Impressionist Algorithms for Autonomous Multi-Robot Systems: Flocking as a Case Study

Florian Berlinger[1], Julia T. Ebert[2], and Radhika Nagpal[1]

Supplementary Material*

This document contains additional information and supporting data complementing the main paper.

Contents

1 Algorithms for alignment, dispersion, and milling **2**

2 Methods and metrics **4**

2.1 Alignment: Weights for inferred angles φ 4

2.2 Milling: Circle evaluation 5

2.3 Milling: Control experiments with physically-validated robot dynamics 5

3 Milling results **6**

3.1 Theory of circle formation 6

3.1.1 Sequential circle formation (special case) 7

3.1.2 Parallel circle formation (general case) 12

3.2 Viewing half-angle α and circle radius R 14

3.3 Number of robots N and circle radius R 16

3.4 Sufficient cognition speed f_c 16

3.5 Minimalist perception 18

*This work was supported by the Office of Naval Research (ONR Award No.: N00014-20-1-2320), the Wyss Institute for Biologically Inspired Engineering, and an Amazon AWS Research Award. The authors are with the Department of Mechanical and Aerospace Engineering at Princeton University, Princeton, New Jersey[1], and the John A. Paulson School of Engineering and Applied Science at Harvard University, Cambridge, Massachusetts[2]. E-mail: `fberlinger@gmail.com`.

1 Algorithms for alignment, dispersion, and milling

```
Algorithm 1: Alignment
1 INPUT:
2 Color scheme (e.g., left/right, cf. Figure 3 in the main paper)
3 MAIN:
4 while alignment in progress do
5     Take 2 binary images (left and right).
6     Obtain a list of blob centroids corresponding to fellow robots.
7     For each fellow robot, calculate the angle of detection and judge which color is salient. Set its
        heading direction accordingly.
8     Calculate the average heading of all fellow robots.
9     Use the pectoral fins to turn towards the average heading.
```

```
Algorithm 2: Dispersion
1 INPUT:
2 $d_t$ # target distance (set to values between 1 BL and 10 BL, cf. Figure 6 in the main paper)
3 MAIN:
4 while dispersion in progress do
5     Take 2 binary images (left and right).
6     Obtain a list of blob centroids corresponding to fellow robots.
7     For each fellow robot, judge whether it is too close, ok, or too far based on the size of its blob.
8     Set the distance for close robots to $0.8 \cdot d_t$. Set the distance for ok robots to $d_t$. Set the distance for
        far robots to $1.2 \cdot d_t$.
9     Calculate the next move vector p based on the following equation (cf. Lennard-Jones potential):
```

$$\begin{aligned}\mathbf{p}_i &= \frac{1}{N}\sum_{j=1}^{N} F_{ij} r_{ij} = \frac{1}{N}\sum_{j=1}^{N} \frac{\partial V_{ij}}{\partial |r_{ij}|} r_{ij} \\ &= \frac{1}{N}\sum_{j=1}^{N} -\frac{1}{|r_{ij}|}\left[a\left(\frac{d_t}{|r_{ij}|}\right)^{a} - 2b\left(\frac{d_t}{|r_{ij}|}\right)^{b}\right] r_{ij} \qquad \forall j \neq i\end{aligned} \tag{1}$$

```
        .
10    Use the fins to move along p. Set fin power based on |p| and momentary error, using proportional
        control.
```

Algorithm 3: Milling

```
INPUT:
$\alpha$ # defining angle of the prismatic field of view (FOV) (set to values between $0^\circ$ and $16^\circ$, cf. Figure 8 in the main paper)
$b_{thresh} = 30$ # brightness threshold for neighbor presence (empirically tuned)
MAIN:
Pre-compute masks that isolate the prismatic FOV in the left and right camera images.
Run the caudal fin at 3 Hz.
while milling in progress do
    Take 2 binary images (left and right).
    Multiply the images with the respective masks. All pixels outside of the FOV will be ignored.
    if total brightness in either image > $b_{thresh}$ then
        Neighbor present: turn counterclockwise by running the right pectoral fin at 6 Hz.
    else
        No neighbor present: turn clockwise by running the left pectoral fin at 6 Hz.
```

2 Methods and metrics

2.1 Alignment: Weights for inferred angles φ

To improve alignment, we weighted the inferred angles φ proportionally to their probability $\Pr(\phi)$ of being correct; i.e., pointing leftwards if the actual robot heading is leftwards and rightwards otherwise. An example of an incorrectly inferred angle φ is shown in Figure 1: The centered robot detects red as the prevalent color on the colored robot and assigns a leftwards φ in spite of the actual robot heading pointing rightwards.

Considering a left/right colored robot in the first or fourth quadrant for now, the weighting follows naturally for the left red side: the assigned heading φ linearly decreases from $-\pi/2$ at $\Pr(\phi = 0) = 1$ to 0 at $\Pr(\phi = \pi/2) = 0.5$ and increases to $\pi/2$ at $\Pr(\phi = \pi) = 1$, pointing less strongly to either cardinal direction as uncertainty increases (Figure 1). For the blue right side, however, φ would naturally reach a maximum value of $\pm\pi$ at $\Pr(\phi = \pi/2) = 0.5$. To avoid having the largest weight where uncertainty is highest, φ is instead assigned as follows: $\varphi \leftarrow -\phi + \pi/2$. The weightings for quadrants II and III follow by symmetry. Nonlinear weightings did not improve convergence in simulations that considered the dynamics and perception of Bluebots.

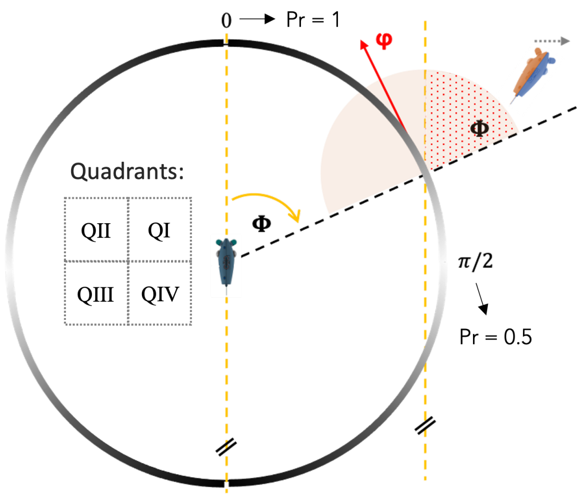

Figure 1: The probability of assigning a heading φ that matches a robot's cardinal direction (left/right) depends on ϕ and is color-coded on the unit circle: darker segments correspond to higher probabilities for left/right (and lower probabilities for anterior/posterior) colored robots. For instance, in the first quadrant when seeing more red on a left/right colored robot, the probability is $1 - |\phi|/\pi$; robots in the dotted area get assigned a heading φ that does not match their cardinal direction. The average probability of a matching heading for bi-colored robots is 0.75.

2.2 Milling: Circle evaluation

For milling simulations, we fitted circles through the participating robots at every half-second, using linear least squares. This allowed us to evaluate the quality of circle formation and declare success.

Here we show 12 exemplary instances during circle formation with $N = 15$ robots and viewing half-angle $\alpha = 0^\circ$ (Figure 2); the corresponding animation is in the supplementary video.

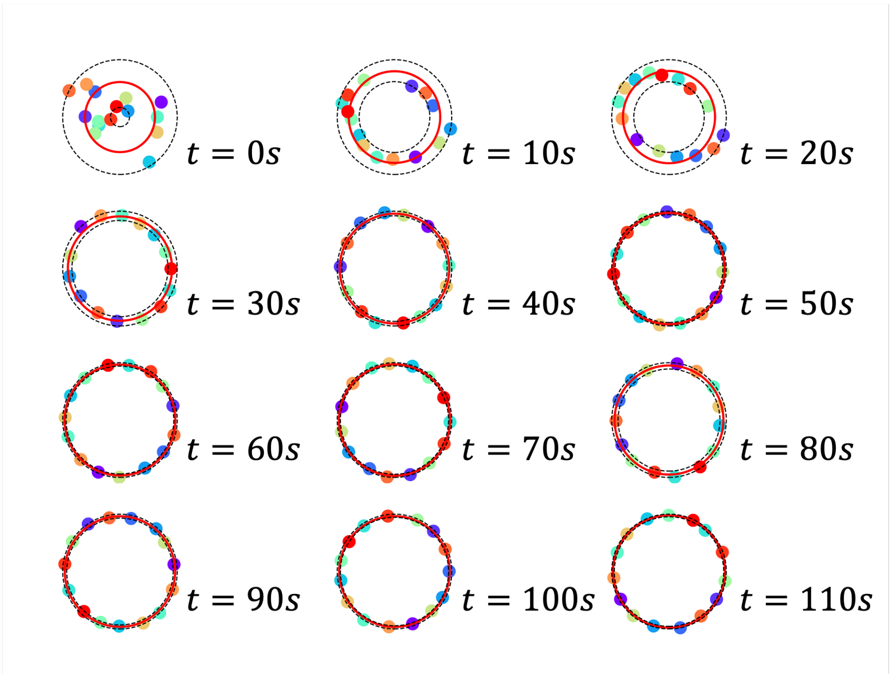

Figure 2: $N = 15$ robots with plane-of-sight sensors ($\alpha = 0^\circ$) during 12 instances of circle formation. Red solid line: fitted circle; black dashed lines: min. and max. circles.

2.3 Milling: Control experiments with physically-validated robot dynamics

We validated the simplified dynamics used for circle formation against fully simulated robot dynamics, for which we solved the robot's equations of motion using Euler integration (see our own previous work). In 10 control experiments each, $N = 10$ and $N = 20$ simulated robots with full dynamics successfully formed and maintained circles of average radii $R = 228\,\mathrm{mm}$ and $R = 1622\,\mathrm{mm}$, respectively. The robots had a viewing half-angle $\alpha = 0$ and a cognition speed $f_c = 2\,\mathrm{Hz}$. Due to robot inertia, the actuation strengths of the caudal and pectoral fins had to be tuned to achieve $r_{swim} \approx R$. Such tuning was not necessary and circle formation was less sensitive to r_{swim} in simulations with simplified dynamics because instantaneous changes between clockwise and counterclockwise turning were possible.

3 Milling results

3.1 Theory of circle formation

We proved that a circle formation can be maintained indefinitely in our previous work. Here we ran several thousand successful simulations that led to such formation from random initializations. Each robot had a line-of-sight (or a field-of-view) binary sensor that can detect the presence of other robots in the front. The sensor returned 0 if it did not detect other robots and 1 if it did.

The circle formation rule

We now provide evidence why circles are formed reliably, when each robot applies the following circle formation rule:

- If the sensor returns 0, rotate clockwise along a circle of radius r_0 with a fixed speed u.
- If the sensor returns 1, rotate counterclockwise along a circle of radius r_1 with a fixed speed u.

Simplifications

In the following analysis, we make the same simplifications as in our previous work:

- The sensor-actuator loop is fast enough that it can be considered continuous.
- There is no noise on the robots' sensor readings and locomotion, and the robots' sensing range is unlimited.
- The robots are capable of moving in a planar circle of a fixed radius at a fixed speed in either direction (i.e., clockwise or counterclockwise).
- The robots have no inertia and can change directions (i.e., from rotating clockwise to counterclockwise or vice versa) instantaneously. This simplification does not hold strongly for the Bluebots, but it makes the analysis tractable, and simulation and physical results show that milling with the Bluebots is indeed possible.
- The robots are on the same horizontal plane. This simplification leads to no loss of generality as long as the robots' sensors are implemented in such a way that they are invariant to other robots' altitudes/depths.

Observations

Our line of argument follows three observations:

1. The circle formation rule leads to a following behavior.
2. Given the same circle formation rule, several robots follow behind each other, i.e., in line.
3. In the absence of any leading robot, every robot follows a single other robot right in front of itself. A circle forms.

3.1.1 Sequential circle formation (special case)

To gain intuition, let's first look at a special case of circle formation. Here we initially have a leading robot, and following robots are added one at a time to a growing formation. All robots have line-of-sight sensors and body radius ρ.

Proposition 1. *Given is a robot (the leader) moving along a straight-line trajectory with a fixed linear velocity u_l. A second robot (the follower) is initialized at a random position in a random orientation and applies the circle formation rule. W.l.o.g., we introduce a Cartesian coordinate system such that the leader moves along the positive x-axis. Let's further denote the leader's and follower's position at time t as $[x_l(t),\, 0]$ and $[x_f(t),\, y_f(t)]$.*

Theorem 1.1. *The follower detects the leader.*

Proof. As long as the follower's sensor returns 0, the follower rotates clockwise on a circle of radius r_0. It is clear that as both robots move along their respective trajectories, the follower's sensor must at some point intersect the leader (Figure 3a). We can quantify how long it takes for such intersection to occur. Here we derive an equation for the time of detection given the radius r_c and distance y_c of the follower's circle to the leader's trajectory, the leader's initial position $x_l(t=0)$ and linear velocity u_l, as well as the follower's initial position $\theta_f(t=0)$ and angular velocity w_f (Figure 3b). At time t, which we are looking for, the follower's line of sight intersects the x-axis at the position of the leader:

$$\underbrace{\Big(x_l(0)+u_l\cdot t\Big)}_{\text{leader position}} - \underbrace{\left(\frac{y_c}{\tan[\theta_f(0)+w_f\cdot t]}+\frac{r_c}{\sin[\theta_f(0)+w_f\cdot t]}\right)}_{x\text{-axis intersection of follower's line of sight}} = 0 \qquad (2)$$

A numerical solution computes the distances between the leader and the intersection of the follower's line of sight with the x-axis for incremental times t. Instances with no intersection (the follower's line of sight

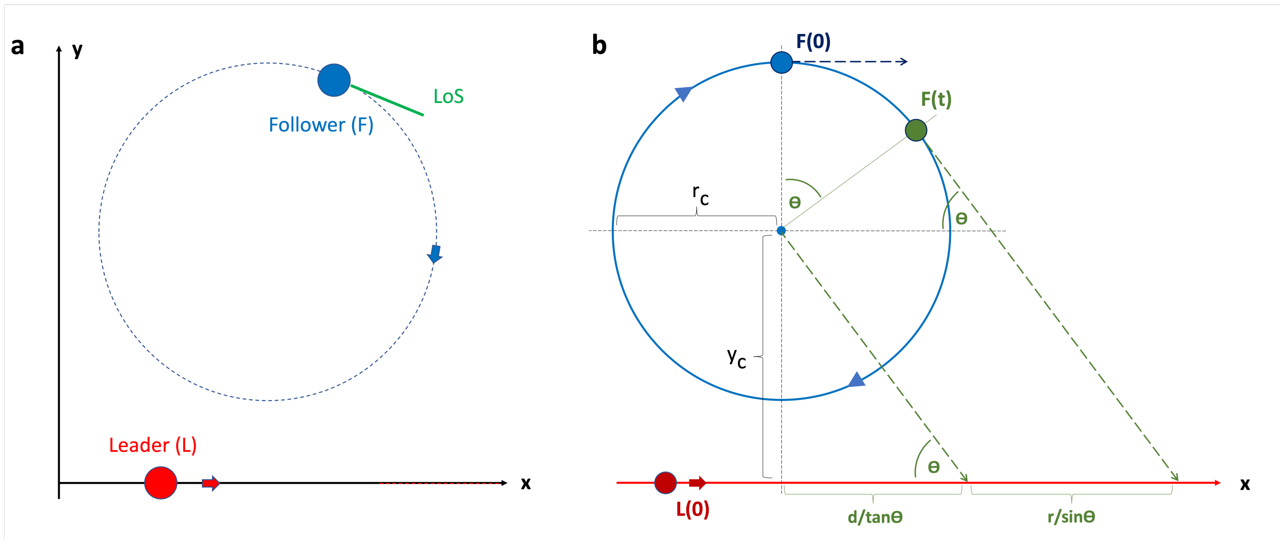

Figure 3: The follower (F) detects the leader (L). (a) The line of sight (LoS) of the follower intersects with every point on the x-axis successively as it searches for the leader, resulting in discovery. (b) The time of discovery can be derived using simple geometry (see Eq. 2). W.l.o.g., we depict here the simple case with $\theta_f(0) = 0$.

has a positive y-component) are assiged a distance value of infinity. The time of detection is the t corresponding to the minimum distance. The only restriction to guarantee successful detection given parameters $r_c, y_c, x_l(0), u_l, \theta_f(0), w_f$ is that w_f can not be zero.

□

Theorem 1.2. *The follower moves toward the leader.*

Proof. Upon detection, the follower's sensor returns 1 and the follower rotates counterclockwise on a circle of radius r_1 until it no longer detects the leader. The follower now alternates its direction of rotation every time its sensor reading changes. As outlined in the simplifications, the sensor-actuator loop is considered continuous and noise-free, and directional changes are instantaneous. On average, the follower moves in the direction of the leader. □

Theorem 1.3. *The follower positions itself behind the leader.*

Proof. As stated previously, the leader moves along the positive x-axis (Proposition 1) and the follower in the direction of the leader (Theorem 1.2). As long as $x_f(t) > x_l(t)$, leader and follower move in opposite x-directions and $(x_f(t) - x_l(t)) \rightarrow 0$. If at initialization $x_f(0) \leq x_l(0)$, the follower is already considered behind the leader. □

Theorem 1.4. *The follower settles behind the leader, following on the same trajectory.*

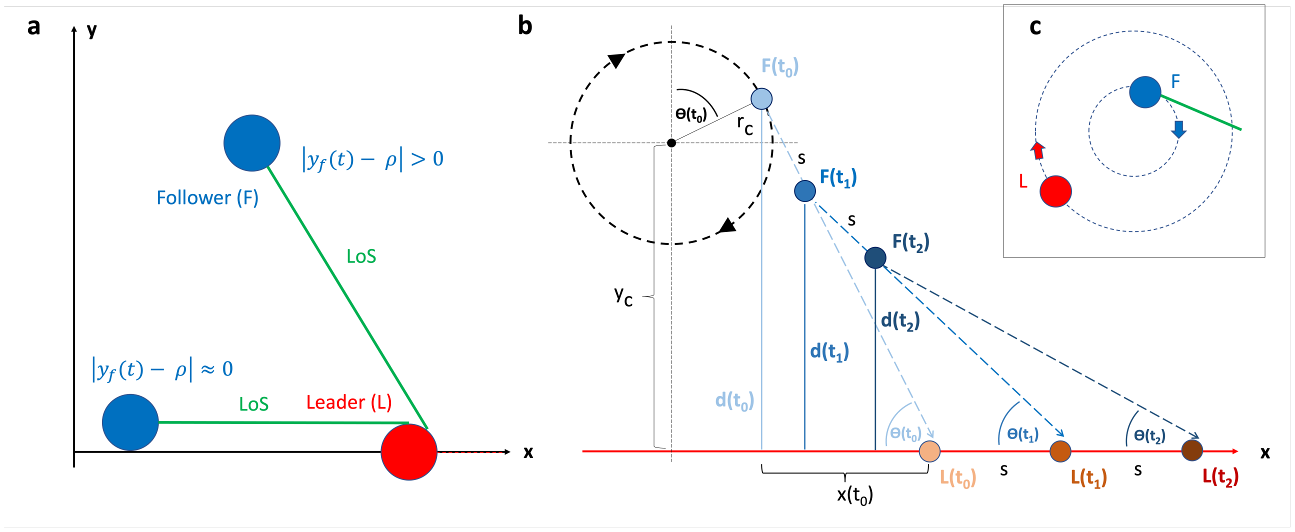

Figure 4: The follower (F) settles behind the leader (L) following on the same trajectory. (a) In moving toward the leader, the follower continuously corrects deviations from the leader's course. Shown are two different follower scenarios. (b) The follower's distance $d(t)$ and angle $\theta(t)$ can be derived using simple geometry (see Eq. 4). s is the distance the follower and leader cover during one time step. (c) The follower detects the leader because it advances quicker on its circle.

Proof. We show that the following configuration exists and is stable:

$$\exists\ t \quad \text{s.t.} \begin{cases} x_f(t) < x_l(t), & \text{the follower is behind the leader (by Theorem 1.3)} \\ y_f(t) \approx \rho, & \text{the follower closely follows the straight-line trajectory of the leader} \\ x_l(t+1) > x_l(t), & \text{the leader advances (by Proposition 1)} \\ x_f(t+1) > x_f(t), & \text{the follower advances} \end{cases} \tag{3}$$

As long as $|y_f(t) - \rho| > 0$, the follower moves toward the leader (Theorem 1.2) and $y_f(t) \to \rho$ (Figure 4a). As a result, the follower follows the trajectory of the leader. Given equal nominal speeds, the follower never overtakes the leader because the alternating rotations limit its speed to at most the speed of the leader. Therefore, the leader-follower configuration is stable.

Letting the leader and follower move in discrete time steps and assuming equal and constant distances s covered per time step, we can derive the orthogonal distance $d(t)$ between the follower and the leader's trajectory and the angle $\theta(t)$ between the follower's and leader's heading, as well as the settling time, i.e., the duration until $d(t) < \rho + \varepsilon_1$ or $\theta(t) < \varepsilon_2$ (Figure 4b). We use the following recurrence relationships (visualized in Figure 5):

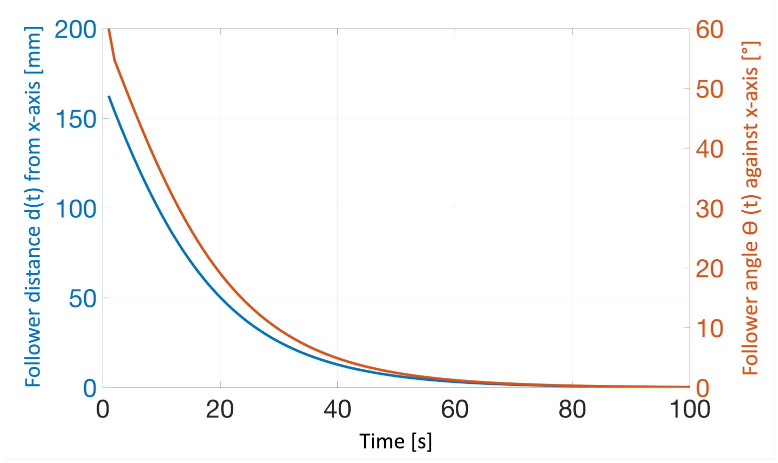

Figure 5: The follower's orthogonal distance $d(t)$ to and angle $\theta(t)$ against the x-axis (trajectory of the leader) decay exponentially over time. This example was visualized following the recurrence relationship in Eq. 4 with $y_c = 150\,\mathrm{mm}$, $r_c = 25\,\mathrm{mm}$, $\theta = 60^\circ$, and $s = 10\,\mathrm{mm}$.

$$
\begin{aligned}
d(t_0) &= y_c + r \cdot \cos\theta, && \text{initial } d \text{ at detection, see Fig. 3}b \\
\theta(t_0) &= \theta, && \text{initial } \theta \text{ at detection, see Fig. 3}b \\
x(t_0) &= \frac{d(t_0)}{\tan\theta(t_0)}, && \text{initial } x\text{-distance between leader and follower} \\
d(t_i) &= d(t_{i-1}) - s \cdot \sin\theta(t_{i-1}), \\
\theta(t_i) &= \arctan\left(\frac{d(t_i)}{x(t_0) + s\left(i - \sum_{n=0}^{i-1}\cos\theta(t_n)\right)}\right)
\end{aligned}
\tag{4}
$$

□

Corollary 1.1. *The straight-line trajectory is a special case of a circular trajectory with $r = \infty$. Theorems 1.1-1.4 hold for all clockwise or counterclockwise circular leader trajectories with $r > r_0$ or $r > r_1$, respectively, where r_0 and r_1 are the clock- and counterlockwise swimming radii followed according to the circle formation rule. Imagine, for instance, the leader and follower moving clockwise on concentric circles. Moving at the same fixed speed u, the follower will complete its circle more quickly and eventually detect the leader (Figure 4c).*

Proposition 2. *Given are a leading robot and n robots following in line (the group), whereby each robot except the leader follows the robot in front of itself. An additional robot (the follower) is added and applies the circle formation rule.*

Theorem 2.1. *The follower follows the group in between other robots, i.e., in line.*

Proof. The follower detects any robot in the group (the *detected* robot) and moves toward it (Theorems 1.1-1.2). From here, two cases are possible:

1. $x_f < x_{detected}$: At the time of detection, the distance between the follower and the detected robot is $d_f = \sqrt{(x_{detected} - x_f)^2 + (y_{detected} - y_f)^2}$ (Figure 6a). The follower has to cover *at least* this distance d_f as it approaches the trajectory of the detected robot laterally ($y_f \to y_{detected} + \rho$) and gets behind it. In the meantime, other potential following robots will have moved as well and may now be in front of the follower. Because robots move at the same speed, the follower detects *at least* all other robots (*other*) which have to cover less ground, i.e., all robots for which $(x_{detected} - x_{other}) < d_f$ at the time of detection. The follower enters the group behind the last other robot it detects. If a robot has been moving behind that last other robot previously, it now moves behind the follower instead. The in-line formation is preserved.

2. $x_f \geq x_{detected}$: The follower first has to position itself behind the detected robot (as described in Theorem 1.3). If it does so without loosing the detected robot, we are in case 1. Because the follower is subject to restricted turning radii and rates, it is possible that it cannot rotate quickly enough to align with the oncoming detected robot. In this case, it looses the detected robot and performs a clockwise rotation. We have two sub-cases:

   (a) If it does not cross the line of sight of the robot behind the detected robot during the clockwise rotation, it now detects this robot and we are again in case 2. However, if there were no other robots behind the detected robot, the follower would keep turning and detect a robot in front of itself and we are in case 1.

   (b) If it does cross the line of sight of the robot behind the detected robot (the *crossed* robot), we have two additional sub-sub-cases (Figure 6b):

      i. $y_f \geq y_{detected}$ upon detection: The crossed robot moves further counterclockwise as the follower crosses before re-detecting the robot in front of itself. The follower detects either the last robot in line and gets behind that robot (Theorems 1.1-1.4), or the leader and we are in case 1.

      ii. $y_f < y_{detected}$ upon detection: If the crossed robot cannot turn quickly enough to follow the crossing follower, it keeps following the robot in front of itself. The follower naturally detects the leader and we are in case 1. If the crossed robot can turn quickly enough, it follows the follower together with all robots behind itself. The follower naturally detects the leader. Because $x_f < x_{crossed+}$ at this point (crossed+ denotes the robot in front of the crossed robot), the follower takes these robots back to swimming in line behind the the detected robot (similar to case 1).

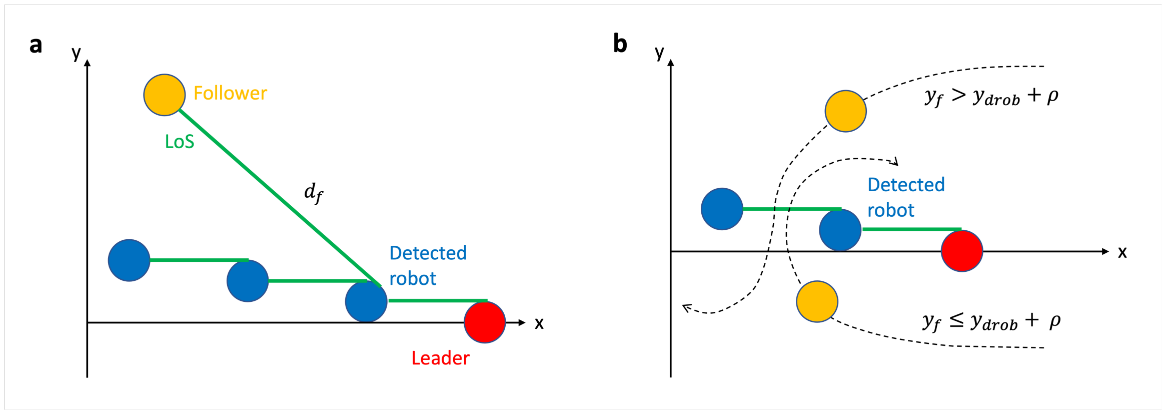

Figure 6: A follower joins a group. (a) $x_f < x_{detected}$: The follower detects the detected robot at distance d_f. (b) $x_f \geq x_{detected}$: The follower crosses the line of sight of the next robot behind the detected robot. Shown are two different follower scenarios.

□

Proposition 3. *Given are a leading robot (the leader) and n robots following in line (the followers). The leading robot changes its behavior to the circle formation rule.*

Theorem 3.1. *A circle formation forms.*

Proof. The leader rotates clockwise until it detects any follower. Naturally, this will be the last robot in line (the *last* robot). Note that the followers follow the leader on its clockwise trajectory (Theorem 1.4). If the leader can follow the last robot without loosing it, a closed circle forms. If the leader does loose the last robot because it cannot rotate quickly enough, it turns further clockwise (i.e, in the direction of the circle). It approaches the circle from the outside, may detect another robot in front of the last robot, but ultimately settles behind the last robot because $(x_{detected} - x_{last}) < d_f$ (see Theorem 2.1). A closed circle forms. □

Proposition 4. *Given is a circle formation with n robots. One robot (the emerging leader) changes its behavior from the circle formation rule to moving along a continuous trajectory.*

Theorem 4.1. *The circle formation transitions into leader following.*

Proof. The emerging leader exits the circle tangentially without intersecting any other robot's trajectory and line of sight except the one of its direct follower. All other robots keep following uninterruptedly. □

3.1.2 Parallel circle formation (general case)

We now generalize the above results to the case of n robots being initialized in random positions and taking parallel actions according to the circle formation rule (i.e., the scenario in Bluesim simulations and Blueswarm experiments).

Proposition 5. *A number of n robots are initialized in random positions and all of them apply the circle formation rule.*

Theorem 5.1. *The n robots form a circle.*

Proof. We summarize what we have established so far:

- Robots are able to detect another robot (Theorem 1.1).
- Robots are able to get behind another robot (Theorem 1.2 and 1.3).
- Robots are able to follow another robot (Theorem 1.4).
- Robots are able to follow a group of other robots in line (Theorem 2.1).
- Given a group in which every robot follows and is followed by exactly one other robot, a circle can form (Theorem 3.1).

Several groups cannot co-exist as robots keep rotating and detecting non-group robots. We now show that a single group in which every robot follows and is followed by exactly one other robot will emerge. Note that a robot never follows multiple robots at a time because the sensor-actuator loop is considered continuous and noise-free, and directional changes are instantaneous. Furthermore, an individual detecting an individual or a group detecting an individual does not lead to multiple robots following a single other robot. There are, however, two cases in which multiple robots can temporarily follow a single other robot:

1. An individual detects a group. The individual starts to follow a group robot that has been followed already. The individual robot will join the group in line (Theorem 2.1).
2. A group detects another group. The detecting robot leads its group to join the other group. Four sub-cases are possible and might occur concurrently:
   (a) Both groups are circles. They merge. Note that several circles could only co-exist for a very short time (e.g., after initialization) before detecting each other.
   (b) One group is a partial circle, i.e., a group with an insufficient number of robots to close itself. This partial circle detects a full circle. As the robots of the partial circle are turning on their circular trajectory, the leading (i.e., first) robot has always seen everything already that followers are about to see. Consequently, the leading robot is the first to detect the full circle. It leads its entire group to join the full circle.

(c) A full circle detects a partial circle. Because all robots turn in the same direction (e.g., clockwise), the detecting robot will see the last robot of the partial circle first because it cannot rotate past it without seeing it. The detecting robot leads all circle robots to join the partial circle (and a full circle forms because there are more than enough robots now).

(d) A partial circle detects another partial circle. The detecting robot has to be the leading robot (b) and will see the last robot of the other partial circle first (c). It leads all its following robots to join the other partial circle.

Following the above, we observe that groups do not partition, the situation of multiple robots following a single robot is always resolved, and the number of individual entities (groups or individuals) decreases. A single entity in the shape of a circle emerges. This is true because any other configuration would not allow that every robot follows and is followed by exactly one other robot. In other words, a single circle is the only stable configuration in which every robot can always follow and be followed by exactly one other robot. □

Theorem 5.2. *The n robots are equally distributed along the emerging circle, i.e., they converge to a regular polygon of radius R.*

Proof. If the robots have not already converged to a regular polygon, some robots must be too close to the robot in the front, others too distant. The circle formation rule naturally balances the distances between robots according to these two situations. (1) Sensor returns 0: The robot is too distant and rotates clockwise until it detects any other robot. As a result, it catches up by temporarily moving at $r_0 < R$, i.e., on a circle of smaller radius. (2) Sensor returns 1: The robot is too close and rotates counterclockwise until it no longer detects the robot in the front. It slows down by temporarily moving out of the circle. The resulting regular polygon is of radius R as shown in our previous work. □

3.2 Viewing half-angle α and circle radius R

Intuitively, the more robots N form a circle, the smaller the viewing half-angle α has to be in order not to see some neighbor at all times. This intuition can easily be confirmed by placing N robots on the vertices of a regular N-sided polygon (Figure 7). In practice and given robots with approximately circular bodies of radius ρ, the viewing half-angle α has to be chosen below π/N in order to maintain a circle of radius R.

In all of our 1100 experiments, robots successfully formed and maintained a circle. The convergence time varied significantly across simulation runs (Figure 8). In general, the mean speed of circle formation was not affected largely by different viewing half-angles α.

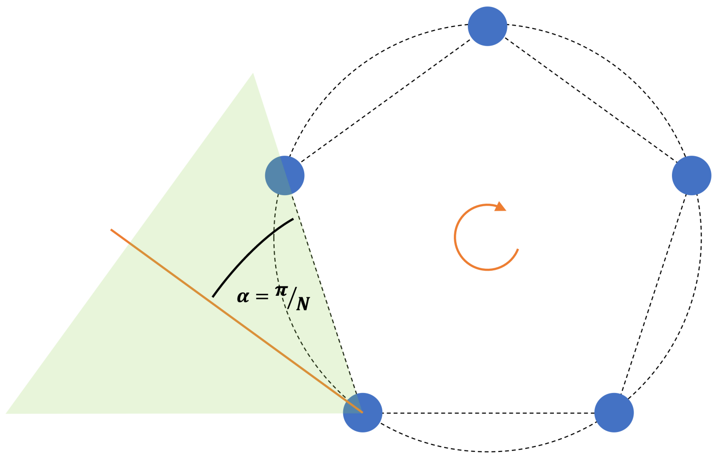

Figure 7: $N = 5$ robots milling in an N-sided polygon formation. Such milling is stable as long as the viewing half-angle α is smaller than π/N.

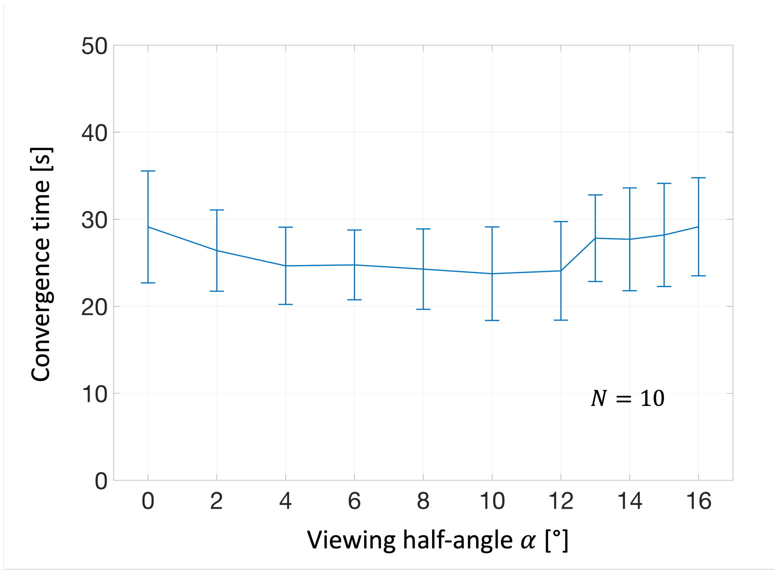

Figure 8: Mean and standard deviation of convergence time from $n = 100$ experiments per viewing half-angle α. Varying α across a large range did not impact the speed of circle formation significantly.

3.3 Number of robots N and circle radius R

In all of our 1100 experiments, robots successfully formed and maintained a circle. The convergence time varied significantly across simulation runs (Figure 9). In general, the more robots N the larger the circle and the more distance to be travelled. However, the mean convergence time was only marginally affected, in part because of parallel action taken by the robots.

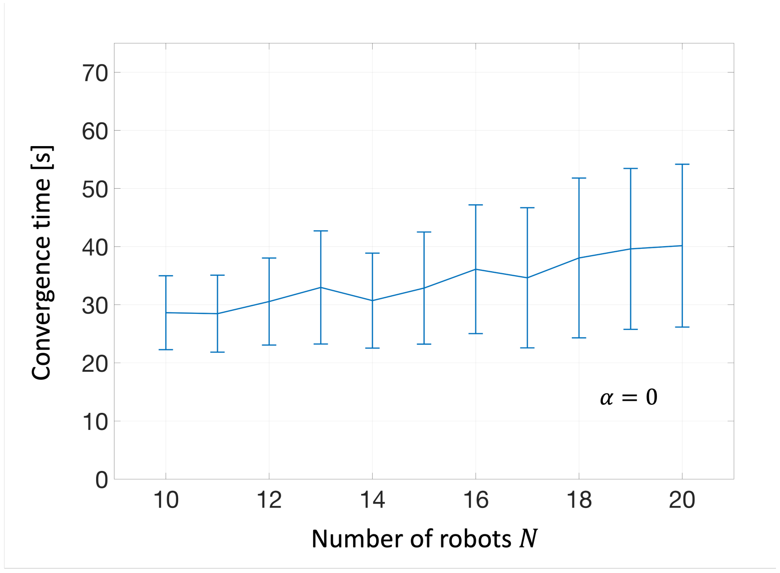

Figure 9: Mean and standard deviation of convergence time from $n = 100$ experiments per number of robots. Doubling the number of robots from 10 to 20 did not slow down circle formation significantly.

3.4 Sufficient cognition speed f_c

For $N = 10$ simulated robots with forward looking plane-of-sight sensors ($\alpha = 0$) moving at a velocity of roughly $130\,\mathrm{mm/s}$, a minimum cognition speed of $f_c = 1.25\,\mathrm{Hz}$ was required to enable circle formation. At lower cognition speeds, neither the average robot distance from the fitted circle (i.e., the per-robot-error) nor the robot distribution σ converged over time (Figures 10 and 11).

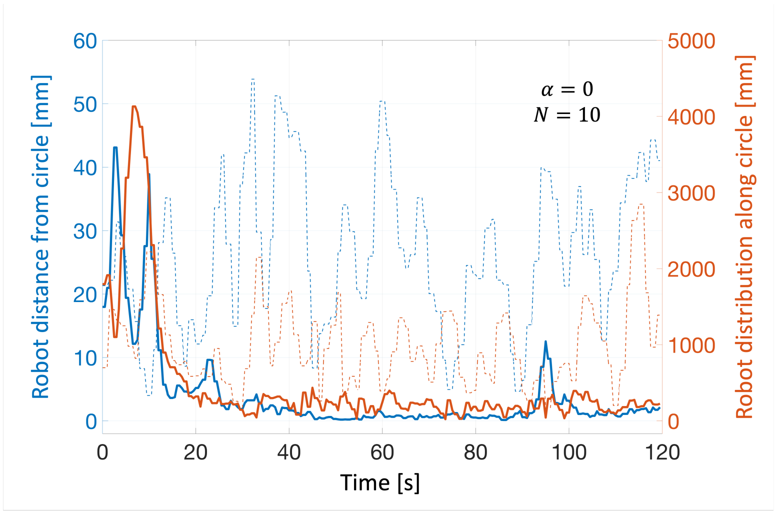

Figure 10: Average robot distance from the fitted circle and robot distribution σ over time for a successful and unsuccessful experiment with $f_c = 1.25\,\mathrm{Hz}$ (solid lines) and $f_c = 0.75\,\mathrm{Hz}$ (dashed lines), respectively.

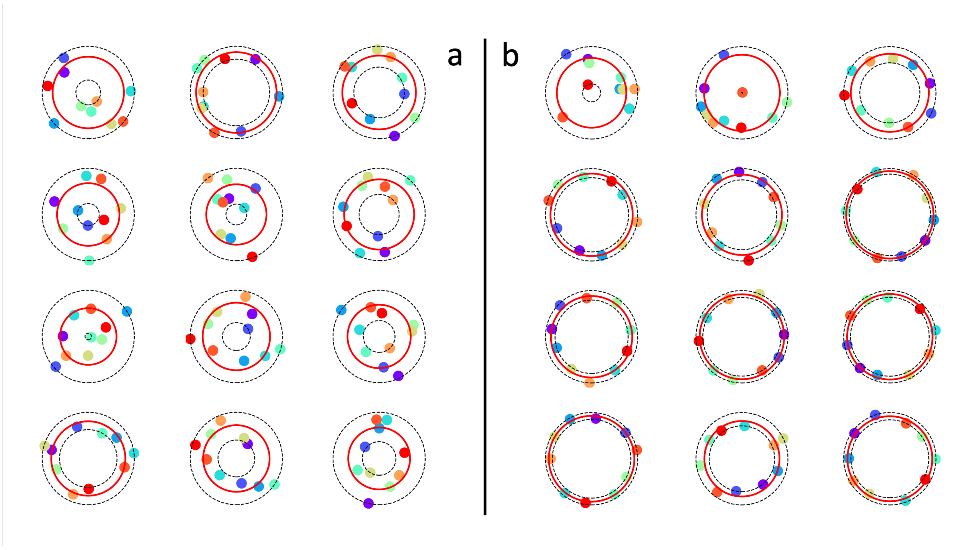

Figure 11: $N = 10$ robots with plane-of-sight sensors ($\alpha = 0$) during 12 instances of (a) unsuccessful and (b) successful circle formation with cognition speeds (a) $f_c = 0.75\,\mathrm{Hz}$ and (b) $f_c = 1.25\,\mathrm{Hz}$.

3.5 Minimalist perception

In our previous work, we had to create a dark underwater environment to reliably detect Bluebot LEDs. Here, we colored the robots black such that they are clearly distinguishable from the white tank environment. In order to apply our rapid blob detection in the usual way, the raw images were masked (the image area above the water surface is ignored) and negated, such that the robots appear bright and the background dark (Figure 12).

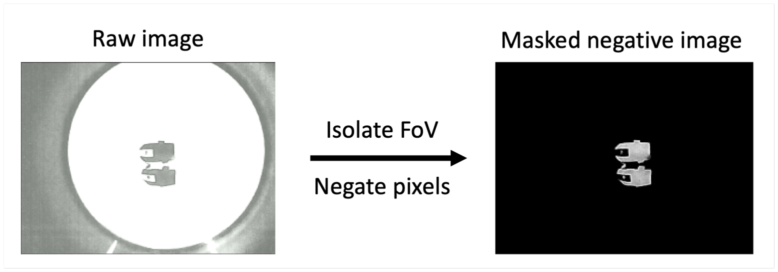

Figure 12: The raw image shows a single neighbor and its reflection on the water surface. A black mask is applied to the peripheral area outside of the spherical field of view (FoV); the pixels inside the FoV are negated. Our blob detection can be applied on the resulting negative.