Mutual Visibility by Robots with Persistent Memory

Subhash Bhagat and Krishnendu Mukhopadhyaya(✉)

Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India
subhash.bhagat.math@gmail.com, krishnendu@isical.ac.in

Abstract. This paper addresses one of the fundamental geometric formation problems, namely the *mutual visibility* problem, for a set of semi-synchronous, opaque robots occupying distinct positions in the Euclidean plane. Since robots are opaque, if three robots lie on a line, the middle robot obstructs the visions of the two other robots. The mutual visibility problem requires the robots to coordinate their movements to form a configuration, within finite time and without collision, in which no three robots are collinear. We assume that robots are endowed with constant bits of persistent memory. We consider the FSTATE computational model [4] in which the persistent memory is used by the robots only to remember their previous internal states. This piece of information is not communicated or visible to the other robots. Except from this persistent memory, robots are oblivious i.e., they do not carry forward any other information from their previous computational cycles. The paper presents a distributed algorithm to solve the mutual visibility problem for a set of semi-synchronous robots using only 1 bit of persistent memory. The proposed algorithm also provides a self-stabilizing solution to the problem. The algorithm does not impose any other restriction on the capability of the robots and guarantees collision-free movements for the robots.

Keywords: Swarm robots · Mutual visibility problem · Semi-synchronous · Persistent memory · Self-stabilizing

1 Introduction

A *swarm* of robots is a multi-robot system consisting of autonomous, homogeneous, small mobile robots which are capable of carrying out some task in a cooperative environment. The robots are modelled as points on the two-dimensional plane. The robots are indistinguishable by their appearances. All of them have identical capabilities and execute same algorithm. They do not share a global coordinate system; each robot has its own local coordinate system. The directions and orientations of the local coordinate axes may vary. Each robot executes the computational cycles consisting of three phases *Look-Compute-Move*. In *Look*

Y. Chen et al. (Eds.): FAW 2019, LNCS 11458, pp. 144–155, 2019.
https://doi.org/10.1007/978-3-030-18126-0_13

phase, a robot takes the snapshot of its surroundings and maps the locations of the other robots w.r.t. its local coordinate system. In *Compute* phase, a robot uses the information gathered in the *Look* phase to compute a destination point. In *Move* phase, it moves towards the computed destination point.

In persistent memory model, robots are endowed with constant amount of persistent memory (the robots are otherwise oblivious) [1]. This persistent memory can be used in three different ways: (i) the robots can set limited communications between themselves using *visible lights* which can assume a constant number of predefined colors to represent their different states and also to retain some constant amount of information about their previous states or (ii) only to remember information about their last states (FState model) or (iii) the robots can use visible lights only to communicate with other robots in the system and they do not remember the colors of the lights of their last computational cycle (FComm model) [4]. Thus, the persistent memory can be used for communication or for internal memory or for both. In this work, robots use persistent memory only for internal memory.

The *mutual visibility* problem is defined as follows: for a set of robots initially occupying distinct positions in the two dimensional plane, the mutual visibility problem asks the robots to form a configuration, within finite time and without collision, in which no three robots are collinear.

1.1 Earlier Works

The mutual visibility problem was first studied by Di Luna et al. [13]. They presented a distributed algorithm to solve the problem for a set of semi-synchronous oblivious robots. Their approach assumes that the robots have the knowledge of total number of robots in the system. Later, their algorithm was analysed and modified by Sharma et al. [11] to improve the round complexity of the algorithm for fully synchronous robots. Di Luna et al. [12] were the first to study the mutual visibility problem for the robots with persistent memory. They solved the problem for the semi-synchronous robots with 3 colors and for asynchronous robots with 3 colors under one axis agreement. Later, Sharma et al. [10] proved that the mutual visibility problem is solvable using only 2 colors for semi-synchronous robots and using 2 colors for asynchronous robots under one-axis agreement. Sharma et al. [8] proposed a solution to the problem which runs in constant time for a set of asynchronous robots using 47 colors. Bhagat and Mukhopadhyay [9] solved the problem for a set of asynchronous robots using 7 colors. In their solution, each robot moves exactly once. The mutual visibility problem has also been considered under different fault models [5,6,14] and also for *fat robots* [7].

The only solution to the mutual visibility problem for oblivious asynchronous robots has been proposed in [2] under the assumption that the robots have an agreement in one coordinate axis and knowledge of total number of robots in the system. Thus, all the existing algorithms for the mutual visibility problem assumes either persistent memory for both communication and internal memory purpose or axis agreement or the knowledge of total number of robots.

1.2 Our Contribution

This paper studies the *mutual visibility* problem for a set of n semi-synchronous robots in the Euclidean plane. A simple but elegant distributed algorithm has been proposed to solve the problem for a set of robots endowed with a constant amount of persistent memory. The proposed algorithm considers FSTATE model which does not have communication overhead of FCOMM model. The persistent memory is used only to remember information about their previous states. The proposed algorithm does not assume any other extra assumptions like agreement on the coordinate axes or chirality, knowledge of n, rigidity of movements. In spite of these weak assumptions, it is shown that the mutual visibility problem is solvable for a set of semi-synchronous robots using only 1 bit of persistent internal memory. The contribution of this paper has following significance.

- While all the existing solutions of the mutual visibility problem for semi-synchronous robots have considered either knowledge of n or persistent memory for both communication and internal memory purposes (combination of FSTATE and FCOMM model), our approach assumes FSTATE model without knowledge of n (this makes system easily scalable).
- In all the existing solutions for the mutual visibility problem, the convex hull of the initial positions of the robots does not remain invariant. The solution of this work maintains the convex hull of the initial robot positions if all the robots initially do not lie on a single line. Furthermore, in all the works with persistent memory, the robots move even if the robots are completely visible to each other. In our algorithm, if the robots are completely visible to each robot, they do not move.
- In our approach, not all robots move. Only the robots which block the vision of the other robots move. Again, the distances they traverse during their movements are kept as small as needed. These help to provide an energy efficient solution to the problem.
- The proposed algorithm is self-stabilizing. Even if robots start with different states, the algorithm achieves its final goal.
- The solution also provides collision free movements for the robots.
- To the best of our knowledge, this paper is the first attempt to study the mutual visibility problem under FSTATE model.

2 Assumptions and Notations

This paper considers a set of n semi-synchronous point robots in the Euclidean plane. The robots are opaque. However, the visibility range of a robot is unlimited. The robots have no knowledge about the total number of robots in the system. The movements of the robots are *non-rigid*. The robots do not have any explicit communication power. Each robot has 1 bit of internal persistent memory. The 1 bit memory stores information about predefined specific states of the robot. This internal bit does not change automatically and it is persistent. Let $s_i(t)$ be the binary variable which denotes the value stored in the internal

memory of the robot r_i at time $t \in \mathbb{N}$. Except for this persistent memory, the robots are oblivious i.e., they do not remember any other data of their previous computational cycles. Initially all the robots occupy distinct locations and they are stationary.

- **Configurations of the robots:** Let $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ denote the set of n robots. The position of robot r_i at time t is denoted by $r_i(t)$. A configuration of robots, $\mathcal{R}(t) = \{r_1(t), \ldots, r_n(t)\}$, is the set of positions occupied by the robots at time t. $\widetilde{C}$ is the set of all such robot configurations.
  We partition $\widetilde{C}$ into two classes: $\widetilde{C}_L$ and $\widetilde{C}_{NL}$, where $\widetilde{C}_L$ is the collection of configurations in which all the robots in $\mathcal{R}$ lie on a straight line and $\widetilde{C}_{NL}$ consists of configurations in which there exist at least three non-collinear robot positions occupied by the robots in $\mathcal{R}$. We say that a robot configuration $\mathcal{R}(t)$ is in *general position* if no three robot positions in $\mathcal{R}(t)$ are collinear. By $\widetilde{C}_{GP}$, we denote the set of all configurations of $\mathcal{R}$ which are in general position. Clearly $\widetilde{C}_{GP} \subset \widetilde{C}_{NL}$.
- **Measurement of angles:** By an angle between two line segments, if not stated otherwise, we mean the angle which is less than or equal to π.
- **Vision of a robot:** If three robots r_i, r_j and r_k are collinear with r_j lying in between r_i and r_k, then r_i and r_k are not visible to each other. We define the vision, $\mathcal{V}_i(t)$, of robot r_i at time t to be the set of robot positions visible to r_i (excluding r_i). The *visibility polygon* of r_i at time t, denoted by $STR(r_i(t))$, is defined as follows: sort the points in $\mathcal{V}_i(t)$ angularly in anti clockwise direction w.r.t. $r_i(t)$, starting from any robot position in $\mathcal{V}_i(t)$. Then connect them in that order to generate the polygon $STR(r_i(t))$.
- A straight line $\mathcal{L}$ is called a *line of collinearity* if it contains more than two distinct robot positions. A robot occupying a position on $\mathcal{L}$ is termed a *collinear* robot. For a robot r_i, let $\mathcal{B}_i(t)$ denote the set of all lines of collinearity on which r_i is a collinear robot at time $t \in \mathbb{N}$.
- Consider a line of collinearity $\mathcal{L}$ at time t. A robot r_i on $\mathcal{L}$ is called a *non-terminal* robot if $r_i(t)$ is a point between two other robot positions on $\mathcal{L}$. A robot which is not a non-terminal robot is called a *terminal* robot.
- A non-terminal robot position $r_i(t)$ on a line of collinearity $\mathcal{L}$ is called a *junction robot position* if there is another line of collinearity $\mathcal{L}_1$ such that $r_i(t)$ lies at the intersection point between $\mathcal{L}$ and $\mathcal{L}_1$.
- By $\overline{pq}$, we denote the closed line segment joining two points p and q, including the end points p and q. Let (p, q) denote the open line segment joining the points p and q, excluding the two end points p and q. Let $|\overline{pq}|$ denote the length of $\overline{pq}$.
- $\boldsymbol{d_{ij}^k(t)}$: Let $\mathcal{L}_{ij}(t)$ denote the straight line joining $r_i(t)$ and $r_j(t)$. The perpendicular distance of the line $\mathcal{L}_{ij}(t)$ from the point $r_k(t)$ is denoted by $d_{ij}^k(t)$.
- $\boldsymbol{D_i(t)}$: $D_i(t)$ is the minimum distance of any two robot positions in $\{r_i(t)\} \cup \mathcal{V}_i(t)$.

3 Algorithm for the Mutual Visibility Problem

Consider an initial configuration $\mathcal{R}(t_0)$ of robots. If $\mathcal{R}(t_0)$ contains no non-terminal robot, then $\mathcal{R}(t_0) \in \widetilde{C}_{GP}$ i.e., all the robots in the system are visible to each other. On the contrary, if $\mathcal{R}(t_0)$ contains at least one non-terminal robot, then there are at least two robots which are not visible to each other. In this scenario, to achieve complete visibility, robots need to coordinate their movements. In this process one has to decide two main things; (i) which are the robots to move: terminals or non-terminals? (ii) what should be their destination point to move? First, we try to give an intuitive idea to resolve these issues and then we describe our algorithm in details.

3.1 Eligible Robots for Movements

Non-terminal robots block the vision of the other robots. In our approach, we choose non-terminal robots for movements. Since one of our main objectives is to maintain the convex hull of the initial robot positions and the robots lying at the vertices of the convex hull are terminal robots, we do not move terminal robots. A robot can easily determine whether it is a terminal robot or non-terminal robot.

3.2 Different Types of Movements

A robot uses its 1 bit of internal memory to remember the information about its last movement. It uses 0 and 1 in its persistent memory for this purpose. Initially all robots have 0 in their respective persistent memory. If the internal bit is 0, a robot moves not along any line of collinearity and this move is called a *type-0* move. If internal bit is 1, a robot moves along a line of collinearity and this move is called a *type-1* move.

3.3 States of a Robot

A robot uses its persistent 1 bit memory to remember information about its last movement. Initially all robots have 0 in their persistent memory.

- If a robot is terminal and its internal bit is 0, it is a terminal robot since the initial configuration.
- If a robot is terminal and its internal bit is 1, it was a non-terminal robot in the initial configuration and has become terminal during the execution of the algorithm.
- If a robot is non-terminal and its internal bit is 0, it is a non-terminal robot since the initial configuration and either it has made no move or has made a type-1 move.
- If a robot is non-terminal and its internal bit is 1, it is a non-terminal robot since the initial configuration and it has made a type-0 move.

3.4 Computation of Destination Point

Destination points of the robots are computed in such way that (i) they do not create new collinearities by moving to these positions and (ii) the total number of collinear robots in the system should decrease within finite number of movements. The algorithm terminates when system contains no non-terminal robot. Let r_i be an arbitrary non-terminal robot at time $t \geq t_0$. To find the new position of r_i, we first decide on the direction of movement and then the amount of displacement along this direction. While computing the new destination point of r_i, two things should be taken care of. The new position of r_i should not block the visibility of the other robots. The movements of the robots should be collision free. Depending on the current configuration $\mathcal{R}(t)$, the destination point for r_i is computed as follows.

– **Case-1:** $\mathcal{R}(\mathbf{t}) \in \widetilde{\mathbf{C}}_{\mathbf{NL}}$
Consider the set of angles $\Gamma_i(t)$ defined as follows:

$$\Gamma_i(t) = \{\angle r_j r_i r_k : r_j, r_k \text{ are two consecutive vertices on } STR(r_i(t))\}$$

- **The direction of movement:** Let $\alpha_i(t)$ denote the angle in $\Gamma_i(t)$ having the maximum value if the maximum value is less than π, otherwise the 2^{nd} maximum value (tie, if any, is broken arbitrarily). The bisector of $\alpha_i(t)$ is denoted by $Bisec_i(t)$. It is a ray from $r_i(t)$. If persistent bit is 0, r_i makes a type-0 move and its direction of movement is along $Bisec_i(t)$. Before starting its movement, r_i changes its persistent bit to 1. It may be noted that any other suitable direction for type-0 move would work fine for robot r_i. If persistent bit is 1, r_i makes a type-1 move. r_i arbitrarily chooses a line of collinearity from $\mathcal{B}_i(t)$ and moves along this line. Before starting a type-1 move, r_i changes its persistent bit to 0.
- **The amount of displacement:** Let $d_i(t) = minimum\{d^k_{ij}(t), d^j_{ik}(t), d^i_{jk}(t) : \forall r_j, r_k \in \mathcal{V}_i(t))\}$. The amount of displacement of r_i at time t is denoted by $\sigma_i(t)$ and it is defined as follows,

$$\sigma_i(t) = \frac{U}{3^{4v_i(t)}}$$

where $U = minimum\{d_i(t), D_i(t)\}$ and $v_i(t) = |\mathcal{V}_i(t))|$.
Three non-collinear robots become collinear when the triangle formed by their positions collapses to a line. The amount $\sigma_i(t)$ is chosen to be a small fraction of $d^k_{ij}(t)$ for all $r_j(t), r_k(t) \in \mathcal{V}_i(t))$ in order to guarantee that no new collinearity is generated during the movements of the robots. Other suitable values will also work.
- **The destination point:** Let $\hat{r}_i(t)$ be the point on $Bisec_i(t)$ at distance $\sigma_i(t)$ from $r_i(t)$ if $s_i(t) = 0$. Otherwise, $\hat{r}_i(t)$ is a point on a line $\mathcal{L} \in \mathcal{B}_i(t)$ at distance $\sigma_i(t)$ from $r_i(t)$ (choose arbitrarily any one of the two directions along $\mathcal{L}$). The destination point of $r_i(t)$ is $\hat{r}_i(t)$.

– **Case-2:** $\mathcal{R}(\mathbf{t}) \in \widetilde{\mathbf{C}}_{\mathbf{L}}$
  There is only one line of collinearity, say $\hat{\mathcal{L}}$, in the system. Only two robots are terminal. Once one of them moves, the present configuration is converted into a configuration in $\widetilde{C}_{NL}$.
  - **The direction of movement:** Let $\mathcal{L}^*$ be the line perpendicular to $\hat{\mathcal{L}}$ at the point $r_i(t)$. The robot r_i arbitrarily chooses a direction along $\mathcal{L}^*$ and moves along that direction. Let $\mathcal{L}_d^*$ denote the direction of movement of r_i. Since all robots are collinear, this movement is a type-0 move. Before starting this move, r_i changes its persistent bit to 1.
  - **The amount of displacement:** In this, the amount of displacement $\hat{\sigma}_i(t)$ is defined as follows:

$$\hat{\sigma}_i(t) = \frac{D_i(t)}{3^4}$$

  - **The destination point:** Let $\bar{r}_i(t)$ be the point on $\mathcal{L}_d^*$ at the distance $\hat{\sigma}_i(t)$ from $r_i(t)$. The destination point of r_i is $\bar{r}_i(t)$.

3.5 Termination

A robot terminates the execution of algorithm $MutualVisibility()$ when it finds itself as a terminal robot. Thus, an initially terminal robot terminates just in one round.

Robots use the algorithm $ComputeDestination()$ to compute its destination point and use algorithm $MutualVisibility()$ to obtain complete visibility.

3.6 Correctness

To prove the correctness of our algorithm, we need to prove the following for any configuration: (i) three non-collinear robots in a particular round do not become collinear in any of the succeeding rounds (ii) within finite number of rounds at least one non-terminal robot becomes terminal and (iii) movements of the robots are collision free. If three non-collinear robots become collinear, then the triangle formed by their positions should collapse into either a line or a point. For arbitrary three non-collinear robots r_i, r_j and r_k, we prove that none of $d_{ij}^k(t)$, $d_{ik}^j(t)$ and $d_{jk}^i(t)$ becomes zero. Without loss of generality, we prove that $d_{ij}^k(t)$ will never vanish, during the execution of our algorithm. We estimate the maximum decrement in the value of $d_{ij}^k(t)$ in a particular round, due to the movements of the robots.

Lemma 1. *Let r_i, r_j and r_k be three arbitrary robots, which are not collinear at time $t \in \mathbb{N}$. During the rest of execution of algorithm $MutualVisibility()$, they do not become collinear.*

Proof. Maximum decrement in the value of $d_{ij}^k(t)$ occurs when all the three robots move simultaneously in a round. Thus, we suppose the three robots move at time t. Depending upon the positions of the robots, we have the following cases.

– **Case-1: r_i, r_j and r_k are mutually visible at t_0**
According to our approach, the displacement of a robot, in a single movement, is bounded above by $\frac{d_{ij}^k(t)}{3^4}$ (since $|\mathcal{V}_i(t)| \geq 1$). Since all the three robots move simultaneously in a round, the total decrement in the value of $d_{ij}^k(t)$ is bounded above by $\frac{3}{3^4}d_{ij}^k(t)$. It is easy to see that this bound also holds for all other scheduling of the actions of the robots. Thus, we have,

$$d_{ij}^k(t+1) > (1 - \frac{3}{3^4})d_{ij}^k(t) \tag{1}$$

Equation (1) implies that the triangle $\triangle_{ijk}(t)$ does not collapses into a line due to the movements of the robots. Since robots are semi-synchronous and t is arbitrary, these three robots never become collinear during the whole execution of the algorithm.
– **Case-2: r_i, r_j and r_k are not mutually visible at t_0**
In this case the triangle $\triangle_{ijk}(t)$ contains a triangle $\triangle_{xyz}(t)$ such that the robots lying at three vertices r_x, r_y and r_z are mutually visible to each other. Case-1 above implies that the triangle $\triangle_{xyz}(t)$ does not vanish during the movements of the robots and so does $\triangle_{ijk}(t)$. □

Lemma 2. *Let r_i be an initially non-terminal robot. During the execution of algorithm MutualVisibility(), $\exists$ a time $t \in \mathbb{N}$ such that r_i becomes a terminal robot at time t and it remains terminal for the rest of the execution of the algorithm.*

Proof. Let $\mathcal{L}_1(t')$ be a line of collinearity in $\mathcal{B}_i(t')$.

– **Case-1: $\mathcal{L}_1(t')$ does not contain a junction robot position**
In this case, r_i is a non-terminal robot on exactly one line. Since both the end robot positions on $\mathcal{L}_1(t')$ are terminal, it takes at most $2k-1$ number of movements for the non-terminal robots on $\mathcal{L}_1(t')$ to become terminal, where k is number of non-terminal robots on $\mathcal{L}_1(t')$ (Fig. 1).
– **Case-2: $\mathcal{L}_1(t')$ contains a junction robot position**
We consider different possible configurations of the robot positions on the line $\mathcal{L}_1(t')$ and show that in each case r_i becomes a terminal robot. Different scenarios are as follows:
  - We first consider a basic scenario in which (i) $\mathcal{L}_1(t')$ contains exactly one junction robot position $r_k(t')$ and (i) r_k lies exactly on two lines of collinearity. Let $\mathcal{L}_2(t')(\neq \mathcal{L}_1(t'))$ be the other line of collinearity of r_k.
    * Suppose $r_k(t')$ is the only junction robot position on $\mathcal{L}_2(t')$. Then, as in case-1, within finite number of rounds r_k becomes a terminal robot (Fig. 2). Once r_k becomes terminal, again by case-1, the collinearity among the robots initially on $\mathcal{L}_1(t')$ are broken within finite number of rounds and r_i becomes terminal.
    * Suppose $\mathcal{L}_2(t')$ contains another junction robot r_m and r_k and r_m are the only two robots which occupy junction position on $\mathcal{L}_2(t')$. Let $\mathcal{L}_3(t')(\neq \mathcal{L}_2(t'))$ be the line of collinearity on which r_m lies. If r_m lies

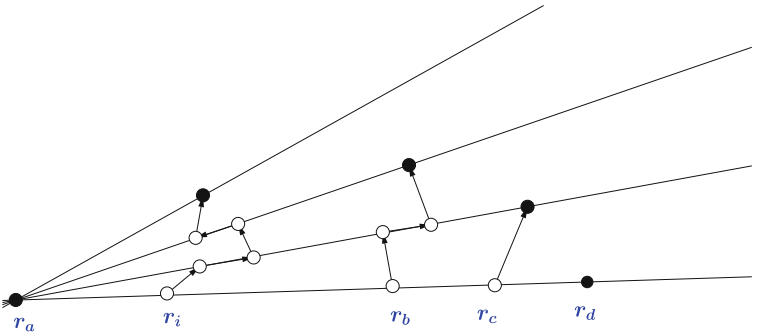

Fig. 1. An illustration of case-1 of Lemma 2: non-terminal robots on a line of collinearity $\mathcal{L}_1(t')$, containing no junction robot position, become terminal within finite number of movements: dark circles are current positions of the robots and white circles are old positions of the robots, the arrows show the directions of movements of the robots

on exactly two lines of collinearity $\mathcal{L}_2(t')$ and $\mathcal{L}_3(t')$ and $\mathcal{L}_3(t')$ does not contain any other junction robot position, by the same arguments as above, within finite number of rounds r_i becomes terminal.

* Suppose $\mathcal{L}_3(t')$ contains another junction robot position. Continuing our arguments as above, we get a sequence $\mathcal{S}$ of lines of collinearity. Since there are finite number of robots, this sequence either ends with a line of collinearity $\mathcal{L}_k(t')$ containing exactly one junction robot position or it contains a *cycle*. If former is true, as above, all the non-terminal robots in this sequence become terminal within finite time. When $\mathcal{S}$ contains a *cycle*, then a type-1 move breaks this *cycle* and r_i becomes terminal within finite time.

Thus, in these basic scenarios within finite number of rounds, r_i becomes terminal.

- Now consider the general scenario, in which a line of collinearity may contain more than two junction robot positions. Starting from $\mathcal{L}_1(t')$, we can get many such sequences of lines of collinearity as described above. Let $\widetilde{\mathcal{S}}$ denote the set of all these sequence. Since the sequences in $\widetilde{\mathcal{S}}$ may have common lines, breaking of collinearities from one line may depend on breaking of collinearities from another line.
  * If no sequence in $\widetilde{\mathcal{S}}$ contains a *cycle*, then type-0 movements i.e., movements not along the lines of collinearity will break all the collinearities in $\widetilde{\mathcal{S}}$.
  * Suppose a sequence in $\widetilde{\mathcal{S}}$ contains a *cycle*, say $\mathcal{C}$. Let r_x be a robot at a critical robot position on a line $\mathcal{L}_u$ in $\mathcal{C}$. If r_x makes a type-1 move along $\mathcal{L}_u$, then r_x does not remain a robot at critical position and cycle $\mathcal{C}$ is broken. Suppose r_x makes a type-1 move along another line of collinearity $\mathcal{L}_v$. If $\mathcal{L}_v$ does not belong to a *cycle*, by case-1, within finite rounds, r_x does not remain non-terminal with the robots on $\mathcal{L}_v$ and when r_x makes a type-1 move along $\mathcal{L}_u$, it breaks the cycle $\mathcal{C}$. If $\mathcal{L}_v$

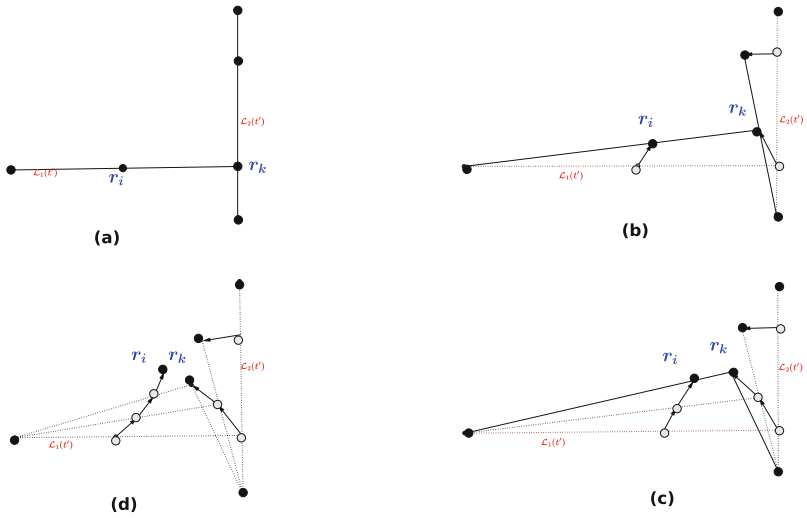

Fig. 2. An illustration of case-2 of Lemma 2: (a) $\mathcal{L}_1(t')$ and $\mathcal{L}_2(t')$ contain exactly one junction robot position $r_k(t')$, (b)-(d) demonstrate a sequence of type-0 movements for the robots to show that r_i becomes terminal within finite number of rounds: dark circles are current positions of the robots and white circles are old positions of the robots, the arrows show the directions of movements of the robots

belongs to a *cycle*, r_x is a robot at a critical robot position on $\mathcal{L}_v$ and a type-1 movement of r_x along $\mathcal{L}_v$ breaks this cycle. Thus, within finite time all the cycles in $\widetilde{\mathcal{S}}$ is broken.

Hence, within finite time, r_i becomes a terminal robot. Since robots are semi-synchronous, by Lemma 1, r_i remains as terminal once it becomes so.

Lemma 3. *The movements of the robots are collision-free.*

Proof. Let r_i and r_j be two arbitrary robots and at least one of them moves. Consider a robot r_k visible to at least one of r_i and r_j. If r_i and r_j collide, then r_i, r_j and r_k would become collinear or remain collinear which contradicts Lemmas 1 and 2. This implies that the movements of the robots are collision free during the whole execution of $MutualVisibility()$.

Lemma 4. *If $\mathcal{R}(t_0) \notin \widetilde{C}_L$, during the whole execution of algorithm $MutualVisibility()$, the convex hull of the robot positions in $\mathcal{R}(t_0)$ remains invariant in size and shape.*

Proof. Let $\mathcal{CH}(t_0)$ denote the convex hull of $\mathcal{R}(t_0)$. The robots occupying the vertices of $\mathcal{CH}(t_0)$ are terminal robots. According to algorithm $MutualVisibility()$, these robots do not move. The robots on the edges of $\mathcal{CH}(t_0)$ move inside the convex hull $\mathcal{CH}(t_0)$ and no robot, lying inside the hull, crosses

any edge of the convex hull (as per definitions of directions of movements and amount of displacement in case-1 of subsection D). Hence, $\mathcal{CH}(t_0)$ remains invariant in size and shape.

Lemma 5. *Algorithm MutualVisibility() is self-stabilizing.*

Proof. Robots use type-1 movements to break the *cycles*. The type-0 movements are used to break the other type of collinearities. In our approach, robots start with type-0 movements. If a robot is not a *critical* robot and starts with a type-1 movement, it would take at most one additional round to become a terminal robot. On the other hand, if a *critical* robot starts with a type-1 movement, it breaks the cycle and would take one round less to become a terminal robot. Thus, our approach works even if robots start with any value in their internal.

From the above lemmas, we have the following theorem:

Theorem 1. *Algorithm MutualVisibility() provides a self-Stabilization solution to the mutual visibility problem without any collision for a set of semi-synchronous, communication-less robots, placed in distinct location, with 1 bit of persistent memory.*

4 Conclusion

This paper presents a self-stabilizing distributed algorithm to solve the mutual visibility problem in finite time for a set of communication-less semi-synchronous robots endowed with a constant amount of persistent memory. The proposed algorithm uses only 1 bit of persistent memory. The robots use their persistent memories only to remember information about their last movements. There is no explicit communication between the robots. The algorithm also guarantees collision-free movements for the robots. The proposed algorithm also maintains the convex hull of the initial robot positions. The results of this paper leave many open questions. How does the internal persistent memory can help to reduce the communication overheads in the existing solutions for the mutual visibility problem, where external lights are used for communicating the internal states of the robots? How to solve the mutual visibility problem for asynchronous robots in this setting? What would be the impact of internal persistent memory in the solutions of other geometric problems?

References

1. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: The power of lights: synchronizing asynchronous robots using visible bits. In: Proceedings of 32nd International Conference on Distributed Computing Systems (ICDCS), pp. 506–515 (2012)
2. Bhagat, S., Chaudhuri, S.G., Mukhopadhyaya, K.: Formation of general position by asynchronous mobile robots under one-axis agreement. In: Kaykobad, M., Petreschi, R. (eds.) WALCOM 2016. LNCS, vol. 9627, pp. 80–91. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30139-6_7

3. Flocchini, P., Prencipe, G., Santoro, N.: Distributed Computing by Oblivious Mobile Robots. Morgan & Claypool, San Rafael (2012)
4. Flocchini, P., Santoro, N., Viglietta, G., Yamashita, M.: Rendezvous of two robots with constant memory. In: Moscibroda, T., Rescigno, A.A. (eds.) SIROCCO 2013. LNCS, vol. 8179, pp. 189–200. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03578-9_16
5. Aljohani, A., Sharma, G.: Complete visibility for mobile robots with lights tolerating faults. Int. J. Netw. Comput. **8**(1), 32–52 (2018)
6. Aljohani, A., Poudel, P., Sharma, G.: Fault-tolerant complete visibility for asynchronous robots with lights under one-axis agreement. In: Rahman, M.S., Sung, W.-K., Uehara, R. (eds.) WALCOM 2018. LNCS, vol. 10755, pp. 169–182. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75172-6_15
7. Sharma, G., Alsaedi, R., Busch, C., Mukhopadhyay, S.: The complete visibility problem for fat robots with lights. In: Proceedings of 19th International Conference on Distributed Computing and Networking (ICDCN 2018), p. 21 (2018)
8. Sharma, G., Vaidyanathan, R., Trahan, J.L.: Constant-time complete visibility for asynchronous robots with lights. In: Spirakis, P., Tsigas, P. (eds.) SSS 2017. LNCS, vol. 10616, pp. 265–281. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69084-1_18
9. Bhagat, S., Mukhopadhyaya, K.: Optimum algorithm for mutual visibility among asynchronous robots with lights. In: Spirakis, P., Tsigas, P. (eds.) SSS 2017. LNCS, vol. 10616, pp. 341–355. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69084-1_24
10. Sharma, G., Busch, C., Mukhopadhyay, S.: Mutual visibility with an optimal number of colors. In: Bose, P., Gąsieniec, L.A., Römer, K., Wattenhofer, R. (eds.) ALGOSENSORS 2015. LNCS, vol. 9536, pp. 196–210. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28472-9_15
11. Sharma, G., Busch, C., Mukhopadhyay, S.: Bounds on mutual visibility algorithms. In: Proceedings of 27th Canadian Conference on Computational Geometry (CCCG 2015) (2015)
12. Di Luna, G.A., Flocchini, P., Gan Chaudhuri, S., Poloni, F., Santoro, N., Viglietta, G.: Mutual visibility by luminous robots without collisions. In: Information and Computation, vol. 254, pp. 392–418 (2017)
13. Di Luna, G.A., Flocchini, P., Poloni, F., Santoro, N., Viglietta, G.: The mutual visibility problem for oblivious robots. In: Proceedings of 26th Canadian Conference on Computational Geometry (CCCG 2014) (2014)
14. Sharma, G.: Mutual visibility for robots with lights tolerating light faults. In: Proceedings of IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 829–836 (2018)