Brief Announcement: Complete Visibility for Oblivious Robots in Linear Time

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

Costas Busch
Louisiana State University
Baton Rouge, LA
busch@csc.lsu.edu

Supratik Mukhopadhyay
Louisiana State University
Baton Rouge, LA
supratik@csc.lsu.edu

ABSTRACT

We consider the distributed setting of N autonomous mobile robots that operate in *Look-Compute-Move* cycles following the well-celebrated *classic oblivious robots* model. We study the fundamental problem where starting from an arbitrary initial configuration, N autonomous robots reposition themselves to a convex hull formation on the plane where each robot is visible to all others (the Complete Visibility problem). We assume obstructed visibility, where a robot cannot see another robot if a third robot is positioned between them on the straight line connecting them. We provide the first $O(N)$ time algorithm for this problem in the fully synchronous setting. Our contribution is a significant improvement over the runtime of the only previously known algorithm for this problem which has a lower bound of $\Omega(N^2)$. Our proposed algorithm is collision-free – robots do not share positions and their paths do not cross.

CCS CONCEPTS

•Theory of computation → Models of computation; Design and analysis of algorithms; Distributed algorithms; •Computing methodologies → Multi-agent systems; Intelligent agents; Mobile agents;

KEYWORDS

Complete visibility; Obstruction; Collisions; Convex hull; Autonomous mobile robots; Oblivious Robots; Runtime

1 INTRODUCTION

The well-celebrated *classic oblivious model* of distributed computing by a finite team of autonomous mobile robots enjoys a long history of research [2]. In this model, the robots are points in a plane, which is also what we assume here. In a large spatial extent, robots can be seen as points relative to the spatial extent in which they operate and the solutions obtained for point robots form the building blocks for the robots that are not points (i.e., robots that occupy certain space such as an unit disk area). Moreover, many robot motion planning algorithms in $\mathbb{R}^2$ (such as bug algorithms [1]) have been studied for the point robots. Point robots are also interesting for exploring the computational efficiency of solving basic robot coordination tasks.

SPAA'17, July 24–26, 2017, Washington, DC, USA.
 978-1-4503-4593-4/17/07.
DOI: 10.1145/3087556.3087591

In this classic model, the point robots are: *autonomous* (no external control), *anonymous* (no unique identifiers), *indistinguishable* (no external identifiers), *oblivious* (do not remember their previous actions or the previous positions of the other robots), *silent* (no direct means of communication), and *disoriented* (no common coordinate system or unit of measure for the distances) [2]. Each robot executes the same algorithm and they all perform their actions following *Look-Compute-Move* (LCM) cycles, i.e., when a robot becomes active, it first observes the positions of other robots (*Look*), then computes a destination point based on that observation (*Compute*), and finally moves towards the destination (*Move*). Many fundamental distributed coordination problems, such as pattern formation, convergence, gathering, scattering, etc., were solved in this model [2].

The classic oblivious robots model makes one important assumption: All the robots in the system are visible to each other at all times. In other words, the visibility is *unobstructed* - three collinear robots are assumed to be mutually visible to each other [2]. The assumption of unobstructed visibility can be easily refuted because the view of the robots that are collinear is blocked in a real setting. Therefore, we remove this assumption which leads to the scenario of *obstructed visibility* under which a robot r_i can see another robot r_j if and only if there is no third robot in the line segment joining their positions. We assume that except the presence of robots, there is no other obstacle for a robot to see another robot. Therefore, we consider the variant of the classic model where robots have obstructed visibility.

Di Luna *et al.* [3] gave the first algorithm for classic oblivious robots to solve the fundamental Complete Visibility problem with obstructed visibility: Given a team of N mobile robots in arbitrary distinct positions in the Euclidean plane $\mathbb{R}^2$, all the robots reach a convex hull configuration in which each robot is in a distinct corner 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 known to the robots. The importance of solving the Complete Visibility problem is that it makes it possible to solve many other robot coordination problems, including gathering, shape formation, and leader election, under obstructed visibility.

Similar to Di Luna *et al.* [3], in our Complete Visibility solution the robots are arranged on corners of a convex polygon. Di Luna *et al.* [3] proved the correctness of their algorithm but gave no runtime analysis (except a proof of finite time termination). The goal of this work is to develop a fast runtime algorithm that solves Complete Visibility for classic oblivious robots.

Contributions. We consider a distributed system of N robots (agents) from a set $Q = \{r_1, \ldots, r_N\}$. Each robot is a (dimensionless) point that can move in an infinite 2-dimensional real plane $\mathbb{R}^2$ following the classic oblivious robots model [2].

In this paper, we prove the following result which, to our knowledge, is the first algorithm for Complete Visibility that achieves linear runtime for classic oblivious robots with obstructed visibility. Time is measured in rounds. A *round* is the smallest number of LCM cycles within which each robot is guaranteed to be active at least once. Since we assume the $\mathcal{FSYNCH}$ setting, a round is a LCM cycle. This result assumes that a robot in motion cannot be stopped by an adversary, i.e., when a robot moves it stops only after it reaches to its destination point (also called *rigid* movements).

Theorem 1.1. *For any initial configuration of $N \geq 3$ classic oblivious robots being in distinct positions in a plane,* Complete Visibility *can be solved in $O(N)$ time without collisions in the fully synchronous setting.*

This is a significant improvement since it can be shown that the only previous algorithm of Di Luna *et al.* [3] for this problem has the time lower bound of $\Omega(N^2)$ in the fully synchronous setting.

The lower bound proof idea is to use an initial configuration where all N robots are on the points of two concentric circles, big and small, with distance between each robot and its two neighbors is the same. Moreover, the robots in the small circle are collinear with the robots in the big circle. Since the algorithm of Di Luna *et al.* [3] only moves the robots in the big circle inward, it can be shown, with appropriately chosen number of robots in the big and small circle, that the big circle does not coincide with the small circle even after executing the algorithm for at least $c \cdot N^2$ rounds, for some constant c. Moreover, collinear robots stay collinear during these rounds. The formal proof will be similar to [4, Theorem 4].

Technique. The main idea is to make robots move autonomously based on their local views (and without communicating with other robots) to become corners of a N-vertex convex hull. When all N robots become corners of a convex hull, the configuration naturally solves Complete Visibility.

Let $\mathbf{H}$ be a convex hull of the given N robots. Initially, the robots are either in the perimeter of $\mathbf{H}$ (i.e., corners and sides of $\mathbf{H}$) or in its interior. The previous algorithm [3] asks robots in the corners of $\mathbf{H}$ to move inward to shrink the hull so that the existing corners of $\mathbf{H}$ remain as corners and the internal robots of $\mathbf{H}$ become new corners of $\mathbf{H}$. The corners of $\mathbf{H}$ do not need to know completely $\mathbf{H}$ to move inward. It is sufficient for a corner robot r of $\mathbf{H}$ to determine all N robots are in a plane with angle $< 180^\circ$ formed by r with the leftmost and the rightmost robot it sees. When a robot that was in the interior of $\mathbf{H}$ becomes a new corner of $\mathbf{H}$, it starts moving inward causing other interior robots new corners of $\mathbf{H}$. Since the robots know N, they eventually recognize the situation of all robots being in the corners of $\mathbf{H}$ and terminate, solving Complete Visibility. However, this approach has the time lower bound of $\Omega(N^2)$.

Our technique is to move internal robots in $\mathbf{H}$ outward towards the perimeter of $\mathbf{H}$ in addition to the moves of the corners of $\mathbf{H}$ inward used in [3]. This is challenging since internal robots in $\mathbf{H}$ may not know which direction is outward and which direction is inward (due to their weak capabilities). We indeed address this challenge and able to show that, in each round, at least an internal robot in $\mathbf{H}$ can correctly move outward towards the perimeter of $\mathbf{H}$ and becomes a new corner of $\mathbf{H}$. We also show that our technique avoids collisions between robots. Solutions avoiding collisions are appealing since colliding of robots may endanger the robots themselves.

2 OVERVIEW OF THE ALGORITHM

In this section, we outline our $O(N)$ time algorithm for Complete Visibility. The algorithm consists of interior depletion (ID) and corner depletion (CD) procedures which work together to make robots reach a configuration where they are positioned in the corners of a N-vertex convex hull (polygon) $\mathbf{H}$ and terminate.

A special case in our algorithm is when initially all robots in Q are collinear. This situation can be detected when a robot r_i sees at most two other robots r_j, r_k, and r_i, r_j, r_k are collinear. If r_i sees two other robots r_j, r_k then r_i is not an endpoint robot of that line. Robot r_i then moves a small distance $\delta > 0$ directly perpendicular to the line $\overline{r_j r_k}$. For $N \geq 3$, we show that this move of r_i ensures that in the resulting configuration not all robots in Q are collinear. (The problem becomes trivial when $N \leq 2$.)

The Interior Depletion Procedure. The ID procedure makes the robots in the interior of $\mathbf{H}$ move outward toward the perimeter of $\mathbf{H}$ and the CD procedure makes the corner robots of $\mathbf{H}$ move inward in $\mathbf{H}$. The robots in Q can easily determine whether they are corners of $\mathbf{H}$ or in its interior. If a robot r_i sees all robots in $\mathbf{C}(r_i)$ (the positions of the robots in Q that r_i sees at any time $t \geq 0$) are within an angle of $< 180^\circ$, r_i realizes that it is a corner robot of $\mathbf{H}$ and executes the CD procedure to move inward. If r_i does not see all robots in $\mathbf{C}(r_i)$ within an angle of $< 180^\circ$, it realizes that it is an interior robot and executes the ID procedure to move outward toward the perimeter of $\mathbf{H}$. The robots which are already on the edges of $\mathbf{H}$ (angle exactly $= 180^\circ$) perform no action until they become corners of $\mathbf{H}$.

The Corner Depletion Procedure. The CD procedure for the corner robots of $\mathbf{H}$ is executed in such a way that they remain as corners of $\mathbf{H}$ and at least one robot that is not the corner of $\mathbf{H}$ (edge or interior) becomes a new corner of $\mathbf{H}$. If there is at least one edge robot in $\mathbf{H}$, it becomes a new corner of $\mathbf{H}$ immediately after all the corners of $\mathbf{H}$ move inward once. If there is no edge robot, at least a robot in the interior of $\mathbf{H}$ becomes a new corner of $\mathbf{H}$ due to the ID procedure executed by the interior robots (simultaneously with the corners of $\mathbf{H}$). This all happens in a single round κ due to the $\mathcal{FSYNCH}$ setting. This is crucial since it allows us to guarantee the claimed runtime of $O(N)$ rounds. Fig. 2 shows how an internal robot r' in $\mathbf{H}$ become a new corner of $\mathbf{H}$ after r' moves to a point z' and the corner v_1 in $\mathbf{H}$ moves to point z''. A robot r_i terminates as soon as it sees N corners in $\mathbf{H}(r_i)$, i.e., all N robots in Q are in the corners of $\mathbf{H}(r_i)$ (r_i can do this decision since it knows N).

Formally, let v_1 be a corner robot of $\mathbf{H}$ and a, b be its left and right neighbors in the boundary of $\mathbf{H}$, respectively. We need following definitions. Let $\Delta a v_1 b$ be the triangle formed by a, v_1, b. Let $\overline{xy}$ be a line parallel to $\overline{ab}$ passing through points $x = \text{length}(\overline{v_1 a})/8$ and $y = \text{length}(\overline{v_1 b})/8$ from v_1 in lines $\overline{v_1 a}$ and $\overline{v_1 b}$, respectively. We say line $\overline{xy}$ is the *triangle line segment* and denote it by TLS_{v_1}.

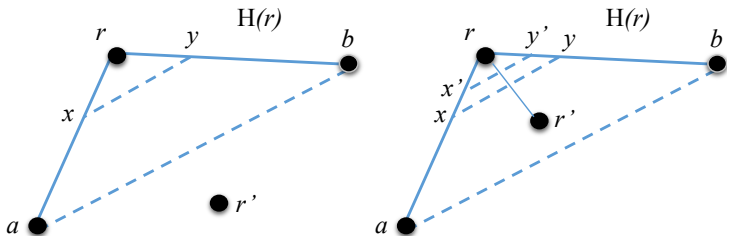

Figure 1: An illustration of (left) triangle line segment, where $\overline{xy}$ is TLS_r and (right) corner line segment, where $\overline{x'y'}$ is CLS_r for a corner robot r.

Let r' be a robot inside Δav_1b. Let z be the point in $\overline{v_1r'}$ at distance length$(\overline{v_1r'})/8$ from v_1. Let $\overline{x'y'}$ be a line parallel to $\overline{ab}$ (or $\overline{xy}$) passing through point z. We say line $\overline{x'y'}$ is the *corner line segment* and denote it by CLS_{v_1}. If there are many robots inside Δav_1b, let r' and r'' be the robots inside Δav_1b that are closest to $\overline{v_1a}$ and $\overline{v_1b}$, respectively. CLS_{v_1} is then computed based on r' or r'' that is closest to v_1. According to the definition of CLS_{v_1} and TLS_{v_1}, CLS_{v_1} is parallel to TLS_{v_1} and CLS_{v_1} is closer to the corner v_1 than TLS_{v_1} (Fig. 1).

Robot v_1 executes the CD procedure as follows.

- **No robot inside** Δav_1b**:** Robot v_1 moves to a position in the triangle line segment TLS_{v_1}.
- **Robots inside** Δav_1b**:** Robot v_1 moves to a position in the corner line segment CLS_{v_1}.

Simultaneously at the same round, an internal robot r' in the interior of **H** executes the ID procedure as follows.

- **Robot** r' **is not inside** Δav_1b**:** Robot r' moves to a position in the triangle line segment TLS_{v_1} (different than the one that will be occupied by v_1).
- **Robot** r' **is inside** Δav_1b**:** Robot r' moves to a position in the corner line segment CLS_{v_1} (different than the one that will be occupied by v_1).

We then prove that when both v_1 and r' move to either TLS_r or CLS_r, v_1 remains as a corner of **H** and r' becomes a new corner of **H**.

For r' to move outward toward v_1, $\overline{v_1a}$ and/or $\overline{v_1b}$ must be the closest edge to r' and r' is closest to v_1 than a and/or b. In situations where there are robots inside the triangular area divided by TLS_{v_1} or CLS_{v_1} towards v_1, r' may not become a new corner of **H** even after it moves to TLS_{v_1} or CLS_{v_1}. In this situation, we are able to show that some robot $\widehat{r'}$ inside that triangular area will become a new corner of **H**. Furthermore, r' may not be able to compute CLS_{v_1} when r' does not see b (or a). But, what we are able to guarantee is that CLS_{v_1} passes through the point that r' moves to and this is sufficient for our algorithm. Fig. 2 shows how a corner robot v_1 of **H** moves inward and a robot r' inside triangle $\Delta v_2v_1v_3$ moves outward toward v_1 and both get positioned in two distinct positions of CLS_{v_1} (shown as line segment $\overline{xy}$ in the figure). The figure also shows the positions z' and z'' that r' and v_1 occupy, respectively, in CLS_{v_1}. The point z' is at distance length$(\overline{v_1r'})/8$ from v_1 in line $\overline{v_1r'}$ and the point z'' is the midpoint of $\overline{z'y}$ with y being the intersection point of CLS_{v_1} and $\overline{v_1v_2}$. Note also that

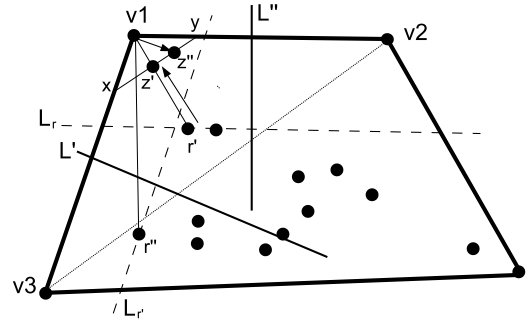

Figure 2: An illustration of how a corner robot $v_1 \in$ H moves inward and an internal robot $r' \in$ H moves outward toward v_1. Both v_1 and r' move to CLS_{v_1} (shown as $\overline{xy}$ in the figure) where the position of r' is the point z' at length$(\overline{v_1r'})/8$ **from v_1 in line $\overline{v_1r'}$ and the position of v_1 is the point z'' that is midpoint of $\overline{z'y}$ with y being the point of intersection of CLS_{v_1} and $\overline{v_1v_2}$.**

if two internal robots r', r'' closest to $\overline{v_1a}$ and $\overline{v_1b}$ move toward corner v_1, then our technique guarantees that at least one of r', r'' and v_1 are positioned in CLS_{v_1}.

Each robot $r_i \in Q$ works autonomously having only the information about $\mathbf{C}(r_i)$. If $\mathbf{H}(r_i)$ is not a line segment for each $r_i \in Q$, then the ID and CD procedures start immediately. However, if $\mathbf{H}(r_i)$ is a line segment, then in one round, the procedure we use for a collinear $\mathbf{C}_0$ transforms $\mathbf{C}_0$ into a non-collinear configuration. The ID and CD procedures then run until all robots of Q become corners of **H**.

Overview of the Analysis. The main goal is to show that in each round $\kappa \geq 0$, at least one robot either on any side of **H** or in the interior of **H** becomes a new corner of **H**. This will immediately give the claimed runtime of $O(N)$ for our algorithm. Since robots know N, after all robots in Q become corners of **H**, each robot can decide on its own (without communicating with other robots) Complete Visibility is solved and terminate its computation.

To prove the above claim, we first show that at least one robot in the interior of **H** moves outward toward **H** in each round $\kappa \geq 0$. We then prove that in the same round κ due to the moves of corner robots of **H** inward in **H**, a robot in any side of **H** or in the interior of **H** becomes a new corner of **H**. We also show that the corner robots of **H** remain as corners of **H** even after they have moved inward in **H**. We then prove that this indeed happens without collisions in every round κ. This altogether provides the $O(N)$ runtime for our algorithm avoiding collisions (Theorem 1.1).

REFERENCES

[1] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. 2005. *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, Cambridge, MA.

[2] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. 2012. Distributed Computing by Oblivious Mobile Robots. *Synthesis Lectures on Distributed Computing Theory* 3, 2 (2012), 1–185.

[3] Giuseppe Antonio Di Luna, Paola Flocchini, Federico Poloni, Nicola Santoro, and Giovanni Viglietta. 2014. The Mutual Visibility Problem for Oblivious Robots. In *CCCG*.

[4] Gokarna Sharma, Costas Busch, and Supratik Mukhopadhyay. 2015. Bounds on Mutual Visibility Algorithms. In *CCCG*. 268–274.