# Limited Visibility Gathering by a Set of Autonomous Mobile Robots

Paola Flocchini* Giuseppe Prencipe† Nicola Santoro‡
Peter Widmayer§

**Keywords:** Distributed Algorithms, Coordination and Control, Mobile Robots.

**Abstract**

We consider a collection of robots which are identical (anonymous), have limited visibility of the environment, and no memory of the past (oblivious); furthermore, they are totally asynchronous in their actions, computations, and movements. We show that, even in such a totally asynchronous setting, it is possible for the robots to gather in the same location in finite time, provided they have a compass.

## 1 Introduction

In current robotics research, both from engineering and behavioral viewpoints, the trend has been to move away from the design and deployment of few, rather complex, usually expensive, application-specific robots. Instead, the interest has shifted towards the design and use of a large number of "generic" robots which are very simple, with very limited capabilities and, thus, relatively inexpensive.

In particular, each robot is only capable of sensing its immediate surrounding, performing computations on the sensed data, and moving towards the computed destination; its behavior is an (endless) cycle of sensing, computing, moving and being inactive (e.g., see [2, 6, 7, 8]). On the other

*School of Information Technology and Engineering, University of Ottawa, flocchin@site.uottawa.ca

†Dipartimento di Informatica - Università di Pisa, Corso Italia, 40 - 56100 - Pisa, e-mail: prencipe@di.unipi.it

‡School of Computer Science, Carleton University, santoro@scs.carleton.ca

§Theoretische Informatik, ETH Zürich, pw@inf.ethz.ch

hand, the robots should be able, together, of performing rather complex tasks. Examples of typical basic tasks are *gathering*, *leader election*, *pattern formation*, *scattering*, etc.

A very important set of questions refer to determining the robots capabilities; that is, how "simple" the robots can be to perform the required task [3]. In computational terms, this question is to identify the factors which influence solvability of a given problem (the task).

These questions have been extensively studied both experimentally and theoretically in the *unlimited visibility* setting, that is assuming that the robots are capable to sense ("see") the entire space (e.g., see [4, 5, 9, 11]).

In general and more realistically, robots can sense only a surrounding with a radius of bounded size. This setting, called the *limited visibility* case, is understandably more difficult, and only few algorithmic results are known [1, 10].

In this paper we are interested in *gathering*: the basic task of having the robots meet in a same location (the choice of the location is arbitrary). Since the robots are modeled as points in the plane, the task of robots gathering is also called the *point formation problem*.

Gathering (or point formation) has been investigated both experimentally and theoretically. In particular, in the limited visibility setting, Ando et al. [1] presented a gathering algorithm for indistinguishable robots which are placed on a plane without any common coordinate system; their algorithm does not require the robots to remember observations nor computations performed in the previous steps. Their result implies that gathering can be performed with limited visibility by very simple robots: *anonymous*, *oblivious* and *disoriented*.

Their solution, however, is based on a very strong "atemporal" assumption on the duration of the robots' actions: their robots must be capable in every cycle to perform all the sensing, computing and moving *instantaneously*.

This assumption has many consequences crucial for its correctness. For example, since movement is instantaneous, a robot can *not* be seen by the others while moving (and its temporary position mistaken for a destination location); since sensing and computing is instantaneous, a robot always has available the correct current situation of its neighborhood.

Note that, since instantaneous movement is not physically realizable, their solution is only of theoretical interest.

In this paper, we study the gathering problem in the most general case of an *asynchronous* system of robots with limited visibility, where both their computations and their movement requires a *finite* but otherwise unpre-

dictable amount of time.

The question motivating our investigation is whether point formation is possible in such a system. Since in these systems gathering is *unsolvable* if the robots are disoriented (i.e., have no common system of coordinates), we shall restrict ourselves to systems with sense of direction (i.e., the robots share the same coordinate system).

In this paper we show that indeed anonymous oblivious robots with limited visibility can gather within a finite number of moves even if they are fully asynchronous. In fact, we describe a new algorithm for solving the point formation problem in the *asynchronous* setting by *anonymous*, *oblivious* robots with limited visibility. We then prove its correctness showing that the robots will gather in a point within a finite amount of time. This result holds not only allowing each activity and inactivity of the robots to be totally unpredictable (but finite) in duration, but also making their movement towards a destination unpredictable in length (but not infinitesimally small).

In other words, we show that gathering can be performed by simpler robots with fewer restrictions than known before, provided they have a common coordinate system.

From a theoretical point of view, this result proves that, with respect to the gathering problem, "sense of direction" has the same computational power as "instantaneous actions".

From a practical point of view, this result has fundamental consequences. In fact, it allows to substitute a theoretically interesting but physically unrealizable motorial and computing capability requirement (instantaneous actions) with a property (sense of direction) which is both simple and inexpensive to provide (e.g., by a compass).

The paper is organized as follows. In Section 2 the model under study is formally presented. In Section 3 the notations used in the paper and some useful geometric lemmas are introduced. The gathering algorithm is described in Section 4, and in Section 5 its correctness is proven.

## 2 The Model

We consider a system of autonomous mobile robots. Each robot is capable of sensing its immediate surrounding, performing computations on the sensed data, and moving towards the computed destination; its behavior is an (endless) cycle of sensing, computing, moving and being inactive.

The robots are modeled as units with computational capabilities, which

are able to freely move in the plane. They are equipped with sensors that let each robot observe the positions of the others with respect to its local coordinate system. Each robot is viewed as a point, and can see only a portion of the plane; more precisely, it can observe whatever is at most at a fixed distance $V$ from it (*limited visibility*).

The robots are fully *asynchronous*. In particular, the amount of time spent in observation, in computation, and in movement, is finite but otherwise unpredictable.

Each robot has its own *local view* of the world. This view includes a local Cartesian coordinate system with origin, unit of length, and the *directions* of two coordinate axes, together with their *orientations*, identified as the positive and negative sides of the axes. In this paper we assume that the robots share the same coordinate system (*sense of direction*); however, they do not necessarily agree on the location of the origin (that we can assume, without loss of generality, to be placed in the view of a robot in its own current position), nor on the unit distance.

The robots are *oblivious*, meaning that they do not remember any previous observation nor computations performed in the previous steps.

The robots are *anonymous*, meaning that they are a priori indistinguishable by their appearances, and they do not have any kind of identifiers that can be used during the computation. Moreover, there are no explicit direct means of communication.

Summarizing, the robots are asynchronous, oblivious, anonymous, and with limited visibility; they do however have a common coordinate system.

They execute the same deterministic algorithm, which takes as input the observed positions of the robots within the visibility radius, and returns a destination point towards which the executing robot moves.

A robot is initially in a *waiting* state (*Wait*); at any point in time, asynchronously and independently from the other robots, it *observes* the environment in its area of visibility (*Look*), it *calculates* its destination point based only on the current locations of the observed robots (*Compute*), it then *moves* towards that point (*Move*), and goes back to a waiting state. The sequence: *Wait* ($\mathbb{W}$) - *Look* ($\mathbb{L}$) - *Compute* ($\mathbb{C}$) - *Move* ($\mathbb{M}$) will be called a *computation cycle* (or briefly *cycle*) of a robot. The operations performed by the robots in each state will be now described in more details.

1. **Wait** The robot is idle. A robot cannot stay infinitely idle (see Assumption A1 below).

2. **Look** The robot observes the world by taking a snapshot of the positions of all other robots with respect to its local coordinate system. Each

robot $r$ is viewed as a point, and therefore its position in the plane is given by its coordinates[1].

3. **Compute** The robot performs a *local computation* according to its deterministic, oblivious algorithm. The result of the computation can be a destination point or `do_nothing()`.

4. **Move** If the result of the computation was `do_nothing()`, the robot does not move; otherwise, it moves towards the computed destination of an unpredictable amount of space, which is however assumed to be neither infinite, nor infinitesimally small (see Assumption A2 below). Hence, the robot can only go towards its goal along a curve, but it cannot know how far it will go in the current cycle, because it can stop anytime during its movement.

In the model there are two basic assumptions which ensure finitness:

**Assumption A1(Cycle Time)** The time for a robot to complete its cycle is neither infinite nor infinitesimally small (i.e., is finite and bounded from below).

**Assumption A2(Minimal Distance)** For each robot $r$, there exists an arbitrary small constant $\delta_r > 0$, representing the minimum distance it travels in the Move state, when the result of the computation is not `do_nothing()`; if the computed destination point is closer than $\delta_r$, $r$ will reach it. Moreover, we define $\delta = \min_r \delta_r$.

As a consequence, the (global) time that passes between two successive movements of the same robot is finite; furthermore, there is no assumption on the maximum distance a robot can travel before observing again (apart from the bound given from the destination point that has to be reached). The only assumption made is that there is a lower bound on such distance: when a robot $r$ moves, if it does not reach its destination, it moves at least some positive, small constant $\delta_r$.

Let us stress that this model with the above assumptions makes the environment *totally asynchronous*: there is no common notion of time; a robot observes the environment at unpredictable time instants; no assumptions on the cycle time of each robot, nor on the time each robot elapses to execute

[1] We do not require the robots to be able to detect *multiplicity* (i.e. whether there is more than one robot on any of the observed points, included the position where the observing robot is).

each phase of a given cycle are made. Thus, each robot can take its own time to compute, or to move towards some point in the plane: in this way, it is possible to model different computational and motorial speeds of the units.

The most important consequence is *asynchronicity within a cycle* [4]: every robot can be seen by other robots *while* it is moving. This feature renders more difficult the design of an algorithm to control and coordinate the robots. An additional complication is due to the unpredictable delay between an observation by a robot and the corresponding move. It is in fact possible that, by the time the robot finally moves, the other robots have in the meanwhile moved, making the environment not "coherent" with the observed one.

# 3 Notations and Geometric Lemmas

We first define sets related to which state a robot is at a given time during the computation.

$\mathbb{W}(t)$ **and** $\mathbb{L}(t)$ are the set of all the robots that are respectively in state $W$ and $L$ at time $t$.

$\mathbb{C}(t) = \mathbb{C}_{\emptyset}(t) \cup \mathbb{C}_{+}(t)$ is the set of all the robots that at time $t$ are computing. The set $\mathbb{C}_{\emptyset}$ contains those robots whose computation's result is to stay still (we say that they execute a *null movement*), while $\mathbb{C}_{+}$ contains those robots whose computation's result is some destination point (we say that they will execute a *real movement*).

$\mathbb{M}(t) = \mathbb{M}_{\emptyset}(t) \cup \mathbb{M}_{+}(t)$ is the set of all the robots that at time $t$ are executing a movement. The set $\mathbb{M}_{\emptyset}(t)$ contains the robots executing a *null movement* (they stay still); $\mathbb{M}_{+}(t)$ contains those executing a *real movement* (they are effectively moving towards a destination).

We define *circle of visibility* $\mathcal{C}_i(t)$ of a robot $r_i$ at time $t$ the circle of radius $V$ centered in $r_i$, if $r_i \in \mathbb{L}(t)$. Otherwise $\mathcal{C}_i(t) = \mathcal{C}_i(t')$, where $t' = \max\{\bar{t} | r_i \in \mathbb{L}(\bar{t})\}$. In other words, if a robot is *Observing*, its circle of visibility is the circle of radius $V$ centered in itself; otherwise, it is the circle of radius $V$ centered in the location of its most recent *Look phase*. Where no ambiguity arises, the parameter $t$ in $\mathcal{C}_i(t)$ will be omitted.

We now introduce some notations and geometrical lemmas which will be needed later. Let $A$ and $B$ be two points; with $\overline{AB}$ we will indicate the

segment starting in $A$ and terminating in $B$. When no ambiguity arises we will also use the notation $\overline{AB}$ to denote the length of such a segment. Let $A$ and $B$ be two points on a circle and not on a diameter; with $arc(AB)$ we indicate the smallest arc on the circle passing through $A$ and $B$; moreover, $sector(AB)$ indicates the region limited by the segment $A\overline{AB}$ and $arc(AB)$. In general, $r$ indicates a generic robot in the system (when no ambiguity arises, $r$ is used also to represent the point in the plane occupied by robot $r$); capital greek letters will represent vertical axes; capital italic letters indicate regions (e.g. $\mathcal{L}$, $\mathcal{R}$); given a region, we denote by $|\cdot|$ the number of robots in that region. All other single letters indicate points in the plane.

**Lemma 3.1.** *Every internal chord of a general triangle has length less or equal to the longest side of the triangle.*

**Lemma 3.2.** *Let $Q$ be a convex quadrilateral. If all the sides and the two internal diagonals have length less or equal to $V$ then every internal chord of $Q$ is less or equal to $V$.*

**Lemma 3.3.** *Let $\overline{OB}$ be the radius of a circle centered in $O$ and $D$ be a point on the circle such that $B\widehat{O}D = \beta$, with $0 \leq \beta \leq 90°$. Then $\overline{pC} \leq \overline{BC}$, $\forall p \in arc(BD)$ and $\forall C \in \overline{OD}$ (see figure 1.b).*

**Proof.** Let $p \in arc(BD)$ and $\beta_p$ the angle $p\widehat{O}C$. from the cosine theorem we have:

$$\overline{pC}^2 = \overline{OC}^2 + \overline{OB}^2 - 2\overline{OC} \cdot \overline{OB} \cos \beta_p.$$

Since $0 \leq \beta_p \leq \beta \leq 90°$, we have that $\cos \beta_p \geq \cos \beta$. Thus,

$$\overline{pC}^2 = \overline{OC}^2 + \overline{OB}^2 - 2 \cdot \overline{OC} \cdot \overline{OB} \cos \beta_p \leq \overline{OC}^2 + \overline{OB}^2 - 2\overline{OC} \cdot \overline{OB} \cos \beta = \overline{BC}^2,$$

and the theorem follows. □

# 4 The Algorithm

Let us call *Universe* ($\mathfrak{U}$) the smallest isothetic rectangle containing the initial configuration of the robots and let us call *Right* and *Bottom* respectively, the rightmost and the bottom most side of $\mathfrak{U}$.

The idea of the algorithm is to make the robots move either towards the bottom or towards the right of the Universe (a robot will never move up or to its left), in such a way that, after a finite number of steps, they will gather at the bottom most lower most corner of the Universe.

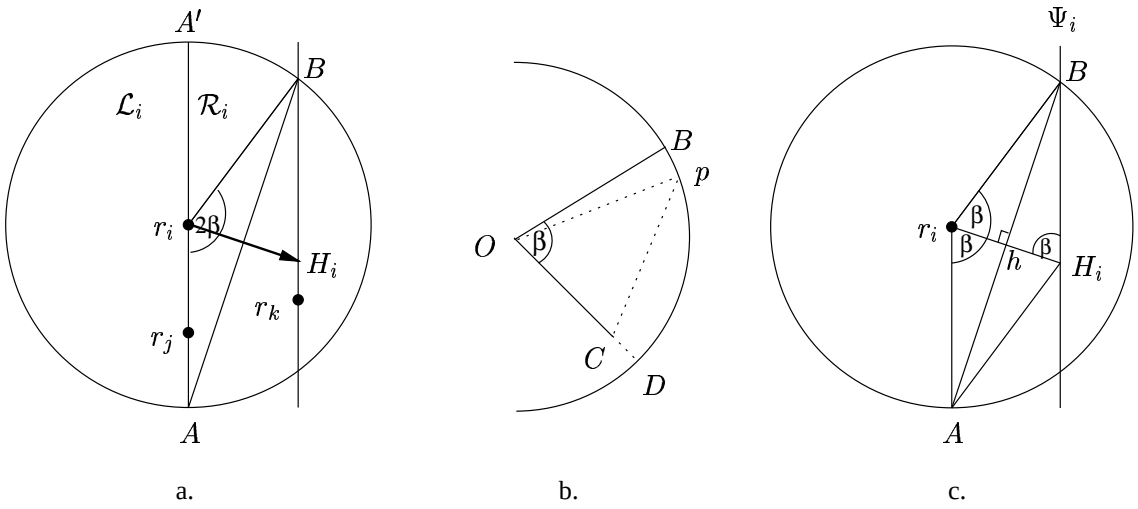


Figure 1: (a) The Notation used in Algorithm 1; (b) Lemma 3.3; (c) Lemma 5.4.

A robot $r$ can move only if it does not see any robot neither to its left nor above on its vertical axis. Several situations could arise depending on the positions of the robots in its area of visibility:

- If $r$ does not see any robot, it does not move;
- If $r$ sees robots only below on its vertical axis, it moves down towards the nearest robot;
- If $r$ sees robots only to its right, it moves horizontally towards the vertical axis of the nearest robot
- If $r$ sees robots both below on its axis and on its right, it computes a destination point and performs a diagonal move towards the right.

Recall that $\mathcal{C}_i$ is the circle of visibility of robot $r_i$. Let $\overline{AA'}$ be the vertical diameter of such region; let $\mathcal{R}_i$ and $\mathcal{L}_i$ denote the regions to the right and to the left of $r_i$, respectively (see Figure 1). Let $S_p = \overline{r_i A'}$ and $S_o = \overline{r_i A}$.

**Algorithm 1 (General).**

$Extrem := (|\mathcal{L}_i| = 0 \wedge |S_p| = 0)$;

**If** I am $\neg Extrem$ **Then**
  `Do_nothing();`
**Else**
  **If** $(|\mathcal{R}_i| = 0 \wedge |S_o| = 0)$ **Then**
    `Do_nothing();`
  **If** $|\mathcal{R}_i| = 0$ **Then**
    $r_j :=$ nearest visible robot on $S_o$;
    `Move(`$r_j$`).`
  **If** $(|\mathcal{R}_i| \neq 0 \wedge |S_o| = 0)$ **Then**
    $\Psi_i :=$ `Nearest();`
    $H_i :=$ `H_Destination(`$\Psi_i$`);`
    `Move(`$H_i$`).`
  **If** $|\mathcal{R}_i| \neq 0$ **Then**
    $\Psi_i :=$ `Nearest();`
    `Diagonal_Movement(`$\Psi_i$`).`

`Nearest()` returns the vertical axis on which the robot in $\mathcal{R}_i$ with the nearest axis to $r_i$ lies.

`H_Destination(`$\Psi_i$`)` returns the intersection between $\Psi_i$ and a line parallel to the $x$ direction and passing through $r_i$.

`Move(p)` terminates the local computation of the calling robot and moves it towards $p$.

In the last case of the Algorithm 1, $r_i$ sees somebody below it and somebody to its right, therefore, to avoid losing some robots, it has to move diagonally, as indicated by the following routine.

**Algorithm 2** (`Diagonal_Movement(`$\Psi_i$`)`).

1: $B :=$ upper intersection between $\mathcal{C}_i$ and $\Psi_i$;
2: $A :=$ point on $S_o$ at distance $V$ from me;
3: $2\beta = A\widehat{r_i}B$;
4: **If** $\beta < 60°$ **Then**
5:  $B :=$ `Rotate(`$r_i, B$`).`
6: $H_i :=$ `D_Destination(`$V, \Psi_i, A, B$`)`;
7: `Move(`$H_i$`).`

`Rotate(`$r_i, B$`)` rotates the segment $\overline{r_iB}$ in such a way that $\beta = 60°$ and returns the new position of $B$.

With `D_Destination(`$V, \Psi_i, A, B$`)`, $r_i$ computes its destination in the following way: the direction of its movement is given by the perpendicular to the segment $\overline{AB}$; $H_i = \min\{$ $V$, the distance of $\Psi_i$ according the direction of movement $\}$.

# 5 Correctness

In this Section we prove the correctness of the algorithm by first showing that the robots which are mutually visible at any point of the computation, will stay mutually visible until the end of the computation, and concluding that at the end of the computation all robots will gather in one point. We first introduce some lemmas.

From Assumptions A1 and A2 it directly follows that:

**Lemma 5.1.** *Let $r_i$ and $r_j$ be two generic robots and let $t$ and $t' > t$ two moment of the computation. If $r_i \in L(t)$, $r_i \in L(t')$, $r_j \in M(t)$, $r_j \in M(t')$, $r_j \in \mathcal{C}_i(t)$ and $r_j \in \mathcal{C}_i(t')$, then $r_j$ can not be in the same point in $t$ and $t'$.*

Moreover, from the algorithm it follows that:

**Lemma 5.2.** *Let $r_j$ and $r_i$ two arbitrary robots, with $r_i$ to the right of $r_j$ at time $t$. If $r_j \in L(t)$ and $\overline{r_j r_i} \leq V$, then $r_j$ can not pass $r_i$ in one step.*

Let us consider a generic robot $r_i$ executing the algorithm. Let $\beta$ be the angle between the vertical axis of $r_i$ and the direction of its movement ($A\widehat{r_i}H_i$ in Figure 1.c).

**Lemma 5.3.** *The segment $\overline{r_i H_i}$ is always smaller or equal to $V$.*

**Proof.** It follows from the fact that $\overline{r_i H_i} = 2V cos\beta$ and that $\beta \geq 60$ by construction. □

**Lemma 5.4.** *$\overline{BH_i} = \overline{AH_i} = V$ and $\overline{pH_i} \leq V$, $\forall\ p \in \overline{r_i A}$.*

**Proof.** By construction, we have that $\overline{r_i B} = V$ (step 2 of Algorithm 2). Since the triangle $\triangle(A, r_i, B)$ is isoscele and $\overline{r_i h}$ is its height, then $A\widehat{r_i}h = \beta$. As a consequence, $r_i\widehat{H_i}B = A\widehat{r_i}h = \beta$ and, thus, also the triangle $\triangle(r_i, B, H_i)$ is isoscele and $\overline{BH_i} = \overline{r_i B} = V$.

Moreover, we know that $\overline{Ah} = \overline{hB}$, since $\triangle(A, r_i, B)$ is isoscele and $\overline{r_i h}$ is perpendicular to $\overline{AB}$, $h\widehat{H_i}B = \beta$ and $\overline{BH_i} = V$. So we can derive:

$$\overline{AH_i} = \sqrt{\overline{hH_i}^2 + \overline{Ah}^2} = \sqrt{(\overline{BH_i} \cdot \cos\beta)^2 + \overline{hB^2}} = \sqrt{V^2 \cdot \cos^2\beta + V^2 \cdot \sin^2\beta} = V.$$

The second part of the statement follows by Lemma 5.3 and Lemma 3.1 on $\triangle(r_i, A, H_i)$. □

Thus, $\diamond(A, r_i, B, H_i)$ is a parallelogram. We now introduce the definition of *visibility graph*.

The *visibility graph* $G = (N, E)$ of the robots is a graph whose node set $N$ is the set of the input robots and, $\forall r_i, r_j \in N$, $(r_i, r_j) \in E$ iff $r_j$ and $r_j$ are initially at distance smaller than the visibility radius $V$.

We first show that the visibility graph must be connected in order for the algorithm to be correct.

**Lemma 5.5.** *If the visibility graph $G$ is disconnected, the problem is unsolvable.*

**Proof.** Let us assume, by contradiction, that there exists a deterministic algorithm $\mathcal{A}$ that solve the problem even if $G$ is disconnected. Let us assume that: there are two robots in input, say $r$ and $r'$; $\mathcal{C}_r \cap \mathcal{C}_{r'} = \emptyset$; the robots move always in a synchronous way; the algorithm $\mathcal{A}$ allow them to gather in a point $d$.

Since the algorithm is deterministic and both $r$ and $r'$ see the world in the same way (namely no other robot is in their circle of visibility), they will always move in the same direction and the distance between them will never decrease. Therefore they can not gather in $d$, having a contradiction. □

Thus, in the following we will always assume that $G$ is connected.

## 5.1 Preserved Visibility

In this Section we prove that the visibility graph is preserved during the entire execution of the algorithm. We prove so by introducing the notion of mutual visibility and by showing that the robots which are connected in the visibility graph (i.e., those which are initially within distance $V$) will eventually become mutually visible, and that two robots that are mutually visible at some point in the algorithm will stay mutually visible until the end of the computation.

Informally speaking, we say that two robots are mutually visible if each robot includes the other one in its computation, namely each of them had seen the other one during its observation phase. Formally, two robots $r_1$ and $r_2$ are *mutually visible* at time $t$ iff

- $r_1 \in (\mathbb{L}(t) \cup \mathbb{C}_\emptyset(t) \cup \mathbb{M}_\emptyset(t)) \wedge r_2 \in \mathcal{C}_1(t) \wedge r_2 \in (\mathbb{W}(t) \cup \mathbb{L}(t))$, or
- $r_2 \in (\mathbb{L}(t) \cup \mathbb{C}_\emptyset(t) \cup \mathbb{M}_\emptyset(t)) \wedge r_1 \in \mathcal{C}_2(t) \wedge r_1 \in (\mathbb{W}(t) \cup \mathbb{L}(t))$.

Since all the robots at the beginning are in $W$, from the above definition we have that the robots that at the beginning are within distance $V$ will become mutually visible in finite time. That is, the following lemma holds:

**Lemma 5.6.** *Let $r_i$ and $r_j$ be two robots that at the beginning are within distance $V$. Robots $r_i$ and $r_j$ will become mutually visible in a finite number of steps.*

**Proof.** Let us assume that $r_i$ enters its first look phase before $r_j$, that is $r_i \in \mathbb{L}(t)$ and $r_j \in \mathbb{W}(t)$, for some $t > 0$. Since they are within distance $V$, the lemma follows. The same argument applies if $r_j$ enters its look phase before $r_i$ or if they look the first time at the same moment. □

We now introduce a couple of Lemmas which will be useful to prove that mutually visible robots will stay so until the end of the algorithm. Let $r_i$ be a generic robot on an axis $\Gamma$. Let $\Gamma'$ and $\Gamma''$ be two vertical axes to the right of $\Gamma$. We will denote by $\overline{\Gamma\Gamma'}$ and $\overline{\Gamma\Gamma''}$ the distances between the corresponding axis.

Then we have:

**Lemma 5.7.** $\overline{\Gamma\Gamma'} < \overline{\Gamma\Gamma''} \Leftrightarrow \beta_{\Gamma'} > \beta_{\Gamma''}$, *where $\beta_{\Gamma'}$ and $\beta_{\Gamma''}$ are respectively the angles computed by the routines* `Diagonal_Movement(`$\Gamma'$`)` *and* `Diagonal_Movement(`$\Gamma''$`)` *(refer to Figure 2.a).*

**Proof.** Let $A$ be a point on $\Gamma$ at distance $V$ below $r_i$. Note that upper intersection between $\mathcal{C}_i$ and $\Gamma'$, say $B'$, is higher than the intersection between $\mathcal{C}_i$ and $\Gamma''$, say $B''$, thus $\overline{AB''}$ is always to the right of $\overline{AB'}$. The lemma follows from the way the final destinations respectively on $\Gamma'$ and $\Gamma''$ are computed by the routine `D_Destination(`$\cdot$`)` called by `Diagonal_Movement(`$\cdot$`)`. □

**Lemma 5.8.** *Let us consider the situation depicted in Figure 2.b), where $r_i$ is a generic robot, $F$ is a point at distance $\leq V$ from $r_i$ on its axis (with $F \neq r_i$), $H_i$ is the destination point of $r_i$. Let $\overline{ps}$ be a segment in $\triangle(F, M, K)$, with $s$ to the right of $p$, and $s'$ the projection of $s$ over $\overline{r_i H_i}$. Then we have*

$$\overline{ll'} \leq V, \quad \forall\ l\ \in \overline{ps},\ \forall\ l' \in \overline{s'H_i}.$$

**Proof.** We distinguish three cases.

*Case i (Figure 2.b): both $p$ and $s$ are below their projections on $\overline{r_i H_i}$*

By Lemma 5.3, we have

$$\overline{s'H_i} \leq \overline{r_i H_i} \leq V. \tag{1}$$

Moreover, let $s''$ be the projection of $s$ on $\overline{FM}$. Since

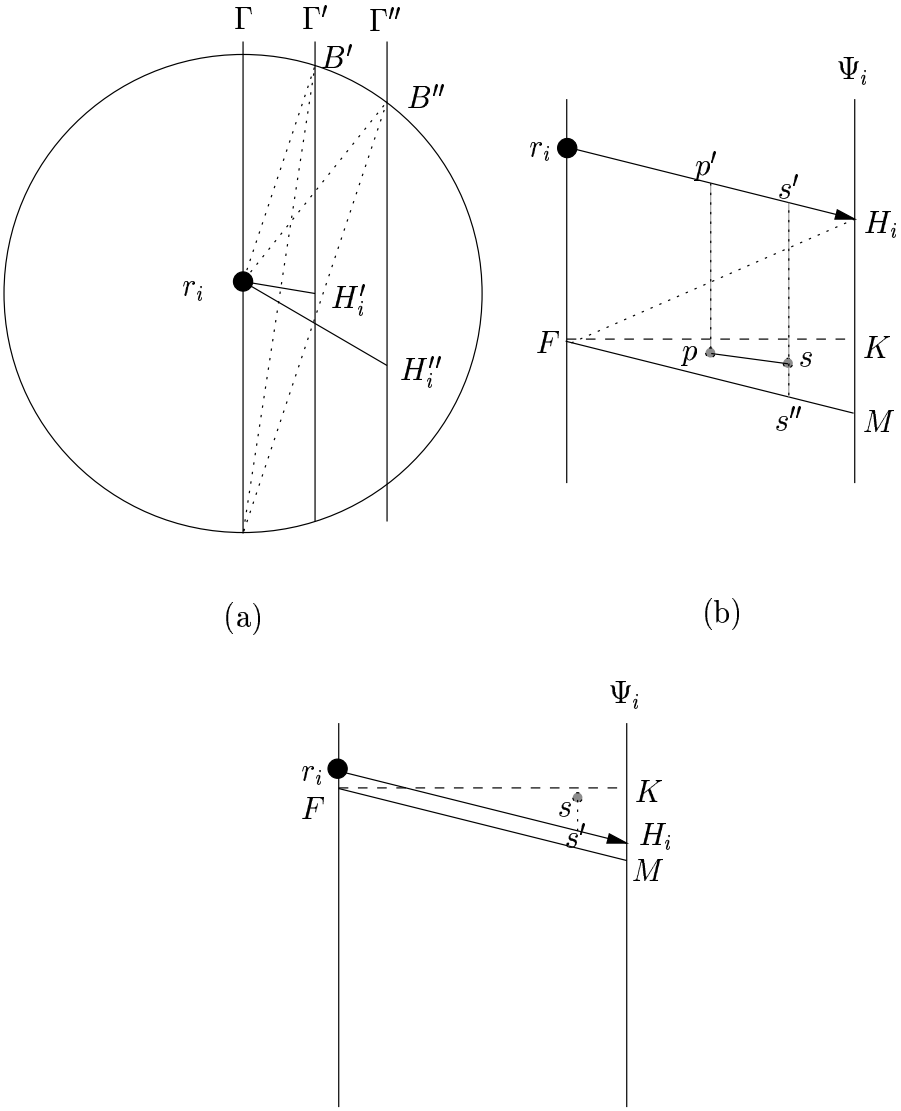


Figure 2: (a) Lemma 5.7; (b) and (c) Lemma 5.8.

- $\overline{FH_i} \leq V$ (by Lemma 5.4),
- $\overline{H_iM} = \overline{r_iF} \leq V$, and
- $\overline{FM} = \overline{r_iH_i} \leq V$ (by Lemma 5.3),

by Lemma 3.1 on $\triangle(F, H_i, M)$, we can conclude that

$$\overline{s''H_i} \leq V, \tag{2}$$

$$\overline{pH_i} \leq V, \tag{3}$$

and that $\overline{FK} \leq V$. By applying again Lemma 3.1 on $\triangle(F, K, M)$, we have that

$$\overline{ps''} \leq V. \tag{4}$$

Moreover, from Lemma 5.3 it follows that:

$$\overline{s's''} \leq \overline{r_iF} \leq V. \tag{5}$$

Let $p'$ the projection of $p$ on $\overline{r_iH_i}$. Since $\overline{p'p} \leq \overline{r_iF} \leq V$, by (3) and Lemma 3.1 on $\triangle(p', p, H_i)$, we have

$$\overline{ps'} \leq V. \tag{6}$$

By (1), (2), (3), (4), (5), (6), and Lemma 3.2 on $\diamond(s', p, s'', H_i)$ the proof follows.

*Case ii (Figure 2.c): $s$ is above $s'$*

Since $\overline{r_iH_i}$ is parallel to and above $\overline{FM}$, and $r_i \neq F$, we clearly have that both $s'$ and $H_i$ are inside $\triangle(F, K, M)$. Since by construction also $p$ is inside this triangle, the lemma follows by Lemma 3.1.

*Case iii (Figure 2.c): $p$ is above $p'$*

Since $p'$ must be inside $\triangle(F, K, M)$, we have that also $s'$ and $H_i$ must be inside it, and the Lemma follows by Lemma 3.1 on $\triangle(F, K, M)$.

□

We are now ready to show that, as soon as two robots becomes mutually visible, they will stay mutually visible. We first prove that this property

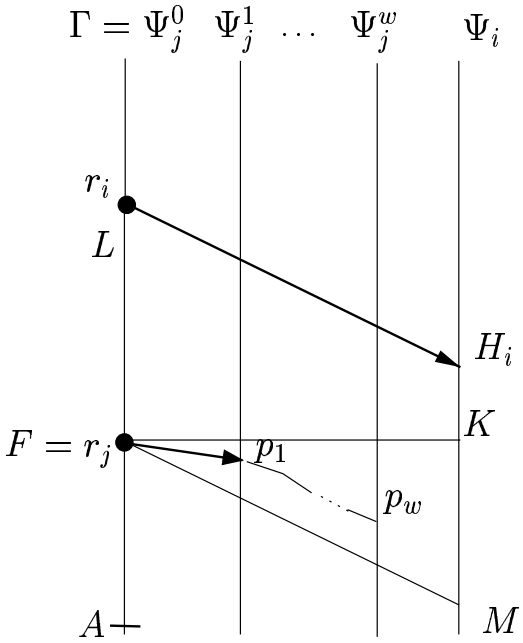


Figure 3: Case ii of Lemma 5.9.

holds when two mutually visible robots lie on the same vertical axis; and then we prove that it holds for two robots lying on different vertical axes.

In the next lemma we will refer to the notation introduced in Figure 1.a and Lemma 5.8.

**Lemma 5.9.** *Let $r_i$ and $r_j$ be robots which are mutually visible at time $t$. Moreover, let they lie, at time $t$, on the same vertical axis with $r_j$ being below $r_i$. There is a time $t' > t$ when $r_i$ and $r_j$ are mutually visible. Moreover, between $t$ and $t'$ $\overline{r_i r_j} \leq V$.*

**Proof.** Let us first consider the case when $\mathcal{R}_i$ is empty. In such a case, $r_i$ would clearly move towards $r_j$ (shortening their distance), while $r_j$ would not move. Since by Algorithm 1 $r_i$ can not pass $r_j$, the first time $r_i$ stops while it is moving towards $r_j$ the mutual visibility definition holds, and the lemma follows.

Let us now consider the more interesting case when $\mathcal{R}_i$ is not empty. In the following we shall consider several situations:

*Case i: $r_j$ does not look until $r_i$ reaches its destination $H_i$.*

We have that $r_i \in W$ while $r_i$ is moving towards $H_i$. Since $\overline{AH_i} = V$ (Lemma 5.4) and $\overline{r_i H_i} \leq V$ (Lemma 5.3), we have that, by Lemma 3.1 on $\triangle(r_i A H_i)$, the distance between $r_i$ and $r_j$ is always $\leq V$ while $r_i$ is moving. Therefore, the first time $r_i$ stops along its path (at most on $H_i$), the mutual visibility definition applies and the lemma follows.

*Case ii: $r_j$ looks while $r_i$ is moving towards its destination $H_i$.* Since $r_i$ is on $r_j$'s right, $r_j$ can not perform a Vertical Move. Hence, $r_j$ can either decide not to move (because it sees some robots above ) or to move. In the first case the proof reduces to the one of Case *i*. On the other hand, $r_j$ can decide to move after having looked. From Case *i* we know that $r_j$ can see $r_i$ on its right. Moreover, it might also see some other robots below it, that can be either on the same axis ($r_j$ perform a Diagonal Move) or not ($r_j$ performs an Horizontal Move). The following applies to both situations.

Let us call $\Psi_j^w$ the $w^{th}$ axis, counting from $\Gamma$, from where $r_j$ looks while $r_i$ is still on its way towards $H_i$, and $p_w$ the points on this axis from where $r_j$ performs the *look* phases. Clearly $\Psi_j^0 = \Gamma$ and $F = p_0$ coincides with the position of $r_j$ on $\Gamma$. In the following we will prove by induction that

**a.** $\Psi_j^w$ is to the left of $\Psi_j^{w+1}$,

**b.** The destination point $d_{w+1}$ that $r_j$ computes when it is on $\Psi_j^w$ is inside $\triangle(F, K, M)$,

**c.** $\overline{p_{w+1}r_i} \leq V$, and $\Psi_j^{w+1}$ is to the left of $r_i$.

**Basis** Let $d_1$ be the first destination point $r_j$ computes. Since $r_i$ is on its right, $r_j$ can only decide to perform a Diagonal Movement, therefore $d_1$ must be to the right of $\Psi_j^0$, and as a consequence $\Psi_j^0$ is to the left of $\Psi_j^1$. Moreover, by Lemma 5.7 we know that $\overline{r_j d_1}$ must lie above $\overline{r_j M}$, hence $p_1$ (that is on $\overline{r_j d_1}$) must be whithin $\triangle(F, K, M)$. Finally, $r_j$ can see $r_i$ by hypothesis and at the beginning $\Psi_j^0$ is to the left of $r_i$, and the basis of the induction follows.

**Inductive Step** Let us assume that all the statements are true for $1, \ldots, w$. Since by inductive hypothesis $\Psi_j^w$ is to the left of $r_i$ and $r_j$ can see $r_i$ from $\Psi_j^w$, $r_j$ can only decide to perform a Diagonal Movement, therefore $d_{w+1}$ must be to the right of $\Psi_j^w$ and can not be after $r_i$ (because of how `Diagonal_Movement(·)` works), and, as a consequence, $\Psi_j^w$ is to the left of $\Psi_j^{w+1}$, and a. follows.

Moreover, since $\overline{\Psi_j^w \Psi_i} < \overline{\Gamma \Psi_i}$ and , by Lemma 5.7, we have that $\overline{d_w p_{d+1}}$ must be above $\overline{FM}$ but cannot be above $\overline{FK}$ (because the algorithm does not allow "up" movements). Therefore the point b. follows.

Furthermore, since b. holds and $\Psi_j^{w+1}$ can not be after $d_{w+1}$, by Lemma 5.8 c. follows, and the induction is proved.

Now we know that all the stop $r_j$ does while $r_i$ is moving towards $H_i$ are inside $\triangle(F, K, M)$, hence, by Lemma 5.8, whithin distance $V$ from $r_i$. Thus we have that, when $r_i$ reaches $H_i$, it can see $r_j$ on its left, therefore, it can not move further. It follows that, until $r_j$ is before it, $r_i$ can be only in $\mathbb{L}(\cdot)$, $\mathbb{C}_\emptyset(\cdot)$, or $\mathbb{M}_\emptyset(\cdot)$. Therefore, the first time that $r_j$ stops after $r_i$ reached $H_i$, say at time $t' > t$, $r_i$ and $r_j$ will be mutual visible. Moreover, between $t$ and $t'$, by Lemma 5.8 $\overline{r_i r_j} \leq V$, and the lemma follows. □

In the following lemma we show that if a robot sees some robots on its right, then it will never lose them during the computations. Let $\mathbb{R}$ be a set of robots which are mutually visible with $r_i$ at time $t$ and that are located to the right of $\Psi_i$, and $r_k$ a robot in $\mathbb{R}$ ((refer to Figure 4). Moreover, let $B$

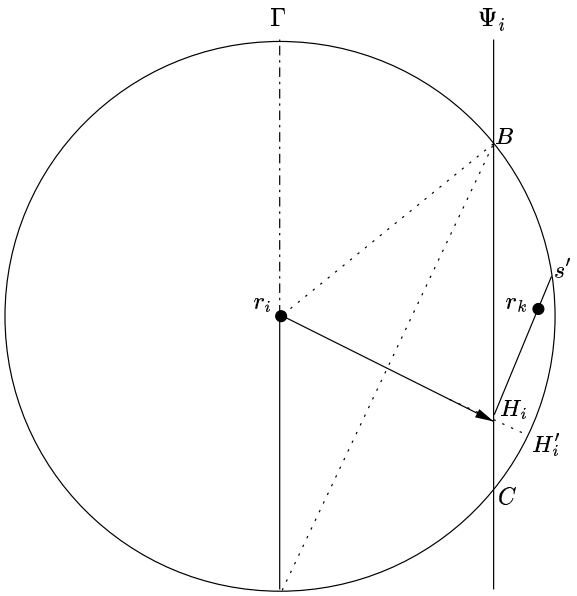


Figure 4: Lemma 5.10.

and $C$ be respectively the upper and lower intersection between $\Psi_i$ and $\mathcal{C}_i$, and $H_i'$ be the intersection between $\mathcal{C}_i$ and the line passing through $\overline{r_i H_i}$.

**Lemma 5.10.** *There exists a time $t' > t$ after which $r_i$ will be always mutually visible with the robots in $\mathbb{R}$. Moreover, $\overline{r_i r^*} \leq V$, $\forall\ r^* \in \mathbb{R}$.*

**Proof.** From Algorithm 1, we know that robots in $\mathbb{R}$ cannot perform any movement while $r_i$ is on their left. Let $t^*$ the time when $r_i$ enters its Look phase and $p$ be the destination point it computes. Clearly, $p$ can not be to the right of any robot in $\mathbb{R}$. In the following, we first prove that $\overline{lr^*} \leq V$, $\forall\ r^* \in \mathbb{R}$ and $\forall l \in \overline{r_i p}$.

From Lemma 3.3, it follows that:

$$\forall p \in arc(BH_i'),\quad pH_i \leq \overline{BH_i} = V. \tag{7}$$

Moreover, $\overline{H_i C} = \overline{BC} - \overline{BH_i} \leq 2V - V = V$ and from Lemma 3.2 we have:

$$\forall p \in arc(H_i'C),\quad pH_i \leq H_iC \leq V. \tag{8}$$

Plugging (7) and (8) we obtain:

$$\forall p \in arc(BC),\quad pH_i \leq V. \tag{9}$$

Let us now consider a robot $r_k \in sector(BCB)$ (that is in the area to the right of $\Psi_i$ and in $\mathcal{C}_i$) and let $s'$ be the intersection between $arc(BC)$ and the line passing through $H_i$ and $r_k$. We have that $\overline{H_i r_k} \leq \overline{H_i s'} \leq V$ (from (9)), $\overline{r_i r_k} \leq V$, and $\overline{r_i H_i} \leq V$. Therefore, applying Lemma 3.1 to $\triangle(r_i, r_k, H_i)$ we have that $\overline{q r_k} \leq V$, $\forall q \in \overline{r_i H_i}$. In conclusion, when $r_i$ stops in $p$, say at time $t' > t$, it will see all the robots in $\mathbb{R}$, that can only be in $L(t')$, $C_\emptyset(t')$, or $M_\emptyset(t')$, and the lemma follows. □

By Lemma 5.6, 5.9 and 5.10 we can conclude that:

**Theorem 5.1.** *The visibility graph $G$ is preserved during the execution of the algorithm.*

## 5.2 Finiteness

In this Section we prove that, after a finite number of steps, the robots will gather in a point.

We know from Assumption A2 that the distance travelled by a robot $r$ is bounded from below by $\delta$; thus, a robot always performs a move of at least length $\delta$, unless the destination point computed by the algorithm is closer than that.

**Lemma 5.11.** *Let us suppose to have several robots on a vertical axis $\Lambda$ and no robots to the left of $\Lambda$. If $r$ is the topmost robot on $\Lambda$ that can see a robot to the right of $\Lambda$, then, in a finite number of steps, either all the robots above $r$ on $\Lambda$ will reach $r$, or one of them will leave $\Lambda$.*

**Proof.** Let $r$ be the topmost robot on $\Lambda$ that can see a robot $r'$ to the right of $\Lambda$. Let us suppose, by contradiction, that no robot above $r$ leaves $\Lambda$ and that there is some robot above $r$ that do not reach $r$ in a finite number of steps. Let $p$ be the topmost point on $\Lambda$ (clearly above $r$) such that there exists a set of robots $\mathbb{R}$ above it that do not pass $p$ in a finite number of steps. Since the cycle time of any robot is finite (Assumption A1), at any stage of the algorithm the topmost robot on $\Lambda$ must perform a movement in finite time; its move must be vertical because the robot is not allowed to leave $\Lambda$ by hypothesis. It follows that all robots in $\mathbb{R}$ will get closer and closer to $p$. After a finite number of steps, all the robots in $\mathbb{R}$ will be at distance smaller than $\delta$ from $p$ since $p$ is the topmost point on $\Lambda$ that robots in $\mathbb{R}$ do not pass. Let us now consider another point $p'$ above $p$ and below the bottom most robot in $\mathbb{R}$. We know that all robots in $\mathbb{R}$ must pass $p'$ (otherwise $p$ wouldn't be the topmost point not passed by the robots in $\mathbb{R}$); let $x$ be the first robot that does it. The move that causes $x$ to pass $p'$,

must be caused by some robots $x'$ below $p'$ that $x$ could see; since there are no robots between $p'$ and $p$, $x'$ must be also below $p$. Moreover, since the distance between $x$ and $p$ is smaller than $\delta$, the robot $x$ will pass $p$ moving towards $x'$ (Assumption A2) leading to a contradiction. Thus, in a finite number of steps, either all the robots in $\mathbb{R}$ reach $r$ or at least one of them leaves $\Lambda$. □

The next Lemma shows that all robots in the system converge to the *Right* axis of the universe $\mathfrak{U}$.

**Lemma 5.12.** *For any given vertical axis $\Psi$ before Right which is at any distance $d > 0$ from it, all the robots that are on the left of $\Psi$ at the beginning of the algorithm, will pass $\Psi$ in a finite number of steps.*

**Proof.** Let us assume, by the sake of contradiction, that the robots do not converge to *Right*. Therefore, there exists at least one vertical axis in $\mathfrak{U}$ before *Right* such that a subset $\mathbb{R}$ of the robots that are before it at the beginning of the algorithm, will not pass it in a finite number of steps. Let $\Psi$ be the leftmost of these axis. Let $\mathbb{R}_a$ be the subset of robots in $\mathbb{R}$ that reach (but do not pass) $\Psi$ in a finite number of steps, and $\mathbb{R}_b$ be the subset of robots that will not reach $\Psi$ in a finite number of steps ($\mathbb{R} = \mathbb{R}_a \cup \mathbb{R}_b$).

First of all, we note that $\Psi$ cannot be the first vertical axis of the system. In fact, let us suppose that $\Psi$ is such a first axis and let $r$ be the topmost robot of $\mathbb{R}$ on this first axis that sees some robots to its right (such a robot must exist since $G$ is connected). By Lemma 5.11 either one of the robot above it will leave the axis, or $r$ will be reached by all the robots above it and then it will be allowed to move to its right passing $\Psi$.

Since, by hypothesis, $\Psi$ is the first axis which is not passed by robots in $\mathbb{R}$, the robots in $\mathbb{R}_b$ must pass, in a finite number of steps, any vertical axis before $\Psi$ getting infinitely closer to $\Psi$. We now consider two cases:

1. $\mathbb{R}_b = \emptyset$.
   In this case, after a finite number of steps, all robots in $\mathbb{R}$ reach $\Psi$. Since $G$ is connected, we have that, unless $\Psi$ is already *Right*, at least one robot $r$ in $\mathbb{R}$ must see some robot after $\Psi$. Let $r$ be the topmost robot with such a property.

   *a.* If $r$ cannot see any robot above it on $\Psi$, it can move either horizontally or diagonally, thus passing $\Psi$ and leading to a contradiction.

   *b.* Otherwise, by Lemma 5.11, we have that, in a finite number of steps, either one of the robots above $r$ on $\Psi$ will leave it or $r$ will be reached by all the robots above it on $\Psi$ and now will be

allowed to move either horizontally or diagonally. In both cases, we have a contradiction.

2. $\mathbb{R}_b \neq \emptyset$.
   In this case the robots in $\mathbb{R}_a$ reach $\Psi$ in a finite number of steps, while the robots in $\mathbb{R}_b$ just converge to it. Let us fix an arbitrary axis $I'$ before $\Psi$ and at a distance smaller than $\delta' < \frac{\sqrt{3}}{2}\delta$ from $\Psi$. We know that, in a finite number of steps, the robots in $\mathbb{R}_b$ will pass $I'$ and we can also assume that they they must have stopped at least once after $I'$, otherwise they would have passed $\Psi$. Let us now consider the computation after the robots in $\mathbb{R}_a$ have reached $\Psi$ and the robots in $\mathbb{R}_b$ have passed $I'$ stopping at least once after it. We now consider another arbitrary axis $I''$, which is after the rightmost robots in $\mathbb{R}_b$ and before $\Psi$ at a distance smaller than $\delta' < \frac{\sqrt{3}}{2}\delta$ from it. Since the robots in $\mathbb{R}_b$ converges to $\Psi$, we know that they will pass $I''$. Let $r$ be the first robot in $\mathbb{R}_b$ that passes $I''$. According to the algorithm, if $r$ has passed $I''$ it must have seen a robot $r'$ on its right and, since $r$ had stopped after $I'$, it must have seen $r'$ when $r$ was between $I'$ and $I''$. Since $r$ is the first robot passing $I''$, there are no robots between $I''$ and $\Psi$. Therefore $r'$ must be on $\Psi$ or on its right. If $r'$ were on $\Psi$, then $r$ would reach $\Psi$ in one step since the longest distance between $r'$ and $\Psi$ is less or equal to $\frac{\delta'}{sin60}$ (by Assumption A2) and since it started to move after $I'$. This is a contradiction since $r \in \mathbb{R}_b$ and it cannot reach $\Psi$ by hypothesis. Therefore, $r'$ must be to the right of $\Psi$ and, for the same reasons as above, $r$ can reach it in one step, passing $\Psi$ and creating a contradiction. □

We now prove that all robots in the system actually reach the *Right* axis of $\mathfrak{U}$.

**Lemma 5.13.** *After a finite number of steps, all the robots in the system reach Right.*

**Proof.** Let us assume that there is a subset $\mathbb{R}$ of robots that will converge to *Right* never reaching it. Thus, in a finite number of steps, the only robots which do not reach *Right* are the robots in $\mathbb{R}$. Let $\Psi$ be a vertical axis before *Right* at distance smaller or equal to $\delta' < \frac{\sqrt{3}}{2}\delta$ from it. Since the robots converge to *Right*, they will pass $\Psi$ in a finite number of steps. Let as call $r$ the first robot that does it. Applying a similar argument to the one of the previous lemma, we have that $r$ must reach *Right* leading to a contradiction. □

The following lemma shows that if all robots lie on the same axis, they will reach the bottom most robot in a finite number of steps.

**Lemma 5.14.** *If all the robots of the system lie on the same vertical axis $\Lambda$, then in a finite number of steps all the robots will reach the bottom most robot on $\Lambda$.*

**Proof.** Let us assume, for sake of contradiction, that there exists points on $\Lambda$ which are not passed by some of the above robots. Let $p$ be the topmost such point and let $\mathbb{R}$ be the set of robots above $p$ which will never pass it. Clearly the topmost robot in $\Lambda$ cannot lie on $p$ because in such a case it would pass it (since $G$ is connected, it would do a vertical move). Moreover, notice that a robot can move only if it is the topmost on $\Lambda$ and, since $G$ is connected and the cycle is finite (Assumption A1), the topmost robot will move. Therefore, after a finite number of steps, all the robot in $\mathbb{R}$ will be at distance smaller than $\delta$ from $p$ and now the same argument of the previous two lemmas can be applied to derive the contradiction. □

We can finally conclude that:

**Theorem 5.2.** *In a finite number of steps, all the robots in the system gather in a point; the rightmost and bottom most corner of the universe.*

**Proof.** From Lemma 5.13, all the robots in the system reach the rightmost axis $\mathfrak{U}$. Since the robots that are on *Bottom* at the beginning of the algorithm could have moved only horizontally, there will be at least one robot on the rightmost and bottom most corner of $\mathfrak{U}$. The robots will now gather in such a corner by Lemma 5.14. □

## References


[1] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A Distributed Memoryless Point Convergence Algorithm for Mobile Robots with Limited Visibility. *IEEE Trans. on Robotics and Automation*, 15(5):818–828, 1999.

[2] G. Beni and S. Hackwood. Coherent Swarm Motion Under Distributed Control. In *Proc. DARS'92*, pages 39–52, 1992.

[3] E. H. Durfee. Blissful Ignorance: Knowing Just Enough to Coordinate Well. In *ICMAS*, pages 406–413, 1995.

[4] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Hard Tasks for Weak Robots: The Role of Common Knowledge in Pattern Formation by Autonomous Mobile Robots. In *10th International Symposium on Algorithm and Computation (ISAAC '99)*, pages 93–102, 1999.

[5] D. Jung, G. Cheng, and A. Zelinsky. Experiments in Realising Cooperation between Autonomous Mobile Robots. In *5th International Symposium on Experimental Robotics (ISER)*, June 1997. Barcelona, Catalonia.

[6] Y. Kawauchi and M. Inaba and. T. Fukuda. A Principle of Decision Making of Cellular Robotic System (CEBOT). In *Proc. IEEE Conf. on Robotics and Automation*, pages 833–838, 1993.

[7] M. J Matarić. *Interaction and Intelligent Behavior*. PhD thesis, MIT, May 1994.

[8] S. Murata, H. Kurokawa, and S. Kokaji. Self-Assembling Machine. In *Proc. IEEE Conf. on Robotics and Automation*, pages 441–448, 1994.

[9] K. Sugihara and I. Suzuki. Distributed Algorithms for Formation of Geometric Patterns with Many Mobile Robots. *Journal of Robotics Systems*, 13:127–139, 1996.

[10] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots. In *Proc. of 3rd International Colloquium on Structural Information and Communication Complexity*, pages 313–330, Siena, 1996.

[11] I. Suzuki and M. Yamashita. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *SIAM J. Comput.*, 28(4):1347–1363, 1999.