CrossMark

Getting close without touching: near-gathering for autonomous mobile robots

Linda Pagli[1] · Giuseppe Prencipe[1] · Giovanni Viglietta[2]

Abstract In this paper we study the NEAR-GATHERING problem for a finite set of dimensionless, deterministic, asynchronous, anonymous, oblivious and autonomous mobile robots with limited visibility moving in the Euclidean plane in Look–Compute–Move 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 solving the NEAR-GATHERING problem is that it makes it possible to overcome the restriction of having robots with limited visibility. Hence it allows to exploit all the studies (the majority, actually) done on this topic in the unlimited visibility setting. Indeed, after the robots get close enough to each other, they are able to see all the robots in the system, a scenario that is similar to the one where the robots have unlimited visibility. We present the first (deterministic) algorithm for the NEAR-GATHERING problem, to the best of our knowledge, which allows a set of autonomous mobile robots to nearly gather within finite time without ever colliding. Our algorithm assumes some reasonable conditions on the input configuration (the NEAR-GATHERING problem is easily seen to be unsolvable in general). Further, all the robots are assumed to have a compass (hence they agree on the "North" direction), but they do not necessarily have the same handedness (hence they may disagree on the clockwise direction). We also show how the robots can detect termination, i.e., detect when the NEAR-GATHERING problem has been solved. This is crucial when the robots have to perform a generic task after having nearly gathered. We show that termination detection can be obtained even if the total number of robots is unknown to the robots themselves (i.e., it is not a parameter of the algorithm), and robots have no way to explicitly communicate.

1 Introduction

Consider a distributed system whose entities are a finite set of dimensionless *robots* or *agents* that can freely move on the Euclidean plane, operating in *Look–Compute–Move* (LCM) cycles. During each cycle, a robot takes a snapshot of the positions of the other robots (*Look*); executes a deterministic protocol, the same for all robots, using the snapshot as an input (*Compute*); and moves towards the computed destination (*Move*). After each cycle, a robot may stay idle 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 this model, time is assumed to be discrete, and at each time instant *all* robots are activated, obtain the same snapshot, compute and move towards the computed destination; thus, no computation or

✉ Giuseppe Prencipe
prencipe@di.unipi.it; giuseppe.prencipe@unipi.it

Linda Pagli
linda.pagli@unipi.it

Giovanni Viglietta
viglietta@gmail.com

[1] Dipartimento di Informatica, Università di Pisa, Pisa, Italy

[2] School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada

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*, if they sense the positions of *all* other robots, or have *limited visibility*, if they sense just a portion of the plane, up to a given distance V [2,12]. With respect to memory, the robots can either be *oblivious*, if they have access only to the information sensed or computed during the current cycle (e.g., [20]), or *non-oblivious*, if they have the capability to store the information sensed or computed since the beginning of the computation (e.g., [3,21,22]). 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., [13], and the one where the robots have *no agreement* on their local coordinate axes, e.g., [21,22]. In the literature, there are studies that tackle also the scenarios in between; for instance, when the robots agree on the direction of only one axis, or there is agreement just on the orientation of the coordinate system (i.e., right-handed or left-handed), e.g., [10]. With respect to direct communication, some recent studies introduced the use of external signals or lights to enhance the capabilities of mobile robots. These were first suggested in [19], and were also referenced in [11], which provided the earliest indication that incorporating some simple means of signaling in the robot model might positively affect the power of the team. Recently, a study that tackles this particular capability more systematically has been presented in [7].

In this paper, we solve the NEAR-GATHERING problem: the robots are required to get close enough to each other, without ever 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 solving the NEAR-GATHERING problem is that it allows to overcome the limitations of having robots with limited visibility, and it makes it possible to exploit all the studies (the majority, actually) done in the unlimited visibility setting, such as, for instance, the *Arbitrary Pattern Formation Problem* [10,13,21,22], or the *Uniform Circle Formation* (e.g., [8,9]). Indeed, if all the robots get close enough, they eventually become able to see one another, reaching a configuration in which they may be assumed to have unlimited visibility (recall that the robots are

dimensionless). Since most of the studies related to the unlimited visibility case assume a starting configuration where no two robots coincide (i.e., they do not share the same location in the plane), it is of crucial importance to ensure that no collision occurs during the process.

A problem that is similar to NEAR-GATHERING is the *gathering* problem, in which the robots have to meet, within finite time, in a point of the plane not agreed upon in advance. Note that the gathering problem requires all robots to actually become coincident, while in NEAR-GATHERING they have to *approach* a point, but they are not allowed to collide with each other. Another related problem is the *convergence* problem, in which the robots have only to approach a point in the plane and converge to it in the limit, but they do not necessarily have to reach it in finite time, and they may collide with each other in the process. Hence, the convergence problem is easier than both gathering and NEAR-GATHERING. For a discussion on previous solutions to the problems of gathering and convergence, and how they fail to solve NEAR-GATHERING, refer to Sect. 3.1.

A preliminary solution to the NEAR-GATHERING problem has been presented by the authors in [18]; however, that solution worked with distances induced by the infinity norm.[1] In this paper we drop that assumption, presenting a more general solution that works with the usual Euclidean distance. We emphasize that the technique used in this paper can be easily adapted to solve the NEAR-GATHERING problem under any p-norm distance with $p \geq 1$, including the infinity norm distance used in [18]. We also note that, in contrast with [18] and other works on limited visibility, such as [12], we only assume that the robots have agreement on one axis (as opposed to both axes). In order to detect termination, the algorithm in [18] requires either the knowledge of the number of robots in the system, or the ability of the robots to communicate through visible lights that can be turned on or off. In the present paper we are able to drop both requirements, and still detect termination.

It is worth mentioning that in [18] a tacit assumption is made on the starting positions of the robots. Namely, we consider the graph on the robot set, with an edge connecting two robots if their initial distance is at most D, where D is a known constant that is smaller than the visibility radius of the robots (but may be arbitrarily close to it). The assumption is that such a graph is connected. Here we make this assumption explicit, and we give a more rigorous proof of our algorithm's correctness. Finally, we remark that, since the algorithm presented here is for the ASYNC model, it solves the problem *a fortiori* also in the SSYNC and FSYNC models.

The organization of the paper is as follows: in Sect. 2 the formal definition of the robot model is presented; in Sect. 3

---

[1] The infinity norm of a vector $(x, y) \in \mathbb{R}^2$ is defined as $\|(x, y)\|_\infty = \max\{|x|, |y|\}$.

the collision-free algorithm that solves the NEAR-GATHE-RING problem is presented, after discussing why previous solutions to related problems fail to solve it; in Sect. 4 the correctness of our algorithm is proven. Finally, Sect. 5 concludes the paper, suggesting some directions for future research.

2 The model

The system is composed of a team of finitely many mobile entities, called *robots*, each representing a computational unit provided with its own local memory and capable of performing local computations. The robots are modeled as points in the Euclidean plane $\mathbb{R}^2$. Let $r(t)$ denote the "absolute" position of robot r at time t (i.e., with respect to an absolute coordinate system), where $0 \leq t \in \mathbb{R}$; also, we will denote by $r(t).x$ and $r(t).y$ the abscissa and the ordinate value of $r(t)$, 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 orthogonal coordinate system, centered at its location, and we assume that the local coordinate systems of the robots agree on the directions of the x- and y-axes. As discussed in Sect. 5, the algorithms that we present in this paper also works in the more restricted model in which the robots agree on the direction of just one axis, as illustrated in Fig. 1a. 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 to 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 visibility range. Also, when combined with the asynchronous behavior of the robots, it introduces a higher level of difficulty in the design of collision-free protocols. For instance, in the example depicted in Fig. 1b, robot s, in transit towards its destination, might be 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 possible 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. Robots are endowed with motorial capabilities, and can move freely in the plane. As a robot moves, its coordinate system is translated accordingly, in such a way the the robot's location is always at the origin.
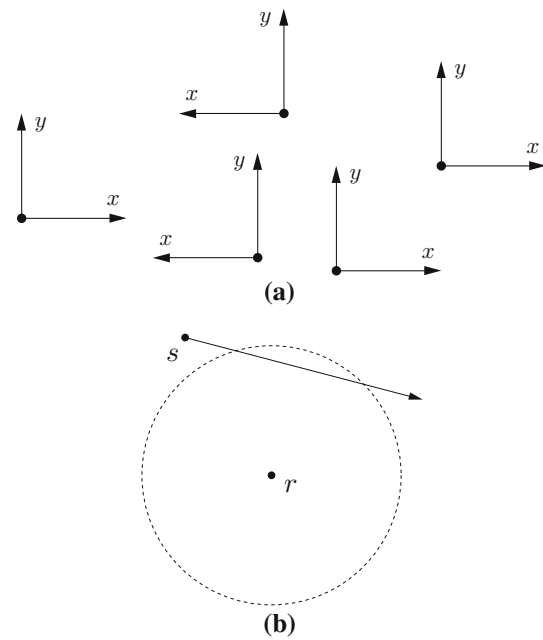

Fig. 1 **a** The robots in the swarm agree on the y-axis but not on the x-axis. **b** In the limited visibility setting a robot can only see robots that are within its radius of visibility. As a consequence, when s starts moving (*the left end of the arrow*), r and s do not see each other. While s is moving, perhaps 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

Each robot continually performs *Look–Compute–Move* (LCM) cycles, each consisting of three different *phases*:

(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 modeled as points, their positions in the plane are just the set of their coordinates).

(ii) *Compute* The robot executes its (deterministic) algorithm, using the snapshot as input. The result of the computation is a destination point, expressed in the robot's own coordinate system. There is no time limit to perform such a computation, although the robot can only compute finite sequences of algebraic functions on the visible robots' coordinates (actually, the algorithm proposed in this paper uses only arithmetic operations and square roots).

(iii) *Move* The robot moves monotonically towards the computed destination along a straight line; if the destination is the current location, the robot stays still (performs a *null movement*). No assumptions are made on the speed of the robot, as it may vary arbitrarily throughout the whole phase.

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, which are erased at each cycle. All robots are initially idle, until they are activated by a scheduler and start executing the *Look* phase of the first cycle. The amount of time to complete a cycle is assumed to be finite, but unpredictably variable from cycle to cycle and from robot to robot (i.e., the scheduler model is ASYNC), but the *Look* phase is assumed to be instantaneous. As a consequence, a robot may even stay still for a long time after it has reached its current destination point, before performing the *Look* phase of the next cycle, or it can stop for a while in the middle of a move and then proceed, etc. All these actions are controlled by the scheduler, which is an entity independent of the robots and their protocol, and may be seen as an "adversary" whose purpose is to prevent the robots from accomplishing their task.

The scheduler may also end the *Move* phase of a robot before it has reached its destination, forcing it to start a new cycle with a new input and a new destination: this feature is intended to model, for instance, a limit to a robot's motion energy. However, there exists a constant $\delta > 0$ such that, if the destination point computed by a robot has distance smaller than δ from the robot's current location, the robot is guaranteed to reach it; otherwise, it will move towards it by at least δ. Note that, without this assumption, the scheduler could make it impossible for a robot to ever reach its destination, even if the robot keeps computing the same destination point. For instance, the scheduler may force the robot to move by smaller and smaller amounts at every cycle, converging to a point that is not the robot's intended destination. Instead, if the robot cannot be interrupted by the scheduler before it has moved by at least δ, and it keeps computing the same destination point, it is guaranteed to reach it in finitely many cycles. The value of δ is not known to the robots, hence it cannot be used in their computations.

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

We stress that robots are modeled as just points in the plane, and as such they do not have an associated vector indicating their "heading" or "forward direction". Likewise, a robot's coordinate system never rotates, but only translates following the robot's movements. Moreover, all robots have the same *visibility radius* V, which is known to them and can be used in their computations. V also serves as a common unit distance for the robots.

2.1 Notation and assumptions

We will denote by $\mathcal{R} = \{r_1, \ldots, r_n\}$ the set of robots in the system. The purpose of this paper is to study the NEAR-GA-THERING problem:

Definition 1 (NEAR-GATHERING) The NEAR-GATHERING *problem* requires all robots to terminate their execution in a configuration such that there exists a disk of radius ε containing all the robots, where ε is a fixed constant, and no two robots occupy the same location.

All the robots are required to execute the same protocol during their *Compute* phase. The input to such a protocol is the snapshot of the robots' locations obtained by the executing robot during its previous *Look* phase, along with the visibility radius V (which is the same for all robots), and of course the value of ε.

The protocol executed by the robots must be independent of the initial configuration of the robots, and must make the robots solve the NEAR-GATHERING problem from any initial configuration. However, in the limited visibility model, this requirement is known to be too strong, and some additional assumptions must be made on the *initial distance graph* in order to make the problem solvable.

Definition 2 (*Initial distance graph* [12]) The *initial distance graph* $I = (\mathcal{R}, E)$ of the robots is the graph such that, for any two distinct robots r and s, $\{r, s\} \in E$ if and only if r and s are initially at distance not greater than the visibility radius V, i.e.,

$$\text{dist}(r(0), s(0)) \leq V.$$

By "dist" we denote the usual Euclidean distance. In [12] it is proven that, if the initial distance graph I is not connected, then the gathering problem may be unsolvable; the same result clearly holds also for the NEAR-GATHERING problem:

Observation 1 *If the initial distance graph I is not connected, the* NEAR-GATHERING *problem may be unsolvable.* $\square$

However, our solution to the NEAR-GATHERING problem requires a slightly more restrictive initial condition. Let σ be an arbitrary small and positive constant, and let $D = V - \sigma$.

Definition 3 (*Initial strong distance graph*) The *initial strong distance graph* $J = (\mathcal{R}, E)$ of the robots is the graph such that, for any two distinct robots r and s, $\{r, s\} \in E$ if and only if r and s are initially at distance not greater than D, i.e., $\text{dist}(r(0), s(0)) \leq D$.

In the following, we will assume that:

Assumption 1 The initial strong distance graph J is connected.

We remark that D (or at least lower bound on D) must be known to the robots. This is not much of a benefit to the robots in terms of raw computational power, since

V is already known to all the robots and can already be used in their computations as a common unit distance. Besides, by choosing σ to be small enough, the set of initial configurations ruled out by Assumption 1 becomes negligible.

The reasons why we need such a slightly more restrictive assumption are technical, and will become apparent in Sect. 4, when the correcntess of our algorithm will be proven. We stress that Assumption 1 only refers to the *initial* strong distance graph, while it does not require such a graph to be connected at all times. However, as we will prove in Sect. 4, our algorithm will indeed preserve the connectedness of a closely related distance graph throughout the execution.

Note that the definition of NEAR-GATHERING does not require the robots to avoid collisions during the execution, but it only requires them to occupy distinct locations when they all have terminated their execution. However, for NEAR-GA-THERING to be solvable, the robots must necessarily occupy distinct locations in the initial configuration, otherwise the scheduler could always activate coinciding robots simultaneously, and never allow them to occupy distinct locations. The algorithm we will describe in this paper is in fact collision-free, that is, it always prevents robots from colliding, provided that they start from distinct locations. As a by-product, our algorithm works regardless of the ability of the robots to detect the presence of more than one robot in the same location (called *multiplicity detection* in the literature [3,12]).

Another necessary assumption is that no robot is moving at time $t = 0$. If the robots are already moving when the execution starts, and two robots have the same destination point, nothing can prevent them from colliding. Moreover, after they have collided, the scheduler can force them to remain coincident forever, by activating them synchronously. If this happens, the NEAR-GATHERING cannot be solved.

Summarizing, we will make Assumption 1 on the initial configuration of the robots, and we will also assume that initially no robot is moving, and no two robots occupy the same location. The protocol executed by the robots in the *Compute* phase takes this input:

- an array of points expressed in the local coordinate system of the executing robot, denoting the locations of the visible robots observed during the previous *Look* phase;
- the visibility radius V (the same for all robots);
- the value of D (the same for all robots);
- the value of ε (required for termination).

Observe that the value of δ is *not* part of the input, and therefore the robots do not have a lower bound on the minimum distance that they are guaranteed to cover in a single *Move* phase.

3 The NEAR-GATHERING problem and its solution

In Sect. 3.1 we discuss some previous solutions to the gathering and convergence problems, explaining why they cannot be easily adapted to solve NEAR-GATHERING. Then, in Sect. 3.2 we give our solution to the NEAR-GATHERING problem.

3.1 Previous solutions to related problems

Gathering Of course, since the gathering problem requires all robots to collide, no solution to this problem is a valid solution to NEAR-GATHERING. However, we may wonder if a simple modification of an existing gathering algorithm may solve NEAR-GATHERING.

The gathering problem has been studied in the literature in all models but, to the best of our knowledge, the most pertinent paper is [12], which considers robots with limited visibility in the ASYNC setting. The algorithm in [12] assumes all robots to agree on the direction of both axes, and ideally it makes the leftmost and topmost robots move first, rightwards and downwards, until all the robots gather. According to the protocol, a robot r will occasionally compute a destination point that coincides with another visible robot s's location. To avoid this type of move, we may make r move toward s without reaching it. If we consider an initial configuration in which all robots lie on the same vertical line, the only robot that is allowed to move according to the algorithm in [12] is the topmost robot r. Moreover, if r moves downward without ever reaching the next robot, then no robot other than r will ever be able to move. Therefore, we ought to let robots other than r move, as well. Unfortunately, the proof of correctness of the algorithm, given in [12], strongly depends on the fact that the robots in the swarm move in a strictly ordered fashion. If we let any robot move, then we have to make sure that the visibility graph remains connected throughout the execution, and that the robots still converge to a single point. Clearly, even if a suitable adaptation of this idea can be effectively applied to solve NEAR-GATHERING, the modified protocol would require a radically new analysis and proof of correctness.

Convergence Several solutions to the convergence problem have been proposed, as well. If we manage to obtain a solution that also avoids collisions, we can successfully apply it to NEAR-GATHERING.

Perhaps the most natural strategy, at least in the unlimited visibility model, is to make all robots move to their center of gravity. This simple protocol has been analyzed in [4], and it has been proven correct even in the ASYNC model. In the limited visibility setting, the only relevant work, to the best of out knowledge, is [2], which gives a convergence algorithm that assumes the SSYNC scheduler. However, in the special case in which the robots' locations are the vertices of

a regular polygon and they are all mutually visible, both the center-of-gravity algorithm and the algorithm in [2] behave in the same way, and make any active robot move to the center of the polygon. Hence, if two robots are activated simultaneously from this configuration, they collide and fail to solve NEAR-GATHERING.

Therefore, we may modify the protocol and make each robot approach the center of gravity by, say, moving halfway towards it. We show that this protocol may still cause collisions in the ASYNC model, even in the very simple case in which the system consists of only two robots. Let r and s be two mutually visible robots, such that $r(0) = (0, 0)$ and $s(0) = (3124, 0)$. Let the scheduler activate r, which observes that the center of gravity is point $(1562, 0)$, and therefore computes the destination point $(781, 0)$ (i.e., the point half-way toward the center of gravity). Now the scheduler lets r start moving and, as soon as it reaches point $(52, 0)$, it temporarily delays the remaining part of the move and makes s quickly perform five complete cycles. As r is always seen in $(52, 0)$, s moves first to $(2356, 0)$, then to $(1780, 0)$, $(1348, 0)$, $(1024, 0)$, and finally to $(781, 0)$. Now the scheduler lets r finish its original move, and this causes a collision with s in $(781, 0)$. Observe that, even if the protocol does not make the robots move half-way toward the center of gravity, but to some other fraction of the distance, similar examples can be constructed in which the robots collide.

Further literature Several other papers considered the gathering or the convergence problems in various models, but these results are either not relevant to NEAR-GATHERING in our model, or they can be reduced to solutions already discussed above, and therefore discarded.

In [20], the gathering problem is studied for robots with limited visibility, the SSYNC scheduler, and *temporarily unreliable compasses*. In the special case in which the robots are close enough and their compasses are reliable, the proposed algorithm becomes equivalent to that of [12], which has already been analyzed and discussed.

The gathering problem is studied in [14] in the context of non-convex environments and limited visibility, but with the FSYNC scheduler. However, if the robots are close enough and they all see each other, the algorithm makes them all move to the center of the smallest enclosing circle. Hence, in the special case in which they form a small-enough regular polygon, they move to the center of gravity, and therefore the algorithm becomes equivalent to those of [2,4], which have already been discussed.

The convergence problem with limited visibility has been studied also for robots whose level of asynchronicity lies strictly between SSYNC and ASYNC. In [17], it is assumed that the time spent in a *Look* or *Move* phase is bounded, and the algorithm is a slight modification of that of [2]. In particular, it suffers from the issues that have already been discussed for [2].

On the other hand, in [16] the scheduler is *1-bounded* ASYNC, which means, roughly, that no robot can perform more than one *Look* phase between two consecutive *Look* phases of another robot. As it turns out, if the number of robots is even and they are vertices of a small-enough regular polygon, the algorithm makes them move to the center of gravity. Once again, this type of move has already been analyzed and discarded.

In [15], the gathering problem is considered for the SSYNC scheduler and the unlimited visibility setting. Here the focus is on the expected termination time of a randomized algorithm where the robots have some sort of *multiplicity detection*, i.e., the ability to detect the presence of more than one robot in the same location. Unfortunately, both algorithms presented in this paper make all robots move to the center of the smallest enclosing circle, except in some special cases. When applied to the NEAR-GATHERING problem, this approach suffers from the same issues of the center-of-gravity approach.

In [6], the gathering problem in ASYNC is studied for *fat* robots, i.e., robots that are modeled as solid discs rather than dimensionless points. Unfortunately, the problem is solved only for a swarm of at most four robots, and the technique involves a case analysis that does not generalize to bigger swarms. Therefore this solution is irrelevant to our problem.

The above result has been generalized in [1], which solves the gathering problem for any number of fat robots. The robots considered have an unlimited visibility radius, and therefore the limitations posed by a bounded visibility radius are not addressed in the paper. Additionally, letting fat robots collide is not an issue, but instead it is a necessary event that is sought by the algorithm. For these reasons, the approach of this paper can hardly be adapted to our problem.

Another work that considers the gathering problem for fat robots is [5], which works in the unlimited visibility setting and the FSYNC scheduler. Moreover, the gathering point is given as input to all the robots. Because of these differences with our model, it is impossible to extract a sound algorithm for NEAR-GATHERING from this work: indeed, the task of making such fat robots touch each other is simple, and the paper focuses on how to make robots slide around each other in order to occupy a small area. All these issues are meaningless in our model, and the real issues of our model become meaningless with fat robots.

3.2 Solving the NEAR-GATHERING problem

We conjecture that no solution to NEAR-GATHERING exists in the ASYNC model in which the robots do not agree at least on one axis. Therefore, in the following we will assume to have agreement on both axes, and in Sect. 5 we will observe that our solution works even in the case of agreement on just one axis.

The general high-level idea of the algorithm is to make the robots move upward and to the right, until they aggregate around the top-right corner of the smallest box that contains all of them. A robot's destination point is carefully computed, taking into account several factors. To avoid collisions, robots try to move in order, never "passing" each other, and never getting in each other's way. This is not a trivial task, because the visibility of the robots is limited, and they cannot predict the moves of the robots they cannot see. On the other hand, robots try to preserve mutual visibility by not moving too far from other visible robots, avoiding to leave them behind. This is supposed to prevent the robots from separating into different groups, which may be unaware of each other and aggregate around different points. As it turns out, the robots are unable to always preserve mutual visibility, but they can indeed preserve "mutual awareness", which is a concept that will be introduced shortly. These different behaviors are blended together and balanced in such a way that the robots are not only guaranteed to avoid collisions and remain mutually aware, but also to effectively aggregate around some point, and never "get stuck" or converge to different limit points. This is obtained by always making robots move by the greatest possible amount, compatibly with the above restrictions.

The details of our NEAR-GATHERING protocol are reported in Fig. 2. The protocol is executed by each robot during every *Compute* phase. In the following, we denote by r^* the robot that is currently executing the algorithm. The returned value is dp, which is the destination point for r^*. The algorithm computes separately the horizontal and the vertical components of the movement of r^*, i.e., $dp.x$ and $dp.y$. Note that the computation of the horizontal component $dp.x$ is symmetrical to the computation of the vertical component $dp.y$, hence any proposition that holds for the x coordinate holds symmetrically for the y coordinate.

Referring to the NEAR-GATHERING protocol and to Fig. 3, let D_1 and D_2 be the (closed) disks with radius $V - \rho/2$ and $V - \rho$, respectively, and center in the current position of r^*. Also, let S be the closed square circumscribed around D_2 (with sides parallel to the x- and y-axes), and $R = D_1 \cap S$. Finally, let H_1 and H_2 be the *halt zones* of r^*, and $\mathcal{NW}$ and $\mathcal{SE}$ the sets of visible robots in Q_1 and Q_2, respectively (note that Q_1 contains its right border, but not the bottom one; similarly, Q_2 contains its top border, but not the left one).

Because no robot ever moves leftwards or downwards, we give the following definition:

Definition 4 (*Move space*) The *Move Space* of a robot r at time t, denoted by $\mathcal{MS}(r, t)$, is the set

$$\left\{ (x', y') \in \mathbb{R}^2 \mid x' \geq r(t).x \land y' \geq r(t).y \right\}.$$

State *Look*

Take the snapshot of the positions of the visible robots, which returns, for each robot $r \in \mathcal{R}$ at distance at most V, $\mathtt{Pos}[r]$, the position in the plane of robot r, according to my coordinate system (i.e., my position is $(0, 0)$).

State *Compute* (returns destination point $dp = (dp.x, dp.y)$)

$\rho = \min \{V/4, \, V - D\}$;
$\varepsilon' = \min\{\varepsilon, \, \rho/2\}$;
$\mathcal{Z} = $ Set of visible robots (including myself);
If $\forall r_1, r_2 \in \mathcal{Z}$, $\mathrm{dist}(\mathtt{Pos}[r_1], \mathtt{Pos}[r_2]) \leq \varepsilon'$ **Then Terminate**;
$D_0 = $ Closed disk with radius V and center in $(0, 0)$;
$D_1 = $ Closed disk with radius $V - \rho/2$ and center in $(0, 0)$;
$D_2 = $ Closed disk with radius $V - \rho$ and center in $(0, 0)$;
$p_1 = $ Leftmost intersection between D_1 and the horizontal line through $(0, V - \rho)$;
$p_2 = $ Bottommost intersection between D_1 and the vertical line through $(V - \rho, 0)$;
$S = $ Full closed square circumscribed around D_2 with edges parallel to the x- and y-axes;
$R = D_1 \cap S$;
$Q_1 = $ Set of points of D_0 with positive y-coordinate and non-positive x-coordinate;
$Q_2 = $ Set of points of D_0 with positive x-coordinate and non-positive y-coordinate;
$H_1 = $ Set of points of $(R \setminus D_2) \cap Q_1$ whose x-coordinate is lower than $p_1.x$;
$H_2 = $ Set of points of $(R \setminus D_2) \cap Q_2$ whose y-coordinate is lower than $p_2.y$;
$\mathcal{NW} = \{r \in \mathcal{Z} \mid \mathtt{Pos}[r] \in Q_1\}$;
$\mathcal{SE} = \{r \in \mathcal{Z} \mid \mathtt{Pos}[r] \in Q_2\}$;
$dp.x = \min \left\{ \min_{r \in \mathcal{SE}} \{\mathtt{Pos}[r].x\}, \, \max_{r \in \mathcal{Z}} \{\mathtt{Pos}[r].x\}, \, \rho/2 \right\}$;
$dp.y = \min \left\{ \min_{r \in \mathcal{NW}} \{\mathtt{Pos}[r].y\}, \, \max_{r \in \mathcal{Z}} \{\mathtt{Pos}[r].y\}, \, \rho/2 \right\}$;
For Each $r \in \mathcal{Z}$ **Do**
  If $\mathtt{Pos}[r] \in H_1$ **Then** $dp.x = 0$;
  Else If $\mathtt{Pos}[r] \in R$ **Then**
    $s_1 = $ Leftmost intersection between $R \setminus H_1$ and the horizontal line through $\mathtt{Pos}[r]$;
    $dp.x = \min \{dp.x, \, \mathtt{Pos}[r].x - s_1.x\}$;
  If $\mathtt{Pos}[r] \in H_2$ **Then** $dp.y = 0$;
  Else If $\mathtt{Pos}[r] \in R$ **Then**
    $s_2 = $ Bottommost intersection between $R \setminus H_2$ and the vertical line through $\mathtt{Pos}[r]$;
    $dp.y = \min \{dp.y, \, \mathtt{Pos}[r].y - s_2.y\}$;
If $dp.x > dp.y$ **Then** $dp = (dp.x/2, 0)$; **Else** $dp = (0, dp.y/2)$;

State *Move*

$\mathtt{Move}(dp)$.

Fig. 2 The NEAR-GATHERING protocol

The destination point of r^* is computed according to the rules below:

1. r^* only moves rightward or upward (not diagonally) at every move. It moves by the greatest possible amount, compatibly with the following restrictions (this is needed for the algorithm's convergence, see Sect. 4.4).
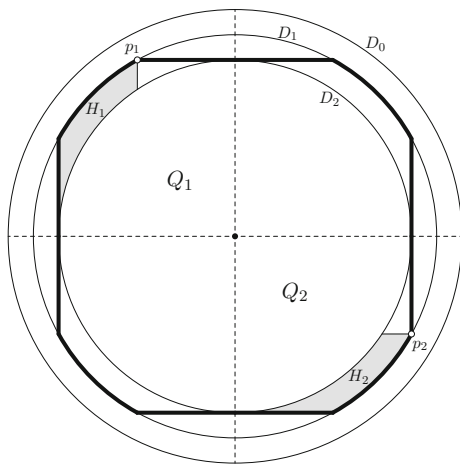
Fig. 3 Some of the elements computed by the NEAR-GATHERING protocol. The computing robot lies in the center, and the *thick line* represents the boundary of R

2. r^* never enters the *move space* of a visible robot, unless it already is in its move space (this is required for collision avoidance, Sect. 4.3);
3. r^* never moves to the right of (resp. above) the rightmost (resp. topmost) robot it can see (needed for convergence, Sect. 4.4);
4. If r^* sees a robot in the halt zone to its left (i.e., H_1 in Fig. 3), r^* does not move rightward. Symmetrically, if r^* sees a robot in the bottom halt zone (i.e., H_2 in Fig. 3), r^* does not move upward. This is needed for the preservation of mutual awareness, see Sect. 4.2;
5. If r^* sees a robot r in $R \setminus H_1$ (resp. $R \setminus H_2$), it moves so that r stays inside $R \setminus H_1$ (resp. $R \setminus H_2$) (preservation of mutual awareness, Sect. 4.2). Note that this does not guarantee *a priori* that r will actually stay inside $R \setminus H_1$ (resp. $R \setminus H_2$), since r moves asynchronously and independently of r^*;
6. The length of the so-computed movement is capped at $\rho/2$ (where $\rho = \min\{V/4, V - D\}$), and then halved (this is needed for both mutual awareness preservation and collision avoidance, see Sects. 4.2 and 4.3).

To correctly detect termination, we make sure that ε is not greater than $\rho/2$, by setting $\varepsilon' = \min\{\varepsilon, \rho/2\}$. This is necessary to prove Lemma 6.

4 Correctness

In this section, we will prove that the protocol reported in Fig. 2 correctly solves the NEAR-GATHERING problem. The proof will be articulated in three parts: first, we will prove that a suitably-defined distance graph remains connected during the execution; second, we will prove that no collisions occur during the movements of the robots; finally, we show that

all the robots converge to the same limit point, and correctly terminate their execution.

4.1 Preliminary definitions and observations

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

Observation 2 *No robot's x- or y-coordinate may ever decrease. No robot's x- and y-coordinates can both increase during the same move. Furthermore, a robot can move rightward (resp. upward) only if there is another robot strictly to the right of (resp. strictly above) its destination point.* □

Observation 3 *During each cycle, a robot travels a distance not greater than $\rho/4 \leq V/16$.* □

We may assume that, in the last line of the algorithm, if $dp.x$ and $dp.y$ are equal, then one of the two values $(dp.x/2, 0)$ and $(0, dp.y/2)$ is chosen arbitrarily as the destination point dp. With this assumption, the following holds.

Observation 4 *The algorithm is symmetric with respect to x- and y-coordinates.* □

Definition 5 (*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:

Definition 6 (*Destination point*) Given a robots r, we define the *destination point* $\mathrm{DP}(r, t)$ of r at time t as follows:

– If $r \in \mathbb{L}(t)$, then $\mathrm{DP}(r, t)$ is the point dp as computed in the next *Compute* phase after t (in the current cycle).
– If $r \in \mathbb{C}(t)$, then $\mathrm{DP}(r, t)$ is the point dp as computed in the current *Compute* phase.
– If $r \in \mathbb{M}(t)$, then $\mathrm{DP}(r, t)$ is the point dp as computed in the last *Compute* phase before t (in the current cycle).

From the previous definition, we can state the following:

Observation 5 *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, $\mathrm{DP}(r, t)$ is the point dp as computed in the next *Compute* phase after t (in the current cycle). Also, the destination point does not change in the next *Compute* and *Move* phases of r. □

The following proposition states a straightforward geometric fact (refer also to Fig. 3): among the segments contained in the annulus $D_1 \setminus D_2$, with one endpoint on the boundary of D_1 and the other endpoint on the boundary of D_2, the shortest are those that are collinear with the center of D_2. This will be often used in conjunction with Observation 3, to show that robots cannot lose visibility to each other under certain conditions.

Proposition 1 *The length of a segment contained in the annulus $D_1 \setminus D_2$, with one endpoint on the boundary of D_1 and the other endpoint on the boundary of D_2, is at least $\rho/2$.*

Proof Due to the rotational symmetry of the annulus, it is enough to prove the proposition for vertical segments only. The claim is equivalent to saying that, if $x \in [0, V - \rho] \subset \mathbb{R}$, then

$$\sqrt{(V - \rho/2)^2 - x^2} - \sqrt{(V - \rho)^2 - x^2} \geq \rho/2.$$

Let $f(x) = \sqrt{(V - \rho/2)^2 - x^2} - \sqrt{(V - \rho)^2 - x^2}$. Then, $f(0) = \rho/2$, and $f(x)$ is monotonically increasing on $[0, V - \rho]$. Indeed, the derivative of $f(x)$ on $(0, V - \rho)$ is

$$\frac{d}{dx} f(x) = x \left(\frac{1}{\sqrt{(V - \rho)^2 - x^2}} - \frac{1}{\sqrt{(V - \rho/2)^2 - x^2}} \right),$$

which is positive. $\square$

Let Q_1 be defined as in the NEAR-GATHERING protocol reported in Fig. 2; in the following, we will denote by $Q_1(r, t)$ the set Q_1 as robot r would compute it if it were in a *Compute* phase at time t (this set is expressed in the global coordinate reference system). A similar notation will be used for the other sets and points computed in our protocol (e.g., D_0, D_1, Q_2, etc.).

4.2 Preservation of mutual awareness

We define yet another notion of distance graph on the robots. This is useful, because in Corollary 2 we will prove that this graph remains connected throughout the execution of our NEAR-GATHERING protocol.

Definition 7 (*Intermediate distance graph*) The *intermediate distance graph* at time $t \geq 0$ is the graph $G(t) = (\mathcal{R}, E(t))$ such that, for any two distinct robots r and s, $\{r, s\} \in E(t)$ if and only if r and s are at distance not greater than $V - \rho/2$ at time t, i.e.,

$$\text{dist}(r(t), s(t)) \leq V - \rho/2,$$

where $\rho = \min\{V/4, V - D\}$.

Recall that, by assumption, the initial strong distance graph J is connected. This implies that $G(0)$ is connected, because $V - \rho/2 > D$, and hence $J \subseteq G(0)$. We will now prove that the connectedness of the intermediate distance graph is preserved during the entire execution of the algorithm. We will do so after introducing the notion of *mutual awareness*, in Definition 9.

First we define the auxiliary relation $\text{AW}(p, q)$.

Definition 8 Given two points $p, q \in \mathbb{R}^2$, we denote by $\text{AW}(p, q)$ the (symmetric) relation[2]

$$\|p - q\|_2 \leq V - \rho/2 \ \wedge \ \|p - q\|_\infty \leq V - \rho.$$

A simple fact to observe is the following (recall that $r(t)$ denotes the position of robot r at time t).

Observation 6 *For any two robots r and s, $\text{AW}(r(t), s(t))$ is equivalent to $s(t) \in R(r, t)$, which is equivalent to $r(t) \in R(s, t)$.* $\square$

Recall that, according to the algorithm, if a robot r sees a robot s in R, it will make its next move in such a way that s, as it was observed, does not exit R (see how s_1 and s_2 are computed in the algorithm). This is stated in the next observation.

Observation 7 *If r and s are two robots, $r \in \mathbb{L}(t)$ and $\text{AW}(r(t), s(t))$, then $\text{AW}(\text{DP}(r, t), s(t))$.* $\square$

Before introducing the next lemmas, let us recall that D_1 and D_2 are the closed disks with radius $V - \rho/2$ and $V - \rho$, respectively, and center in $(0, 0)$; S is the full closed square circumscribed around D_2 with sides parallel to the x- and y-axes; and that $R = D_1 \cap S$ (refer to the NEAR-GATHERING protocol, and to Fig. 3).

The next two lemmas are technical, and will be used in the proof of Lemma 3.

Lemma 1 *Let two robots r and s be given, with $r \in \mathbb{L}(t)$. If $\text{AW}(r(t), s(t))$ and $\text{AW}(r(t), \text{DP}(s, t))$, then*

$$\text{AW}(\text{DP}(r, t), s(t)) \text{ and } \text{AW}(\text{DP}(r, t), \text{DP}(s, t)).$$

Proof From Observation 7 it immediately follows that $\text{AW}(\text{DP}(r, t), s(t))$. Next we prove that $\text{AW}(\text{DP}(r, t), \text{DP}(s, t))$.

Without loss of generality we may assume that s is not moving horizontally at time t, that is, $s(t).x = \text{DP}(s, t).x$ and $0 \leq \text{DP}(s, t).y - s(t).y \leq \rho/4$ (cf. Observations 2–4). Let $\Delta = \text{DP}(r, t) - r(t)$; first observe that $\text{AW}(\text{DP}(r, t), \text{DP}(s, t))$ is equivalent to $\text{AW}(r(t), \text{DP}(s, t) - \Delta)$. Hence we have to

---

[2] By $\|a\|_2 = \sqrt{a.x^2 + a.y^2}$ we denote the usual Euclidean norm; by $\|a\|_\infty = \max\{|a.x|, |a.y|\}$ we denote the infinity norm.
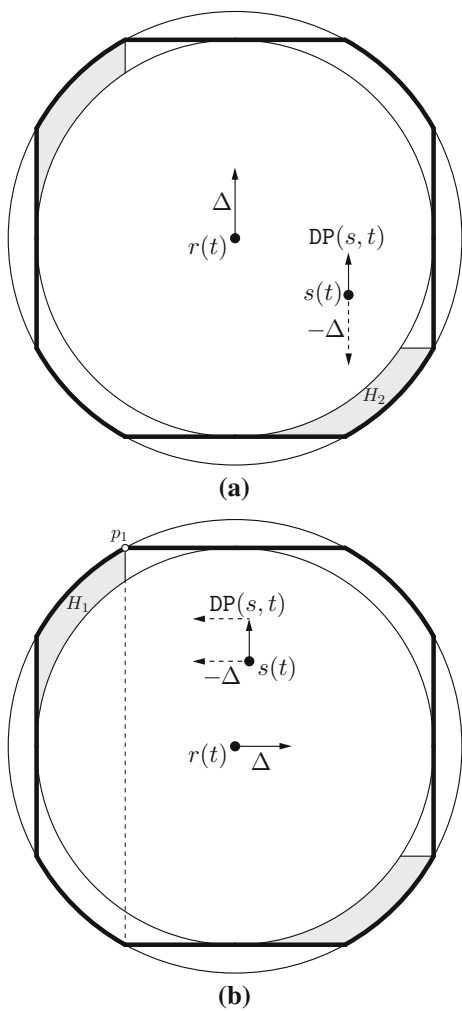
Fig. 4 Proof of Lemma 1. The *thick line* is the border of *R*. In **a**, *r* moves *vertically*. In **b**, *r* moves *horizontally* and *s* is to the *right* of *r* at time *t*
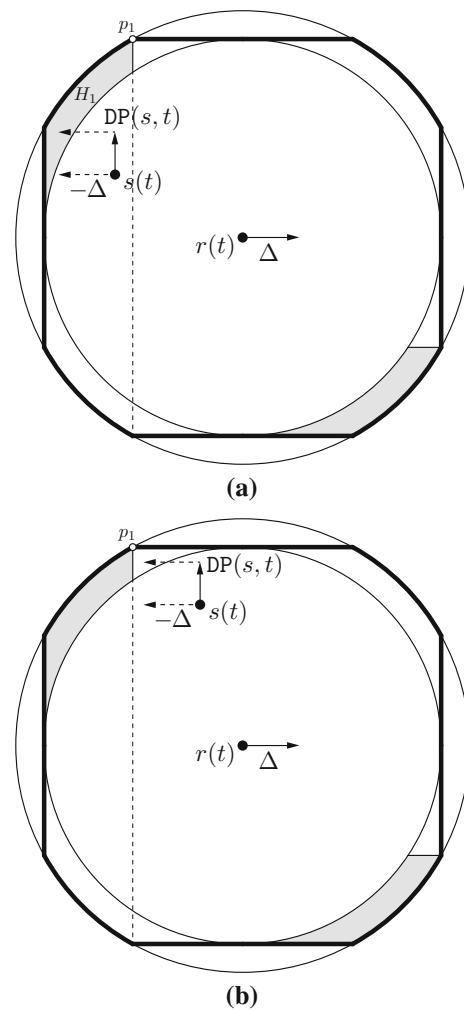

Fig. 5 Proof of Lemma 1. The *thick line* is the border of *R*. In **a**, *r* moves *horizontally* and *s* is to the *left* of p_1 at time *t*. In **b**, *s* is to the *right* of p_1 at time *t*

prove that the point $\mathrm{DP}(s, t) - \Delta$ lies in $R(r, t) = R$, provided that $s(t)$ and $\mathrm{DP}(s, t)$ do.

If Δ is the null vector, there is nothing to prove. So, let us assume first that $\Delta.x = 0$ and $\Delta.y > 0$. Referring to Fig. 4a, and by the convexity of R, it is sufficient to prove that $s(t) - \Delta$ lies in R, which is equivalent to $\mathrm{AW}(\mathrm{DP}(r, t), s(t))$, which has already been proven.

Otherwise, $\Delta.x > 0$ and $\Delta.y = 0$. Referring to Fig. 4b, if $s(t).x \geq r(t).x$, our claim that $\mathrm{DP}(s, t) - \Delta$ lies in $R(r, t)$ is trivially true, due to Proposition 1 and recalling that $\Delta.x \leq \rho/4$: indeed, $s(t)$ and $\mathrm{DP}(s, t)$ move leftward in the coordinate system of r by at most $\rho/4$, hence they stay to the right of p_1. Moreover, $s(t)$ cannot lie in H_1 or else r would not move rightward.

The only case left is that in which $s(t)$ belongs to $R \backslash H_1$ and lies to the left of $r(t)$. Recall that, according to the algorithm, $s(t) - \Delta$ belongs to $R \backslash H_1$ as well. Since $\mathrm{DP}(s, t).y - s(t).y \leq \rho/4$, and due to Proposition 1, it is

clear that $\mathrm{DP}(s, t) - \Delta$ lies in R, provided that $s(t) - \Delta$ lies to the left of p_1 (see Fig. 5a). Otherwise (see Fig. 5b), if $p_1.x \leq s(t).x - \Delta.x < r(t).x$, the claim follows from the fact that $\mathrm{DP}(s, t).y \leq p_1.y$ (because by assumption $\mathrm{AW}(r(t), \mathrm{DP}(s, t)))$, and therefore $\mathrm{DP}(s, t) - \Delta$ lies below p_1 and to its right. □

Lemma 2 *Let two robots r and s be given, with $r \in \mathbb{L}(t_r)$ and $t_s = Last(s, t_r)$. If $\mathrm{AW}(r(t_s), s(t_s))$ and $\mathrm{AW}(r(t_r), s(t_r))$, then $\mathrm{AW}(r(t_r), \mathrm{DP}(s, t_r))$.*

Proof From $t_s = Last(s, t_r)$ it follows that $t_s \leq t_r$. If $t_s = t_r$, then $s \in \mathbb{L}(t_r)$ and, due to Observation 7, $\mathrm{AW}(\mathrm{DP}(s, t_r), r(t_r))$, which is our claim. So let us assume that $t_s < t_r$. If $\mathrm{DP}(s, t_s) = s(t_s)$, there is nothing to prove, because in this case $\mathrm{DP}(s, t_r) = \mathrm{DP}(s, t_s) = s(t_s) = s(t_r)$. So we may assume that s moves strictly vertically (cf. Observation 4), and therefore $\mathrm{DP}(s, t_s).x = s(t_s).x$ and $s(t_s).y <$

$\mathrm{DP}(s, t_s).y \leq s(t_s).y + \rho/4$. Let $\Delta = \mathrm{DP}(s, t_s) - s(t_s)$. Also observe that, by definition of t_s, $\mathrm{DP}(s, t_r) = \mathrm{DP}(s, t_s)$.

We reason by considering the "point of view" of robot r. Let $\Delta' = r(t_r) - r(t_s)$. Hence $\mathrm{DP}(s, t_r) - r(t_r) = \mathrm{DP}(s, t_s) - \Delta' - r(t_s)$. In other terms, as a consequence of r moving upward and rightward (by Δ') between t_s and t_r, $\mathrm{DP}(s, t)$ moves downward and leftward in the coordinate system of r, as t varies from t_s to t_r.

Recall that $\mathrm{AW}(r(t_r), s(t_r))$ by hypothesis, and hence $s(t_r) \in R(r, t_r)$. If $s(t_r).y \leq r(t_r).y$, then, by Proposition 1, $\mathrm{DP}(s, t_r) \in R(r, t_r)$, as desired. Therefore, assume that $s(t_r).y > r(t_r).y$. This also implies that

$$\mathrm{DP}(s, t).y > r(t).y, \quad \forall t \in [t_s, t_r].$$

Note that $|s(t).x - r(t).x| \leq V - \rho$, for every $t \in [t_s, t_r]$. Indeed, the inequality holds at times t_s and t_r by the hypotheses of the lemma, and moreover $s(t).x$ is independent of $t \in [t_s, t_r]$, while $r(t).x$ may only increase.

Let $t' = First(r, t_s)$. We claim that both $s(t')$ and $\mathrm{DP}(s, t')$ belong to $R(r, t')$. Assume first that r moves upward (or stays still) between t_s and t'. Then, by Observation 7 and the convexity of R, the segment with endpoints $s(t_s)$ and $\mathrm{DP}(s, t_s)$ lies in $R(r, t_s)$. If such a segment moves downward in the coordinate system of r (as a consequence of r moving upward), and, at time t', s lies strictly below $R(r)$, this implies that $\mathrm{DP}(s, t').y$ cannot be greater than $r(t').y$, due to Proposition 1 (recall that $\mathrm{DP}(s, t').y - s(t').y \leq \rho/4$). This contradicts the assumption on $\mathrm{DP}(s, t').y$ made in the previous paragraph.

So, let r move rightward, and let $r(t') = r(t_s) + \Delta''$, with $0 < \Delta''.x \leq \rho/4$. Hence, if $s(t_s).x \geq r(t_s).x$, our claim is once again easily proven. Indeed, by Observation 7, $\mathrm{DP}(s, t_s)$ lies in $R(r, t_s)$, as well as $s(t_s)$. Then, by Proposition 1, these two points cannot move outside of $R(r)$ as r moves rightward by at most $\rho/4$, provided that $s(t_s).x = \mathrm{DP}(s, t_s).x \geq r(t_s).x$. So, let us assume that $s(t_s).x < r(t_s).x$.

Since by hypothesis s moves strictly upward based on a *Look* performed at time t_s, it means that $r(t_s) \notin H_2(s, t_s)$. Equivalently, $s(t_s)$ does not belong to the region symmetric to $H_2(r, t_s)$ with respect to $r(t_s)$, which we denote by $-H_2(r, t_s)$ (see Fig. 6a). As a consequence of the algorithm (in particular, by Rule 5 of Sect. 3.2), s does not compute a destination point that would make r enter the region H_2. Equivalently, in r's coordinate system, $\mathrm{DP}(s, t_s) \notin -H_2(r, t_s)$.

In particular, as illustrated in Fig. 6b, if $s(t_s) \in H_1(r, t_s) \backslash - H_2(r, t_s)$, then also $\mathrm{DP}(s, t_s) \in H_1(r, t_s) \backslash - H_2(r, t_s)$. Hence both $s(t')$ and $\mathrm{DP}(s, t')$ belong to $R(r, t')$ (recall that $|s(t').x - r(t').x| \leq V - \rho$).

Suppose now that $s(t_s) \in D_2(r, t_s)$. Note that, as a consequence of the algorithm (again, by Rule 5 of Sect. 3.2), $\mathrm{DP}(s, t_s).y \leq r(t_s).y + V - \rho$. Additionally, s has to move by
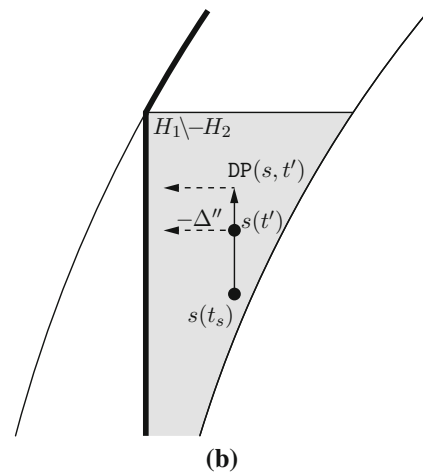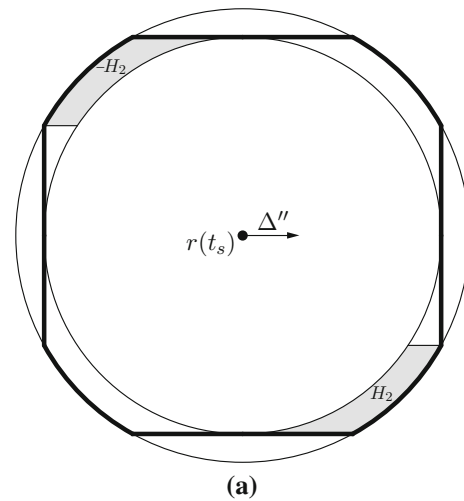

Fig. 6 Proof of Lemma 2. In **a**, the *gray area* in the *upper-left* corner is $-H_2(r, t_s)$. In **b**, a detail of the set difference $H_1 \backslash - H_2$ is shown

more than $\rho/2$ in the coordinate system of r in order to cross the boundary of $D_1(r)$. But $\|\Delta + \Delta''\|_2 \leq \rho/4 + \rho/4 = \rho/2$. As a consequence, both $s(t')$ and $\mathrm{DP}(s, t')$ still belong to $D_1(r, t')$, and therefore also to $R(r, t')$ (note that we already proved that $|s(t').x - r(t').x| \leq V - \rho$).

The only case left is when $s(t_s)$ lies in the lower-left area bounded by $R(r, t_s)$ and $D_2(r, t_s)$. By Proposition 1 and because $\mathrm{DP}(s, t_s).y - s(t_s).y \leq \rho/4$, $\mathrm{DP}(s, t')$ certainly lies in $R(r, t')$. However, we also know that $\mathrm{DP}(s, t').y > r(t').y$, and that $\mathrm{DP}(s, t').y - s(t').y \leq \rho/4$. Hence, again by Proposition 1, $s(t')$ must lie in $R(r, t')$ as well.

Now our claim is proven. If $t' = t_r$, we are done. Otherwise, we apply Lemma 1 by setting $t := t'$. As a result, $\mathrm{AW}(\mathrm{DP}(r, t'), s(t'))$ and $\mathrm{AW}(\mathrm{DP}(r, t'), \mathrm{DP}(s, t'))$. Let $t'' = First(r, t')$. By the convexity of R, it follows that both $s(t'')$ and $\mathrm{DP}(s, t'')$ belong to $R(r, t'')$ (recall that $\mathrm{DP}(s, t)$ does not depend on $t \in [t_s, t_r]$). If $t'' = t_r$, we are done. Otherwise, we keep applying Lemma 1 (with $t := t''$, etc.)

and repeating the previous reasoning, until we prove that $\mathrm{DP}(s, t_r) \in R(r, t_r)$, which concludes the proof. □

Now we are ready to give the full definition of *mutual awareness* and the related graph.

Definition 9 (*Mutual awareness*) Two distinct robots r and s are *mutually aware* at time t if both conditions hold:

1. $\mathrm{AW}(r(t_r), s(t_r))$, with $t_r = Last(r, t)$, and
2. $\mathrm{AW}(r(t_s), s(t_s))$, with $t_s = Last(s, t)$.

Definition 10 (*Mutual awareness graph*) The *mutual awareness graph* at time $t \geq 0$ is the graph $\widetilde{G}(t) = (\mathcal{R}, E(t))$ such that, for any two distinct robots r and s, $\{r, s\} \in E(t)$ if and only if r and s are mutually aware at time t.

We recall that $D = V - \sigma$, with $\sigma > 0$ arbitrary small. By definition of *Last* and of mutual awareness, we have the following.

Observation 8 *All the pairs of robots that are at* (*Euclidean*) *distance not greater than D from each other at time $t = 0$ are initially mutually aware. Hence $J \subseteq \widetilde{G}(0)$, and therefore $\widetilde{G}(0)$ is connected.* □

In the following lemma, we will prove that any two robots that are mutually aware at some point keep being so during the entire execution.

Lemma 3 *If robots r and s are mutually aware at time t, they are mutually aware at any time $t' \geq t$.*

Proof Let $(t_i)_{i \geq 0}$ be the strictly increasing sequence of time instants at which either r or s executes a *Look*; if both r and s execute a *Look* simultaneously, such a time instant appears only once in the sequence. Without loss of generality, we may assume that r and s first become mutually aware at time t_m, when r enters a *Look* phase.

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

1. $\mathrm{AW}(r(t_i), s(t_i))$,
2. $\mathrm{AW}(\mathrm{DP}(r, t_i), s(t_i))$,
3. $\mathrm{AW}(r(t_i), \mathrm{DP}(s, t_i))$,

which will clearly imply our claim (Condition 1 actually suffices).

Let $i = m$, and observe that Condition 1 holds by definition of mutual awareness. Moreover, by Lemma 2 with $t_r := t_m$, Condition 3 holds, too. Finally, Condition 2 is implied by Conditions 1 and 3 and by Lemma 1 with $t := t_m$.

Suppose now that $i > m$, and let the three conditions hold at every time t_j with $m \leq j \leq i - 1$. Without loss of generality, we may assume that $r \in \mathbb{L}(t_{i-1})$ (if $s \in \mathbb{L}(t_{i-1})$,

we just exchange r and s in our proof). By Conditions 1 and 3 on t_{i-1}, we have

$\mathrm{AW}(r(t_{i-1}), s(t_{i-1}))$ and

$\mathrm{AW}(r(t_{i-1}), \mathrm{DP}(s, t_{i-1}))$.

By Lemma 1 with $t := t_{i-1}$, we have also

$\mathrm{AW}(\mathrm{DP}(r, t_{i-1}), s(t_{i-1}))$

and

$\mathrm{AW}(\mathrm{DP}(r, t_{i-1}), \mathrm{DP}(s, t_{i-1}))$.

These are equivalent, respectively, to

$\mathrm{AW}(r(t_{i-1}), s(t_{i-1}) - \mathrm{DP}(r, t_{i-1}) + r(t_{i-1}))$ and

$\mathrm{AW}(r(t_{i-1}), \mathrm{DP}(s, t_{i-1}) - \mathrm{DP}(r, t_{i-1}) + r(t_{i-1}))$.

Collectively, $s(t_{i-1})$, $\mathrm{DP}(s, t_{i-1})$, $s(t_{i-1}) - \mathrm{DP}(r, t_{i-1}) + r(t_{i-1})$ and $\mathrm{DP}(s, t_{i-1}) - \mathrm{DP}(r, t_{i-1}) + r(t_{i-1})$ are four points whose convex hull C is either a rectangle or a segment (depending if r and s move orthogonally or parallel to each other between t_{i-1} and t_i). Because the vertices of C are contained in $R(r, t_{i-1})$ (cf. the definition of R in the algorithm), and because R is convex, C is entirely contained in $R(r, t_{i-1})$ (refer to Fig. 3).

Moreover, $r(t_i)$ (resp. $s(t_i)$) lies on the segment with endpoints in $r(t_{i-1})$ and $\mathrm{DP}(r, t_{i-1})$ (resp. $s(t_{i-1})$ and $\mathrm{DP}(s, t_{i-1})$). Let $r(t_i) = r(t_{i-1}) + \Delta_r$ and $s(t_i) = s(t_{i-1}) + \Delta_s$. So, the point $s(t_{i-1}) + \Delta_s - \Delta_r$ belongs to C, and therefore to $R(r, t_{i-1})$. In other terms,

$\mathrm{AW}(r(t_{i-1}), s(t_{i-1}) + \Delta_s - \Delta_r)$,

which is equivalent to $\mathrm{AW}(r(t_{i-1}) + \Delta_r, s(t_{i-1}) + \Delta_s)$, and to $\mathrm{AW}(r(t_i), s(t_i))$. Hence Condition 1 holds at t_i.

Once again, without loss of generality, we may assume that $r \in \mathbb{L}(t_i)$. Then, Condition 3 at t_i follows from Condition 1 and Lemma 2 with $t_r := t_i$. Condition 2, on the other hand, follows from Conditions 1 and 3, and from Lemma 1 with $t := t_i$. □

Corollary 1 $\widetilde{G}(t)$ *is connected at any time $t \geq 0$.*

Proof $\widetilde{G}(0)$ is connected by Observation 8. By Lemma 3, $\widetilde{G}(0)$ is a subgraph of $\widetilde{G}(t)$, and therefore $\widetilde{G}(t)$ is connected. □

Corollary 2 $\widetilde{G}(t) \subseteq G(t)$, *and therefore $G(t)$ is connected at any time $t \geq 0$.*

Proof Suppose that robots r and s are mutually aware at time t. Then, by Lemma 3, they are mutually aware at any

time after t, regardless of the scheduler's choices. Moreover, observe that the proof of Lemma 3 goes through even if the scheduler can stop the robots before they have moved by at least δ (recall that the fairness assumption of our robot model normally forbids the scheduler to interrupt a robot's *Move* phase before it has moved by at least δ).

Let us therefore modify the execution of r and s, and let the scheduler interrupt their *Move* phase precisely at time t, regardless of how much they have actually moved during that phase. By the previous observations, r and s are mutually aware at time $t' = First(r, t)$, and additionally $r(t) = r(t')$ and $s(t) = s(t')$. Hence, by definition of mutual awareness, r and s are at (Euclidean) distance not grater than $V - \rho/2$ at time t', and therefore also at time t.

This implies that $\widetilde{G}(t) \subseteq G(t)$, and hence that $G(t)$ is connected, by Corollary 1. □

4.3 Collision avoidance

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

Lemma 4 *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 during their execution. Because $r(t)$ and $s(t)$ are continuous functions, there exists a minimum time instant $t > 0$ at which $r(t) = s(t) = p$. At least one robot, say r, must make a strictly positive movement toward p, at some point. Let $t' < t$ be the last time at which r performs a *Look* phase such that $r(t') \neq p$. Recall that, by Observation 2, r and s move either upward or rightward at each move. Without loss of generality (cf. Observation 4), let us assume that r moves strictly rightward between t' and t. Then, by Observation 3, $0 < p.x - r(t').x \leq V/16$. Several cases arise.

If $s(0) = p$, then $s(t') = p \in Q_2(r, t')$, which is a contradiction because, by the algorithm (specifically, by Rule 2 of Sect. 3.2), $\text{DP}(r, t').x$ must be less than the x-coordinate of every robot in $Q_2(r, t')$, and therefore r cannot be found in p at time t.

If $s(0) \neq p$, then s performs at least one positive movement to reach p. Let $t'' < t$ be the last time at which s performs a *Look* phase such that $s(t'') \neq p$. By symmetry between r and s, we may assume that $t'' \leq t' < t$.

Suppose that s moves strictly upward between t'' and t (see Fig. 7a). Hence $0 < p.y - s(t'').y \leq V/16$. Because $t'' \leq t'$, it follows that $s(t') \in Q_2(r, t')$, which contradicts the fact that r reaches p in the next move.

Otherwise, s moves strictly rightward between t'' and t. Since $t'' \leq t'$, it follows that $s(t'').y = s(t').y = p.y$ (see Fig. 7b). $s(t')$ cannot lie to the right of $r(t')$, otherwise it would be in $Q_2(r, t')$, yielding a contradiction with the algorithm (Rule 2 of Sect. 3.2). Hence $s(t'').x \leq$
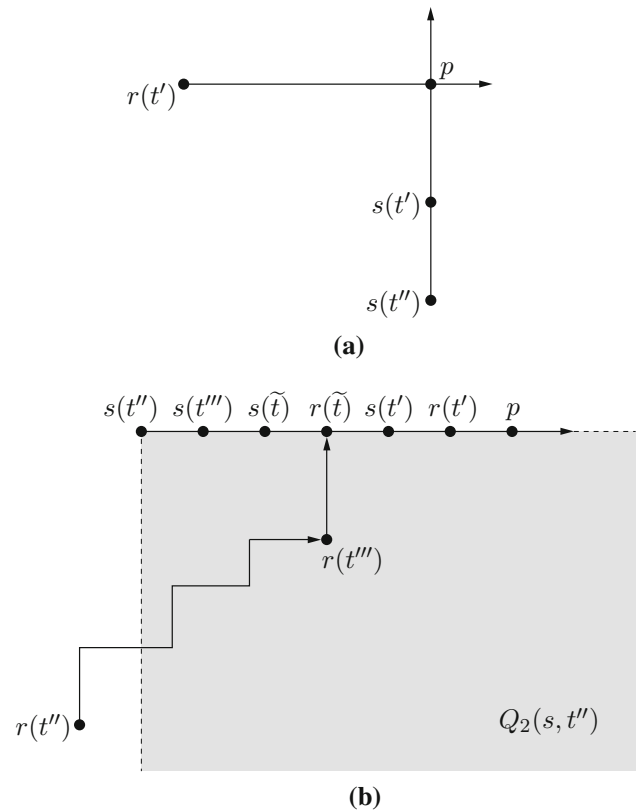

Fig. 7 Proof of Lemma 4. In **a**, s moves upward between t'' and t. In **b**, s moves rightward

$s(t').x \leq r(t').x < p.x$. We claim that $r(t'').y < s(t'').y$. Indeed, suppose by contradiction that $r(t'').y = s(t'').y$. If $r(t'').x > s(t'').x$, then $r(t'') \in Q_2(s, t'')$ and s computes a destination point that is not to the left of r, which again contradicts Rule 2 of the algorithm. Otherwise $r(t'').x \leq s(t'').x$, which implies that r and s collide between t'' and t', contradicting the minimality of t.

Because $r.y < p.y$ at time t'' and $r.y = p.y$ at time t', there is a time $\widetilde{t} \in (t'', t']$ at which $r.y$ first becomes equal to $p.y$. Note that $r(\widetilde{t}).x > s(\widetilde{t}).x$, otherwise r and s would collide between $\widetilde{t}$ and t'. Hence $r(\widetilde{t}) \in Q_2(s, \widetilde{t})$. Note also that $r(t'') \notin Q_2(s, t'')$, because $\text{DP}(s, t'').x \geq r(t'').x$. Since each move covers at most $V/16$, r performs more than one move between t'' and $\widetilde{t}$: if r enters $Q_2(s)$ for the last time from below, it must move vertically more than once; if r enters $Q_2(s)$ for the last time from the left, then it must turn upwards at some point (refer to Fig. 7b). More precisely, r performs at least one *Look* phase in $[t'', \widetilde{t})$, the last of which at time t''', and r moves strictly upward between t''' and $\widetilde{t}$. Then

$$s(t'').x \leq s(t''').x \leq s(\widetilde{t}).x < r(\widetilde{t}).x = r(t''').x.$$

It follows that $0 < r(t''').x - s(t''').x \leq V/16$. Moreover, $0 < s(t''').y - r(t''').y \leq V/16$, hence $s(t''') \in$

$Q_1(r, t''')$. This contradicts Rule 2 of the algorithm, because $\text{DP}(r, t''').y \geq s(t''').y$. ☐

4.4 Convergence and termination

In this final section, we will prove that the robots will converge to the same limit point (Lemma 5), and then finally that our NEAR-GATHERING algorithm is correct (Theorem 1).

Let ℓ be the point having the x-coordinate of the rightmost point in $\mathcal{I}$, and the y-coordinate of the topmost point in $\mathcal{I}$. That is,

$$\ell = \left(\max_{r \in \mathcal{R}} \{r(0).x\}, \max_{r \in \mathcal{R}} \{r(0).y\} \right).$$

Lemma 5 *If no robot ever terminates its execution, then all robots converge towards point ℓ.*

Proof Let an execution of the robot set $\mathcal{R}$ be fixed, in which no robot ever terminates. By Observation 2, the movement of each robot is monotonically increasing with respect to both the x-coordinate and the y-coordinate. Also, at any time, each robot's coordinates are bounded from above by the coordinates of ℓ. It follows that each robot r converges towards a point, denoted by $\text{LIM}(r)$, such that $\text{LIM}(r).x \leq \ell.x$ and $\text{LIM}(r).y \leq \ell.y$.

If all robots have the same convergence point, then this point must be ℓ, because there is a robot whose x-coordinate is constantly $\ell.x$ and a (possibly distinct) robot whose y-coordinate is constantly $\ell.y$. Hence, in this case the lemma follows. Thus, let us assume that there is more than one convergence point. Let $\lambda \in \mathbb{R}^+$ be any positive number such that:

- $\lambda \leq \text{LIM}(r).x - \text{LIM}(s).x$ for every $r, s \in \mathcal{R}$ with $\text{LIM}(r).x > \text{LIM}(s).x$;
- $\lambda \leq \text{LIM}(r).y - \text{LIM}(s).y$ for every $r, s \in \mathcal{R}$ with $\text{LIM}(r).y > \text{LIM}(s).y$;
- $\lambda \leq V - \text{dist}(\text{LIM}(r), \text{LIM}(s))$ for every $r, s \in \mathcal{R}$ with $\text{dist}(\text{LIM}(r), \text{LIM}(s)) < V$;
- $\lambda \leq \min\{\rho/2, \delta\}$.

Because $\mathcal{R}$ is a finite set, there is a time t_0 at which, for every $r \in \mathcal{R}$,

$$\text{dist}(r(Last(r, t_0)), \text{LIM}(r)) < \lambda/3.$$

By definition of λ, if $\text{dist}(\text{LIM}(r), \text{LIM}(s)) < V$, then $\text{dist}(r(t), s(t)) < V$ for all $t \geq t_0$. On the other hand, if $\text{AW}(r(t), s(t))$ for some $t \geq t_0$, then in particular $\text{dist}(r(t), s(t)) \leq V - \rho/2$, and therefore

$$\text{dist}(\text{LIM}(r), \text{LIM}(s)) < V.$$

Let us choose $t_1 > t_0$ such that every robot in $\mathcal{R}$ executes at least one complete cycle between t_0 and t_1 (i.e., from a *Look* phase to the next). We further assume that, for every $r \in \mathcal{R}$, if $r(t_0).x < \text{LIM}(r).x$ (resp. $r(t_0).y < \text{LIM}(r).y$), then in at least one such cycle (i.e., executed between t_0 and t_1) r moves strictly rightward (resp. upward). Note that we can make this assumption because $\text{LIM}(r).x$ must be approached indefinitely by $r.x$, and therefore, if $r(t_0).x < \text{LIM}(r).x$, then r must make a rightward move at some point after t_0 (and similarly for y-coordinates and upward moves).

Let a be the lowest among the leftmost convergence points of the robots in $\mathcal{R}$, and let $\mathcal{A} \subset \mathcal{R}$ be the set of robots that converge towards a.

Suppose first that there exists some robot $s \in \mathcal{R} \setminus \mathcal{A}$ converging to $b \neq a$, such that $\text{dist}(a, b) < V$ and $b.x > a.x$. Let r be any rightmost robot of $\mathcal{A}$ at time t_1. As observed three paragraphs above, r and s can see each other at any time since t_0.

If $r.x < a.x$ then, by construction, there exists a time $t^* \in [t_0, t_1]$ at which r performs a *Look* phase, such that $r(t^*).x < \text{DP}(r, t^*).x$ and $r(First(r, t^*)).x = r(t_1).x$ (see Fig. 8a). According to the algorithm (specifically, by Rule 2 of Sect. 3.2), if r is able to compute such a destination point, it means that no robot of $\mathcal{A}$ lies in $Q_2(r)$ at time t^*. Therefore, by definition of λ and by construction, every robot in $Q_2(r, t^*)$ has an x-coordinate that is greater than $a.x + 2\lambda/3$. Because $s(t^*).x > a.x + 2\lambda/3$ as well, it follows that $\text{DP}(r, t^*).x > r(t^*).x + \lambda/3$ (observe that no robot in $Q_1(r, t^*)$ can prevent r from moving rightward by at least $\rho/4 > \lambda/3$, due to Proposition 1). Additionally, $\lambda/3 < \delta$, hence r actually moves by more than $\lambda/3$. But $r(t^*).x + \lambda/3 > a.x$, contradicting the fact that $r.x$ monotonically converges to $a.x$.

Otherwise, $r.x = a.x$ holds. Then, let $t^* = First(r, t_1)$. At time t^*, r sees no robot q with $r(t^*).x < q(t^*).x \leq r(t^*).x + 2\lambda/3$. Moreover, r sees s, and $s(t^*).x > r(t^*).x + 2\lambda/3$. Hence, $\text{DP}(r, t^*)$ has distance greater than $\lambda/3$ from $r(t^*)$, and r actually moves rightward or upward by more than $\lambda/3 < \delta$. When r is done moving, either $r.x > a.x$ or $r.y > a.y$, contradicting the fact that $\text{LIM}(r) = a$.

Suppose now that there is no limit point $b \neq a$ such that $\text{dist}(a, b) < V$ and $b.x > a.x$. By Corollary 1, $\widetilde{G}(t_0)$ is connected, hence there exist robots $r' \in \mathcal{A}$ and $s' \in \mathcal{R} \setminus \mathcal{A}$ that are mutually aware at time t_0 (and also at any time $t \geq t_0$, by Lemma 3). Let $b = \text{LIM}(s')$. Then, either $\text{dist}(a, b) \geq V$ or $a.x = b.x$ (recall that a is a leftmost convergence point). However, observe that if $\text{dist}(a, b) \geq V$, then r' and s' cannot be mutually aware at any time $t \geq t_0$, because $\text{dist}(r'(t), s'(t)) > V - \rho/2$. Therefore, $a.x = b.x$ and $a.y < b.y < a.y + V$.

Let r'' be any topmost robot of $\mathcal{A}$ at time t_1. By Corollary 2, r' sees s' at any time $t \geq t_0$. Then, by definition of λ and t_0, also r'' sees s' at any time $t \geq t_0$.
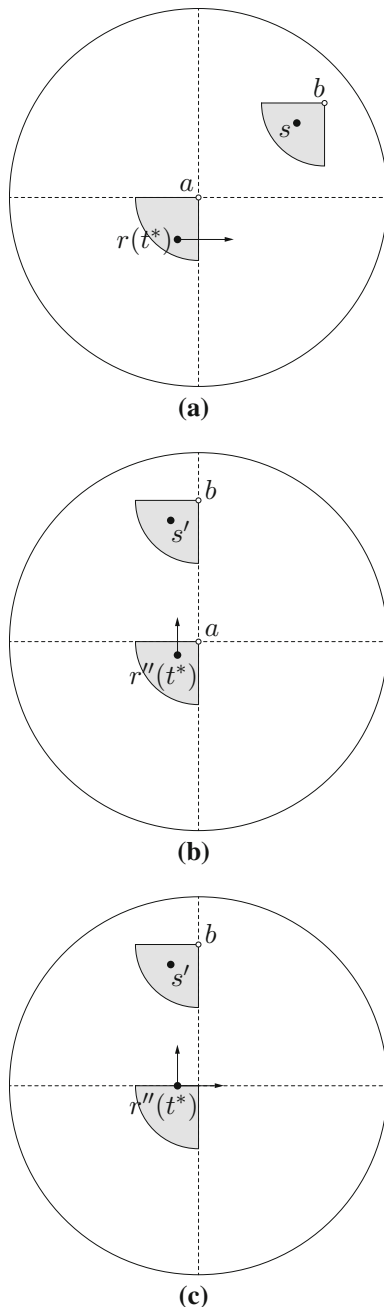
(a)

(b)

(c)

Fig. 8 Proof of Lemma 5. In **a**, b lies strictly to the right of a. In **b**, $a.x = b.x$ and $r''(t_1).y < a.y$. In **c**, $a.x = b.x$ and $r''(t_1).y = a.y$

Suppose first that $r''(t_1).y < a.y$ (see Fig. 8b). By definition of t_1, there exists a time $t^* \in [t_0, t_1]$ at which r'' performs a *Look* phase, such that $r''(t^*).y < \mathrm{DP}(r'', t^*).y$ and $r''(First(r'', t^*)).y = r''(t_1).y$. According to the algorithm (specifically, by Rule 2 of Sect. 3.2), if r'' is able to compute such a destination point, it means that no robot of $\mathcal{A}$ lies in $Q_1(r'')$ at time t^*. Therefore, by definition of λ and by construction, every robot in $Q_1(r'', t^*)$ has a y-coordinate that is greater than $a.y + 2\lambda/3$. On the other

hand, r'' sees no robot q with $q(t^*).y < r''(t^*).y$ and $\mathrm{dist}(r''(t^*), q(t^*)) \leq V - \rho/2$, hence r'' is able to move upward by more than $\lambda/3$. But indeed, r'' does see s', and $s'(t^*).y > a.y + 2\lambda/3$, hence $\mathrm{DP}(r'', t^*).y > r''(t^*).y + \lambda/3$. Once again, this contradicts the fact that $\mathrm{LIM}(r'') = a$.

Finally, suppose that $r''(t_1).y = a.y$ (see Fig. 8c), and let $t^* = First(r'', t_1)$. At time t^*, r'' sees no robot q in $Q_1(r'', t^*)$ with $r''(t^*).y < q(t^*).y \leq r'(t^*).y + 2\lambda/3$. Similarly to the previous paragraph's case, no robot below r'' can prevent r'' from moving upward, and the presence of s' makes r'' compute a destination point that is more than $\lambda/3 < \delta$ away from r''. Thus, r'' moves either upward of rightward by more than $\lambda/3$, which is in contradiction with the fact that $\mathrm{LIM}(r'') = a$. □

To prove that termination is correctly detected, we need one last lemma.

Lemma 6 *A robot r terminates its execution at time t only if it sees all the robots in $\mathcal{R}$ at time $Last(r, t)$.*

Proof Let r terminate its execution at time t, and let $\mathcal{Z}$ be the set of robots that are at distance at most ε' from r at time $t' = Last(r, t)$. Note that $\mathcal{Z}$ is not empty, because $r \in \mathcal{Z}$. Because r terminates, it follows that every robot in $\mathcal{R} \setminus \mathcal{Z}$ has distance greater than V from r at time t'.

Assume for a contradiction that $\mathcal{R} \setminus \mathcal{Z}$ is not empty. By Corollary 2, $G(t')$ is connected, and therefore there are a robot $s \in \mathcal{Z}$ and a robot $s' \in \mathcal{R} \setminus \mathcal{Z}$ such that $\mathrm{dist}(s(t'), s'(t')) \leq V - \rho/2$. Since $\mathrm{dist}(s'(t'), r(t')) > V$, then $\mathrm{dist}(s(t'), r(t')) > \rho/2$, by the triangle inequality. But $s \in \mathcal{Z}$, hence $\mathrm{dist}(s(t'), r(t')) \leq \varepsilon' \leq \rho/2$, which yields a contradiction. □

By putting together all the previous results, we obtain the following.

Theorem 1 *The algorithm in Fig. 2 correctly solves the* NEAR-GATHERING *problem under Assumption 1.*

Proof Assume for a contradiction that no robot ever terminates its execution. Due to Lemma 5, all the robots converge to the same point ℓ. Therefore, when all the robots are contained in a square Q with diagonal length ε' and upper-right vertex ℓ, they all see each other at distance not greater than ε'. From this time onward, whenever a robot executes a *Look* and then a *Compute* phase, it terminates, contradicting our assumption.

Hence, at least one robot r will terminate its execution at some point in time $t > 0$. Due to Lemma 6, r sees all the robots in $\mathcal{R}$ at time $t' = Last(r, t)$. This means that, at time t', all the robots are within distance ε' from each other. Due to Observation 2 and Lemma 5, at any time $t'' \geq t'$, all the robots are contained in a square Q with diagonal length ε' and upper-right vertex ℓ. Then, by the same reasoning used in the

previous paragraph, we conclude that every robot eventually terminates its execution while lying in Q.

By Lemma 4, no two robots ever collide, and hence they correctly solve Near-Gathering. □

5 Conclusions

In this paper we presented the first algorithm that solves the Near-Gathering problem using the standard Euclidean distance (in contrast with [18]) for a set of autonomous mobile robots with limited visibility. The protocol presented here is collision-free and handles termination: this allows to potentially combine our protocol with solutions to other problems designed for the unlimited visibility setting. This is achieved without assuming that the total number of robots in the system is known to the robots themselves, and without allowing them to explicitly communicate.

We remark that our algorithm can be easily modified to solve the Near-Gathering problem in the robot models that use any p-norm distance as opposed to the Euclidean distance, including the infinity norm distance.

Moreover, our algorithm is perfectly symmetric with respect to the x- and y-axes. This implies that our solution works also when the robots agree only on the direction of one of the two axes, say, the y-axis, and not necessarily on the orientation of the other axis.

Corollary 3 *The* Near-Gathering *problem is solvable under Assumption 1 even if the robots agree only on the direction of one of the two axes.*

Proof Suppose without loss of generality that the robots agree on the y-axis. Then, the following algorithm is employed: the input snapshot is first rotated clockwise by 45°, then the algorithm in Fig. 2 is applied to the resulting snapshot, and finally the computed destination point dp is rotated counterclockwise by 45°.

Indeed, the two rotations effectively tilt the coordinate systems of all robots, in such a way that their y-axes become actually parallel to the line $y = x$ in the "global" coordinate system. This is equivalent to having the robots agree only on the positive direction of the line $y = x$, but allowing them to disagree on which is the x-axis and which is the y-axis. The algorithm in Fig. 2 still works because, due to Observation 4, it is symmetric with respect to x- and y-coordinates. □

Therefore, under Assumption 1, the Near-Gathering protocol can also be used to solve the classical gathering problem in the limited visibility scenario, when the robots have only this form of partial agreement on their local coordination systems, thus improving on [12], which requires total agreement on both axes and does not avoid collisions. Indeed, it is sufficient to convert the termination command

in the algorithm in Fig. 2 with a move to point ℓ, as defined in Sect. 4.4.

We conjecture that no algorithm can solve Near-Gathering with no agreement on at least one axis, and we leave this as an open problem. Another direction for future research would be to solve Near-Gathering from any initial configuration in which the distance graph is connected, with no further assumption on the initial strong distance graph (cf. Assumption 1). Again, we conjecture this problem to be unsolvable in Async; note that in this case some extra assumption is required, for instance that the total number of robots is known, or that robots are able to communicate. Finally, the more general model in which robots do not necessarily have the same visibility radius, and hence do not share a common unit distance, should be considered in conjunction with both the gathering problem and Near-Gathering.

References

1. Agathangelou, C., Georgiou, C., Mavronicolas, M.: A distributed algorithm for gathering many fat mobile robots in the plane. In: Proceedings of the 32nd annual ACM symposium on principles of distributed computing (PODC), pp. 250–259 (2013)
2. Ando, H., Oasa, Y., Suzuki, I., Yamashita, M.: A distributed memoryless point convergence algorithm for mobile robots with limited visibility. IEEE Trans. Robot. Autom. **15**(5), 818–828 (1999)
3. Cieliebak, M.: Gathering non-oblivious mobile robots. In: 6th Latin American conference on theoretical informatics (LATIN), LNCS 2976, pp. 577–588 (2004)
4. Cohen, R., Peleg, D.: Convergence properties of the gravitational algorithms in asynchronous robots systems. SIAM J. Comput. **34**(6), 1516–1528 (2005)
5. Cord-Landwehr, A., Degener, B., Fischer, M., Hüllmann, M., Kempkes, B., Klaas, A., Kling, P., Kurras, S., Märtens, M., auf der Heide, F.M., Raupach, C., Swierkot, K., Warner, D., Weddemann, C., Wonisch, D.: Collision-less gathering of robots with an extent. In: Proceedings of the 37th international conference on current trends in theory and practice of computer science (SOFSEM), pp. 178–189 (2011)
6. Czyzowicz, J., Gasieniec, L., Pelc, A.: Gathering few fat mobile robots in the plane. Theor. Comput. Sci. **410**(6–7), 481–499 (2009)
7. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: The power of lights: synchronizing asynchronoys robots using visibile bits. In: Proceedings of the 32nd international conference on distributed computing systems (ICDCS), pp. 506–515 (2012)
8. Défago, X., Souissi, S.: Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. Theor. Comput. Sci. **396**(1–3), 97–112 (2008)
9. Dieudonné, Y., Labbani-Igbida, O., Petit, F.: Circle formation of weak mobile robots. ACM Trans. Auton. Adapt. Syst. **3**(4), 1–20 (2008)
10. Dieudonné, Y., Petit, F., Villain, V.: Leader election problem versus pattern formation problem. In: Proceedings of the 24th interna-

tional symposium on distributed computing (DISC), LNCS 6343, pp. 267–281 (2010)

11. Efrima, A., Peleg, D.: Distributed models and algorithms for mobile robot systems. In: Proceedings of the 33rd international conference on current trends in theory and practice of computer science (SOF-SEM), LNCS 4362, pp. 70–87 (2007)

12. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Gathering of robots with limited visibility. Theor. Comput. Sci. **337**(1–3), 147–168 (2005)

13. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Arbitrary pattern formation by asynchronous oblivious robots. Theor. Comput. Sci. **407**(1–3), 412–447 (2008)

14. Ganguli, A., Cortés, J., Bullo, F.: Multirobot rendezvous with visibility sensors in nonconvex environments. IEEE Trans. Robot. **25**(2), 340–352 (2009)

15. Izumi, T., Izumi, T., Kamei, S., Ooshita, F.: Feasibility of polynomial-time randomized gathering for oblivious mobile robots. IEEE Trans. Parallel Distrib. Syst. **24**(4), 716–723 (2013)

16. Katreniak, B.: Convergence with limited visibility by asynchronous mobile robots. In: Proceedings of the 18th international colloquium on structural information and communication complexity (SIROCCO), pp. 125–137 (2011)

17. Lin, J., Morse, A., Anderson, B.: The multi-agent rendezvous problem—part 2: the asynchronous case. SIAM J. Control Optim. **46**(6), 2120–2147 (2007)

18. Pagli, L., Prencipe, G., Viglietta, G.: Getting close without touching. In: Proceedings of the 19th colloquium on structural information and communication complexity (SIROCCO), LNCS 7355, pp. 315–326 (2012)

19. Peleg, D.: Distributed coordination algorithms for mobile robot swarms: new directions and challenges. In: Proceedings of the 7th international workshop on distributed computing (IWDC), LNCS 3741, pp. 1–12 (2005)

20. Souissi, S., Défago, X., Yamashita, M.: Using eventually consistent compasses to gather memory-less mobile robots with limited visibility. ACM Trans. Auton. Adapt. Syst. **4**(1), 1–27 (2009)

21. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: formation of geometric patterns. SIAM J. Comput. **28**(4), 1347–1363 (1999)

22. Yamashita, M., Suzuki, I.: Characterizing geometric patterns formable by oblivious anonymous mobile robots. Theor. Comput. Sci. **411**(26–28), 2433–2453 (2010)