Rendezvous of Two Robots with Constant Memory

Paola Flocchini[1], Nicola Santoro[2],
Giovanni Viglietta[2], and Masafumi Yamashita[3]

[1] EECS, University of Ottawa, Ottawa, Canada
flocchin@site.uottawa.ca
[2] SCS, Carleton University, Ottawa, Canada
{santoro,viglietta}@scs.carleton.ca
[3] Kyushu University, Fukuoka, Japan
mak@csce.kyushu-u.ac.jp

Abstract. We study the impact that persistent memory has on the classical *rendezvous* problem of two mobile computational entities, called robots, in the plane. It is well known that, without additional assumptions, rendezvous is impossible if the entities have no persistent memory, even if the system is semi-synchronous and movements are rigid. It has been recently shown that if each entity is endowed with $O(1)$ bits of persistent visible memory (called lights), they can rendezvous even if the system is asynchronous.

In this paper we investigate the rendezvous problem in two weaker settings in systems of robots endowed with visible lights: in FState, a robot can only see its own light, while in FComm a robot can only see the other robot's light. Among other things, we prove that, with rigid movements, finite-state robots can rendezvous in semi-synchronous settings, and finite-communication robots are able to rendezvous even in asynchronous ones. All proofs are constructive: in each setting, we present a protocol that allows the two robots to rendezvous in finite time.

1 Introduction

1.1 Framework and Background

Rendezvous is the process of two computational mobile entities, initially dispersed in a spatial universe, meeting within finite time at a location, non known *a priori*. When there are more than two entities, this task is known as *Gathering*. These two problems are core problems in distributed computing by mobile entities. They have been intensively and extensively studied when the universe is a connected region of $\mathbb{R}^2$ in which the entities, usually called *robots*, can freely move; see, for example, [1,3,4,8,9,11,14,15,16,17,18].

Each entity is modeled as a point, it has its own local coordinate system of which it perceives itself as the centre, and has its own unit distance. Each entity operates in cycles of Look, Compute, Move activities. In each cycle, an entity observes the position of the other entities expressed in its local coordinate system (Look); using that observation as input, it executes a protocol (the same

T. Moscibroda and A.A. Rescigno (Eds.): SIROCCO 2013, LNCS 8179, pp. 189–200, 2013.

for all robots) and computes a destination point (COMPUTE); it then moves to the computed destination point (MOVE). Depending on the activation schedule and the synchronization level, three basic types of systems are identified in the literature: a *fully synchronous* system (FSYNCH) is equivalent to a system where there is a common clock and at each clock tick all entities are activated simultaneously, and COMPUTE and MOVE are instantaneous; a *semi-synchronous* system (SSYNCH) is like a fully synchronous one except that, at each clock tick, only some entities will be activated (the choice is made by a fair scheduler); in a *fully asynchronous* system (ASYNCH), there is no common notion of time, each COMPUTE and MOVE of each robot can take an unpredictable (but finite) amount of time, and the interval of time between successive activities is finite but unpredictable. The focus of almost all algorithmic investigations in the continuous setting has been on *oblivious* robots, that is when the memory of the robots is erased at the end of each cycle, in other words the robots have no persistent memory (e.g., for an overview see [10]).

The importance of *Rendezvous* in the continuous setting derives in part from the fact that it separates FSYNCH from SSYNCH for oblivious robots. Indeed, *Rendezvous* is trivially solvable in a fully synchronous system, without any additional assumption. However, without additional assumptions, *Rendezvous* is *impossible* for oblivious robots if the system is semi-synchronous [19]. Interestingly, from a computational point of view, *Rendezvous* is very different from the *Gathering* problem of having $k \geqslant 3$ robots meet in the same point; in fact, *Gathering* of oblivious robots is always *possible* for any $k \geqslant 3$ even in ASYNCH without any additional assumption other than multiplicity detection [3]. Furthermore, in SSYNCH, $k \geqslant 3$ robots can gather even in spite of a certain number of faults [1,2,7], and converge in spite of inaccurate measurements [5]; see also [12]. The *Rendezvous* problem also shows the impact of certain factors. For example, the problem has a trivial solution if the robots are endowed with *consistent compasses* even if the system is fully asynchronous. The problem is solvable in ASYNCH even if the local compasses have some degree of inconsistency of an appropriate angle [13]; the solution is no longer trivial, but does exist.

In this paper, we are interested in determining what type and how much persistent memory would allow the robots to rendezvous. What is known in this regard is very little. On the one hand, it is well known that, in absence of additional assumptions, without persistent memory rendezvous is impossible even in SSYNCH [19]. On the other hand, a recent result shows that rendezvous is possible even in ASYNCH if each robot has $O(1)$ bits of persistent memory *and* can transmit $O(1)$ bits in each cycle *and* can remember (i.e., can persistently store) the last received transmission [6] (see also [20] for size-optimal solutions). The conditions of this result are overly powerful. The natural question is whether the simultaneous presence of these conditions is truly necessary for rendezvous.

1.2 Main Contributions

In this paper we address this question by weakening the setting in two different ways, and investigate the *Rendezvous* problem in these weaker settings. Even

though its use is very different, in both settings, the amount of persistent memory of a robot is constant.

We first examine the setting where the two robots have $O(1)$ bits of *internal* persistent memory but cannot communicate; this corresponds to the *finite-state* (FState) robots model. Among other contributions, we prove that FState robots with rigid movements can rendezvous in SSynch, and that this can be done using only six internal states. The proof is constructive: we present a protocol that allows the two robots to rendezvous in finite time under the stated conditions.

We then study the *finite-communication* (FComm) setting, where a robot can transmit $O(1)$ bits in each cycle and remembers the last received transmission, but it is otherwise oblivious: it has no other persistent memory of its previous observations, computations and transmissions. We prove that two FComm robots with rigid movements are able to rendezvous even in ASynch; this is doable even if the different messages that can be sent are just 12. We also prove that only three different messages suffice in SSynch. Also for this model all the proofs are constructive.

Finally, we consider the situation when the movement of the robots is not rigid, that is it can be interrupted by an adversary. The only constraint on the adversary is that a robot moves at least a distance $\delta > 0$ (otherwise, rendezvous is clearly impossible). We show that, with knowledge of δ, three internal states are sufficient to solve *Rendezvous* by FState robots in SSynch, and three possible messages are sufficient for FComm robots in ASynch.

These results are obtained modeling both settings as a system of robots endowed with a constant number of *visible lights*: a FState robot can see only its own light, while a FComm robot can see only the other robot's light. Our results seem to indicate that "it is better to communicate than to remember". In addition to the specific results on the *Rendezvous* problem, an important contribution of this paper is the extension of the classical model of oblivious silent robots into two directions: adding finite memory, and enabling finite communication.

Due to space limitations, several details and proofs are omitted; they can be found in `http://arxiv.org/abs/1306.1956`.

2 Model and Terminology

The general model we employ is the standard one, described in [10]. The two robots are autonomous computational entities modeled as points moving in $\mathbb{R}^2$. Each robot has its own coordinate system and its own unit distance, which may differ from each other, and it always perceives itself as lying at the origin of its own local coordinate system. Each robot operates in cycles that consist of three phases: Look, Compute, and Move. In the Look phase it gets the position (in its local coordinate system) of the other robot; in the Compute phase, it computes a destination point; in the Move phase it moves to the computed destination point, along a straight line. Without loss of generality, the Look

phase is assumed to be instantaneous. The robots are anonymous and oblivious, meaning that they do not have distinct identities, they execute the same algorithm in each COMPUTE phase, and the input to such algorithm is the snapshot coming from the previous LOOK phase.

We study two settings; both can be described as restrictions of the model of visibile lights introduced in [6]. In that model, each robot carries a persistent memory of constant size, called *light*; the value of the light is called *color* or *state*, and it is set by the robot during each COMPUTE phase. Other than their own light, the robots have no other memory of past snapshots and computations.

In the first setting, that of silent finite-state (FSTATE) robots, the light of a robot is visible only to the robot itself; i.e., the colored light merely encodes an *internal state*. In the second setting, of oblivious finite-communication (FCOMM) robots, the light of a robot is visible only to the other robot; i.e., they can communicate with the other robot through their colored light, but by their next cycle they forget even the color of their own light (since they do not see it). The color a robot sees is used as input during the computation.

In the *asynchronous* (ASYNCH) model, the robots are activated independently, and the duration of each COMPUTE, MOVE and inactivity is finite but unpredictable. As a consequence, the robots do not have a common notion of time, they can be seen while moving, and computations can be made based on obsolete observations. In the *semi-synchronous* (SSYNCH) model the activationsof robots can be logically divided into global rounds; in each round, one or both robots are activated, obtain the same snapshot, compute, and perform their move. It is assumed that the activation schedule is fair, i.e., each robot is activated infinitely often.

Depending on whether or not the adversary can stop a robot before it reaches its computed destination, the movements are called *non-rigid* and *rigid*, respectively. In the case of non-rigid movements, there exists a constant $\delta > 0$ such that if the destination point's distance is smaller than δ, the robot will reach it; otherwise, it will move towards it by at least δ. Note that, without this assumption, an adversary could make it impossible for any robot to ever reach its destination, following a classical Zenonian argument.

The two robots solve the *Rendezvous* problem if, within finite time, they move to the same point (not determined *a priori*) and do not move from there. A rendezvous algorithm for SSYNCH (resp., ASYNCH) is a protocol that allows the robots to solve the *Rendezvous* problem under any possible schedule in SSYNCH (resp., ASYNCH). A particular class of algorithms, denoted by $\mathcal{L}$, is that where each robot may only compute a destination point of the form $\lambda \cdot other.position$, for some $\lambda \in \mathbb{R}$ obtained as a function only of the light of which the robot is aware (i.e., its internal state in the FSTATE model, or the other robot's color in the FCOMM model). The algorithms of this class are of interest because they operate also when the coordinate system of a robot is not self-consistent (i.e., it can unpredictably rotate, change its scale or undergo a reflection).

3 Finite-State Robots

We fist consider FSTATE robots and we start by identifying a simple impossibility result for algorithms in $\mathcal{L}$.

Theorem 1. *In* SSYNCH*,* Rendezvous *of two* FSTATE *robots is unsolvable by algorithms in $\mathcal{L}$, regardless of the amount of their internal memory.*

Thus the computation of the destination must take into account more than just the lights (or states) of which the robot is aware.

The approach we use to circumvent this impossibility result is to have each robot use its own unit of distance as a computational tool; recall that the two robots might have different units, and they are not known to each other. We propose Algortihm 1 for *Rendezvous* in SSYNCH. Each robot has six internal states, namely S_{start}, S_1, S_2^{left}, S_2^{right}, S_3, and S_{finish}. Both robots are assumed to begin their execution in S_{start}. Each robot lies in the origin of its own local coordinate system and the two robots have no agreement on axes orientations or unit distance. Intuitively, the robots try to reach a configuration in which they both observe the other robot at distance not lower than 1 (their own unit). From this configuration, they attempt to meet in the midpoint. If they never meet because they are never activated simultaneously, at some point one of them notices that its observed distance is lower than 1. This implies a breakdown of symmetry that enables the robots to finally gather. In order to reach the desired configuration in which they both observe a distance not lower than 1, the two robots first try to move farther away from each other if they are too close. If they are far enough, they memorize the side on which they see each other (left or right), and try to switch positions. If only one of them is activated, they gather; otherwise they detect a side switch and they can finally apply the above protocol. This is complicated by the fact that the robots may disagree on the distances they observe. To overcome this difficulty, they use their ability to detect a side switch to understand which distance their partner observed. If the desired configuration is not reached because of a disagreement, a breakdown of symmetry occurs, which is immediately exploited to gather anyway. As soon as the two robots coincide at the end of a cycle, they never move again, and *Rendezvous* is solved.

Theorem 2. *In* SSYNCH*,* Rendezvous *of two* FSTATE *robots is solvable with six internal states. This result holds even without unit distance agreement.*

4 Finite-Communication Robots

4.1 Asynchronous

It is not difficult to see that algorithms in $\mathcal{L}$ are not sufficient to solve the problem.

Theorem 3. *In* ASYNCH*,* Rendezvous *of two* FCOMM *robots is unsolvable by algorithms in $\mathcal{L}$, regardless of the amount of colors employed.*

Algorithm 1. Rendezvous for rigid SSYNCH with no unit distance agreement and six internal states

```
1: dist ← ‖other.position‖
2: if dist = 0 then
3:     terminate
4: if other.position.x > 0 then
5:     dir ← right
6: else if other.position.x < 0 then
7:     dir ← left
8: else if other.position.y > 0 then                    ▷ other.position.x = 0
9:     dir ← right
10: else
11:     dir ← left
12: if me.state = S_start then
13:     if dist < 1 then
14:         me.state ← S_1
15:         me.destination ← other.position · (1 − 1/dist)
16:     else
17:         me.state ← S_2^dir
18:         me.destination ← other.position
19: else if me.state = S_1 then
20:     if dist ⩽ 1 then
21:         me.state ← S_finish
22:         me.destination ← (0, 0)
23:     else
24:         me.state ← S_2^dir
25:         me.destination ← other.position
26: else if me.state = S_2^d then
27:     if dir = d then
28:         me.state ← S_finish
29:         me.destination ← other.position
30:     else if dist < 1/2 then                         ▷ side switch detected
31:         me.state ← S_finish
32:         me.destination ← (0, 0)
33:     else
34:         me.destination ← other.position/2
35:         if dist < 1 then
36:             me.state ← S_3
37: else if me.state = S_3 then
38:     me.state ← S_finish
39:     if dist < 1/4 then
40:         me.destination ← (0, 0)
41:     else                                            ▷ 1/4 ⩽ d < 1/2
42:         me.destination ← other.position
43: else                                                ▷ me.state = S_finish
44:     if dist ⩽ 1 then
45:         me.destination ← (0, 0)
46:     else
47:         me.destination ← other.position
```

We now describe an algorithm (which is not in $\mathcal{L}$) that solves the problem. Also this algorithm uses the local unit distance as a computational tool, but in a rather different way since a robot cannot remember and has to infer information by observing the other robot's light.

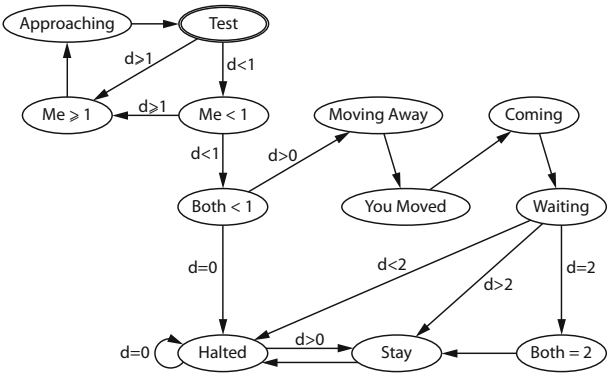

Fig. 1. State transitions in Algorithm 2

Intuitively, the two robots try to reach a configuration in which both robots see each other at distance lower than 1. To do so, they first communicate to the other whether or not the distance they observe is smaller than 1 (recall that they may disagree, because their unit distances may differ). If one robot acknowledges that its partner has observed a distance not smaller than 1, it reduces the distance by moving toward the midpoint.

The process goes on until both robots observe a distance smaller than 1. At this point, if they have not gathered yet, they try to compare their distance functions, in order to break symmetry. They move away from each other in such a way that their final distance is the sum of their respective unit distances. Before proceeding, they attempt to switch positions. If, due to asynchrony, they failed to be in the same state at any time before this step, they end up gathering. Instead, if their execution has been synchronous up to this point, they finally switch positions. Now, if the robots have not gathered yet, they know that their distance is actually the sum of their unit distances. Because each robot knows its own unit, they can tell if one of them is larger. If a robot has a smaller unit, it moves toward its partner, which waits.

Otherwise, if their units are equal, they apply a simple protocol: as soon as a robot wakes up, it moves toward the midpoint and orders its partner to stay still. If both robots do so, they gather in the middle. If one robot is delayed due to asynchrony, it acknowledges the order to stay still and tells the other robot to come.

Theorem 4. *In* ASYNCH, Rendezvous *of two* FCOMM *robots is solvable with 12 colors. This result holds even without unit distance agreement.*

Algorithm 2. Rendezvous for rigid ASYNCH with no unit distance agreement and 12 externally visible states

```
 1: dist ← ‖other.position‖
 2: if other.state = (TEST) then                                   ▷ testing distances
 3:     if dist ⩾ 1 then
 4:         me.state ← (ME ⩾ 1)
 5:     else
 6:         me.state ← (ME < 1)
 7: else if other.state = (ME ⩾ 1) then                            ▷ reducing distances
 8:     me.state ← (APPROACHING)
 9:     me.destination ← other.position/2
10: else if other.state = (APPROACHING) then                       ▷ test distances again
11:     me.state ← (TEST)
12: else if other.state = (ME < 1) then
13:     if dist ⩾ 1 then
14:         me.state ← (ME ⩾ 1)
15:     else
16:         me.state ← (BOTH < 1)
17: else if other.state = (BOTH < 1) then
18:     if dist = 0 then                                          ▷ we have gathered
19:         me.state ← (HALTED)
20:     else
21:         me.state ← (MOVING AWAY)
22:         if dist < 1 then                                      ▷ moving away by 1 − dist/2
23:             me.destination ← other.position · (1/2 − 1/dist)
24: else if other.state = (MOVING AWAY) then
25:     me.state ← (YOU MOVED)
26: else if other.state = (YOU MOVED) then
27:     me.state ← (COMING)
28:     me.destination ← other.position
29: else if other.state = (COMING) then
30:     me.state ← (WAITING)
31: else if other.state = (WAITING) then
32:     if dist > 2 then                                          ▷ my unit is smaller
33:         me.state ← (STAY)
34:         me.destination ← other.position
35:     else if dist = 2 then                                     ▷ our units are equal
36:         me.state ← (BOTH = 2)
37:     else                                          ▷ my unit is bigger or we have gathered
38:         me.state ← (HALTED)
39: else if other.state = (BOTH = 2) then
40:     me.state ← (STAY)
41:     if dist = 2 then                                          ▷ moving to the midpoint
42:         me.destination ← other.position/2
43: else if other.state = (STAY) then
44:     me.state ← (HALTED)
45: else                                                          ▷ other.state = (HALTED)
46:     if dist = 0 then                                          ▷ we have gathered
47:         me.state ← (HALTED)
48:         terminate
49:     else                                          ▷ maintain position while I come
50:         me.state ← (STAY)
51:         me.destination ← other.position
```

Proof. We show that Algorithm 2, also depicted in Figure 1, correctly solves *Rendezvous*. Both robots start in state (TEST), and then update their state to (ME $\geqslant 1$) or (ME < 1), depending if they see each other at distance greater or lower than 1 (they may disagree, because their distance functions may be different). If robot r sees robot s set to (ME $\geqslant 1$), it starts approaching it by moving to the midpoint, in order to reduce the distance. No matter if r approaches s several times before s is activated, or both robots approach each other at different times, one of them eventually sees the other set to (APPROACHING). When this happens, their distance has reduced by at least a half, and at least one robot turns (TEST) again, thus repeating the test on the distances. At some point, both robots see each other at distance lower than 1 during a test, and at least one of them turns (BOTH < 1). If they have not gathered yet, they attempt to break symmetry by comparing their distance functions. To do so, when a robot sees the other set to (BOTH < 1), it turns (MOVING AWAY) and moves away by its own unit distance minus half their current distance. This move will be performed at most once by each robot, because if one robot sees the other robot still set to (BOTH < 1), but it observes a distance not lower than 1, then it knows that it has already moved away, and has to wait. When a robot sees its partner set to (MOVING AWAY), it shares this information by turning (YOU MOVED). If only one robot turns (YOU MOVED), while the other is still set to (MOVING AWAY), then the second robot turns (COMING) and reaches the other robot, which just turns (WAITING) and stays still until they gather. Otherwise, if both robots see each other set to (YOU MOVED), they both turn (COMING) and switch positions. At least one of them then turns (WAITING). Now, if a robot sees its partner set to (WAITING) and they have not gathered yet, it knows that their current distance is the sum of their unit distances. If such distance is greater than 2, then the robot knows that its partner's unit distance is bigger, and it moves toward it, while ordering it to stay still. Vice versa, if the distance observed is smaller than 2, the observing robot stays still and orders its partner to come. Finally, if the distance observed is exactly 2, the observing robot knows that the two distance functions are equal, and turns (BOTH $= 2$). In this case, a simple protocol allows them to meet. If a robot sees the other set to (BOTH $= 2$) at distance 2, it turns (STAY) and moves to the midpoint. If both robots do so, they eventually gather. Indeed, even if the first robot reaches the midpoint while the other is still set to (BOTH $= 2$), it now sees its partner at distance 1, and knows that it has to wait. On the other hand, whenever a robot sees its partner set to (STAY), it turns (HALTED), which tells its partner to reach it. This guarantees gathering even if only one robot attempts to move to thee midpoint.

4.2 Semi-Synchronous

In SSYNCH the situation is radically different from the ASYNCH case. In fact, it is possible to find a simple solution in $\mathcal{L}$ that uses the minimum number of colors possible, and operates correctly without unit distance agreement, starting from any arbitrary color configuration, and with interruptable movements (see Algorithm 3).

Algorithm 3. Rendezvous for non-rigid SSYNCH with three externally visible states

```
1: if other.state = A then
2:     me.state ← B
3:     me.destination ← other.position/2
4: else if other.state = B then
5:     me.state ← C
6: else                                          ▷ other.state = C
7:     me.state ← A
8:     me.destination ← other.position
```

Theorem 5. *In* SSYNCH, Rendezvous *of two* FCOMM *robots is solvable by an algorithm in $\mathcal{L}$ with only three distinct colors. This result holds even if starting from an arbitrary color configuration, without unit distance agreement, and with non-rigid movements.*

Note that the number of colors used by the algorithm is optimal. This follows as a corollary of the impossibility result when lights are visible to both robots:

Lemma 1. *[20] In* SSYNCH, Rendezvous *of two robots with persistent memory visible by both of them is unsolvable by algorithms in $\mathcal{L}$ that use only two colors.*

5 Movements: Knowledge vs. Rigidity

In this section, we consider the *Rendezvous* problem when the movement of the robots can be interrupted by an adversary; previously, unless otherwise stated, we have considered rigid movements, i.e., in each cycle a robot reaches its computed destination point. Now, the only constraint on the adversary is that a robot, if interrupted before reaching its destination, moves by at least $\delta > 0$ (otherwise, rendezvous is clearly impossible). We prove that, for rendezvous with lights, knowledge of δ has the same power as rigidity of the movements. Note that knowing δ implies also that the robots can agree on a unit distance.

5.1 FState Robots

Both robots start in state A. If a robot sees its partner at distance lower than $\delta/2$, it moves in the opposite direction, to the point at distance $\delta/2$ from its partner. On the other hand, if the distance observed is not lower than δ, it moves toward the point located $\delta/4$ before the midpoint.

It is easy to see that after sufficiently many turns, the robots find themselves at a distance in the interval $[\delta/2, \delta)$, and both in state A. From now on, all their movements are rigid.

Theorem 6. *In non-rigid* SSYNCH, Rendezvous *of two* FSTATE *robots with knowledge of δ is solvable with three colors.*

5.2 FComm Robots

The idea of the Algorithm is simple. Both robots begin their execution in state Start, and attempt to position themselves at a distance in the interval $(\delta, 2\delta]$. To do so, they adjust their position by moving by $\delta/2$ at each step. When a robot sees its partner at the desired distance, it turns Ready and stops. It is easy to show that, even if its partner is still moving, it will end its move at a distance in the interval $(\delta, 2\delta]$. When a robot sees its partner set to Ready, it turns Come and moves to the midpoint; the midpoint is eventually reached, because the distance traveled is not greater than δ.
We can conclude that:

Theorem 7. *In non-rigid* ASynch, Rendezvous *of two* FComm *robots with knowledge of δ is solvable with three colors.*

6 Open Problems

Our results, showing that rendezvous is possible in SSynch for FState robots and in ASynch for FComm robots, seem to indicate that "it is better to communicate than to remember". However, determining the precise computational relationship between FState and FComm is an open problem. To settle it, it must be determined whether or not it is possible for FState robots to rendezvous in ASynch.

Although minimizing the amount of constant memory was not the primary focus of this paper, the number of states employed by our algorithms is rather small. An interesting research question is to determine the smallest amount of memory necessary for the robots to rendezvous when rendezvous is possible, and devise optimal solution protocols.

The knowledge of δ in non-rigid scenarios is quite powerful and allows for simple solutions. It is an open problem to study the *Rendezvous* problem for FState and FComm robots when δ is unknown or not known precisely.

This paper has extended the classical models of oblivious silent robots into two directions: adding finite memory, and enabling finite communication. It thus opens the investigation in the FState and FComm models of other classical robots problems (e.g., *Pattern Formation*, *Flocking*, etc.); an exception is *Gathering* because, as mentioned in the introduction, it is already solvable without persistent memory and without communication [3].

Acknowledgments. This work has been supported in part by NSERC, and by Prof. Flocchini's URC.

References

1. Agmon, N., Peleg, D.: Fault-tolerant gathering algorithms for autonomous mobile robots. SIAM Journal on Computing 36, 56–82 (2006)
2. Bouzid, Z., Das, S., Tixeuil, S.: Gathering of mobile robots tolerating multiple crash faults. In: Proceedings of 33rd IEEE International Conference on Distributed Computing Systems, ICDCS (2013)

3. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Distributed computing by mobile robots: Gathering. SIAM Journal on Computing 41(4), 829–879 (2012)
4. Cohen, R., Peleg, D.: Convergence properties of the gravitational algorithm in asynchronous robot systems. SIAM Journal on Computing 34, 1516–1528 (2005)
5. Cohen, R., Peleg, D.: Convergence of autonomous mobile robots with inaccurate sensors and movements. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 549–560. Springer, Heidelberg (2006)
6. Das, S., Flocchini, P., Prencipe, G., Santoro, N., Yamashita, M.: The power of lights: synchronizing asynchronous robots using visible bits. In: Proceedings of the 32nd International Conference on Distributed Computing Systems (ICDCS), pp. 506–515 (2012)
7. Défago, X., Gradinariu, M., Messika, S., Raipin-Parvédy, P.: Fault-tolerant and self-stabilizing mobile robots gathering. In: Dolev, S. (ed.) DISC 2006. LNCS, vol. 4167, pp. 46–60. Springer, Heidelberg (2006)
8. Degener, B., Kempkes, B., Langner, T., Meyer auf der Heide, F., Pietrzyk, P., Wattenhofer, R.: A tight runtime bound for synchronous gathering of autonomous robots with limited visibility. In: Proceedings of 23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), pp. 139–148 (2011)
9. Dieudonné, Y., Petit, F.: Self-stabilizing gathering with strong multiplicity detection. Theoretical Computer Science 428(13) (2012)
10. Flocchini, P., Prencipe, G., Santoro, N.: Distributed Computing by Oblivious Mobile Robots. Morgan & Claypool (2012)
11. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Gathering of asynchronous robots with limited visibility. Theo. Comp. Sci. 337(1-3), 147–168 (2005)
12. Izumi, T., Bouzid, Z., Tixeuil, S., Wada, K.: Brief Announcement: The BG-simulation for Byzantine mobile robots. In: Peleg, D. (ed.) Distributed Computing. LNCS, vol. 6950, pp. 330–331. Springer, Heidelberg (2011)
13. Izumi, T., Souissi, S., Katayama, Y., Inuzuka, N., Defago, X., Wada, K., Yamashita, M.: The gathering problem for two oblivious robots with unreliable compasses. SIAM Journal on Computing 41(1), 26–46 (2012)
14. Kamei, S., Lamani, A., Ooshita, F., Tixeuil, S.: Asynchronous mobile robot gathering from symmetric configurations without global multiplicity detection. In: Kosowski, A., Yamashita, M. (eds.) SIROCCO 2011. LNCS, vol. 6796, pp. 150–161. Springer, Heidelberg (2011)
15. Lin, J., Morse, A.S., Anderson, B.D.O.: The multi-agent rendezvous problem. parts 1 and 2. SIAM Journal on Control and Optimization 46(6), 2096–2147 (2007)
16. Pagli, L., Prencipe, G., Viglietta, G.: Getting close without touching. In: Even, G., Halldórsson, M.M. (eds.) SIROCCO 2012. LNCS, vol. 7355, pp. 315–326. Springer, Heidelberg (2012)
17. Prencipe, G.: Impossibility of gathering by a set of autonomous mobile robots. Theoretical Computer Science 384(2-3), 222–231 (2007)
18. Souissi, S., Défago, X., Yamashita, M.: 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)
19. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM Journal on Computing 28, 1347–1363 (1999)
20. Viglietta, G.: Rendezvous of two robots with visible bits. Technical Report arXiv:1211.6039 (2012)