

# Rendezvous of two robots with visible bits

Giovanni Viglietta  
viglietta@gmail.com

November 7, 2018

## Abstract

We study the rendezvous problem for two robots moving in the plane (or on a line). Robots are autonomous, anonymous, oblivious, and carry colored lights that are visible to both. We consider deterministic distributed algorithms in which robots do not use distance information, but try to reduce (or increase) their distance by a constant factor, depending on their lights' colors.

We give a complete characterization of the number of colors that are necessary to solve the rendezvous problem in every possible model, ranging from fully synchronous to semi-synchronous to asynchronous, rigid and non-rigid, with preset or arbitrary initial configuration.

In particular, we show that three colors are sufficient in the non-rigid asynchronous model with arbitrary initial configuration. In contrast, two colors are insufficient in the rigid asynchronous model with arbitrary initial configuration and in the non-rigid asynchronous model with preset initial configuration.

Additionally, if the robots are able to distinguish between zero and non-zero distances, we show how they can solve rendezvous and detect termination using only three colors, even in the non-rigid asynchronous model with arbitrary initial configuration.

## 1 Introduction

### 1.1 Models for mobile robots

The basic robot model we employ has been thoroughly described in [1, 2, 3, 4]. Robots are modeled as points freely moving in  $\mathbb{R}^2$  (or  $\mathbb{R}$ ). Each robot has its own coordinate system and its own unit distance, which may differ from the others. Robots operate in cycles that consist of four phases: WAIT, LOOK, COMPUTE, and MOVE.

In a WAIT phase a robot is idle; in a LOOK phase it gets a snapshot of its surroundings (including the positions of the other robots); in a COMPUTE phase it computes a destination point; in a MOVE phase it moves toward the destination point it just computed, along a straight line. Then the cycle repeats over and over.

Robots are anonymous and oblivious, meaning that they do not have distinct identities, they all execute the same algorithm in each COMPUTE phase, and the only input to such algorithm is the snapshot coming from the previous LOOK phase.

In a MOVE phase, a robot may actually reach its destination, or it may be stopped before reaching it. If a robot always reaches its destination by the end of each MOVE phase, then the model is said to be *rigid*. If a robot can unpredictably be stopped before, the model is *non-rigid*. However, even in non-rigid models, during a MOVE phase, a robot must always be found on the line segment between its starting point and the destination point. Moreover, there is a constant distance  $\delta > 0$  that a robot is guaranteed to walk at each cycle. That is, if the destination point that a robot computes is at most  $\delta$  away (referred to some global coordinate system), then the robot is guaranteed to reach it by the end of the next MOVE phase. On the other hand, if the destination point is more than  $\delta$  away, the robot is guaranteed to approach it by at least  $\delta$ .

In the basic model, robots cannot communicate in conventional ways or store explicit information, but a later addition to this model allows each robot to carry a “colored light” that is visible to every robot (refer to [2]). There is a fixed amount of possible light colors, and a robot can compute its destination and turn its own light to a different color during a COMPUTE phase, based on the light colors that it sees on other robots and on itself. Usually, when robots start their execution, they have all their lights set to a predetermined color. However, we are also interested in algorithms that work regardless of the initial color configurations of the robots.

In the fully synchronous model (FSYNCH) all robots share a common notion of time, and all their phases are executed synchronously. The semi-synchronous model (SSYNCH) is similar, but not every robot may be active at every cycle. That is, some robots are allowed to “skip” a cycle at unpredictable times, by extending their WAIT phase to the whole cycle. However, the robots that are active at a certain cycle still execute it synchronously. Also, no robot can remain inactive for infinitely many consecutive cycles. Finally, in the asynchronous model (ASYNCH) there is no common notion of time, and each robot’s execution phase may last any amount of time, from a minimum  $\epsilon > 0$  to an unboundedly long, but finite, time.

Figure 1 shows all the possible models arising from combining synchronousness, rigidity, and arbitrariness of the initial light colors. The trivial inclusions between models are also shown.

Without loss of generality, in this paper we will assume LOOK phases in ASYNCH to be instantaneous, and we will assume that a robot’s light’s color may change only at the very end of a COMPUTE phase.

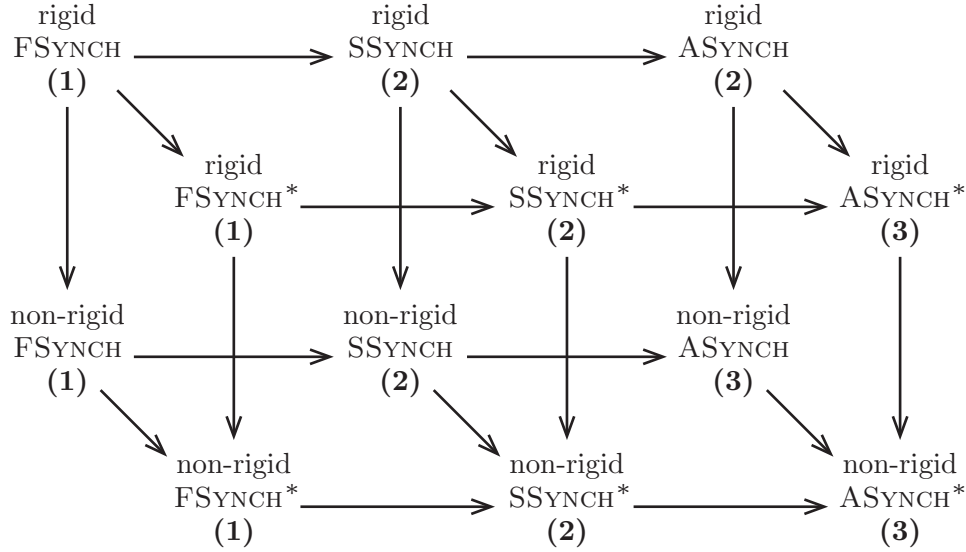


Figure 1: Robot models with their trivial inclusions. An asterisk means that the initial color configuration may be arbitrary; no asterisk means that it is fixed. The numbers indicate the minimum amounts of distinct colors that are necessary to solve RENDEZVOUS in each model (cf. Theorem 5.1).

## 1.2 Gathering mobile robots

GATHERING is the problem of making a finite set of robots in the plane reach the same location in a finite amount of time, and stay there forever, regardless of their initial positions. Such location should not be given as input to the robots, but they must implicitly determine it, agree on it, and reach it, in a distribute manner. Note that this problem is different from CONVERGENCE, in which robots only have to approach a common location, but may never actually reach it.

For any set of more than two robots, GATHERING has been solved in non-rigid ASYNCH, without using colored lights (see [1]). The special case with only two robots is also called RENDEZVOUS, and it is easily seen to be solvable in non-rigid FSYNCH but unsolvable in rigid SSYNCH, if colored lights are not used (see [5]).

**Proposition 1.1.** *If only one color is available, RENDEZVOUS is solvable in non-rigid FSYNCH and unsolvable in rigid SSYNCH.*

*Proof.* In non-rigid FSYNCH, consider the algorithm that makes each robot move to the midpoint of the current robots' positions. At each move, the distance between the two robots is reduced by at least  $2\delta$ , until it becomes less than  $2\delta$ , and the robots gather.

Suppose that an algorithm exists that solves RENDEZVOUS in rigid SSYNCH by using just one color. Let us assume that the two robots' axes are oriented

symmetrically, in opposite directions. This implies that, if we activate both robots at each cycle, they obtain isometric snapshots, and thus they make moves that are symmetric with respect to their current midpoint. Therefore, by doing so, the robots can never meet unless they compute the midpoint. If they do it, we just activate one robot for that cycle (and each time this happens, we pick a different robot, alternating). As a result, the robots never meet, regardless of the algorithm.  $\heartsuit$

However, in [2] it was shown how RENDEZVOUS can be solved even in non-rigid ASYNCH using lights of four different colors, and starting from a preset configuration of colors. Optimizing the amount of colors was left as an open problem.

### 1.3 Our contribution

In this paper, we will determine the minimum number of colors required to solve RENDEZVOUS in all models shown in Figure 1, with some restrictions on the class of available algorithms.

Recall that robots do not necessarily share a global coordinate system, but each robot has its own. If the coordinate system of a robot is not even self-consistent (i.e., it can unpredictably change from one cycle to another), then the only reliable reference for each robot is the position of the other robot(s) around it. In this case, the only type of move that is consistent will all possible coordinate systems is moving to a linear combination of the robots' positions, whose coefficients may depend on the colored lights. In particular, when the robots are only two, we assume that each robot may only compute a destination point of the form

$$(1 - \lambda) \cdot me.position + \lambda \cdot other.position,$$

for some  $\lambda \in \mathbb{R}$ . In turns,  $\lambda$  is a function of *me.light* and *other.light* only. This class of algorithms will be denoted by  $\mathcal{L}$ .

In Section 2, we will prove that two colors are sufficient to solve RENDEZVOUS in non-rigid SSYNCH with arbitrary initial configuration and in rigid ASYNCH with preset initial configuration, whereas three colors are sufficient in non-rigid ASYNCH with arbitrary initial configuration. All the algorithms presented are of class  $\mathcal{L}$ .

On the other hand, in Section 3 we show that even termination detection can be achieved in non-rigid ASYNCH with arbitrary initial configuration using only three colors, although our algorithm is not of class  $\mathcal{L}$  (indeed, no algorithm of class  $\mathcal{L}$  can detect termination in RENDEZVOUS).

In contrast, in Section 4 we prove that no algorithm of class  $\mathcal{L}$  using only two colors can solve RENDEZVOUS in rigid ASYNCH with arbitrary initial configuration or in non-rigid ASYNCH with preset initial configuration.

Finally, in Section 5 we put all these results together and we conclude with a complete characterization of the minimum amount of colors that are needed to solve RENDEZVOUS in every model (see Theorem 5.1).

## 2 Algorithms for rendezvous

### 2.1 Two colors for the non-rigid semi-synchronous model

For non-rigid SSYNCH, we propose Algorithm 1, also represented in Figure 2. Labels on arrows indicate the color that is seen on the other robot, and the destination of the next MOVE with respect to the position of the other robot. “0” stands for “do not move”, “1/2” means “move to the midpoint”, and “1” means “move to the other robot”. The colors used are only two, namely  $A$  and  $B$ .

---

**Algorithm 1:** Rendezvous for non-rigid SSYNCH and rigid ASYNCH

---

```

 $me.destination \leftarrow me.position$ 
if  $me.light = A$  then
  if  $other.light = A$  then
     $me.light \leftarrow B$ 
     $me.destination \leftarrow (me.position + other.position)/2$ 
  else
     $me.destination \leftarrow other.position$ 
else if  $other.light = B$  then
   $me.light \leftarrow A$ 

```

---

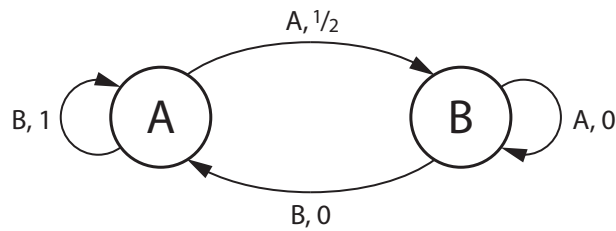


Figure 2: Illustration of Algorithm 1

**Lemma 2.1.** *If the two robots start a cycle with their lights set to opposite colors, they eventually gather.*

*Proof.* Both robots retain their colors at every cycle, and the  $A$ -robot keeps computing the other robot’s location, while the  $B$ -robot keeps waiting. Hence, their distance decreases by at least  $\delta$  for every cycle in which the

$A$ -robot is active, until the distance becomes smaller than  $\delta$ , and the robots gather.  $\heartsuit$

**Theorem 2.2.** *Algorithm 1 solves RENDEZVOUS in non-rigid SSYNCH, regardless of the colors in the initial configuration.*

*Proof.* If the robots start in opposite colors, they gather by Lemma 2.1. If they start in the same color, they keep alternating colors until one robot is active and one is not. If this happens, they gather by Lemma 2.1. Otherwise, the two robots are either both active or both inactive at each cycle, and they keep computing the midpoint every other active cycle. Their distance decreases by at least  $2\delta$  each time they move, until it becomes smaller than  $2\delta$ , and they finally gather.  $\heartsuit$

## 2.2 Two colors for the rigid asynchronous model

We prove that Algorithm 1 solves RENDEZVOUS in rigid ASYNCH as well, provided that the initial color is  $A$  for both robots.

**Lemma 2.3.** *If, at some time  $t$ , the two robots have opposite colors and neither of them is in a COMPUTE phase that will change its color, they will eventually solve RENDEZVOUS.*

*Proof.* Each robot retains its color at every cycle after time  $t$ , because it keeps seeing the other robot in the opposite color at every LOOK phase. As soon as the  $A$ -robot performs its first LOOK after time  $t$ , it starts chasing the other robot. On the other hand, as soon as the  $B$ -robot performs its first LOOK after time  $t$ , it stops forever. Eventually, the two robots will gather and never move again.  $\heartsuit$

**Theorem 2.4.** *Algorithm 1 solves RENDEZVOUS in rigid ASYNCH, provided that both robots start with their lights set to  $A$ .*

*Proof.* Let  $r$  be the first robot to perform a LOOK. Then  $r$  sees the other robot  $s$  set to  $A$ , and hence it turns  $B$  and computes the midpoint  $m$ . Then, as long as  $s$  does not perform its first LOOK,  $r$  stays  $B$  because it keeps seeing  $s$  set to  $A$ . Hence, if  $s$  performs its first LOOK after  $r$  has turned  $B$ , Lemma 2.3 applies, and the robots will solve RENDEZVOUS.

On the other hand, if  $s$  performs its first LOOK when  $r$  is still set to  $A$  (hence still in its starting location),  $s$  will turn  $B$  and compute the midpoint  $m$ , as well. If some robot reaches  $m$  and performs a LOOK while the other robot is still set to  $A$ , the first robot waits until the other turns  $B$ . Without loss of generality, let  $r$  be the first robot to perform a LOOK while the other robot is set to  $B$ . This must happen when  $r$  is in  $m$  and set to  $B$ , hence it will turn  $A$  and stay in  $m$ . If  $r$  turns  $A$  before  $s$  has reached  $m$ , then Lemma 2.3 applies. Otherwise,  $r$  turns  $A$  when  $s$  is already in  $m$ , and both

robots will stay in  $m$  forever, as they will see the other robot in  $m$  at every LOOK. ♡

### 2.3 Three colors for the non-rigid asynchronous model

For non-rigid ASYNCH, we propose Algorithm 2, also represented in Figure 3. The colors used are three, namely  $A$ ,  $B$ , and  $C$ .

---

**Algorithm 2:** Rendezvous for non-rigid ASYNCH

---

```

 $me.destination \leftarrow me.position$ 
if  $me.light = A$  then
  if  $other.light = A$  then
     $me.light \leftarrow B$ 
     $me.destination \leftarrow (me.position + other.position)/2$ 
  else if  $other.light = B$  then
     $me.destination \leftarrow other.position$ 
else if  $me.light = B$  then
  if  $other.light = B$  then
     $me.light \leftarrow C$ 
  else if  $other.light = C$  then
     $me.destination \leftarrow other.position$ 
else
  if  $other.light = C$  then
     $me.light \leftarrow A$ 
  else if  $other.light = A$  then
     $me.destination \leftarrow other.position$ 

```

---

**Observation 2.5.** *A robot retains its color if and only if it sees the other robot set to a different color.*

**Lemma 2.6.** *If, at some time  $t$ , the two robots are set to different colors, and neither of them is in a COMPUTE phase that will change its color, they will eventually solve RENDEZVOUS.*

*Proof.* The two robots keep seeing each other set to different colors, and hence they never change color, by Observation 2.5. One of the two robots will eventually stay still, and the other robot will then approach it by at least  $\delta$  at every MOVE phase, until their distance is less than  $\delta$ , and they gather. As soon as they have gathered, they will stay in place forever. ♡

**Theorem 2.7.** *Algorithm 2 solves RENDEZVOUS in non-rigid ASYNCH, regardless of the colors in the initial configuration.*

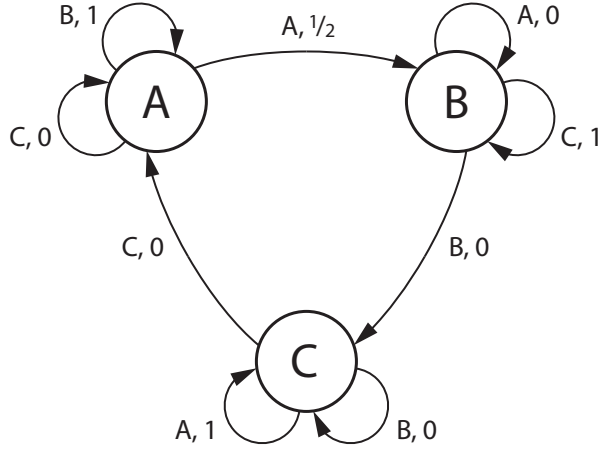


Figure 3: Illustration of Algorithm 2

*Proof.* If the robots start the execution at different colors, they solve RENDEZVOUS by Lemma 2.6.

If they both start in  $A$ , then let  $r$  be the first robot to perform a LOOK.  $r$  plans to turn  $B$  and move to the midpoint. If it turns  $B$  before the other robot  $s$  has performed a LOOK, then Lemma 2.6 applies.

Otherwise,  $s$  plans to turn  $B$  and move to the midpoint, as well. If a robot stops and sees the other robot still set to  $A$ , it waits. Without loss of generality, let  $r$  be the first robot to perform a LOOK and see the other robot set to  $B$ .  $r$  now plans to turn  $C$ , but if it does so before  $s$  has performed a LOOK, Lemma 2.6 applies.

So, let us assume that both robots have seen each other in  $B$  and they both plan to turn  $C$ . Once again, if a robot turns  $C$  and sees the other robot still in  $B$ , it waits. Without loss of generality, let  $r$  be the first robot to see the other robot in  $C$ .  $r$  plans to turn  $A$ , but if it does so before  $s$  has performed a LOOK, Lemma 2.6 applies.

Assume that both robots see each other in  $C$  and they both plan to turn  $A$ . If a robot turns  $A$  and sees the other robot still in  $C$ , it waits. At some point, both robots are in  $A$  again, in a WAIT phase, but they have approached each other. They both moved toward the midpoint in their first cycle, and then they just made null moves. As a consequence, if their distance was smaller than  $2\delta$ , they have gathered. Otherwise, the distance has decreased by at least  $2\delta$ . As the execution goes on and the same pattern of transitions repeats, the distance keeps decreasing until the robots gather. As soon as they have gathered, they never move again, hence RENDEZVOUS is solved.

The cases in which the robots start both in  $B$  or both in  $C$  are resolved with the same reasoning. Note that all the states with both robots set to the same color and in a WAIT phase have been reached in the analysis above. ♡



### 3 Termination detection

Suppose we wanted our robots to acknowledge that they have gathered, in order to turn off, or “switch gears” and start performing a new task.

**Observation 3.1.** *If the model is SSYNCH, termination detection is trivially obtained by checking at each cycle if the robots’ locations coincide.*

Unfortunately, in ASYNCH, correct termination detection is harder to obtain. Observe that both Algorithm 1 (for rigid ASYNCH) and Algorithm 2 (for non-rigid ASYNCH) fail to guarantee termination detection. Indeed, suppose that robot  $r$  is set to  $A$  and sees the other robot  $s$  set to  $B$ , and that the two robots coincide. Then  $r$  cannot tell if  $s$  is still moving or not. If  $s$  is not moving, it is safe for  $r$  to terminate, but if  $s$  is moving, then  $r$  has still to “chase”  $s$ , and cannot terminate yet.

To guarantee correct termination detection in non-rigid ASYNCH, we propose Algorithm 3, also represented in Figure 4. Note that different rules may apply depending on the distance between the two robots, indicated by  $d$  in the picture. However, robots need only distinguish between zero and non-zero distances. The colors used are again three, namely  $A$ ,  $B$ , and  $C$ .

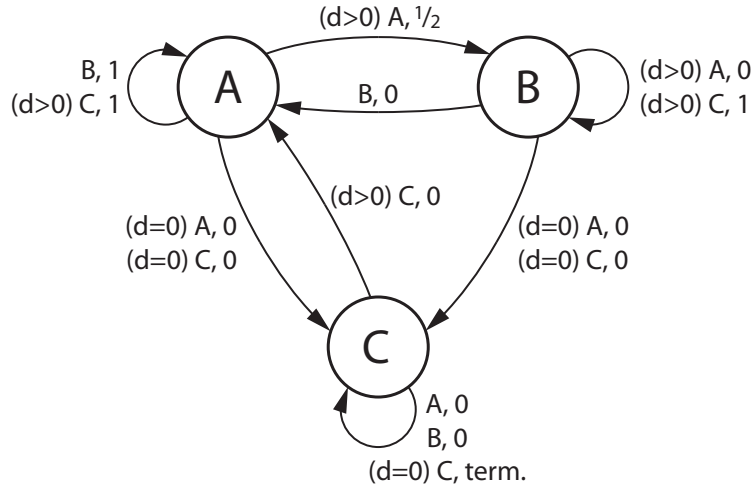


Figure 4: Illustration of Algorithm 3

**Observation 3.2.** *No robot can move while it is set to  $C$ .*

**Lemma 3.3.** *If some robot ever turns  $C$  from a different color, the two robots will gather and their execution will terminate correctly.*

*Proof.* A robot can turn  $C$  only if it performs a LOOK while the other robot is in the same location. If robot  $r$  performs a LOOK at time  $t$  that makes it turn  $C$ , then  $r$  stays  $C$  forever after, unless it sees the other robot  $s$  set to  $C$

---

**Algorithm 3:** Rendezvous for non-rigid ASYNCH with termination

---

```
me.destination  $\leftarrow$  me.position
if me.light = A then
  if other.light = A then
    if other.position  $\neq$  me.position then
      me.light  $\leftarrow$  B
      me.destination  $\leftarrow$  (other.position + me.position)/2
    else
      me.light  $\leftarrow$  C
  else if other.light = B then
    me.destination  $\leftarrow$  other.position
  else if other.position  $\neq$  me.position then
    me.destination  $\leftarrow$  other.position
  else
    me.light  $\leftarrow$  C
else if me.light = B then
  if other.light = A and other.position = me.position then
    me.light  $\leftarrow$  C
  else if other.light = B then
    me.light  $\leftarrow$  A
  else if other.position  $\neq$  me.position then
    me.destination  $\leftarrow$  other.position
  else
    me.light  $\leftarrow$  C
else if other.light = C then
  if other.position  $\neq$  me.position then
    me.light  $\leftarrow$  A
  else
    terminate
```

---

as well, in a different location. Let  $t' > t$  be the first time this happens. Due to Observation 3.2,  $r$  does not move between  $t$  and  $t'$ . On the other hand,  $s$  coincides with  $r$  at time  $t$ . Then  $s$  must turn some other color and move away from  $r$ , and then turn  $C$  at some time  $t''$  such that  $t < t'' \leq t'$ . But, in order to turn  $C$ ,  $s$  would have to coincide with  $r$ , which is a contradiction.

Hence  $r$  will stay  $C$  and never move after time  $t$ . As soon as  $s$  sees  $r$  set to  $C$ , it starts moving toward it (after turning  $A$ , if  $s$  is also set to  $C$  and not coincident with  $r$ ), covering at least  $\delta$  at each MOVE phase, until their distance becomes less than  $\delta$  and  $s$  finally reaches  $r$ . Then  $s$  will turn  $C$  as well, and both robots will terminate correctly after seeing each other again.  $\heartsuit$

**Lemma 3.4.** *If, at some time  $t$ , the two robots are set to  $A$  and  $B$  respectively, and neither of them is in a COMPUTE phase that will change its color, they will eventually gather and terminate correctly.*

*Proof.* If some robot ever turns  $C$  after time  $t$ , gathering and termination are ensured by Lemma 3.3. Otherwise, the two robots keep seeing each other set to opposite colors, and hence they never change color. The  $B$ -robot will eventually stay still, and the  $A$ -robot will then approach it by at least  $\delta$  at every MOVE phase, until their distance is less than  $\delta$ , and they gather. The  $B$ -robot then turns  $C$ , and Lemma 3.3 applies again.  $\heartsuit$

Let  $r(t)$  denote the position of robot  $r$  at time  $t \geq 0$ .

**Lemma 3.5.** *Let  $t$  be a time instant at which both robots are set to  $A$ , and neither of them is in a COMPUTE phase. Let us assume that robot  $r$  will stay still until the end of its current phase (even if it is a MOVE phase), and that robot  $s$  will either stay still until the end of its current phase, or its destination point is  $r$ 's current location. Then  $r$  and  $s$  will eventually gather and terminate correctly.*

*Proof.* If  $s$  is not directed toward  $r$  at time  $t$ , let  $d$  be the distance between  $r(t)$  and  $s(t)$ . Otherwise, let  $t'$  be the time at which  $s$  performed its last LOOK, and let  $d$  be the distance between  $s(t')$  and  $r(t)$ . Furthermore, let  $k = \lceil d/\delta \rceil$ . We will prove our claim by well-founded induction on  $k$ , so let us assume our claim to hold for every  $k'$  such that  $0 \leq k' < k$ .

The first robot to perform a LOOK after time  $t$  sees the other robot set to  $A$ . If they coincide (i.e., if  $s$  has reached  $r$  or if  $k = 0$ ), the first robot turns  $C$ , and Lemma 3.3 applies. If they do not coincide, the first robot turns  $B$ . If it turns  $B$  before the other robot has performed a LOOK, then Lemma 3.4 applies. Otherwise, when the second robot performs its first LOOK after time  $t$ , it sees the first robot still set to  $A$ . Once again, if they coincide, the second robot turns  $C$  and Lemma 3.3 applies. At this point, if  $k = 1$  and  $s$  was directed toward  $r$  at time  $t$ , the robots have gathered and terminated correctly.

Hence, if  $r$  and  $s$  perform their first LOOK at times  $t_r$  and  $t_s$  respectively, we may assume that both will turn  $B$ ,  $r$  computes the midpoint  $m_r$  of  $r(t_r)$  and  $s(t_r)$ , and  $s$  computes the midpoint  $m_s$  of  $r(t_s)$  and  $s(t_s)$ . Observe that, if  $s$ 's destination was not  $r$  at time  $t$ , then  $m_r = m_s$ .

Without loss of generality, let  $r$  be the first robot to perform the second LOOK.  $r$  sees  $s$  set to  $B$ , hence it turns  $A$ . If  $s$  performs the second LOOK after  $r$  has already turned  $A$ , then  $s$  necessarily sees  $r$  in  $A$  (because  $r$  keeps seeing  $s$  in  $B$ ), and Lemma 3.4 applies.

Otherwise, both robots see each other in  $B$ , and both eventually turn  $A$ . Without loss of generality, let  $s$  be the first robot to perform the third LOOK. If  $k = 1$  and  $s$  was not directed toward  $r$  at time  $t$ , the robots have indeed gathered in  $m_r = m_s$ , so  $s$  turns  $C$  and Lemma 3.3 applies.

At this point we may assume that  $k \geq 2$ , hence  $\delta \leq (k-1)\delta < d \leq k\delta$ . We claim that the distance  $d'$  between  $r$  and  $s$  is now at most  $(k-1)\delta$ . Indeed, if  $s$  was not directed toward  $r$  at time  $t$ , then each robot has either reached  $m_r = m_s$ , or has approached it by at least  $\delta$ . In any case,  $d' \leq d - \delta \leq (k-1)\delta$ . Otherwise, if  $s$  was directed toward  $r$  at time  $t$ , then observe that both  $m_r$  and  $m_s$  lie between  $r(t)$  and  $m = (r(t) + s(t))/2$ . Moreover,  $s$  has performed its first LOOK while at distance at most  $d - \delta$  from  $r(t)$ , and subsequently it has further approached  $r(t)$ . On the other hand,  $r$  is found between  $r(t)$  and  $m$ , thus at distance not greater than  $d/2$  from  $r(t)$ . Hence,  $d' \leq \max\{d - \delta, d/2\} \leq (k-1)\delta$ .

Now,  $s$  is the first robot to perform the third LOOK, and sees  $r$  either already in  $A$  or still in  $B$ . In the first case, the inductive hypothesis applies, because  $r$  is not in a COMPUTE phase, and its destination is  $r$  itself. In the second case,  $s$  computes  $r$ 's location, and it keeps doing so until  $r$  turns  $A$ . When this happens, the inductive hypothesis applies again.  $\heartsuit$

**Corollary 3.6.** *If, at some time  $t$ , both robots are set to  $B$  and are both in a WAIT or in a LOOK phase, they will eventually gather and terminate correctly.*

*Proof.* The reasoning in the proof of Lemma 3.5 also implicitly addresses this case. Indeed, the configuration in which both robots are set to  $B$  and in a WAIT or a LOOK phase is reached during the analysis, and is incidentally resolved, as well.  $\heartsuit$

**Theorem 3.7.** *Algorithm 3 solves RENDEZVOUS in non-rigid ASYNCH and terminates correctly, regardless of the colors in the initial configuration.*

*Proof.* If both robots start in  $A$ , Lemma 3.5 applies. If they both start in  $B$ , Corollary 3.6 applies. If one robot starts in  $A$  and the other one starts in  $B$ , then Lemma 3.4 applies.

If exactly one robot starts in  $C$ , it will stay still forever, and the other robot will eventually reach it, turn  $C$  as well, and both will terminate.

If both robots start in  $C$  and they are coincident, they will terminate. If they are not coincident, let  $r$  be the first robot to perform a LOOK.  $r$  will then turn  $A$  and move toward the other robot  $s$ . If  $s$  performs its first LOOK when  $r$  has already turned  $A$ , it will wait,  $r$  will eventually reach it, turn  $C$ , and both will terminate. Otherwise,  $s$  performs its LOOK when  $r$  is still set to  $C$ , hence  $s$  will turn  $A$  as well, and move toward  $r$ .

Then, one robot will keep staying  $A$  and moving toward the other one, until both have turned  $A$ . Without loss of generality, let  $r$  be the first robot to see the other one set to  $A$ . If they are coincident,  $r$  turns  $C$  and Lemma 3.3 applies. Otherwise,  $r$  turns  $B$ . If this happens before  $s$  has seen  $r$  in  $A$ , then Lemma 3.4 applies. Otherwise, both robots will turn  $B$ . As long as only one robot has turned  $B$ , it stays  $B$  and does not move. At some point, one robot sees the other in  $B$  and Corollary 3.6 applies.  $\heartsuit$

## 4 Impossibility of rendezvous with two colors

Observe that Algorithms 1, 2 and 3 only produce moves of three types: stay still, move to the midpoint, and move to the other robot. It turns out that, regardless of the number of available colors, any algorithm for RENDEZVOUS must use those three moves under some circumstances.

**Proposition 4.1.** *For any algorithm solving RENDEZVOUS in rigid FSYNCH, there exist a color  $X$  and a distance  $d > 0$  such that any robot set to  $X$  that sees the other robot at distance  $d$  and set to  $X$  moves to the midpoint.*

*Proof.* Assume both robots start with the same color and in distinct positions. We may assume that both robots get isometric snapshots at each cycle, so they both turn the same colors, and compute destination points that are symmetric with respect to their midpoint. If they never compute the midpoint and their execution is rigid and fully synchronous, they never gather.  $\heartsuit$

**Proposition 4.2.** *For any algorithm solving RENDEZVOUS in rigid SSYNCH, there exist two colors  $X$  and  $Y$  and a distance  $d > 0$  such that any robot set to  $X$  that sees the other robot at distance  $d$  and set to  $Y$  moves to the other robot's position.*

*Proof.* We activate one robot on even cycles, and the other robot on odd cycles. If no robot ever computes the other robot's position and they perform rigid movements, they never gather.  $\heartsuit$

**Proposition 4.3.** *For any algorithm solving RENDEZVOUS in rigid SSYNCH, there exist two colors  $X$  and  $Y$  and a distance  $d > 0$  such that any robot set to  $X$  that sees the other robot at distance  $d$  and set to  $Y$  does not move.*

*Proof.* We keep activating only one robot at each cycle (alternately), except when one robot computes the other robot's position. Whenever this happens, we activate both robots for that cycle. If no robot ever performs a null move, they never gather.  $\heartsuit$

The above observations partly justify the choice to restrict our attention to a specific class of algorithms: from now on, every algorithm we consider computes only destinations of the form

$$(1 - \lambda) \cdot \textit{me.position} + \lambda \cdot \textit{other.position},$$

where the parameter  $\lambda \in \mathbb{R}$  depends only on *me.light* and *other.light*. Similarly, a robot's next light color depends only on the current colors of the two robots' lights, and not on their distance. Recall from Section 1 that this class of algorithms is denoted by  $\mathcal{L}$ . Notice that Algorithms 1 and 2 both belong to  $\mathcal{L}$ , but Algorithm 3 does not, because it may output a different color depending if the two robots coincide or not.

A statement of the form  $X(Y) = (Z, \lambda)$  is shorthand for “if a robot is set to  $X$  and sees the other robot set to  $Y$ , it turns  $Z$  and makes a move with parameter  $\lambda$ ”, where  $\{X, Y, Z\} \subseteq \{A, B\}$  and  $\lambda \in \mathbb{R}$ . The negation of  $X(Y) = (Z, \lambda)$  will be written as  $X(Y) \neq (Z, \lambda)$ , whereas a transition with an unspecified move parameter will be denoted by  $X(Y) = (Z, \star)$ .

#### 4.1 Preliminary results

Here we assume that the model is rigid ASYNCH, that only two colors are available, namely  $A$  and  $B$ , and that the initial configuration is with both robots set to  $A$ . All our impossibility results for this very special model are then applicable to both non-rigid ASYNCH with preset initial configuration and rigid ASYNCH with arbitrary initial configuration.

So, let an algorithm that solves RENDEZVOUS in this model be given. If the algorithm belongs to class  $\mathcal{L}$ , then the following statements hold.

**Lemma 4.4.**  $A(A) = (B, \star)$ .

*Proof.* If the execution starts with both robots in  $A$ , and  $A(A) = (A, \star)$ , then no robot ever transitions to  $B$ , and RENDEZVOUS is not solvable, due to Proposition 1.1.  $\heartsuit$

**Lemma 4.5.** If  $A(A) = (B, 1/2)$ , then  $B(A) = (B, \star)$ .

*Proof.* Let us assume by contradiction that  $B(A) = (A, \star)$ . If  $B(A) = (A, \lambda)$  with  $\lambda \neq 1$ , we let the two robots execute two cycles each, alternately. As a result, each robot keeps seeing the other robot in  $A$ , and their distance is multiplied by  $|1 - \lambda|/2 \neq 0$  at every turn. Hence the robots never gather.

If  $B(A) = (A, 1)$ , we let robot  $r$  perform a whole cycle and the LOOK and COMPUTE phases of the next cycle, while the other robot  $s$  waits. At

this point, their distance has halved,  $r$  is set to  $A$ , and is about to move to  $s$ 's position. Now  $s$  performs two whole cycles, reaching  $r$ 's position with its light set to  $A$ . Finally, we let  $r$  finish its cycle. As a result, the distance between the two robots has halved, both robots have performed at least a cycle, they are in a WAIT phase, and they are both set to  $A$ . Hence, by repeating the same pattern of moves, they never gather.  $\heartsuit$

**Lemma 4.6.** *If  $A(A) = (B, 1/2)$  and  $B(B) = (A, \star)$ , then  $B(B) = (A, 0)$ .*

*Proof.* Assume by contradiction that  $A(A) = (B, 1/2)$  and  $B(B) = (A, \lambda)$  with  $\lambda \neq 0$ . We let both robots perform a LOOK and a COMPUTE phase simultaneously. Both turn  $B$  and compute the midpoint  $m$ . Then we let robot  $r$  finish the current cycle and perform a new LOOK. As a result,  $r$  will turn  $A$  and will move away from  $m$ . Now let the other robot  $s$  finish its first cycle and perform a whole new cycle.  $s$  reaches  $m$ , sees  $r$  still set to  $B$  and still in  $m$ , hence  $s$  turns  $A$  and stays in  $m$ . Finally, we let  $r$  finish the current cycle. At this point, both robots are set to  $A$ , they are in a WAIT phase, both have performed at least one cycle, and their distance has been multiplied by  $|\lambda|/2 \neq 0$ . Therefore, by repeating the same pattern of moves, they never gather.  $\heartsuit$

**Lemma 4.7.** *If  $A(A) = (B, 1/2)$  and  $B(B) = (A, 0)$ , then  $B(A) = (B, 0)$ .*

*Proof.* By Lemma 4.5,  $B(A) = (B, \star)$ . Assume by contradiction that  $B(A) = (B, \lambda)$  with  $\lambda \neq 0$ . We let both robots perform a LOOK simultaneously, so both plan to turn  $B$  and move to the midpoint  $m$ . We let robot  $r$  finish the cycle, while the other robot  $s$  waits. Then we let  $r$  perform a whole other cycle. So  $r$  sees  $s$  still in  $A$ , and moves away from  $m$ , while staying  $B$ . Now we let  $s$  finish its first cycle and move to  $m$ . Finally, we let both robots perform a new cycle simultaneously. As a result, both robots are set to  $A$  and are in a WAIT phase, both have performed at least one cycle, and their distance has been multiplied by  $|\lambda|/2 \neq 0$ . By repeating the same pattern of moves, they never gather.  $\heartsuit$

**Lemma 4.8.** *If  $A(A) = (B, 1/2)$  and  $B(B) = (A, 0)$ , then  $A(B) = (A, 1)$ .*

*Proof.* Let us first assume that  $A(B) = (B, \lambda)$  with  $\lambda \neq 1$ . We let one robot perform a whole cycle, thus turning  $B$  and moving to the midpoint. Then we let the other robot perform a cycle, at the end of which both robots are set to  $B$ . Finally, we let both robots perform a cycle simultaneously, after which they are back to  $A$  and in a WAIT phase. Because their distance has been multiplied by  $|1 - \lambda|/2 \neq 0$ , by repeating the same pattern of moves they never gather.

Assume now that  $A(B) = (B, 1)$ . We let robot  $r$  perform a LOOK and a COMPUTE phase, thus turning  $B$  and computing the midpoint. Now we let the other robot  $s$  perform a whole cycle, at the end of which it is set to

$B$  and has reached  $r$ . Then we let  $r$  finish its cycle, moving away from  $s$ . Finally, we let both robots perform a new cycle simultaneously, which takes them back to  $A$ . Their distance has now halved, and by repeating the same pattern of moves they never gather.

Assume that  $A(B) = (A, \star)$ , and let robot  $r$  perform an entire cycle, thus turning  $B$  and moving to the midpoint. Due to Lemma 4.7,  $B(A) = (B, 0)$ , which means that, from now on, both robots will retain colors. Hence,  $r$  will always stay still, and  $s$  will never reach  $r$  unless  $A(B) = (A, 1)$ .  $\heartsuit$

## 4.2 Rigid asynchronous model with arbitrary initial configuration

**Lemma 4.9.** *Algorithm 1 does not solve RENDEZVOUS in rigid ASYNCH, if both robots are set to  $B$  in the initial configuration.*

*Proof.* Let both robots perform a LOOK phase, so that both will turn  $A$ . We let robot  $r$  finish the current cycle and perform a new LOOK, while the other robot  $s$  waits. Hence,  $r$  will stay  $A$  and move to  $s$ 's position. Now we let  $s$  finish the current cycle and perform a new LOOK. So  $s$  will turn  $B$  and move to the midpoint  $m$ . We let  $r$  finish the current cycle, thus reaching  $s$ , and perform a whole new cycle, thus turning  $B$ . Finally, we let  $s$  finish the current cycle, thus turning  $B$  and moving to  $m$ . As a result, both robots are again set to  $B$ , they are in a WAIT phase, both have executed at least one cycle, and their distance has halved. Thus, by repeating the same pattern of moves, they never gather.  $\heartsuit$

**Theorem 4.10.** *There is no algorithm of class  $\mathcal{L}$  that solves RENDEZVOUS using two colors in rigid ASYNCH from all possible initial configurations.*

*Proof.* Because robots may start both in  $A$  or both in  $B$ , the statement of Lemma 4.4, holds also with  $A$  and  $B$  exchanged. Hence  $A(A) = (B, \star)$ , but also  $B(B) = (A, \star)$ . Moreover, by Proposition 4.1, either  $A(A) = (B, 1/2)$  or  $B(B) = (A, 1/2)$ . By symmetry, we may assume without loss of generality that  $A(A) = (B, 1/2)$ . Now, by Lemma 4.6,  $B(B) = (A, 0)$ . Additionally, by Lemma 4.7 and Lemma 4.8,  $B(A) = (B, 0)$  and  $A(B) = (A, 1)$ . These rules define exactly Algorithm 1, which is not a solution, due to Lemma 4.9.  $\heartsuit$

## 4.3 Non-rigid asynchronous model with preset initial configuration

**Theorem 4.11.** *There is no algorithm of class  $\mathcal{L}$  that solves RENDEZVOUS using two colors in non-rigid ASYNCH, even assuming that both robots are set to a predetermined color in the initial configuration.*

*Proof.* Let both robots be set to  $A$  in the initial configuration, and let  $d \geq 0$  be given. By Lemma 4.4,  $A(A) = (B, \lambda)$ , for some  $\lambda \in \mathbb{R}$ . If  $\lambda \neq 1/2$ , we



place the two robots at distance  $d/|1 - 2\lambda|$  from each other, and we let them perform a whole cycle simultaneously. If  $\lambda = 1/2$ , we place the robots at distance  $d + 2\delta$ , and we let them perform a cycle simultaneously, but we stop them as soon as they have moved by  $\delta$ . As a result, both robots are now set to  $B$ , and at distance  $d$  from each other. This means that any algorithm solving RENDEZVOUS with both robots set to  $A$  must also solve it with both robots set to  $B$ , as well.

Similarly, we can place the two robots at distance  $d/|1 - \lambda|$  or  $d + \delta$ , depending if  $\lambda \neq 1$  or  $\lambda = 1$ . Then we let only one robot perform a full cycle, and we let it finish or we stop it after  $\delta$ , in such a way that it ends up at distance exactly  $d$  from the other robot. At this point, one robot is set to  $A$  and the other is set to  $B$ .

It follows that any algorithm for RENDEZVOUS must effectively solve it from all possible initial configurations. But this is impossible, due to Theorem 4.10. ♡

## 5 Conclusions

We considered deterministic distributed algorithms for RENDEZVOUS for mobile robots that cannot use distance information, but can only reduce (or increase) their distance by a constant factor, depending on the color of the lights that both robots are carrying. We called this class of algorithms  $\mathcal{L}$ .

We gave several upper and lower bounds on the number of different colors that are necessary to solve RENDEZVOUS in different robot models. Based on these results, we can now give a complete characterization of the number of necessary colors in every possible model, ranging from fully synchronous to semi-synchronous to asynchronous, rigid and non-rigid, with preset or arbitrary initial configuration.

**Theorem 5.1.** *To solve RENDEZVOUS with an algorithm of class  $\mathcal{L}$  from a preset starting configuration,*

- *one color is sufficient for rigid and non-rigid FSYNCH;*
- *two colors are necessary and sufficient for rigid SSYNCH, non-rigid SSYNCH, and rigid ASYNCH;*
- *three colors are necessary and sufficient for non-rigid ASYNCH.*

*To solve RENDEZVOUS with an algorithm of class  $\mathcal{L}$  from an arbitrary starting configuration,*

- *one color is sufficient for rigid and non-rigid FSYNCH;*
- *two colors are necessary and sufficient for rigid and non-rigid SSYNCH;*
- *three colors are necessary and sufficient for rigid and non-rigid ASYNCH.*

*Proof.* All the optimal color values derive either from previous theorems or from the model inclusions summarized in Figure 1.

That just one color is (necessary and) sufficient for all FSYNCH models follows from Proposition 1.1.

Proposition 1.1 also implies that, for all the other models, at least two colors are necessary. Therefore, by Theorem 2.2, two colors are necessary and sufficient for all SSYNCH models.

Similarly, Theorem 2.4 states that two colors are necessary and sufficient for rigid ASYNCH with preset initial configuration. On the other hand, by Theorem 4.10 and Theorem 4.11, three colors are necessary in the three remaining models, and by Theorem 2.7 three colors are also sufficient. ♡

In the three models in which three colors are necessary and sufficient, it remains an open problem to determine whether using distance information to its full extent would make it possible to use only two colors.

An interesting variation on this model is when the light on a robot can be seen only by the other robot(s). In this case, algorithms of class  $\mathcal{L}$  are inadequate to solve RENDEZVOUS even in rigid ASYNCH with preset initial configuration, regardless of the number of available colors. In contrast, three colors are necessary and sufficient for all SSYNCH models.

On the other hand, if the light is visible only to the robot that is carrying it (i.e., internal memory), then no algorithm of class  $\mathcal{L}$  can solve RENDEZVOUS, even in rigid SSYNCH with preset initial configuration, regardless of the number of colors.

## References

- [1] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing for mobile robots: gathering. *SIAM Journal on Computing*, to appear.
- [2] 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*, pp. 506–515, 2012.
- [3] P. Flocchini, G. Prencipe, and N. Santoro. *Distributed computing by oblivious mobile robots*. Morgan & Claypool, 2012.
- [4] G. Prencipe and N. Santoro. Distributed algorithms for mobile robots. In *Proceedings of the 5th IFIP International Conference on Theoretical Computer Science*, pp. 47–62, 2006.
- [5] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: formation of geometric patterns. *SIAM Journal on Computing*, vol. 28, pp. 1347–1363, 1999.