Rendezvous with Constant Memory*

P. Flocchini* N. Santoro† G. Viglietta† M. Yamashita‡

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 are oblivious (i.e., have no persistent memory) even if the system is semi-synchronous (SSynch). It has been recently shown that rendezvous is possible even if the system is asynchronous (ASynch) if each robot is endowed with $O(1)$ bits of persistent memory, can transmit $O(1)$ bits in each cycle, and can remember (i.e., can persistently store) the last received transmission. This setting is overly powerful.

In this paper we weaken that setting in two different ways: (1) by maintaining the $O(1)$ bits of persistent memory but removing the communication capabilities, a setting we call *finite-state* (FState); and (2) by maintaining the ability of transmitting $O(1)$ bits and remembering the last received message, but removing the ability of an agent to remember its previous activities, a setting we call *finite-communication* (FComm). Note that, even though its use is very different, in both settings, the amount of persistent memory of a robot is a constant number of bits.

We investigate the rendezvous problem in these two weaker settings. We model both settings as a system of robots endowed with visible lights, each with a constant number of colors: 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 SSynch, and that finite-communication robots are able to rendezvous even in ASynch. All proofs are constructive: in each setting, we present a protocol that allows the two robots to rendezvous in finite time.

Keywords: autonomous mobile robots; finite states robots; robots with lights; rendezvous; gathering; obliviousness and memory; synchrony vs. asynchrony.

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

*School of Electrical Engineering and Computer Science, University of Ottawa, Canada.

†School of Computer Science, Carleton University, Canada.

‡Kyushu University, Fukuoka, Japan.

*This research has been supported in part by the Natural Sciences and Engineering Research Council (Canada) under the Discovery Grant program, by Prof. Flocchinis University Research Chair, and by the Scientific Grant in Aid by the Ministry of Education, Sports, Culture and Technology of Japan. A preliminary version of this paper has been presented at the 20th International Colloquium on Structural Information and Communication Complexity

is a connected region of $\mathbb{R}^2$ in which the entities, usually called *robots*, can freely move; see, for example, [1, 3, 5, 6, 8, 11, 13, 15, 18, 19, 20, 21, 22].

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 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; for a recent detailed review, see [14].

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 [23].

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 [5]. Furthermore, in SSYNCH, $k \geqslant 3$ robots can gather even in spite of a certain number of faults [1, 2, 10], and converge in spite of inaccurate measurements [7]; see also [16].

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 (a tilt of an appropriate angle) [17]; the solution is no longer trivial, but does exist.

In this paper, we are concerned with the impact that *memory* has on the solvability of the *Rendezvous* problem. In particular, we are interested in determining how many bits of persistent memory and what type of use would allow the robots to rendezvous. Little is known in this regard.

If a robot can store a constant number of observed coordinates of the other robot (i.e., real numbers), the problem is solvable in SSYNCH [12, 23]; by applying the general technique of [4], this result holds even in ASYNCH. However, to remember a real number requires an *unbounded* number of bits of persistent memory. This opens the question of whether it is possible to solve *Rendezvous* with a bounded number of bits, and under what conditions.

A recent result shows that $O(1)$ bits of persistent memory suffice, even in ASYNCH, if the robots are also able to transmit $O(1)$ bits in each cycle and remember (i.e., persistently store) the last received message [9] (see also [24] for size-optimal solutions). These conditions (called finite-state and finite-communication) 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, a robot uses only a constant amount of persistent bits.

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 (hence, three persistent bits suffice). 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 using 12 message types (hence four persistent bits suffice). We also prove that only three different message types (hence two persistent bits) suffice in SSYNCH. Also for this model all the proofs are constructive.

A summary of these results is shown in Table 1; the only outstanding question is whether finite-state robots can rendezvous in ASYNCH.

	FSTATE	FCOMM
SSYNCH	3 bits	2 bits
ASYNCH	?	4 bits

Table 1: Summary of main results.

Finally, we consider the *Rendezvous* problem when the movement of the robots are non-rigid: a robot can be stopped by an adversary before it reaches its destination. The only constraint on the adversary is that, if it does not reach its destination, a robot moves by at least a distance $\delta > 0$ (otherwise, rendezvous is clearly impossible). The protocol we have developed for FCOMM robots in SSYNCH allows the robots to rendezvous even in presence of this type of adversarial attacks. We show that, with knowledge of δ, three internal states are sufficient to solve *Rendezvous* by FSTATE robots in SSYNCH, and three distinct messages are sufficient for FCOMM robots in ASYNCH. In other words, we prove that knowledge of δ is able to overcome malicious interruptions of movements by an adversary, and two persistent bits suffice.

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 a finite number of bits of memory, and enabling communication of a finite number of bits.

2 Model and Terminology

The general model we employ is the standard one, described in [14]. 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 an algorithm is the snapshot coming from the previous LOOK phase.

Here we study two settings; both settings can be described as restrictions of the model of visibile lights introduced in [9]. 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 persistent memory of past snapshots and computations.

In the first setting, that of silent finite-state (or simply, 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 (or simply 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 result, the robots do not have a common notion of time, robots can be seen while moving, and computations can be made based on obsolete observations. In the *semi-synchronous* (SSYNCH) models the activations of 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 and do not move from there; the meeting point is not determined *a priori*. A rendezvous algorithm for SSYNCH (resp., ASYNCH) is a protocol that allows the robots to solve the *Rendezvous* problem under any possible execution schedule in SSYNCH (resp., ASYNCH). A particular class of algorithms, denoted by $\mathcal{L}$, is that in which each robot may only compute a destination point of the form $\lambda \cdot \mathit{other.position}$, 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 first consider FSTATE robots in the semi-synchronous setting, 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.*

Proof. For each robot, the destination point and the next state are a function of the internal state only. Assuming that both robots start in the same state, we keep activating them one at a time, alternately. Hence, every other turn they are in the same state. As soon as the first robot attempts to move to the other robot's location, we activate both robots simultaneously, making them switch positions. By repeating this pattern, the robots never gather. □

Thus the computation of the destination must take into account more than just the lights (or states) of which each 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 Algorithm 1 for *Rendezvous* in SSYNCH, also illustrated in Figure 1. 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.

To analyze the correctness of Algorithm 1, some terminology is needed. In the following, the two robots will be called r and s, respectively. An expression of the form (S_r, S_s, I_r, I_s) denotes a configuration in which robot r (resp. s) is in state S_r (resp. S_s), and the distance at which it sees the other robot lies in the interval I_r (resp. I_s), according to its own distance function. Therefore, the starting configuration of r and s is $(S_{\text{start}}, S_{\text{start}}, [0, +\infty), [0, +\infty))$.

With abuse of notation, we will say that a robot is in state $S_2^{=}$ if it is in state S_2^{left} (resp. S_2^{right}) and it sees the other robot on its left (resp. right). Analogously, a robot is said to be in state $S_2^{\neq}$ if its state is S_2^{left} or S_2^{right} and it has detected a switch.

The unit distances of robots r and s, as measured in a global reference system, will be denoted by u_r and u_s, respectively. We will also let $u = (u_r + u_s)/2$. For a configuration C and a function $f(\cdot, \cdot)$, the expression $C \to f(d, u)$ means that, whenever the two robots reach C and their distance

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

1: $dist \leftarrow \|other.position\|$
2: **if** $dist = 0$ **then**
3: terminate
4: **if** $other.position.x > 0$ **then**
5: $dir \leftarrow$ right
6: **else if** $other.position.x < 0$ **then**
7: $dir \leftarrow$ left
8: **else if** $other.position.y > 0$ **then** ▷ $other.position.x = 0$
9: $dir \leftarrow$ right
10: **else**
11: $dir \leftarrow$ left
12: **if** $me.state = S_{\text{start}}$ **then**
13: **if** $dist < 1$ **then**
14: $me.state \leftarrow S_1$
15: $me.destination \leftarrow other.position \cdot (1 - 1/dist)$
16: **else**
17: $me.state \leftarrow S_2^{dir}$
18: $me.destination \leftarrow other.position$
19: **else if** $me.state = S_1$ **then**
20: **if** $dist \leqslant 1$ **then**
21: $me.state \leftarrow S_{\text{finish}}$
22: $me.destination \leftarrow (0, 0)$
23: **else**
24: $me.state \leftarrow S_2^{dir}$
25: $me.destination \leftarrow other.position$
26: **else if** $me.state = S_2^d$ **then**
27: **if** $dir = d$ **then**
28: $me.state \leftarrow S_{\text{finish}}$
29: $me.destination \leftarrow other.position$
30: **else if** $dist < 1/2$ **then** ▷ side switch detected
31: $me.state \leftarrow S_{\text{finish}}$
32: $me.destination \leftarrow (0, 0)$
33: **else**
34: $me.destination \leftarrow other.position/2$
35: **if** $dist < 1$ **then**
36: $me.state \leftarrow S_3$

```
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
```

is d, they eventually gather and solve *Rendezvous*, after covering a combined distance of at most $f(d, u)$.

Observation 2. *$(S_a, S_b, I_a, I_b) \to f(d, u)$ if and only if $(S_b, S_a, I_b, I_a) \to f(d, u)$.*

Observation 3. *If $(S_r, S_s, I_r, I_s) \to f(d, u)$ and $I'_r \subseteq I_r$, then $(S_r, S_s, I'_r, I_s) \to f(d, u)$.*

Lemma 4. $(S_{\text{finish}}, S_3, [0, 1], [1/4, 1/2)) \to d$.

Proof. Robot r keeps staying still, while robot s moves to r as soon as it is activated. Hence the total distance covered is d. □

Lemma 5. $(S_{\text{finish}}, S_2^{\neq}, [0, 1], [1/2, +\infty)) \to d$.

Proof. Robot s keeps moving to the midpoint, while robot r never moves, because it keeps observing a distance not greater than 1. As soon as s observes a distance smaller than 1 (hence in $(1/2, 1)$), its state becomes S_3 and it moves to the midpoint again. At this point the distance covered is $d' < d$, and the new distance is $d - d'$. Now the configuration is $(S_{\text{finish}}, S_3, [0, 1/2], [1/4, 1/2))$, and Lemma 4 applies. The total distance covered is therefore $d' + (d - d') = d$. □

Lemma 6. $(S_{\text{finish}}, S_2^{=}, [0, 1], [0, +\infty)) \to d$.

Proof. Robot r keeps staying still, while robot s moves to r as soon as it is activated, covering a distance of d. □

Lemma 7. $(S_3, S_2^{\neq}, [1/4, 1/2), [0, 1/2)) \to d$.

Proof. If only robot s is activated, it stays still and its state becomes S_{finish}. Therefore Lemma 4 applies. Otherwise r moves over s's location, traveling a distance of d, while s does not move. □

Lemma 8. $(S_3, S_2^{\neq}, [1/4, 1/2), [1/2, +\infty)) \to 2d$.

Proof. If only robot r is activated, it moves to s, and *Rendezvous* is solved with a covered distance of d.

If only robot s is activated, it moves to the midpoint (and possibly switches to S_3). As a consequence, the distance observed by r becomes less than $1/4$, hence it stays still forever (it only switches to S_{finish} as soon as it is activated). On the other hand, s keeps moving to the midpoint, until it observes a distance lower than 1, switches to S_3, and finally moves to r, solving *Rendezvous* after having covered a distance of d.

If both robots are activated on the first cycle, two cases arise.

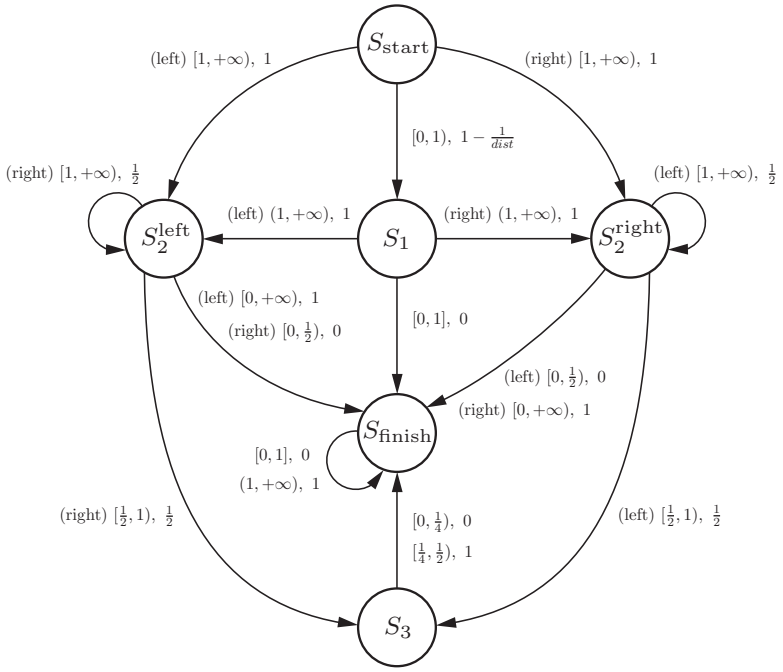

Figure 1: Illustration of Algorithm 1. A label of the form $(d)I, \lambda$ denotes a transition that applies when the other robot is seen in direction $d \in \{\text{left}, \text{right}\}$ and its observed distance lies in the interval $I \subset \mathbb{R}$. The computed destination point is $\lambda \cdot \mathit{other.position}$. For example, a robot in state S_{start} perceiving the other at distance $\geqslant 1$ on the right will move to the position of the other robot and will change state to S_2^{right}.

- If the distance observed by s lies in $[1/2, 1)$, configuration $(S_{\text{finish}}, S_3, [1/8, 1/4), [1/4, 1/2))$ is reached, and Lemma 4 applies.
- If the distance observed by s is at least 1, configuration $(S_{\text{finish}}, S_2^{=}, [1/8, 1/4), [1/2, +\infty))$ is reached (the two robots switch sides), and Lemma 6 applies.

In both cases, at the first move s reaches r's position, while r moves to the midpoint. Hence the total distance covered is $3d/2$, and the new distance is $d/2$. This distance is finally covered by s, and the total distance covered becomes $2d$. □

Lemma 9. $(S_2^{\neq}, S_2^{\neq}, [0, 1/2), [1/2, +\infty)) \to d$.

Proof. If both robots are activated, two cases arise. If the distance observed by s is less than 1, configuration $(S_{\text{finish}}, S_3, [0, 1/4), [1/4, 1/2))$ is reached, and Lemma 4 applies. Otherwise, if the distance is at least 1, configuration $(S_{\text{finish}}, S_2^{\neq}, [0, 1/4), [1/2, +\infty))$ is reached, and Lemma 5 applies. In both cases, s moves by $d/2$ on its first turn while r stays still, and then they move again by $d/2$.

If only r is activated, configuration $(S_{\text{finish}}, S_2^{\neq}, [0, 1/2), [1/2, +\infty))$ is reached, and Lemma 5 applies.

If only robot s is activated, two cases arise. If the distance observed by s is less than 1, configuration $(S_2^{\neq}, S_3, [0, 1/4), [1/4, 1/2))$ is reached, Lemma 7 applies, and the total distance covered is d. Otherwise, if the distance is at least 1, the configuration remains $(S_2^{\neq}, S_2^{\neq}, [0, 1/2), [1/2, +\infty))$, but the distance between the two robots halves. As the execution progresses, this case cannot repeat itself forever, because eventually the distance observed by s becomes less than 1, or r is activated.

When this happens, s has just approached r at every move, covering a distance of $d' < d$. Then the remaining $d - d'$ is covered as detailed above. □

Lemma 10. $(S_2^{\neq}, S_2^{\neq}, [1, +\infty), [1, +\infty)) \to d + u$.

Proof. If both robots are activated, they compute the midpoint and they gather, covering a distance of d. If only one robot is activated at each cycle, configuration $(S_2^{\neq}, S_2^{\neq}, [1, +\infty), [1, +\infty))$ keeps repeating itself for finitely many cycles, until the distance observed by some robot, say r, becomes less than 1. The configuration then becomes $(S_2^{\neq}, S_2^{\neq}, [1/2, 1), [1/2, +\infty))$. At this point, the distance covered is $d' < d$, and the new distance is $d - d' < u_r$.

Once again, if both robots are activated at the next cycle, they gather in the midpoint, and the total distance covered is d. If only s is activated, two cases arise. If the distance observed by s is less than 1, configuration $(S_2^{\neq}, S_3, [1/4, 1/2), [1/4, 1/2))$ is reached, and Lemma 7 applies. If the distance is at least 1, then configuration $(S_2^{\neq}, S_2^{\neq}, [1/4, 1/2), [1/2, +\infty))$ is reached, and Lemma 9 applies. In both cases, the total distance covered is d.

Finally, if only r is activated, it moves to the midpoint, and configuration $(S_3, S_2^{\neq}, [1/4, 1/2), [1/4, +\infty))$ is reached. The distance covered so far is $d' + (d - d')/2$, and the new distance is $(d - d')/2 < u_r/2$. Now Lemmas 7 and 8 apply so the two robots gather, and the distance they cover in this step is at most $d - d'$. Hence the total distance covered is at most $d + (d - d')/2 < d + u_r/2 < d + u$. □

Lemma 11. $(S_1, S_{\text{finish}}, [0, 1], (1, +\infty)) \to d$ *and* $(S_1, S_{\text{start}}, [0, 1], [1, +\infty)) \to d$.

Proof. Robot r switches to S_{finish} as soon as it is activated, and keeps staying still. Robot s moves to r as soon as it is activated. The total distance covered is therefore d. □

Lemma 12. $(S_1, S_{\text{start}}, \{1\}, [0, 1)) \to 4u - d$.

Proof. We distinguish three cases.

- If both robots are activated on the first cycle, they reach configuration $(S_{\text{finish}}, S_1, (1, +\infty), \{1\})$. While doing that, r stays still and s moves by $u_s - d$, and the new distance is u_s. Then Lemma 11 applies, and the final distance covered is $u_s - d + u_s = 2u_s - d < 4u - d$.

- If only robot r is activated on the first cycle, configuration $(S_{\text{finish}}, S_{\text{start}}, \{1\}, [0, 1))$ is reached. Now r keeps staying still and in state S_{finish}. As soon as s is activated, configuration $(S_{\text{finish}}, S_1, (1, +\infty), \{1\})$ is reached, and Lemma 11 applies. The total distance covered is once again $u_s - d + u_s = 2u_s - d < 4u - d$.

- If only robot s is activated on the first cycle, configuration $(S_1, S_1, (1, +\infty), \{1\})$ is reached. From now on, s keeps staying still (possibly switching to S_{finish}), whereas r moves to s as soon as it is activated. The distance covered is again $2u_s - d < 4u - d$.

□

Lemma 13. $(S_1, S_1, (1, +\infty), (1, +\infty)) \to 3d + u$.

Proof. If only one robot is activated, it moves to the other robot, and *Rendezvous* is solved with a distance covered of d. If both robots are activated, they turn S_2^{left} or S_2^{right} and switch positions, covering a distance of $2d$. Hence they reach configuration $(S_2^{\neq}, S_2^{\neq}, (1, +\infty), (1, +\infty))$, and Lemma 10 applies. The final distance covered is $2d + d + u = 3d + u$. □

Theorem 14. *In* SSYNCH, Rendezvous *of two* FSTATE *robots is solvable with six internal states. This result holds even without unit distance agreement. Moreover, if the average unit distance of the two robots is u, and their initial distance is d, they combinedly cover a distance of at most*

$$\max\{\, 9u - 5d,\ 3d + u \,\}.$$

Proof. We prove that $(S_{\text{start}}, S_{\text{start}}, [0, +\infty), [0, +\infty)) \to \max\{\, 9u - 5d,\ 3d + u \,\}$. Three cases arise.

- Let the configuration be $(S_{\text{start}}, S_{\text{start}}, [0, 1), [0, 1))$. If both robots are activated, configuration $(S_1, S_1, (1, +\infty), (1, +\infty))$ is reached. The distance covered is $(u_r - d) + (u_s - d) = 2u - 2d$ and the new distance is $u_r + u_s - d = 2u - d$. Now Lemma 13 applies, and the total distance covered is $2u - 2d + 3(2u - d) + u = 9u - 5d$.

  If only one robot is activated, say r, then configuration $(S_1, S_{\text{start}}, \{1\}, [0, +\infty))$ is reached. The distance covered is $u_r - d$ and the new distance is u_r. Now, if the configuration is $(S_1, S_{\text{start}}, \{1\}, [0, 1))$, Lemma 12 applies. The final distance covered is $u_r - d + 4u - u_r = 4u - d$. Instead, if the configuration is $(S_1, S_{\text{start}}, \{1\}, [1, +\infty))$, Lemma 11 applies, and the final distance covered is $u_r - d + u_r < 4u - d$.

- Let the configuration be $(S_{\text{start}}, S_{\text{start}}, [1, +\infty), [0, 1))$ (the symmetric case is equivalent, due to Observation 2). If only robot r is activated, it moves to s and *Rendezvous* is solved with a distance covered of d. If only s is activated, configuration $(S_{\text{start}}, S_1, (1, +\infty), \{1\})$ is reached, and Lemma 11 applies. The total distance covered is $u_s - d + u_s < 4u - d$. Finally, if both robots are activated, configuration $(S_2^{=}, S_1, [0, +\infty), [0, 1))$ is reached. The combined distance covered is $(d) + (u_s - d) = u_s$, and the new distance is $u_s - d$. Next, if only robot s is activated, configuration $(S_2^{=}, S_{\text{finish}}, [0, +\infty), [0, 1))$ is reached, and Lemma 6 applies. In any other case, r moves to s and *Rendezvous* is solved. In all cases, the total distance covered is $u_s + u_s - d < 4u - d$.

- Let the configuration be $(S_{\text{start}}, S_{\text{start}}, [1, +\infty), [1, +\infty))$. If only one robot is activated, it moves to the other robot, and *Rendezvous* is solved with a total distance covered of d. If both robots move, they switch positions, and the configuration becomes $(S_2^{\neq}, S_2^{\neq}, [1, +\infty), [1, +\infty))$. The combined distance covered is $2d$, and the new distance is again d. Then Lemma 10 applies, and the final distance covered is $2d + d + u = 3d + u$.

Hence the total distance covered is at most

$$\max\{\, 4u - d,\ 9u - 5d,\ 3d + u \,\}.$$

However, note that $4u - d < 9u - 5d$ if $d \leqslant u$ and $4u - d < 3d + u$ if $d > u$. Therefore,

$$\max\{\, 4u - d,\ 9u - 5d,\ 3d + u \,\} = \max\{\, 9u - 5d,\ 3d + u \,\}.$$

□

The upper bound we computed on the distance covered by robots executing Algorithm 1 is in fact asymptotically tight, as d/u grows.

Proposition 15. *Two robots executing Algorithm 1 may cover a combined distance that is arbitrarily close to the upper bound given by Theorem 14, for arbitrarily large values of d/u.*

Proof. Indeed, let n be a positive integer, let ε be a small-enough positive number, let $u_s = \varepsilon \cdot u_r$, and let $d = 2^n(1-\varepsilon)u_r$. Hence $u = u_r(1+\varepsilon)/2$ and the initial configuration is $(S_{\text{start}}, S_{\text{start}}, [1, +\infty), [1, +\infty))$. On the first turn we activate both robots, which switch positions and cover a total distance of $2d$. The new configuration is $(S_2^{\neq}, S_2^{\neq}, [1, +\infty), [1, +\infty))$. Now we activate only robot r, n times in a row. r keeps observing a distance greater than 1, hence it keeps moving to the midpoint. Eventually it reaches a distance of $(1-\varepsilon)u_r$, having covered a distance of $d - (1-\varepsilon)u_r$. Now, assuming that ε is small enough, the configuration is $(S_2^{\neq}, S_2^{\neq}, [1/2, 1), [1, +\infty))$. Once again we activate only robot r, which moves to the midpoint, covering a distance of $(1-\varepsilon)u_r/2$. The configuration becomes $(S_3, S_2^{\neq}, [1/4, 1/2), [1/2, +\infty))$, and now we activate both robots. r moves to s's position and gets state S_{finish}, while s moves to the midpoint, becoming either S_3 or $S_2^{=}$. The combined distance covered is $(1-\varepsilon)3 \cdot u_r/4$, and the new distance is $(1-\varepsilon)u_r/4$. Now, if only s is activated, regardless of its state, it reaches the other robot, covering a distance of $(1-\varepsilon)u_r/4$. In total, the distance covered is

$$2d + d - (1-\varepsilon) \cdot u_r + \frac{1-\varepsilon}{2} \cdot u_r + \frac{3(1-\varepsilon)}{4} \cdot u_r + \frac{1-\varepsilon}{4} \cdot u_r = 3d + \frac{1-\varepsilon}{2} \cdot u_r = 3d + \frac{1-\varepsilon}{1+\varepsilon} \cdot u,$$

which converges to $3d + u$ as ε vanishes. □

4 Finite-Communication Robots

We now focus on FCOMM robots, distinguishing the asynchronous and the semi-synchronous cases.

4.1 Asynchronous

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

Theorem 16. *In* ASYNCH, Rendezvous *of two* FCOMM *robots is unsolvable by algorithms in $\mathcal{L}$, regardless of the amount of colors employed.*

Proof. For each robot, the destination point and the next state are a function of the state of the other robot only. Assuming that both robots start in the same state, we let them perform their execution synchronously. As soon as both robots compute the midpoint m as a result of seeing each other in state A, we let only robot r complete its cycle. Meanwhile, s has computed m but still has not updated its state, nor moved. Therefore, r keeps seeing s set to A, and computes the new midpoint without changing its own state. We let r complete another cycle, and then we let s update its state and reach m. As a result, both robots are back in the same state and have not gathered. By repeating this pattern, the robots never solve *Rendezvous*. □

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.

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.

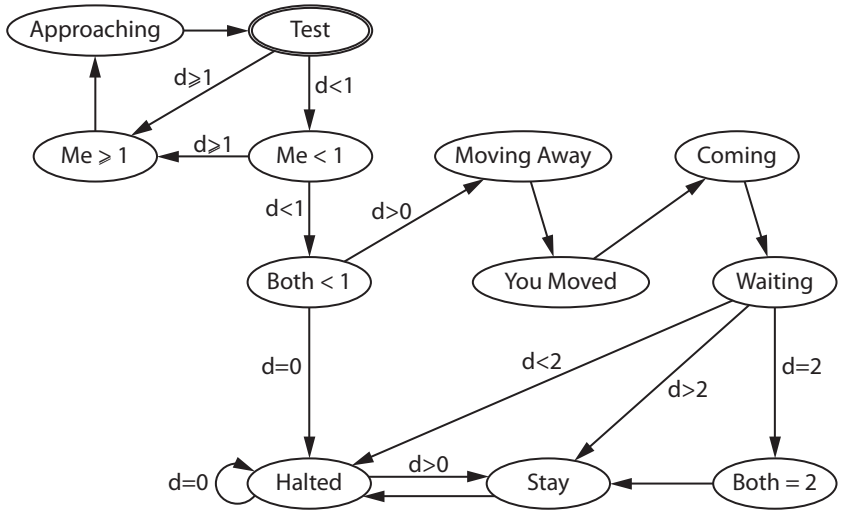

Figure 2: State transitions in Algorithm 2.

Theorem 17. *In* ASynch, Rendezvous *of two* FComm *robots is solvable with 12 colors. This result holds even without unit distance agreement.*

Proof. We show that Algorithm 2, also depicted in Figure 2, correctly solves *Rendezvous*. Both robots start in state (Test), and then update their state to (Me $\geqslant 1$) or (Me < 1), depending on 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 a 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

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
```

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 a 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 the midpoint. □

Proposition 18. *Two robots executing Algorithm 2 cover a combined distance of at most*

$$\max\{\, 8u - d,\ 3d + 6u \,\}.$$

Moreover, the distance covered may be arbitrarily close to this upper bound, for arbitrarily large and arbitrarily small values of d/u.

Proof. If $u_s \leqslant d < u_r$, no robot moves until s turns (Me $\geqslant 1$), r sees it, and starts approaching it. s never moves in this phase, because r keeps communicating that its observed distance is lower than 1. Instead, s keeps communicating that its observed distance is at least 1, until r has approached enough, and their distance is $d' < u_s < u_r$. At this point, s turns (Test), r turns (Me < 1), and s turns (Both < 1). When r sees this, it moves away by $u_r - d'/2$, while s turns (You Moved). Hence r moves to s's position while s waits, and the robots gather. The total distance covered is

$$d - d' + u_r - \frac{d'}{2} + u_r + \frac{d'}{2} = d - d' + 2u_r < 3u_r < 6u.$$

If $u_r \leqslant d < u_s$, the analysis is similar.

If both u_r and u_s are greater than d, no robot moves until one of them turns (Moving Away). If only one robot does so, say, r, the analysis is similar to that of the previous case: r moves away by $u_r - d/2$ and then reaches s. The total distance covered is $2u_r < 4u$. Instead, if both robots turn (Moving Away), they collectively move by $2u - d$, ending up at distance $2u$. Now, either one robot reaches the other and they gather, or they switch positions. In the first case, the final distance covered is $4u - d < 8u - d$. In the other case, the robots switch positions, covering a

distance of $4u$, and then they gather, covering a distance of $2u$. The total distance covered in this case is exactly $8u - d$. Clearly, there exists a schedule that makes the robots cover exactly this distance, for arbitrarily small values of d/u. Also note that $8u - d > 7u$. Since the upper bound obtained in the previous paragraph was less than $6u$, the distance covered by the robots (in terms of u) is strictly greater in this case.

Finally, assume that neither u_r's nor u_s's unit distance is greater than d. At least one of the two robots eventually turns (Approaching) and moves to the midpoint. After they have approached, at least one will eventually see the other set to (Test), and the test will be repeated. This process will continue until at least one of the two robots will see the other at a distance not greater than its unit distance. Let d' be the distance of the two robots at this point. By the previous paragraphs' reasoning, the distance the robots cover from this time onward is at most $8u - d' > 7u$.

Let d_i be the distance between the two robots as they perform the i-th test, with $d_1 = d$ and $d_k = d'$. Between two consecutive tests, either only one robot approaches the other, or both robots approach each other. In the first case, only one robot will turn (Test) afterwards, hence the symmetry will be broken and it will never be restored (that is, the two robots will never be found in the same state). Therefore, at any subsequent test, only one robot will actually turn (Test) and only one robot will approach the other. Eventually, after the i-th test, they will cover a distance of at most $d_i + 4u$.

Otherwise, if both robots approach each other between the i-th and the $(i+1)$-th test, they may gather in the midpoint. However, if this happens, they do not cover the maximum distance possible, therefore we ignore this case. The only other behavior that the robots may have is the following: one robot, say r, sees s set to (Me $\geqslant 1$), so it plans to turn (Approaching) and move to the midpoint. Before this happens, s sees r still set to (Me $\geqslant 1$), hence it moves to the midpoint. Then s performs another cycle, and keeps moving towards r, to the new midpoint. This continues until r actually moves to the old midpoint, and then a new test is performed. At the end of this process, the new distance d_{i+1} is such that $d_i/4 \leqslant d_{i+1} < d_i/2$. The total distance covered during this phase is exactly $d_i + d_{i+1}$.

From the above analysis it follows that, if the symmetry is never broken (i.e., if both robots approach each other between every pair of consecutive tests), the maximum distance they may cover is

$$d + 2\sum_{i=2}^{k-1} d_i + d_k + 8u - d_k = d + 2\sum_{i=2}^{k-1} d_i + 8u < d + 2\sum_{i=2}^{k-1} \frac{d}{2^{i-1}} + 8u = 3d - \frac{d}{2^{k-3}} + 8u.$$

This also means that d_{k-1} at least as large as u_r and u_s (because both robots approach each other after the $(k-1)$-th test), and therefore $d_{k-1} \geqslant u$. This implies that $d = d_1 > 2^{k-2}u$, hence

$$3d - \frac{d}{2^{k-3}} + 8u < 3d - 2u + 8u = 3d + 6u.$$

There actually exist schedules that yield covered distances that are arbitrarily close to $3d + 6u$, for arbitrarily large values of d/u. Indeed, in the previous analysis, we may assume that $u_r = u_s = u$ and $d = 2^{k-2}(1+\varepsilon)u$, for a sufficiently small $\varepsilon > 0$. Every time a test is performed, both robots approach each other and one of them performs sufficiently many turns before the other one moves, in such a way that $d_i \geqslant 2^{k-i-1}(1+\varepsilon/i)u$. Clearly, as ε vanishes, the resulting distance covered by the robots tends to $3d + 6u$.

To conclude, we observe that, if the symmetry is broken after the j-th test, the total distance covered is at most

$$d + 2\sum_{i=2}^{j} d_i + 4u < d + 2\sum_{i=2}^{j} \frac{d}{2^{i-1}} + 4u = 3d - \frac{d}{2^{j-2}} + 4u < 3d + 4u < 3d + 6u.$$

□

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 and Figure 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
```

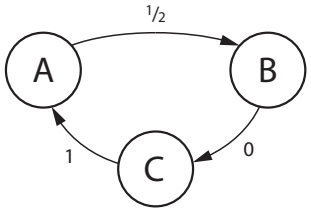

Figure 3: State transitions in Algorithm 3.

Theorem 19. *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.*

Proof. We show that Algorithm 3 (see also Figure 3) correctly solves *Rendezvous* from any initial configuration. Assume first that both robots start in the same state and both are activated at each turn. Then they keep having equal states, and they cycle through states A, B, and C forever. Every time they are both set to A, they move toward the midpoint and their distance reduces by at least 2δ, until it becomes so small that they actually gather.

Otherwise, if at some point the two robots are in different states, they will keep staying in different states forever. In this case their distance will never increase, and they will periodically be found in states B and C, respectively. Whenever this happens, the robot set to C retains its state and stays still until the other robot is activated and moves toward it by at least δ. As soon as their distance becomes not greater than δ and they turn again B and C, they finally gather. □

It is straightforward to observe the following.

Observation 20. *If movements are rigid, two robots executing Algorithm 3, initially at distance d and both in state A, cover a combined distance of exactly d.*

However, if movements are not rigid, the algorithm may be very inefficient.

Proposition 21. *Two robots executing Algorithm 3, initially at distance d and both in state A, cover a combined distance of at most*

$$(d + 2\delta) \left\lceil \frac{d}{2\delta} \right\rceil - d.$$

This upper bound is tight.

Proof. If both robots are activated at each turn, they keep having the same internal state. They alternate between moving towards the midpoint and moving towards each other's position. In order to make them cover the maximum possible distance, the scheduler must make them move as little as possible when their target is the midpoint (hence by at most δ), and let them switch positions otherwise. Then, every three turns, the robots' distance decreases by 2δ and they switch position. Eventually, the total distance covered by the two robots combined is

$$d + 2\sum_{i=1}^{k-1}(d - 2i\delta),$$

where $k = \lceil d/(2\delta) \rceil$. Let d' be such that $d = 2(k-1)\delta + d'$ (hence $0 < d' \leqslant 2\delta$). Then the total distance covered becomes

$$d + 2(k-1)d - 4\delta \, \frac{k(k-1)}{2} = 2kd - d - (d - d')k = (d + d')k - d \leqslant (d + 2\delta) \left\lceil \frac{d}{2\delta} \right\rceil - d.$$

Whenever $d = 2k\delta$, that is, $d' = 2\delta$, the above expression becomes equal to kd, the inequality becomes an equality, and there exists an activation schedule that makes the robots cover exactly that distance, implying that the upper bound is matched for arbitrarily large values of d.

If, at any time, only one robot is activated, then the two robots' states become different, and will remain different forever. Then, the sequence of movements will cycle through these three state configurations: one robot in A and one in B, one robot in B and one in C, and one robot in C and one in A. In the first phase, the robot in B keeps moving toward the midpoint until the robot in A is activated. In the second phase the robot in B moves toward the other robot exactly once. In the third phase, (they either gather or) the robot in C moves exactly once toward the midpoint, and the robot in A may move toward the other robot's position.

Clearly, the largest possible distance covered is obtained by repeating the following activation scheme: no robot moves in the first phase, one robot moves by at most δ in the second phase, one robot moves to the other robot's position, while the other robot moves to the midpoint, in the third phase. At each cycle, if the starting distance is d, the robots cover a distance of $\delta + 3(d - \delta)/2 < d + d/2$, and the new distance is $(d - \delta)/2$. Hence, the total distance covered when the robots gather is strictly smaller than

$$d + 2\sum_{i=1}^{\infty} \frac{d}{2^i} = 3d.$$

Let us compare this bound with the previous one. Clearly, if $k = \lceil d/(2\delta) \rceil \geqslant 3$, then

$$(d + 2\delta) \left\lceil \frac{d}{2\delta} \right\rceil - d \geqslant dk + d - d \geqslant 3d.$$

Therefore, if the two robots start behaving asymmetrically when their distance is greater than 6δ, they fail to cover the maximum possible distance. Otherwise, if the distance is smaller, it is easy to verify manually that making them move symmetrically is still better. Indeed, note that when the robots start being asymmetric, regardless of their stats, they must first reduce their distance by at least δ, before being able to switch sides. □

If the robots' initial state is not necessarily A, the upper bounds of Observation 20 and Proposition 21 increase by $2d$, as it is easy to infer.

Note that the number of colors used by Algorithm 3 is optimal. This follows as a corollary of the impossibility result when lights are visible to both robots:

Lemma 22. *[24] 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

Theorem 23. *In non-rigid* SSynch, Rendezvous *of two* FState *robots with knowledge of δ is solvable with three colors. Moreover, if the initial distance is d, the combined distance covered by the two robots is at most*

$$\begin{cases} 4\delta - 5d & \text{if } d < \frac{\delta}{2}, \\ 3d & \text{if } \frac{\delta}{2} \leqslant d < \delta, \\ d + 2\delta & \text{if } d \geqslant \delta. \end{cases}$$

Proof. We show that Algorithm 4 correctly solves *Rendezvous*. Both robots start in state A. Suppose first that the initial distance is in the interval $[\delta/2, \delta)$. Then the robots' movements are rigid, i.e., the robots always reach the destinations they compute. If only one robot is activated, it reaches its partner and *Rendezvous* is solved, with a distance covered of d. Otherwise, they both turn B and switch positions, covering a total distance of $2d$. Then, if both robots are activated, they gather in the midpoint. Otherwise, one of them turns C and moves to the midpoint. Now, the robot still in B keeps staying still because it observes a distance lower than $\delta/2$. On the other hand, the robot set to C moves to its partner as soon as it is activated. In all cases, the total distance covered is $3d$.

Suppose now that the initial distance is $d \geqslant \delta$. Then, any robot that is activated moves toward the point located $\delta/4$ before the midpoint, until the distance becomes smaller than δ. Let d' be the first distance lower than δ observed by some robot. It is immediate to see that d' lies in the interval $[\delta/2, \delta)$. At this point, the combined distance covered is $d - d'$. Now the previous reasoning applies, and the robots cover at most an additional distance of $3d'$. In total, they cover $d - d' + 3d' \leqslant d + 2\delta$.

Finally, assume that the initial distance is $d < \delta/2$ (we let $d > 0$, because if $d = 0$ both robots immediately terminate). Then, as soon as a robot is activated, it moves away from its partner, to the point at distance $\delta/2$. The distance is lower than δ, hence these movements are rigid. If only one robot is activated, it covers a distance of $\delta/2 - d$, and the new distance is exactly $\delta/2$. Since both robots are still in state A, the first paragraph's reasoning applies, and the total distance covered is at most $\delta/2 - d + 3\delta/2 = 2\delta - d$. In turn, $2\delta - d < 4\delta - 5d$. Instead, if both robots are activated on the first turn, they move by $\delta - 2d$, and the new distance is $d + \delta - 2d = \delta - d$,

which lies in the interval $[\delta/2, \delta)$. Hence the first paragraph's argument applies again, and the final distance covered is at most $\delta - 2d + 3(\delta - d) = 4\delta - 5d$. □

Algorithm 4 Rendezvous for non-rigid SSYNCH with knowledge of δ and three internal states

```
1: dist ← ‖other.position‖
2: if dist = 0 then
3:     terminate
4: if me.state = A then
5:     if dist < δ/2 then                          ▷ reach the point at distance δ/2 from the other
6:         me.destination ← other.position · (1 − δ/(2 · dist))
7:     else if δ/2 ⩽ dist < δ then                 ▷ gather or switch positions
8:         me.state ← B
9:         me.destination ← other.position
10:    else                                        ▷ dist ⩾ δ, reach the point at distance δ/4 from the midpoint
11:        me.destination ← other.position · (1/2 − δ/(4 · dist))
12: else if me.state = B then
13:    if δ/2 ⩽ dist < δ then
14:        me.state ← C
15:        me.destination ← other.position/2
16: else                                           ▷ me.state = C
17:    me.destination ← other.position
```

5.2 FComm Robots

Theorem 24. *In non-rigid* ASYNCH, Rendezvous *of two* FCOMM *robots with knowledge of δ is solvable with three colors. Moreover, if the initial distance is d, the combined distance covered by the two robots is at most*

$$\begin{cases} 6\delta - 4d & \text{if } d < \delta, \\ 2d & \text{if } \delta \leqslant d < 2\delta, \\ d + 2\delta & \text{if } d \geqslant 2\delta. \end{cases}$$

Proof. We show that Algorithm 5 correctly solves *Rendezvous*. Suppose that, at some point during the execution, say at time t, robot r is in state READY and it is not moving, and robot s sees it. Suppose that the distance between r and s at this time is d', with $\delta \leqslant d' < 2\delta$. Then, s turns COME and moves to the midpoint. The midpoint is eventually reached, because the distance traveled is not greater than δ. Assume first that both robots are in state READY at time t, both see each other, and move toward the midpoint. If some robot, say r, reaches the midpoint and sees its partner s still on its way and set to COME, r turns READY and keeps chasing s. When s reaches its destination (i.e., the old midpoint) and sees r set to READY and at distance at most δ, it stays still and waits until *Rendezvous* is solved. In this case, the distance covered by s is $d'/2$, and the distance covered by r is at most $3d'/2$ (achieved when r reaches s before s starts moving, and then r follows s toward the former midpoint). Hence the total is at most $2d'$. Similarly, assume that only s sees r set to READY at time t, and s turns COME without ever being seen by r in state READY. s will reach the midpoint and stay there, while r will start chasing s until they meet in the midpoint. Once again, the combined distance covered is at most $2d'$.

Algorithm 5 Rendezvous for non-rigid ASYNCH with knowledge of δ and three externally visible states

```
1: dist ← ‖other.position‖
2: if other.state = START then
3:     if dist = 0 then                                   ▷ we have already gathered
4:         me.state ← COME
5:     else if dist < δ then          ▷ moving to the point at distance δ from the other
6:         me.state ← START
7:         me.destination ← other.position · (1 − δ/dist)
8:     else if dist ⩾ 2δ then                  ▷ moving by δ/2 toward the other
9:         me.state ← START
10:        me.destination ← other.position · δ/(2 · dist)
11:    else                                    ▷ δ ⩽ dist < 2δ, ready to gather
12:        me.state ← READY
13: else if other.state = READY then
14:    me.state ← COME
15:    if δ ⩽ dist < 2δ then                              ▷ reaching the midpoint
16:        me.destination ← other.position/2
17: else                                                  ▷ other.state = COME
18:    if dist = 0 then                                      ▷ we have gathered
19:        me.state ← COME
20:        terminate
21:    else
22:        me.state ← READY
23:        me.destination ← other.position
```

As the execution begins, both robots are in state START. If the initial distance d lies in the interval $[\delta, 2\delta)$, as soon as a robot is activated it turns READY and stays still, and at some point it will be seen by the other robot. Hence the previous reasoning applies, and the robots gather after covering a distance of at most $2d$.

If the initial distance is $d < \delta$, the first robot that is activated, say r, moves away by $\delta - d$ (unless $d = 0$, in which case no robot ever moves). Because this distance is lower than δ, r actually reaches its destination. If s is never activated before r reaches its target, then the new distance is exactly δ, the first robot to be activated turns READY, and the first paragraph's reasoning applies. The resulting distance covered is at most $\delta - d + 2\delta = 3\delta - d$, which is less than $6\delta - 4d$. Otherwise, if s sees r before it has reached its target, say at distance d', with $d \leqslant d' < \delta$, then s moves away by $\delta - d'$. Hence the two destination points are exactly $2\delta - d'$ apart. Therefore, the first robot to reach its destination and perform a new LOOK observes a distance in the interval $[\delta, 2\delta)$, turns READY, and waits. The second robot to perform a LOOK observes a distance of exactly $2\delta - d'$, which is again in the interval $[\delta, 2\delta)$, and hence the first paragraph's reasoning applies again. In total, the robots cover at most a distance of $(\delta - d) + (\delta - d') + 2(2\delta - d') \leqslant 6\delta - 4d$.

Finally, if the initial distance is $d \geqslant 2\delta$, the robots keep approaching each other by $\delta/2$ at each step, until they observe a distance lower than 2δ. Let r be the last robot to observe the other at a distance not lower than 2δ, at time t. Then, the destination of r is $\delta/2$ toward robot s's position. In turn, the destination of s at time t is up to $\delta/2$ toward robot r's position. Hence, when the first robot reaches its destination point and performs a LOOK, it observes a distance lower than 2δ (by definition of t), which also happens to be in the interval $[\delta, 2\delta)$. Hence it turns READY and waits. When also the other robot reaches its destination, the distance d' is still in the interval $[\delta, 2\delta)$, and hence the first paragraph's reasoning applies. The total distance covered is therefore at most $d - d' + 2d' = d + d' < d + 2\delta$. □

6 Open Problems

We have shown that rendezvous can be obtained both in FSTATE and FCOMM, two models substantially weaker than the one of [9], where both internal memory and communication memory capabilities are present. Our results open several new problems and research questions.

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 [5].

References

[1] N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36:56–82, 2006.

[2] Z. Bouzid, S. Das, and S. Tixeuil. Gathering of Mobile Robots Tolerating Multiple Crash Faults. In *Proceedings of 33rd IEEE International Conference on Distributed Computing Systems* (ICDCS), 337-346, 2013.

[3] Z. Bouzid, M. Gradinariu Potop-Butucaru, and S. Tixeuil. Byzantine convergence in robot networks: The price of asynchrony. In *Proceedings of 13th Int. Conference Principles of Distributed Systems* (OPODIS), 54–70, 2009.

[4] Z. Bouzid, S. Dolev, M. Gradinariu Potop-Butucaru, and S. Tixeuil. RoboCast: Asynchronous Communication in Robot Networks. In *Proceedings of 14th Int. Conference on Principles of Distributed Systems* (OPODIS), 16–31, 2010.

[5] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: Gathering. *SIAM Journal on Computing*, 41(4): 829-879, 2012.

[6] R. Cohen and D. Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM Journal on Computing*, 34:1516–1528, 2005.

[7] R. Cohen and D. Peleg. Convergence of autonomous mobile robots with inaccurate sensors and movements. In *Proceedings of 23rd International Symposium on Theoretical Aspects of Computer Science* (STACS), 549—560, 2006.

[8] A. Cord-Landwehr, B. Degener, M. Fischer, M. Hüllmann, B. Kempkes, A. Klaas, P. Kling, S. Kurras, M. Mrtens, F. Meyer auf der Heide, C. Raupach, K. Swierkot, D. Warner, C. Weddemann, and D. Wonisch. A new approach for analyzing convergence algorithms for mobile robots. In *Proceedings of 38th International Colloquium on Automata, Languages and Programming* (ICALP), 650–661. Springer, 2011.

[9] S. Das, P. Flocchini, G. Prencipe, N. Santoro, and M. Yamashita. The power of lights: synchronizing asynchronous robots using visible bits. In *Proceedings of the 32nd International Conference on Distributed Computing Systems* (ICDCS), 506–515, 2012.

[10] X. Défago, M. Gradinariu, S. Messika, and P. Raipin-Parvédy. Fault-tolerant and self-stabilizing mobile robots gathering. In *Proc. 20th Intl. Symp. on Distributed Computing* (DISC), 46–60, 2006.

[11] B. Degener, B. Kempkes, T. Langner, F. Meyer auf der Heide, P. Pietrzyk, and R. Wattenhofer. 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), 139–148, 2011.

[12] Y. Dieudonné, S. Dolev, F. Petit, and M. Segal. Deaf, dumb, and chatting asynchronous robots. In *Proceedings 13th International Conference on Principles of Distributed Systems* (OPODIS), 71–85, 2009.

[13] Y. Dieudonné and F. Petit. Self-stabilizing gathering with strong multiplicity detection. *Theoretical Computer Science*, 428(13), 2012.

[14] P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Oblivious Mobile Robots*. Morgan & Claypool, 2012.

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

[16] T. Izumi, Z. Bouzid, S. Tixeuil, and K. Wada. Brief Announcement: The BG-simulation for Byzantine mobile robots. *Proceedings of 25-th International Symp. on Distributed Computing* (DISC), 330–331, 2011.

[17] T. Izumi, S. Souissi, Y. Katayama, N. Inuzuka, X. Defago, K. Wada, and M. Yamashita. The gathering problem for two oblivious robots with unreliable compasses. *SIAM Journal on Computing*, 41(1):26–46, 2012.

[18] S. Kamei, A. Lamani, F. Ooshita, and S. Tixeuil. Asynchronous mobile robot gathering from symmetric configurations without global multiplicity detection. In *Proceedings of 18th International Colloquium on Structural Information and Communication Complexity* (SIROCCO), 150–161, 2011.

[19] J. Lin, A.S. Morse, and B.D.O. Anderson. The multi-agent rendezvous problem. parts 1 and 2. *SIAM Journal on Control and Optimization*, 46(6):2096–2147, 2007.

[20] L. Pagli, G. Prencipe, and G. Viglietta. Getting close without touching. In *Proceedings of 19th International Colloquium on Structural Information and Communication Complexity* (SIROCCO), 315–326, 2012.

[21] G. Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theoretical Computer Science*, 384(2-3):222–231, 2007.

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

[23] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, vol. 28, pp. 1347–1363, 1999.

[24] G. Viglietta. Rendezvous of two robots with visible bits. In *Proceedings of 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics* (ALGOSENSORS), 291-306, 2013.