1 is not violated after applying the technique of Section 4.2.1.

After non-collinear $\mathbf{C}_{feasible}$ is reached, each corner of $\mathbf{P}$ colors itself `Red` (without moving). After colored `Red`, the corners of $\mathbf{P}$ move toward the interior of $\mathbf{P}$ to shrink $\mathbf{P}$ until all robots in $\mathcal{Q}$ become corners of $\mathbf{P}$ as in Section 4.2.2. The side and interior robots of $\mathbf{P}$ do nothing (no change in color and they do not move) until they become corners of $\mathbf{P}$.

We are now ready to describe how a corner r_c of $\mathbf{P}$ moves to shrink $\mathbf{P}$. The goal is to make both u, v (faulty robots) corners of $\mathbf{P}$, along with the remaining robots of $\mathcal{Q}$. After that from Definition 1, we can argue on the correctness of the algorithm on solving Complete Visibility. Let r_c be a corner of $\mathbf{P}$ colored `Red`. It differentiates the following three cases to move when becomes active in some round.

- **Robot r_c sees at least three robots with color `Off`:** Robot r_c moves toward the interior of $\mathbf{P}$ as in Section 4.2.2, keeping its color `Red`. Robot r_c moves to either CLS_c or TLS_c.
- **Robot r_c sees exactly two robots r_i, r_j with color `Off`:** Robot r_c differentiates the following two sub-cases:
    - (i) **Robot r_i and/or r_j is in the interior or side of $\mathbf{P}(r_c)$:** Robot r_c moves toward the interior of $\mathbf{P}$ as in the (above) case of seeing at least three robots with lights `Off`. Note that r_c moves to either CLS_c or TLS_c, and keeps its color `Red`.

(ii) **Both r_i, r_j are corners of $\mathbf{P}(r_c)$:** Robot r_c moves to a point in the eligible area $EA(r_c)$ and assumes color `Green`.

- **Robot r_c sees exactly one robot r_i with color `Off`:** Robot r_c differentiates the following two sub-cases:

  (i) **Robot r_i is in the interior or side of $\mathbf{P}(r_c)$:** Robot r_c moves toward the interior of $\mathbf{P}$ as in the (above) case of seeing at least three robots with lights `Off`. Note that r_c moves to either CLS_c or TLS_c, and keeps its color `Red`.

  (ii) **Robot r_i is a corner of $\mathbf{P}(r_c)$:** Robot r_c moves to a point in the eligible area $EA(r_c)$ and assumes color `Green`.

There might be scenarios similar to Section 4.2.2 where after r_c colored `Green`, it sees either (a) three or more robots with color `Off` or (b) at least a robot with color `Off` in the interior or side of $\mathbf{P}$. In this case, r_c changes its color back to `Red` and continue shrinking $\mathbf{P}$ based on which case above applies for r_c.

We now discuss how a corner r_c terminates its computation. Robot r_c terminates if and only if all three conditions below satisfy simultaneously.

(i) r_c is colored `Green`,

(ii) all robots in $\mathbf{C}(r_c)$ are on the corners of $\mathbf{P}(r_c)$, i.e., there is no side or interior robot in $\mathbf{P}(r_c)$, and

(iii) all corners of $\mathbf{P}(r_c)$, in addition to r_c, are colored `Green`, except at most 2 corner robots of $\mathbf{P}$ colored $\in \{$`Red`, `Off`$\}$.

Analysis of the Algorithm We now analyze the correctness of the algorithm. We proceed by proving the following lemma which is crucial to show that all robots of any feasible initial configuration $\mathbf{C}_{feasible}$ become corners of $\mathbf{P}$ and Complete Visibility is solved.

Lemma 7.2 *Let a corner robot r_c, after colored `Green`, sees at most 2 robots r_i, r_j with light `Off` in the corners of $\mathbf{P}(r_c)$ and there is no side or interior robot in $\mathbf{P}$. If r_i, r_j are in fact the interior robots of $\mathbf{P}$, then (i) they block r_c from seeing at most two corners of $\mathbf{P}$, and (ii) they are inside triangles formed by three consecutive corners of $\mathbf{P}$.*

Proof. Since r_c is colored `Green`, it must have seen r_i, r_j on the corners of $\mathbf{P}(r_c)$ with color `Off`, when it was colored `Red`. Otherwise, r_c would not assume color `Green`. When r_c assumed color `Green`, it must have moved to a point in $EA(r_c)$. We have from Lemma 7.1 that r_c sees all interior and side robots of $\mathbf{P}$ after moving to a point in $EA(c_i)$. Therefore, since r_c sees no interior robot besides r_i, r_j even after moving to $EA(r_c)$, then either (a) r_i, r_j are corners of $\mathbf{P}$ or (b) r_i, r_j are the only robots in the interior of $\mathbf{P}$. In Case (a), we are done. In Case (b), each r_i, r_j can block only one corner of $\mathbf{P}$. Therefore, we have the part (i) of lemma. For part (ii), it is easy to observe that if r_i (or r_j) was not in the triangle formed by three consecutive corners of $\mathbf{P}$, r_c would have seen it as internal in $\mathbf{P}(r_c)$. □

We are now ready to prove the main result of this section.

Theorem 7.3 *Given a set of $N \geq 3$ robots (with lights) being in the distinct positions in a plane satisfying Definition 1, there is an algorithm that solves* Complete Visibility *tolerating two crash-faulty robots using 3 colors and without collisions in the semi-synchronous setting.*

Proof. Similar to Lemma 5.1, it can be shown that any collinear feasible initial configuration $\mathbf{C}_{feasible}$ correctly transforms to a non-collinear feasible configuration $\mathbf{C}_{feasible}$. The collision-free execution of the algorithm is also immediate similar to Lemma 5.2. Furthermore, similar to Lemma 5.3, it is easy to see that corners of $\mathbf{P}$ remain as corners and the corners of $\mathbf{P}$ monotonically increase throughout the execution of the algorithm.

We have from Lemma 7.2 that a corner r_c of **P** never terminates if there are more than two robots in the interior, side, or on the corners of **P** with color `Off`. We again have from Lemma 7.2 that if only (at most) two robots r_i, r_j are in the interior of **P** with color `Off`, and a corner r_c terminates, then they are inside the triangles formed by three consecutive corners of **P**. Since the corners are moving inside, once inside the triangle of a corner, an interior robot never gets outside of that triangle until it becomes a corner (since that robot does not move until it becomes a corner). Therefore, even if r_c terminates, r_i, r_j become corners of **P** through the moves of the other corners of **P** (except r_c). This is because all other corners of **P** (except r_c) will definitely see both r_i, r_j internal in their convex hulls (one robot cannot block the same robot from two or more different robots) and two corners of **P** see one of r_i, r_j inside the triangle they form with their neighbors. If r_i, r_j were inside the triangles of the corners but not blocking r_c to see the corner of triangle they are in, r_c sees r_i, r_j as internal in $\mathbf{P}(r_c)$ and it terminates after both r_i, r_j become corners of **P**. Therefore, the techniques used to prove Theorem 1.1 can be extended to obtain this theorem.

Only three colors `Off`, `Red`, and `Green` are used throughout the algorithm. □

8 Concluding Remarks

We have presented, to our best knowledge, the first fault-tolerant algorithm for the Complete Visibility problem using 3 colors in the robots with lights model under the semi-synchronous setting, tolerating one crash-faulty robot, not known a priori. The algorithm terminates in finite time avoiding collisions. All previous algorithms were not fault-tolerant (except handling some special cases in [11]). We then provided an impossibility result on solving the Complete Visibility problem tolerating a Byzantine faulty robot in a system of $N = 3$ robots. Furthermore, we provided a Complete Visibility algorithm that tolerates two crash-faulty robots in a system of $N \geq 3$ robots using 3 colors in the semi-synchronous setting for a certain subset of arbitrary initial configurations.

Many questions remain for future work. It will be interesting to extend our algorithm to handle non-rigid movements and also to the asynchronous setting. It will also be interesting to either (i) minimize the number of colors from 3 to 2 as a 2-color algorithm is optimal w.r.t. the number of colors in the fault-free robots with lights model (for Complete Visibility) [11, 16] when N is not known; note that in our algorithm robots have no knowledge of N, or (ii) prove that any 3-color solution is optimal in the faulty robots with lights model. Most importantly, it will be interesting to tolerate 3 or more faults in the crash-fault model.

References

[1] Chrysovalandis Agathangelou, Chryssis Georgiou, and Marios Mavronicolas. A distributed algorithm for gathering many fat mobile robots in the plane. In *PODC*, pages 250–259, 2013.

[2] Noa Agmon and David Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM J. Comput.*, 36(1):56–82, July 2006.

[3] Subhash Bhagat and Krishnendu Mukhopadhyaya. Optimum algorithm for mutual visibility among asynchronous robots with lights. In *SSS*, pages 341–355, 2017.

[4] Kálmán Bolla, Tamás Kovacs, and Gábor Fazekas. Gathering of fat robots with limited visibility and without global navigation. In *SIDE*, pages 30–38, 2012.

[5] Reuven Cohen and David Peleg. Local spreading algorithms for autonomous robot systems. *Theor. Comput. Sci.*, 399(1-2):71–82, June 2008.

[6] Andreas Cord-Landwehr, Bastian Degener, Matthias Fischer, Martina Hüllmann, Barbara Kempkes, Alexander Klaas, Peter Kling, Sven Kurras, Marcus Märtens, Friedhelm Meyer auf der Heide, Christoph Raupach, Kamil Swierkot, Daniel Warner, Christoph Weddemann, and Daniel Wonisch. Collisionless gathering of robots with an extent. In *SOFSEM*, pages 178–189, 2011.

[7] Jurek Czyzowicz, Leszek Gasieniec, and Andrzej Pelc. Gathering few fat mobile robots in the plane. *Theor. Comput. Sci.*, 410(6-7):481–499, 2009.

[8] Shantanu Das, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Masafumi Yamashita. Autonomous mobile robots with lights. *Theor. Comput. Sci.*, 609:171–184, 2016.

[9] Ayan Dutta, Sruti Gan Chaudhuri, Suparno Datta, and Krishnendu Mukhopadhyaya. Circle formation by asynchronous fat robots with limited visibility. In *ICDCIT*, pages 83–93, 2012.

[10] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed computing by oblivious mobile robots. *Synthesis Lectures on Distributed Computing Theory*, 3(2):1–185, 2012.

[11] Giuseppe Antonio Di Luna, Paola Flocchini, Sruti Gan Chaudhuri, Federico Poloni, Nicola Santoro, and Giovanni Viglietta. Mutual visibility by luminous robots without collisions. *Inf. Comput.*, 254:392–418, 2017.

[12] Giuseppe Antonio Di Luna, Paola Flocchini, Sruti Gan Chaudhuri, Nicola Santoro, and Giovanni Viglietta. Robots with lights: Overcoming obstructed visibility without colliding. In *SSS*, pages 150–164, 2014.

[13] Giuseppe Antonio Di Luna, Paola Flocchini, Federico Poloni, Nicola Santoro, and Giovanni Viglietta. The mutual visibility problem for oblivious robots. In *CCCG*, pages 348–354, 2014.

[14] Linda Pagli, Giuseppe Prencipe, and Giovanni Viglietta. Getting close without touching: Near-gathering for autonomous mobile robots. *Distrib. Comput.*, 28(5):333–349, October 2015.

[15] David Peleg. Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In *IWDC*, pages 1–12, 2005.

[16] Gokarna Sharma, Costas Busch, and Supratik Mukhopadhyay. Bounds on mutual visibility algorithms. In *CCCG*, pages 268–274, 2015.

[17] Gokarna Sharma, Costas Busch, and Supratik Mukhopadhyay. Mutual visibility with an optimal number of colors. In *ALGOSENSORS*, pages 196–210, 2015.

[18] Gokarna Sharma, Costas Busch, and Supratik Mukhopadhyay. Brief announcement: Complete visibility for oblivious robots in linear time. In *SPAA*, pages 325–327, 2017.

[19] Gokarna Sharma, Ramachandran Vaidyanathan, and Jerry L. Trahan. Constant-time complete visibility for asynchronous robots with lights. In *SSS*, pages 265–281, 2017.

[20] Gokarna Sharma, Ramachandran Vaidyanathan, Jerry L. Trahan, Costas Busch, and Suresh Rai. Complete visibility for robots with lights in O(1) time. In *SSS*, pages 327–345, 2016.

[21] Gokarna Sharma, Ramachandran Vaidyanathan, Jerry L. Trahan, Costas Busch, and Suresh Rai. Logarithmic-time complete visibility for asynchronous robots with lights. In *IPDPS*, pages 513–522, 2017.

[22] Ramachandran Vaidyanathan, Costas Busch, Jerry L. Trahan, Gokarna Sharma, and Suresh Rai. Logarithmic-time complete visibility for robots with lights. In *IPDPS*, pages 375–384, 2015.