Getting Close Without Touching

Linda Pagli[†] Giuseppe Prencipe[†] Giovanni Viglietta[†]

Abstract. In this paper we study the NEAR-GATHERING problem for a set of asynchronous, anonymous, oblivious and autonomous mobile robots with limited visibility moving in Look-Compute-Move (LCM) cycles: In this problem, the robots have to get close enough to each other, so that every robot can see all the others, without touching (i.e., colliding) with any other robot. The importance of this problem might not be clear at a first sight: Solving the NEAR-GATHERING problem, it is possible to overcome the limitations of having robots with limited visibility, and it is therefore possible to exploit all the studies (the majority, actually) done on this topic, in the unlimited visibility setting. In fact, after the robots get close enough, they are able to see all the robots in the system, a scenario similar to the one where the robots have unlimited visibility.

Here, we present a collision-free algorithm for the NEAR-GATHERING problem, the first to our knowledge, that allows a set of autonomous mobile robots to nearly gather within finite time: The collision-free feature of our solution is crucial in order to combine it with an unlimited visibility protocol. In fact, the majority of the algorithms that can be found on the topic assume that all robots occupy distinct positions at the beginning. Hence, only providing a collision-free NEAR-GATHERING algorithm, as the one presented here, is it possible to successfully combine it with an unlimited visibility protocol, hence overcoming the natural limitations of the limited visibility scenario.

In our model, distances are induced by the infinity norm. A discussion on how to extend our algorithm to models with different distance functions, including the usual Euclidean distance, is also presented.

1 Introduction

Consider a distributed system whose entities are a set of *robots* or *agents* that can freely move on a two-dimensional plane, operating in *Look-Compute-Move* (LCM) cycles. During a cycle, a robot takes the snapshot of the position of the other robots (*Look*); executes the protocol, the same for all robots, using the snapshot as an input (*Compute*); and moves towards the computed destination, if any (*Move*). After each cycle, a robot may be inactive for some time. With respect to the LCM cycles, the most common models used in these studies are the *fully synchronous* (FSYNC), the *semi-synchronous* (SSYNC), and the *asynchronous* (ASYNC). In the *asynchronous* (ASYNC) model, each robot acts independently from the others and the duration of each cycle is finite but unpredictable; thus, there is no common notion of time, and robots can compute and move based on *obsolete* observations. In contrast, in the *fully synchronous* (FSYNC) model, there is a common notion of time, and robots execute their cycles synchronously. In particular, time is assumed to be discrete, and at each time instant

[†]Dipartimento di Informatica, Università di Pisa, {pagli,prencipe,viglietta}@di.unipi.it

all robots are activated, obtain the same snapshot, compute and move towards the computed destination; thus, no computation or move can be made based on obsolete observations. The last model, the *semi-synchronous* (SSYNC), is like FSYNC where, however, not all robots are necessarily activated at each time instant.

In the last few years, the study of the computational capabilities of such a system has gained much attention, and the main goal of the research efforts has been to understand the relationships between the capabilities of the robots and their power to solve common tasks. The main capabilities of the robots that, to our knowledge, have been studied so far in this distributed setting are *visibility*, *memory*, *orientation*, and *direct communication*. With respect to visibility, the robots can either have *unlimited visibility*, by sensing the positions of *all* other robots, or have *limited visibility*, by sensing just a portion of the plane, in particular up to a given distance V [1, 8]. With respect to memory, the robots can either be *oblivious*, by having access only to the information sensed or computed during the current cycle (e.g., [12]), or *non-oblivious*, by having the capability of storing the information sensed or computed since the beginning of the computation (e.g., [2, 13, 14]). With respect to orientation, the two extreme settings studied are the one where the robots have *total agreement*, and agree on the orientation and direction of their local coordinate systems (i.e., they agree on a *compass*), e.g., [9], and the one where the robots have *no agreement* on their local coordinate axes, e.g., [13, 14]; in the literature, there are studies that tackle also the scenarios in between; for instance, when the robots agree on the direction and orientation just of the y coordinate, or there is agreement just on the chirality of the coordinate system, e.g., [6]. With respect to direct communication, the direction so far has been towards the use of external signals or lights to enhance the capabilities of mobile, first suggested in [10], and also referenced in [7], which provided the earliest indication that incorporating in the robot model some simple means of signalling might positively affect the power of the team. Recently, a study that tackles more systematically this particular capability has been presented in [3].

In this paper, we solve the Near-Gathering problem: The robots are required to get close enough to each other, without touching or colliding during their movements. Here, the team of robots under study executes the cycles according to the ASYNC model, the robots are oblivious and have limited visibility. The importance of this problem might not be clear at a first sight: With a solution to the Near-Gathering problem it would be possible to overcome the limitations of having robots with limited visibility, and it would be possible to exploit all the studies (the majority, actually) done in the unlimited visibility setting. In fact, after the robots get close enough, they are able to see all the robots in the system, a scenario similar to the one where the robots have unlimited visibility. Since most of the solutions to the unlimited visibility case assume a starting configuration where no two robots *touch* (i.e., they do not share the same position in the plane), it is of crucial importance to ensure that no collision occurs during the near gathering.

A problem close to Near-Gathering is the *gathering* problem, where the robots have to meet, within finite time, in a point of the plane not agreed in advance. This problem has been studied in the literature in all models; in particular, a study in SSYNC with limited visibility has been presented in [1]: Actually, this solution could be easily modified to solve also the Near-Gathering problem, just imposing a termination condition; however, it has been shown that this solution does not work in ASYNC [11]. Another solution for the limited visibility case is in [12], where the coordinate systems are assumed to be consistent only after a period of instability (i.e., the robots agree on the coordinate system only after an arbitrary long period); however, also this solution is designed for the SSYNC model. In the asynchronous model, the only solution to the gathering problem with robots having limited visibility has been presented in [8]: This protocol, however, is not collision-free; hence, it cannot be used to solve our problem. We note that, as in the protocol in [8], we also assume that the robots have total agreement. Also, we remark that, since the algorithm presented here is for the ASYNC model, it solves the problem also in the SSYNC and FSYNC models.

As stated above, solutions to problems studied in the unlimited visibility setting can be potentially used to solve the same problems in the limited visibility setting, by exploiting the Near-Gathering protocol presented in this paper. Among these, we can cite for instance the *Arbitrary Pattern Formation Problem* [13, 9, 6, 14], or the *Uniform Circle Formation* (e.g., [4, 5]).

The organization of the paper is as follows: In Section 2 the formal definition of the robot model is presented; in Section 3 the collision-free algorithm that solves the Near-Gathering problem is presented; in Section 4 the correctness of the protocol is shown. Due to space constraints, one of the proofs has been moved to the appendix, and we thoroughly discuss only the scenario in which distances are induced by the infinity norm. However, some extensions of our algorithm to models with different distance functions, including the usual Euclidean distance, are also briefly discussed in Section 5.

2 The Model

The system is composed of a team of mobile entities, called *robots*, each modeled as a computational unit provided with its own local memory and capable of performing local computations. The robots are (viewed as) points in the plane. Let $r(t)$ denote the absolute position of robot r at time t (i.e., with respect to an absolute reference frame); also, we will denote by $r(t).x$ and $r(t).y$ the abscissa and the ordinate value of position $r(t)$, respectively. When no ambiguity arises, we shall omit the temporal indication; also, the *configuration of the robots at time t* is the set of robots' positions at time t.

Each robot has its own local coordinate system, and we assume that the local coordinate systems of the robots are consistent with each other: In other words, they agree on where the North, South, East and West are. A robot is endowed with sensorial capabilities and it observes the world by activating its sensors, which return a snapshot of the positions of all other robots with respect to its local coordinate system. The visibility radius of the robots is limited: Robots can sense only points in the plane within distance V. This setting, referred in the literature as *limited visibility*, is understandably more difficult; for example, a robot with limited visibility might not even know the total number of robots nor where they are located if outside its radius of visibility. Also, combined with the asynchronous behavior of the robots, introduces a higher level of difficulty in the design of collision-free protocols. For instance, in the example depicted in Figure 1.a, robot s, in transit towards its destination, is seen by r; however, s is not aware of r's existence and, if it starts the next cycle before r starts moving, s will continue to be unaware of r; hence, since r does not see s when s starts its movement, it must take care of the "potential" arrival of s when computing its destination.

All robots are identical: They are indistinguishable from their appearance and they execute the same protocol. Robots are autonomous, without a central control. Robots are silent, in the sense that they have no means of direct communication (e.g., radio, infrared) of information to other robots. Each robot is endowed with motorial capabilities, and can move freely in the plane. A move may end before the robot reaches its destination, e.g., because of limits to its motion energy. The distance traveled in a move is neither infinite nor infinitesimally small. More precisely, there exists a constant $\delta > 0$ such that, if the destination point is closer than δ,

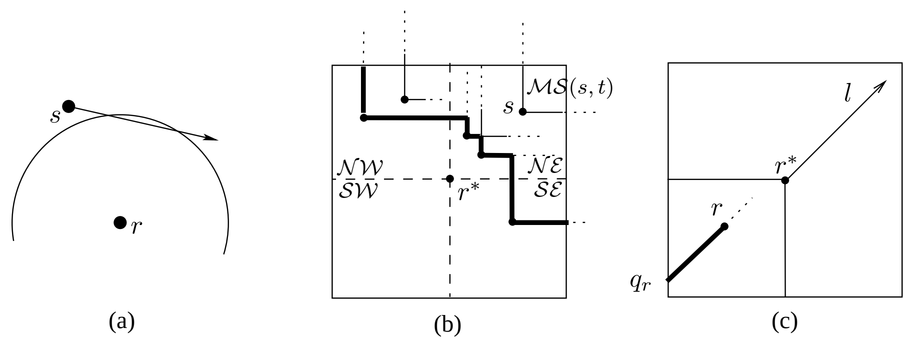

Figure 1: (a) When s starts moving (the left end of the arrow), r and s do not see each other. While s is moving, r *Look*s and sees s; however, s is still unaware of r. After s passes the area of visibility of r, it is still unaware of r. (b) The area above and to the right of s defines the *Move Space* of s. The fat line is the *Contour* of r^*. (c) Computation of the length of the movement in the algorithm.

the robot will reach it; otherwise, it will move towards it of at least δ. Note that, without this assumption, an adversary would make it impossible for any robot to ever reach its destination, following a classical Zenonian argument. The quantity δ might not be known to the robots.

The robots do not have persistent memory, that is, memory whose content is preserved from one cycle to the next; they are said to be *oblivious*. The only available memory they have is used to store local variables needed to execute the algorithm at each cycle.

At any point in time, a robot is either *active* or *inactive*. When *active*, a robot r executes a *Look-Compute-Move* (LCM) cycle performing the following three operations, each in a different state:

(i) Look: The robot observes the world by activating its sensor, which returns a snapshot of the positions of all robots within its radius of visibility with respect to its own coordinate system (since robots are viewed as points, their positions in the plane are just the set of their coordinates).

(ii) Compute: The robot executes its algorithm, using the snapshot as input. The result of the computation is a destination point.

(iii) Move: The robot moves towards the computed destination; if the destination is the current location, the robot stays still (performs a *null movement*).

When *inactive*, a robot is idle. All robots are initially inactive. The amount of time to complete a cycle is assumed to be finite, and the *Look* is assumed to be instantaneous.

We will denote by $\mathbb{W}(t)$, $\mathbb{L}(t)$, $\mathbb{C}(t)$, $\mathbb{M}(t)$ the sets of robots that are, respectively, inactive, in a *Look* phase, in a *Compute* phase and in a *Move* phase at time t.

In the following, we will assume that all distances are induced by the *infinity norm*: $\|p\|_\infty = \max\{p.x, p.y\}$. Different distance functions, including the usual Euclidean distance, will be briefly discussed in Section 5.

2.1 Notation

We will denote by $\mathcal{R} = \{r_1, \ldots, r_n\}$ the set of robots in the system. First note that, in order to achieve explicit termination, it is necessary that all robots share the knowledge of n. In

Section 4.5 we will show how to overcome this by making use of visible bits [3].

We will denote by $G(t) = (N, E(t))$ the *distance graph* at time $t \geq 0$, where N is the set of the input robots and, for any two distinct robots r and s, $(r, s) \in E(t)$ iff $0 \leq \|r(t) - s(t)\|_\infty \leq V$.

In [8] it was proved that the initial distance graph $G(0)$ must be connected for the gathering problem to be solvable; the same result clearly holds also for the Near-Gathering problem:

Lemma 1. *If the distance graph $G(0)$ is disconnected, the* Near-Gathering *problem is unsolvable.*

Thus, in the following we will always assume that $G(0)$ is connected.

Let r be a robot, and let us divide its visible area into four quadrants, denoted by $\mathcal{NW}(r)$, $\mathcal{NE}(r)$, $\mathcal{SE}(r)$, and $\mathcal{SW}(r)$ (see the example depicted in Figure 1.b). For technical reasons, the vertical and the horizontal segment of length V starting from r and going South and West, respectively (including the location of r itself), are part of $\mathcal{SW}(r)$; the vertical (resp. horizontal) segment of length V passing through r and going North (resp. East) is part of $\mathcal{NW}(r)$ (resp. $\mathcal{SE}(r)$). When not necessary, the reference to r will be dropped. Similarly, a reference to time may be added.

Next, we define the *Move Space* of a robot (refer to the example depicted in Figure 1.b):

Definition 1 (Move Space)**.** *The* Move Space *of a robot r at time t, denoted by $\mathcal{MS}(r, t)$, is the set $\{(x', y') \in \mathbb{R}^2 \mid x' \geq r(t).x \wedge y' \geq r(t).y\}$.*

Based on the previous definition, we introduce the *Contour* of a robot (refer again to Figure 1.b):

Definition 2 (Contour)**.** *The* Contour *of a robot r at time t, denoted by $\mathcal{CT}(r, t)$, is the boundary of the set $\bigcup_s \mathcal{MS}(s, t)$, where s ranges through all the robots in $\mathcal{NW}(r, t) \cup \mathcal{NE}(r, t) \cup \mathcal{SE}(r, t)$.*

We will call a *peak* of the contour any convex corner of $\mathcal{CT}(r)$; the concave corners will be called *valleys*. An easy property of $\mathcal{CT}(r, t)$ is stated in the following

Observation 1. *If there are robots in both $\mathcal{NW}(r)$ and in $\mathcal{SE}(r)$, and no robot in $\mathcal{NE}(r)$, then $\mathcal{CT}(r)$ has exactly one valley in $\mathcal{NE}(r)$.*

3 The Near-Gathering Problem and Its Solution

In the Near-Gathering problem, at the beginning a set of n robots is arbitrarily placed in the plane, on distinct positions such that $G(0)$ is connected: We will call this the *initial configuration*, denoted by $\mathcal{I}$. In finite time, the robots are required to move within distance ε from each other, for a given $0 < \varepsilon < V/4$: We will call this the *final configuration*, denoted by $\mathcal{F}$.

In our solution (reported in Figure 2), a robot moves only when it sees robots in $\mathcal{NW} \cup \mathcal{NW} \cup \mathcal{SE}$. Informally, at each cycle, robot r^* first computes the direction of movement according to the following rules:

- If r^* can see robots only in $\mathcal{SW}$, then it will not move; that is, in this case the destination point is the point of coordinates $(0, 0)$.
- If r^* can see robots only in $\mathcal{NW} \cup \mathcal{SW}$, then its direction of movement is given by the half-line l starting in r^* and going North.
- If r^* can see robots only in $\mathcal{SW} \cup \mathcal{SE}$, then its direction of movement is given by the half-line l starting in r^* and going East.

- Otherwise, the direction of movements of r is decided based on the shape of the Contour of r^*. In particular, if in $\mathcal{NE}$ there is at least a robot, the direction of movement is given by the half-line l starting from r^* and passing through robot in $\mathcal{NE}$ closest to r^*. Otherwise, there must be robots in both $\mathcal{NW}$ and $\mathcal{SE}$; in this case, the direction of movement is given by the half-line l starting from r^* and passing through the only valley in $\mathcal{CT}(r^*)$.

In order to establish the length of the movements along l, r^* checks two main factors: First, it must not enter the Move Space of any robot it can see (this contributes to guarantee collision avoidance); second, the new position must be within distance $V/2$ from any of the robots it is currently seeing (this contributes to guarantee both collision avoidance and the connectedness of the initial distance graph). In order to ensure these two factors, first, for each $r \in \mathcal{NW} \cup \mathcal{NE} \cup \mathcal{SE}$, it computes the intersection p_r between l and $\mathcal{MS}(r)$ (notice that robots move only upward and rightward). Second, for each visible robot r, the intersection q_r between the visible area of r^* and the line parallel to l and passing through r is computed: The distance d_r between r and q_r is the maximum distance r^* is allowed to move in order to not lose visibility with r (assuming r does not move). Thus, if p is the point closest to r^* among the points in $\{p_r\} \cup \{d_r\}$, the destination point of r^* is the median point dp on the segment between r^* and p.

As we will prove in the following, a consequence of the computation of dp as described above is that the distance graph never gets disconnected; also, collisions are avoided. Termination is achieved using the knowledge of n that the robots are assumed to have. In fact, it is easy to see that, since the robots operate in a totally asynchronous environment, without knowledge of n, explicit termination would not be possible. In particular, in our solution, a robot terminates its execution as soon as it sees n robots at distance less than a given tolerance ε.

4 Correctness

In this section, we will prove that the Algorithm reported in Figure 2 correctly solves the Near-Gathering problem. In particular, the proof will be articulated in three parts: First, we will prove that the initial distance graph is preserved during the execution; second, we will prove that no collision occurs during the movements of the robots; finally, the correctness proof concludes by showing that the algorithm terminates.

4.1 Preliminary Definitions and Observations

Before presenting the correctness proof, we will introduce a few preliminary definitions and observations. First, by construction, it is easy to observe the following:

Observation 2. *Each robot can only move rightward and upward. Furthermore, the robots on the rightmost vertical axis never move right, and the robots on the topmost horizontal axis never move up.*

Observation 3. *During each cycle, a robot travels a distance of at most $V/2$.*

Definition 3 (First and Last). *Given a robot r, let $First(r, t) = \min\{t' > t | r \in \mathbb{L}(t')\}$ be the first time, after time t, at which r performs a Look operation. Also, let $Last(r, t) = \max\{t' \leq t | r \in \mathbb{L}(t')\}$ be the last time, from the beginning up to time t, at which r has performed a Look operation; if r has not performed a Look yet, then $Last(r, t) = 0$.*

Now, we define the *Destination Point* of a robot at a time t as follows:

```
State Look
        Take the snapshot of the positions of the visible robots, which returns, for each robot $r \in \mathcal{R}$ within
        distance $V$, Pos[r], the position in the plane of robot $r$ (according to my coordinate system); (Note:
        I am robot $r^*$)
```

```
State Compute
  $Z_\varepsilon$ = Robots in Pos[] within distance $\leq \varepsilon$;
  If $|Z_\varepsilon| = n$ Then Terminate.
  $l, p_1, \ldots, p_n, p'_1, \ldots, p'_n, b$ = nil;
  Let $\mathcal{NW}$, $\mathcal{NE}$, $\mathcal{SE}$, and $\mathcal{SW}$ be the quadrants of my visible area;
  $\mathcal{CT}$ = Contour of the robots in $\mathcal{NW} \cup \mathcal{NE} \cup \mathcal{SE}$;
  If I see robots only in $\mathcal{SW}$ Then $dp = (0, 0)$;
  Else
    If I see robots only in $\mathcal{NW} \cup \mathcal{SW}$ Then
      $l$ = Half-line from me going North;
    Else If I see robots only in $\mathcal{SE} \cup \mathcal{SW}$ Then
      $l$ = Half-line from me going East;
    Else
      If There is at least one robot in $\mathcal{NE}$ Then
        $l$ = Half-line from me to the closest robot in $\mathcal{NE}$;
      Else
        $l$ = Half-line from me to the only valley of $\mathcal{CT}$ in $\mathcal{NE}$;
    For Each robot $r \in \mathcal{NW} \cup \mathcal{NE} \cup \mathcal{SE}$ Do
      $p_r$ = Intersection between $l$ and $\mathcal{MS}(r)$;
    For Each visible robot $r$ Do
      $l_r$ = Line parallel to $l$ and passing through $r$;
      $q_r$ = Lowest or leftmost intersection between $l_r$ and my visible area;
      $d_r$ = Distance between $r$ and $q_r$;
      $p'_r$ = Point on $l$ at distance $d_r$ from me;
    $b$ = Point on $l$ at distance $V$ from me;
    $p$ = Point closest to me among points in $\{p_r\} \cup \{p'_r\} \cup \{b\}$;
    $dp$ = Median point on the segment between my position and $p$.
```

```
State Move
  Move($dp$).
```

Figure 2: The NEAR-GATHERING Protocol

Definition 4 (Destination Point)**.** *Given a robots r, we define the* Destination Point $\texttt{DP}(r,t)$ *of r at time t as follows:*

- *If $r \in \mathbb{W}(t)$, then: if r is in its first cycle, then $\texttt{DP}(r,t) = r(0)$ (i.e., the starting position of r); otherwise, $\texttt{DP}(r,t)$ is the point p as computed in the last Compute state before t (in the previous cycle).*
- *If $r \in \mathbb{L}(t)$, then $\texttt{DP}(r,t)$ is the point p as computed in the next Compute state after t (in the current cycle).*
- *If $r \in \mathbb{C}(t)$, then $\texttt{DP}(r,t)$ is the point p as computed in the current Compute state.*
- *If $r \in \mathbb{M}(t)$, then $\texttt{DP}(r,t)$ is the point p as computed in the last Compute state before t (in*

the current cycle).

From the previous definition, we can state the following:

Lemma 2. *Let r be a robot. During the time strictly between two consecutive Looks, the Destination Point of r does not change.*

Proof. Let t be any time when r executes a *Look*; then, by definition, $\texttt{DP}(r, t)$ is the point p as computed in the next *Compute* state after t (in the current cycle). Also, the destination point does not change in the next *Compute* and *Move* state of r. □

4.2 Preservation of Mutual Awareness

We will now prove that the connectedness of the initial distance graph is preserved during the entire execution of the algorithm. We do so by first introducing the notion of *mutual awareness.*

Definition 5 (Mutual Awareness). *Two distinct robots r and s are* mutually aware *at time t iff both conditions hold:*

1. $\|r(t_r) - s(t_r)\|_\infty \leq V$*, with* $t_r = \mathit{Last}(r, t)$*, and*
2. $\|r(t_s) - s(t_s)\|_\infty \leq V$*, with* $t_s = \mathit{Last}(s, t)$*.*

Since initially all robots are inactive, then by definition of mutual awareness we have

Lemma 3. *All the pairs of robots that are within distance V from each other at time $t = 0$ are initially mutually aware.*

In the following lemma, we will prove that two robots that are mutually aware at the beginning of the computation keep the awareness during the execution.

Lemma 4. *If robots r and s are mutually aware at time t, they are mutually aware at any time $t' > t$.*

Proof. Let $\{t_i\}_{i \geq 0}$ be the weakly increasing sequence of time instants at which either r or s execute a *Look*; if both r and s *Look* simultaneously, then such time instant appears twice in the sequence. Without loss of generality, we may assume that r and s first become mutually aware at time t_m, when r starts a *Look* state, whereas s started a *Look* at time t_{m-1}.

We will prove by induction that, for any $i \geq m$, the following conditions hold:

1. $\|r(t_i) - s(t_i)\|_\infty \leq V$,
2. $\|\texttt{DP}(r, t_i) - s(t_i)\|_\infty \leq V$,
3. $\|r(t_i) - \texttt{DP}(s, t_i)\|_\infty \leq V$,

which will clearly imply our claim.

Observe that Condition 1 holds for $i = m - 1$ and $i = m$, by definition of mutual awareness. Moreover, by the algorithm, Condition 2 holds for $i = m$ and Condition 3 holds for $i = m - 1$. To show that Condition 3 holds for $i = m$, recall that

$$r(t_{m-1}).x \leq r(t_m).x \leq \texttt{DP}(r, t_{m-1}).x \leq r(t_{m-1}).x + V,$$

$$s(t_{m-1}).x \leq s(t_m).x \leq \texttt{DP}(s, t_{m-1}).x \leq s(t_{m-1}).x + V.$$

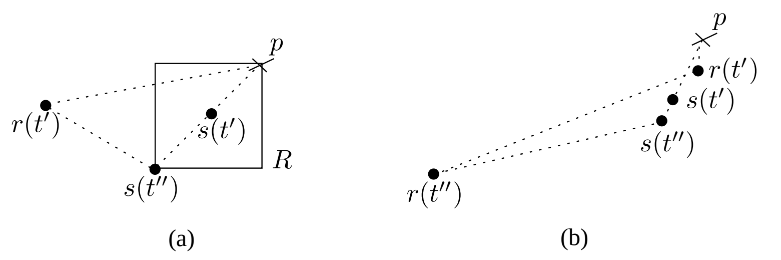

Figure 3: Proof of Lemma 5.

By Condition 1 (for $i = m$), it follows that

$$r(t_m).x \leq s(t_m).x + V \leq \mathtt{DP}(s, t_{m-1}).x + V,$$

and by Condition 3 (for $i = m - 1$)

$$\mathtt{DP}(s, t_{m-1}).x \leq r(t_{m-1}).x + V \leq r(t_m).x + V,$$

which are collectively equivalent to $|r(t_m).x - \mathtt{DP}(s, t_{m-1}).x| \leq V$. A similar argument holds for the y coordinates, implying that $\|r(t_m) - \mathtt{DP}(s, t_{m-1})\|_\infty \leq V$. This yields Condition 3 for $i = m$, since, by Lemma 2, $\mathtt{DP}(s, t_m) = \mathtt{DP}(s, t_{m-1})$. Thus the base of the induction holds.

Let $i > m$, and let the three conditions hold at time t_{i-1}. Again, without loss of generality, we may assume that r starts a *Look* at time t_i, and therefore Condition 2 holds by the algorithm.

By indcutive hypothesis, observe that

$$r(t_i).x \leq \mathtt{DP}(r, t_{i-1}).x \leq s(t_{i-1}).x + V \leq s(t_i).x + V$$

$$s(t_i).x \leq \mathtt{DP}(s, t_{i-1}).x \leq r(t_{i-1}).x + V \leq r(t_i).x + V,$$

which yields $|r(t_i).x - s(t_i).x| \leq V$. A similar argument holds for the y coordinate, thus proving Condition 1.

Since Condition 1 holds, then Condition 3 also holds by the same argument used for $i = m$. □

Based on the previous lemma, we can state the following

Corollary 1. *The connectedness of $G(0)$ is preserved during the execution of the algorithm.*

4.3 Collision Avoidance

In this section, we will prove that no collision occurs during the execution of the algorithm.

Lemma 5. *No collision ever occurs between any pair of robots during the execution of the algorithm.*

Proof. Let us assume by contradiction that two distinct robots r and s collide at a point p at time $t > 0$. Let $t' = \mathit{Last}(r, t)$ and $t'' = \mathit{Last}(s, t)$, and let $t' \geq t''$, without loss of generality; note that, by Observation 2, point p belongs to both $\mathcal{MS}(r, t')$ and $\mathcal{MS}(s, t'')$. Furthermore, by Observation 3, $\|r(t') - p\|_\infty \leq V/2$ and $\|s(t'') - p\|_\infty \leq V/2$, implying $\|r(t') - s(t'')\|_\infty \leq V$ by the triangle inequality. Also, let R be the rectangle passing through $s(t'')$ and p; by construction, all points inside R are within distance $V/2$ (refer to the example depicted in Figure 3.a).

Since $s(t')$ belongs to the segment passing through $s(t'')$ and p, and because of the convexity of the visible area, we have $\|r(t') - s(t')\|_\infty \leq V$. Hence r sees s when performing its *Look* at time t'. By construction, p belongs also to $\mathcal{MS}(s, t')$; however, by the algorithm, r cannot compute a point in $\mathcal{MS}(s, t')$, unless $r(t') \in \mathcal{MS}(s, t')$. On the other hand, by Observation 2, $\mathcal{MS}(r, t') \subseteq \mathcal{MS}(r, t'')$, hence p belongs also to $\mathcal{MS}(r, t'')$; however, s cannot compute a point in $\mathcal{MS}(r, t'')$ at time t'', unless $\|r(t'') - s(t'')\|_\infty > V$. Thus, $r(t'') \notin \mathcal{MS}(s, t'')$, otherwise $r(t'')$ would be inside the rectangle R, and the last inequality would not hold (see Figure 3.b).

Since $\|r(t'')-s(t'')\|_\infty > V$ and $\|r(t')-s(t'')\|_\infty \leq V$, it follows by the triangle inequality that $\|r(t'')-r(t')\|_\infty > V$. Thus, by Observation 3, r must *Look* at least once on its movement between $r(t'')$ and $r(t')$. Let $\tilde{t}$ be the last time at which r executes a *Look* phase and $r(\tilde{t}) \notin \mathcal{MS}(s, \tilde{t})$, and let $t''' = \mathit{First}(r, \tilde{t})$. Clearly, $t'' \leq \tilde{t} < t''' \leq t'$. By hypothesis, $r(t''') \in \mathcal{MS}(s, t''')$, hence $r(t''')$ and $s(\tilde{t})$ lie both inside R; thus, it follows that $\|r(t''') - s(\tilde{t})\|_\infty \leq V/2$. But, by Observation 3, also $\|r(\tilde{t}) - r(t''')\|_\infty \leq V/2$, hence $\|r(\tilde{t}) - s(\tilde{t})\|_\infty \leq V$ due to the triangle inequality. By Observation 2, $\mathcal{MS}(s, t''') \subseteq \mathcal{MS}(s, \tilde{t})$; that is, r computes at time $\tilde{t}$ as destination a point in $\mathcal{MS}(s, \tilde{t})$, while seeing s. However, by the algorithm, this is a contradiction, and the lemma follows. $\square$

4.4 Termination

Let us call *Right* the vertical axis passing throught the righmost robot(s) in $\mathcal{I}$, and *Top* the horizontal axis passing throught the topmost robot(s) in $\mathcal{I}$; also, let f be the intersection point between *Right* and *Top*. By Observation 2, and by the algorithm, we can easily observe that

Observation 4. *If at any time t a robot is at position f, then it never moves from there.*

Next, we introduce a definition that will be useful to prove the convergence of the algorithm.

Definition 6 (Convergence Point)**.** *Given a point a, let Ψ and Γ be the vertical and the horizontal axes passing through it, respectively. We say that a is a* convergence point *for robot r (or that r converges towards a) if, within finite time, r passes any vertical axis to the left of Ψ and any horizontal axis below Γ, and never passes neither Ψ or Γ.*

Note that, by Observation 2, all robots that converge towards a point a are below and to the left of a (refer to Figure 4). The following lemma, whose proof can be found in the appendix, shows that f is the only converge point.

Lemma 6. *All robots converge towards point f.*

From the previous lemma, and by the termination condition of the algorithm, we can state the following

Corollary 2. *After finite time, all robots terminate their execution, being at distance ε from each other.*

By Corollaries 1 and 2, and by Lemma 5, we can state the following

Theorem 3. *Algorithm 2 correctly solves the* Near-Gathering *problem.*

4.5 On the Knowledge of n

In the solution that we presented, in order for the robots to explicitly terminate, the knowledge of n is necessary. However, this assumption can be dropped by using external visible bits, as recently introduced in [3]. In particular, each robot is equipped with a visible light, whose color can be changed during the *Compute* phase. During the *Look*, a robot can retrieve, beside the position, also the value of the light of its fellow robots, which can be stored in a local `Light`[] array (the color of the light of the executing robot is stored in `Light`[`1`]).

With this extra information, the explicit termination of the robots can be achieved by substituting the termination check in Algorithm 2 with the following check, where ε is an arbitrary small constant (any fraction of V):

```
If $|Z \setminus Z_\varepsilon| == 0$ Then
    Light[r*] = 1;
    If $\forall r \in Z_\varepsilon$, Light[r] == 1 Then Terminate.
Else Light[r*] = 0;
```

5 Conclusions

In this paper we presented the first algorithm that solves the Near-Gathering problem for a set of autonomous mobile robots with limited visibility (where the distance function is induced by the infinity norm); the protocol presented here is collision-free: This allows to potentially combine our protocol with solutions designed for the unlimited visibility setting.

We remark that our Algorithm 2 also solves the Near-Gathering problem in the robot model that uses the Manhattan distance (i.e., the distance induced by the 1-norm): Each robot merely has to transform each snapshot that it gets during a *Look* phase by rotating it clockwise by 45° and scaling it by a factor of $\sqrt{2}$. Then the protocol can be applied as it is, and finally the computed point *dp* has to be moved again with the inverse transformation: Scaled by $1/\sqrt{2}$ and rotated counterclockwise by 45°.

Algorithm 2 can also be applied to models that use distances induced by any p-norm, with $p > 1$, including the usual Euclidean distance: Each robot r just "ignores" any point p such that $\|r - p\|_\infty > V$, thus pretending to be in the infinity norm model. Of course, this is guaranteed to terminate correctly only if the initial conditions given in Section 2.1 are met, i.e., if $G(0)$, computed with the infinity norm, is connected.

In particular, when using the Euclidean distance, our protocol and proofs work if $G(t)$ is constructed by connecting pairs of robots that are within Euclidean distance $V/\sqrt{2}$, as opposed to V. Moreover, we are confident that even this constraint on the initial distance graph can be dropped, by a simple adaptation of Algorithm 2 to circular visible areas. Due to space limitations, we are unable to discuss the topic further in this paper.

Acknowledgments

We would like to thank Paola Flocchini, Nicola Santoro, and Peter Widmayer, who contributed to the writing of this paper by sharing their ideas.

References

[1] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. A distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transaction on Robotics and Automation*, 15(5):818–828, 1999.

[2] M. Cieliebak. Gathering non-oblivious mobile robots. In *6th Latin American Conference on Theoretical Informatics (LATIN)*, LNCS 2976, pages 577–588, 2004.

[3] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. The power of lights: Synchronizing asynchronoys robots using visibile bits. In *The* 32^{nd} *International Conference on Distributed Computing Systems (ICDCS)*, 2012 (to appear).

[4] X. Défago and S. Souissi. Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. *Theoretical Computer Science*, 396(1-3):97–112, 2008.

[5] Y. Dieudonné, O. Labbani-Igbida, and F. Petit. Circle formation of weak mobile robots. *ACM Transactions on Autonomous and Adaptive Systems*, 3(4), 2008.

[6] Y. Dieudonné, F. Petit, and V. Villain. Leader election problem versus pattern formation problem. In *International Symposium on Distributed Computing (DISC)*, LNCS 6343, pages 267–281, 2010.

[7] A. Efrima and D. Peleg. Distributed models and algorithms for mobile robot systems. In *33rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, LNCS 4362, pages 70–87, 2007.

[8] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of robots with limited visibility. *Theoretical Computer Science*, 337(1-3):147–168, 2005.

[9] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Arbitrary pattern formation by asynchronous oblivious robots. *Theoretical Computer Science*, 407:412–447, 2008.

[10] D. Peleg. Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In *7th International Workshop on Distributed Computing (IWDC)*, LNCS 3741, pages 1–12, 2005.

[11] Giuseppe Prencipe. The effect of synchronicity on the behavior of autonomous mobile robots. *Theory of Computing Systems (TOCS)*, 38(5):539–558, 2005.

[12] S. Souissi, X. Défago, and M. Yamashita. Using eventually consistent compasses to gather memory-less mobile robots with limited visibility. *ACM Transactions on Autonomous and Adaptive Systems*, 4(1):1–27, 2009.

[13] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: formation of geometric patterns. *Siam Journal on Computing*, 28(4):1347–1363, 1999.

[14] M. Yamashita and I. Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411(26-28), 2010.

Appendix

Definition 7 (Horizontal and Vertical Distances)**.** *Given two distinct robots r and s, we define the* Horizontal Distance *at time t between r and s, shortly $HD(r, s, t)$, the distance between the vertical axes where r and s lie at t. Similarly, we define the* Vertical Distance *between r and s, shortly $VD(r, s, t)$, the distance between the horizontal axes where r and s lie at t. The reference to t will be dropped where not necessary.*

Proof. [**Lemma 6**] By Observation 2, the movement of each robot is monotonically increasing with respect to both the x-coordinate and the y-coordinate. Also, by Observations 2 and 4, no robot can ever pass f; that is, each robot converges towards a point.

If all robots have the same convergence point, then, by previous observations, this point must be f, and the lemma follows. Thus, let us assume that there is more than one convergence point, and let a be the leftmost and bottomost of them, and let $\mathbb{A}$ be the set of robots tha converge towards a. Also, let Ψ_a and Γ_a be the vertical and the horizontal axis passing through a, respectvely. Note that at least one robot $r \in \mathbb{A}$ must be within distance V from a robot s that is not converging towards a. Otherwise, either all robots are converging towards a, or the robots in $\mathbb{A}$ are at a distance greater than V from all the others: The first case is not possible by hypothesis; in the second case, we would have a contradiction by Corollary 1. We distinguish the possible cases.

1. If a is the only convergence point on Ψ_a (refer to the example depicted in Figure 4.a), then, by hypothesis, there must be at least another convergence point to the right of a: Let Ψ be the vertical axis passing through the first convergence point to the right of a, and δ be the distance between Ψ_a and Ψ. Let us also consider the time instant t when all robots are at a distance closer than $\delta/5$ from their respective convergence points; note that, up to time t, r and s are still within distance V. Also, at this time, let Ψ' be any vertical axis between the rightmost robot among those in $\mathbb{A}$, and Ψ_a. By construction, within finite time, at least one robot in $\mathbb{A}$ will pass Ψ'; let r^* be the first one. According to the algorithm, in order for r^* to pass Ψ', it has to see at least a robot to its right, hence to the right of Ψ_a. By construction, all robots to the left of r^* are within distance δ_5, and the robots r^* sees to its right are at a distance d such that $\delta/2 < d < \delta$; that is, the

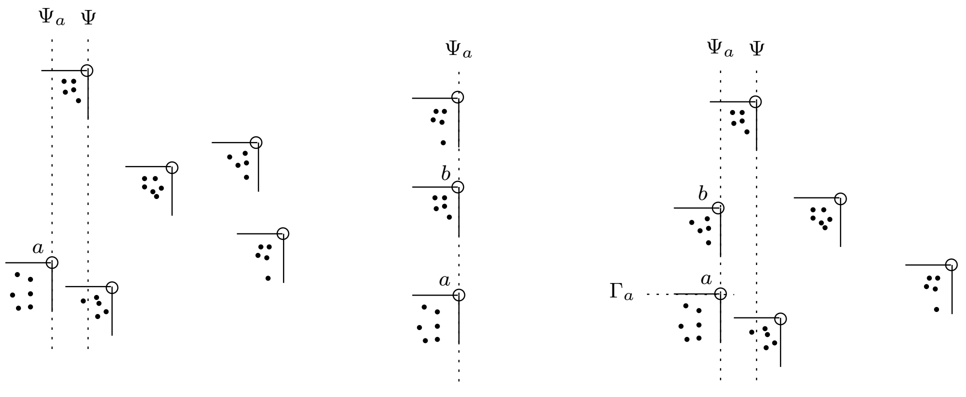

Figure 4: Proof of Lemma 6. Each empty circle represents a convergence point.

movement r^* executes towards its right is greater than $\delta/4$, hence r^* passes Ψ_a, having a contradiction.

2. If there is another convergence point on Ψ_a, and there are no other convergence points to the right of Ψ_a, let b be the closest convergence point to a (refer to the example depicted in Figure 4.b). Then, by construction, b is above a and, by Corollary 1, at least a robot $r \in \mathbb{A}$ must be within distance V from one of the robots converging towards b. In this case, the proof proceeds similarly to the previous case, where the rightward movements are changed into upwards movements.

3. If there is another convergence point on Ψ_a, and there are other convergence points to the right of Ψ_a, then, let b be the closest convergence point to a on Ψ_a (refer to the example depicted in Figure 4.c). In this case, if no robot in $\mathbb{A}$ is within distance V from any other robot to the right of Ψ_a, then the lemma follows by previous Case 2.

   Otherwise, let Ψ be the vertical axis passing through the first convergence point to the right of a, δ' be the distance bewteen Ψ_a and Ψ, $\delta'' = VD(a, b)$, and $\delta = \min\{\delta', \delta''\}$. Let us also consider the time instant t when all robots are at a distance closer than $\delta/5$ from their respective convergence points. Let us now consider at this time the vertical axis Ψ' and the horizontal axis Γ' passing through the rightmost and the topmost robots in $\mathbb{A}$, respectively. By construction, within finite time, at least one robot in $\mathbb{A}$ will pass either Ψ' or Γ'; let r^* be the first one.

   According to the algorithm, in order for r^* to pass either Ψ' or Γ', it has to see robots to the right of Ψ_a and/or above Γ_a. If r^* sees only robots to the right of Ψ_a, then the proof continues similarly to previous Case 1. If r^* sees only robots above Γ_a, then the proof continues similarly to previous Case 2. Otherwise, r^* sees robots both to the right of Ψ_a and above Γ_a; according to the algorithm, r^* chooses as direction of movement either a peak or a valley (in $\mathcal{NE}(r^*)$) of the contour of r^*. By construction, the distance d between such a point and r^* is such that $\delta/2 < d < \delta$; that is, r^* moves inside $\mathcal{NE}(r^*)$ of a distance greater than $\delta/4$, thus passing either Ψ_a or Γ_a, thus having a contradiction.

□