Spatial Coverage Without Computation

Anıl Özdemir[1], Melvin Gauci[2], Andreas Kolling[3], Matthew D. Hall[1], and Roderich Groß[1]

***Abstract*— We study the problem of controlling a swarm of anonymous, mobile robots to cooperatively cover an unknown two-dimensional space. The novelty of our proposed solution is that it is applicable to extremely simple robots that lack run-time computation or storage. The solution requires only a single bit of information per robot—whether or not another robot is present in its line of sight. Computer simulations show that our deterministic controller, which was obtained through off-line optimization, achieves around 71–76% coverage in a test scenario with no robot redundancy, which corresponds to a 26–39% reduction of the area that is not covered, when compared to an optimized random walk. A moderately lower level of performance was observed in 20 experimental trials with 25 physical e-puck robots. Moreover, we demonstrate that the same controller can be used in environments of different dimensions and even to navigate a maze. The controller provides a baseline against which one can quantify the performance improvements that more advanced and expensive techniques may offer. Moreover, due to its simplicity, it could potentially be implemented on swarms of sub-millimeter-sized robots. This would pave the way for new applications in micro-medicine.**

I. INTRODUCTION

We consider the multi-robot spatial coverage problem for which a group of robots is required to cooperatively cover an environment or specific regions of interest within. The problem is relevant for a number of applications. For instance, robots may be required to monitor a given area, perhaps to log data, or to detect abnormal events and relay an alarm to a central station. The robots may also be required to service the environment, such as watering or applying chemicals to a field of crops. Finally, complete coverage constitutes a systematic strategy for search. In this work, we assume that the environment is bounded, and that all parts are to be covered continuously. This differs from the problem of visiting every location once [1], or repeatedly [2], [3].

Howard *et al.* [4] showed that a swarm of robots, by emulating the movements of charged particles in a potential field, could disperse within an office-like environment. Each robot used relative position information about nearby robots and obstacles. While the attained formations may not be uniform, they are guaranteed to be stable. McLurkin and Smith [5] studied a strategy to deploy a swarm of robots in a bounded environment, where each robot moved away from its k nearest neighbors. For $k = 2$, the robots obtained an almost uniform distribution, whereas for $k \gg 2$, they ended up at the boundary. They could also disperse in open space while maintaining connectivity. The robots used an infra-red communication system to obtain relative positions. Schwager *et al.* [6] studied a swarm of robots that, when put in a bounded environment, assume positions that optimize an a priori unknown utility function. This could allow regions of importance to be monitored more densely. The robots used local sensing to sample and approximate the utility function. Their controller achieves near-optimal coverage, and was tested on the same platform as in [5]. Ramaithitima *et al.* [7] proposed a solution to the coverage problem that relies on only touch and bearing sensors. As the environment is not known in advance, the robots get sequentially deployed, until complete coverage is guaranteed.

Gauci *et al.* [8] proposed a *computation-free swarming* concept, which is applicable to robots that lack arithmetic logic units, run-time memory, and communication capabilities. In the simplest form, each robot has a single binary line-of-sight sensor that tells whether another robot is detected or not. The concept was first applied to the problem of multi-robot rendezvous [8], [9], and was further investigated by Brown *et al.* [10], who used novelty search to discover what other behaviors it could produce. One of these behaviors turned out to be dispersion; however, it was not further analyzed. The computation-free swarming concept has also been used with ternary line-of-sight sensors, allowing the robots to detect both robots and objects (though not simultaneously). This enabled swarms of physical robots to collectively cluster a group of objects [11] and to choose collectively between one of multiple options [12]. Recently, Wareham and Vardy [13] formally examined the computational-free swarming concept for grid-based environments, showing that the design problem, given an arbitrary task, cannot be solved in polynomial time, but that efficient solutions exist for a restricted class of problems.

In this paper we present the first solution to the spatial coverage problem that is applicable to anonymous robots that lack the ability to compute, store run-time information, or communicate. Despite their simplicity, the robots are shown to cover around 71–76% of an unknown two-dimensional space in situations without robot redundancy. They outperform random walks—reducing the area that is not covered by 26–39%. Although alternative approaches that require computation, localization, and more elaborate coordination may perform better, our minimalist approach serves a number of important functions. For one, it establishes a baseline on what should be expected from any system and therefore

[1]Anıl Özdemir, Matthew D. Hall, and Roderich Groß are with the Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK {a.ozdemir, m.d.hall, r.gross}@sheffield.ac.uk

[2]Melvin Gauci is with the Wyss Institute for Biologically Inspired Engineering, Harvard University, Boston, MA, USA mgauci@g.harvard.edu

[3]Andreas Kolling is with iRobot, Pasadena, CA, USA akolling@irobot.com

allows a quantification of the performance improvements that any advanced and expensive techniques add to the baseline. Secondly, the approach could be used on extremely simple robot-like systems, such as micro- and nano-scale mobile machines that lack fully-fledged CPUs or wireless radios.

II. Problem Definition

A. Environment and Robots

Consider a two-dimensional, bounded environment, $\mathcal{E} \subset \mathbb{R}^2$, with n autonomous mobile robots. The robots are indistinguishable from each other and execute an identical controller. They lack the capability of performing arithmetic computation, and have no run-time memory. Moreover, they are unable to communicate with each other, cannot localize, and have no knowledge of $\mathcal{E}$ nor of n.

At time t, robot i's position and orientation is written as $x_i(t) \in \mathcal{E}$ and $\theta_i(t) \in [0, 2\pi)$, respectively. Robots are modeled as open disks of radius r which are fully contained in $\mathcal{E}$.

Each robot moves using a differential drive. Its linear velocity in its local reference frame is $\frac{v_\ell(t)+v_r(t)}{2} v_{\max}$, and its angular velocity is $\frac{v_r(t)-v_\ell(t)}{d_{\text{wheel}}} v_{\max}$, where $v_\ell(t), v_r(t) \in [-1, 1]$ are the normalized wheel velocities along the ground, $v_{\max}$ is the maximum velocity, and $d_{\text{wheel}} \leq 2r$ is the inter-wheel distance.

Each robot has an unlimited-range sensor that detects whether another robot is present in the line of sight directly ahead of the robot. It reports $s(t) = 1$ at time t if another robot is present and $s(t) = 0$ otherwise.

The robot executes a deterministic controller, $c : \{0, 1\} \to [-1, 1] \times [-1, 1]$. At time t, c assigns sensor reading $s(t)$ to a pair of wheel velocities. Formally,

$$(v_\ell(t), v_r(t)) = c(s(t)) = \begin{cases} (v_{\ell,0}, v_{r,0}) & \text{if } s(t) = 0, \\ (v_{\ell,1}, v_{r,1}) & \text{otherwise.} \end{cases} \quad (1)$$

Using $(v_{\ell,0}, v_{r,0}, v_{\ell,1}, v_{r,1}) \in [-1, 1]^4$, any reactive control strategy can be expressed.

B. Objective

The coverage literature has used a number of performance criteria for coverage that either relate to the area a robot covers, special positions robots should occupy, or special measures of importance of parts of the space. In our work, we consider the following two performance criteria. The first criterion, *cell coverage*, determines coverage quality by relating robot positions to a given partitioning of the environment into cells. The goal is to occupy every cell with at least one robot. Such a partitioning may be provided by a user, derived from a utility function that represents the importance of the space, or simply be uniform (as in our scenarios). The second criterion, *area coverage*, measures the joint area that is close to some robot.

The *cell coverage* at time t is defined by

$$\mathcal{P}_{\text{cell}}(t) = \frac{m_{\text{occupied}}(t)}{m}, \quad (2)$$

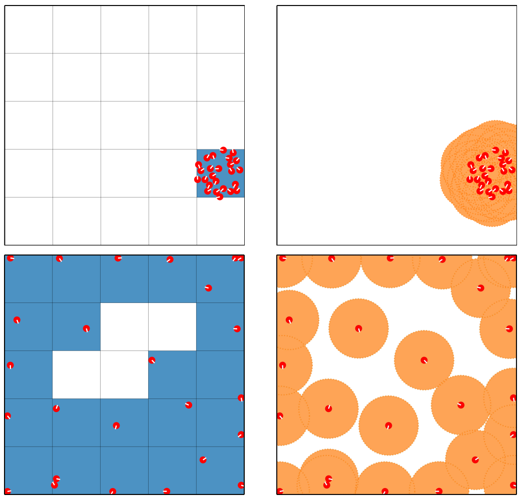


Fig. 1. A group of 25 robots performing coverage with two performance measures illustrated at the beginning (top) and end (bottom) of a trial: (a) Cell coverage uses a decomposition of the environment (square cells), and reports the fraction of cells with at least 1 robot; (b) area coverage assumes that each robot covers all points within a certain range, and reports the fraction of the environment's area that the robots collectively cover. The corresponding coverage (percentage) for (a) is 4.0% (top) and 84.0% (bottom). For (b) it is 11.6% (top) and 69.8% (bottom).

where $m_{\text{occupied}}(t)$ is the number of cells that contain at least one robot at time t. It follows that $1/m \leq \mathcal{P}_{\text{cell}} \leq \min(n/m, 1)$.

The *area coverage* at time t is defined by

$$\mathcal{P}_{\text{area}}(t) = \frac{A\big(\bigcup_{i=1}^{n} \mathcal{N}_i\big)}{A(\mathcal{E})}, \quad (3)$$

where $\mathcal{N}_i = \{p \in \mathcal{E} | : \|p - x_i(t)\| \leq r_{\text{cover}}\}$, $r_{\text{cover}} > r$ denotes the distance up to which the robot covers the environment, and $A(S)$ is the area of S. It follows that $\frac{\pi r_{\text{cover}}^2}{A(\mathcal{E})} < \mathcal{P}_{\text{area}} \leq \min(\frac{n\pi r_{\text{cover}}^2}{A(\mathcal{E})}, 1)$.

Note that the cell partitions and coverage radius have no bearing on the robot's behavior. They are merely used to measure performance. Figure 1 illustrates both performance measures in a square environment with $n = 25$ robots. The environment is partitioned into 25 equally-sized square cells.

III. Controller Design

We use an evolutionary robotics approach [14], [15], [8] for designing the deterministic controller.

A. Evaluation of Candidate Solutions

Candidate solutions are controllers that are considered by the optimization process. They are represented in continuous space, by tuples $\mathbf{v} = (v_{\ell,0}, v_{r,0}, v_{\ell,1}, v_{r,1}) \in [-1, 1]^4$.

To assess the performance of a candidate solution, simulations are conducted with Enki [16], a 2-D rigid bodies physics engine. The robot platform is the e-puck [17]. It is modelled as a disk of radius $r = 3.7\,\text{cm}$ and mass $152\,\text{g}$. Its maximum velocity is $v_{max} = 12.8\,\text{cm/s}$. The inter-wheel distance is $d_{\text{wheel}} = 5.1\,\text{cm}$. Throughout all simulation runs, 5% uniform noise is affecting the velocity of each wheel.

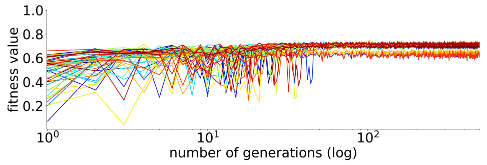


Fig. 2. Fitness dynamics of 50 evolutionary runs.

For each candidate solution, 20 simulation trials are performed using $\mathcal{E} = [0, 300] \times [0, 300]$ (cm) and $n = 25$ robots. Each trial lasts for $T = 120\,\mathrm{s}$, corresponding to 1200 updates of the robot's control cycle.

The environment is decomposed into a 5×5 grid of cells, as shown in Figure 1(a). At the beginning of the trial, all robots are placed with uniformly random position and orientation at a distance of up to $30\,\mathrm{cm}$ from the center of a uniformly randomly chosen cell.[1] All candidate solutions in a generation are evaluated on the same set of initial configurations.

The performance of the swarm in a trial is measured using the cell coverage measure[2]. Formally,

$$F = \frac{2}{T(T+1)} \sum_{t=1}^{T} t\mathcal{P}_{\text{cell}}(k). \tag{4}$$

The performance at time $t < T$ is taken into account, weighted by t, to reward solutions that reach good coverage faster. The constant factor normalizes F to $[0, 1]$.[3] The overall fitness of the candidate solution is the mean performance across 20 trials.

B. Evolutionary Algorithm

Candidate solutions are synthesized using the Covariance Matrix Adaptation-Evolution Strategy, a black-box optimization method that is quasi parameter-free [18]. We use a population of $\lambda = 12$ candidate solutions. Initially the candidate solutions are generated randomly using uniform distributions. In every generation, each candidate solution is evaluated via simulation as described above. The evolutionary run terminates after 500 generations.

C. Controller Selection

In total, 50 evolutionary runs were conducted. The fitness dynamics are shown in Figure 2. In 4 runs, the evolution prematurely converged towards solutions of lower quality than in the other 46 runs.

To choose the *best* controller out of the 50 evolutionary runs, we post-evaluated the candidate solutions from the last generation of each run using 200 additional simulations, and chose the one with the highest mean performance. The following section examines the performance of the best controller across a range of scenarios.

[1]Initializing the robots in a circular region prevents orientation bias.

[2]The area coverage measure was not used during the evolutionary process because it is computationally more demanding.

[3]In principle, a 0-value is not possible, as at least 1 cell has to be covered.

D. Behavioral Analysis

The parameters of the best controller are: $v_{\ell,0} = 0.719558$, $v_{r,0} = 0.412543$, $v_{\ell,1} = -0.998071$, $v_{r,1} = -0.911843$.

As long as the sensor reading does not change, the robot follows a circular trajectory of radius R, with an angular velocity ω [19]. We obtain $R_0 = 9.40\,\mathrm{cm}$ and $\omega_0 = -0.77\,\mathrm{rad/s}$ for $s = 0$, and $R_1 = 56.48\,\mathrm{cm}$ and $\omega_1 = 0.22\,\mathrm{rad/s}$ for $s = 1$. When the robot detects another robot in its line of sight, $s = 1$, it moves rapidly backward (with 95.5% of the maximum linear speed) along a circular trajectory, in a counter-clockwise fashion. Otherwise, it moves forward (with 56.6% of the maximum linear speed), along a circular trajectory, in a clockwise fashion. If the two radii, R_0 and R_1 were the same, the robot would remain on its orbit indefinitely (assuming no collisions), and hence would be unable to spatially disperse. We hypothesize that for any $R_0 \ll R_1$, spatial separation can be achieved.

IV. SIMULATION STUDIES

In this section, we evaluate the performance of the controller (Section III-C) using a series of simulation studies. Unless otherwise stated, we use the same experimental setup as during the optimization process (see Section III-A).

We report the coverage performance observed at the end of the simulation trials using (2) and (3). For area coverage, we use a coverage radius of $r_{\text{cover}} = 37.22\,\mathrm{cm}$—a lower bound for the smallest radius that can obtain complete coverage.[4]

A. Performance Comparison with Different Strategies

We compare the computation-free controller against three other controllers:

- *Open-loop*: The robot moves backward with maximum speed; $(v_{\ell,0}, v_{r,0}) = (v_{\ell,1}, v_{r,1}) = (-1, -1)$.[5]
- *Greedy*: The robot moves backward with maximum speed if another robot is detected, $(v_{\ell,1}, v_{r,1}) = (-1, -1)$, and otherwise turns clockwise on the spot with maximum angular velocity; $(v_{\ell,0}, v_{r,0}) = (1, -1)$.
- *Random walk*: We use the random walk framework studied in [21]. The random walk consists of alternating straight-line segments of random length and on-the-spot rotations by random angles. The nature of the random walk is characterized by a 3-tuple, (ρ, α, β). The first parameter, $\rho \in (0, 1)$ controls the correlation between angles of subsequent segments, while the other two parameters, $\alpha \in (0, 2]$ and $\beta \in (0, \infty)$, control, respectively, the shape and scale of the distribution from which the lengths of the segments are drawn (for details, see [21]). To optimize (ρ, α, β), we follow the same process that was used for optimizing our computation-free controller. As in Section III-C, 50 evolutionary runs were conducted, and the best controller of each run was post-evaluated to select the overall best random walk for the present environment.

[4]This is obtained by solving $n\pi r_{\text{cover}}^2 = (2\pi\sqrt{3}/9)A(\mathcal{E})$ [20].

[5]Due to symmetry, this strategy is identical to $(1, 1)$.

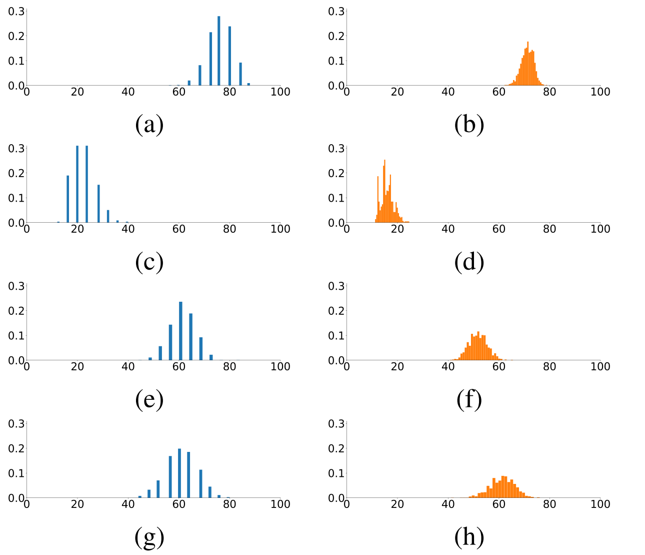


Fig. 3. Histogram showing the frequency of cell coverage (left) and area coverage (right) percentages, as observed in 1000 simulation trials with $n = 25$ robots, controlled by (a)–(b) our optimized controller, (c)–(d) an *open-loop* controller, and (e)–(f) the *greedy* controller, all of which are deterministic and computation-free, as well as (g)–(h) the optimized *random walk* controller, which uses computation to count control cycles while moving forward and turning, and to generate pseudo-random numbers from tailored distributions for the step length and turning angle. The environment has dimensions 300 cm × 300 cm.

For each strategy, 1000 trials were performed. Figure 3 shows the results. Regarding cell coverage (left), the mean performance is 76.0% for the proposed controller, 22.4% for the deterministic *open-loop* controller, 60.8% for the deterministic, *greedy* controller, and 60.8% for the optimized *random walk* controller, respectively. Regarding area coverage (right), the mean performance is 71.2% for the proposed controller, 16.0% for the deterministic *open-loop* controller, 51.8% for the deterministic, *greedy* controller, and 61.3% for the optimized *random walk* controller, respectively. The proposed controller achieves a 25.6–38.8% reduction in the uncovered area when compared to the *random walk* controller. In addition, its performance is more consistent: the variances for cell and area coverage are 4.99% and 6.36% for the proposed controller and *random walk* controller, respectively.

B. Effect of Sensory Noise

We examine the situation that noise is affecting the reading values of the binary line-of-sight sensor. This noise is in addition to the noise affecting the wheel velocities.

We consider three types of noise:

- *False-negative*: For $s(t) = 1$, with probability $p \in [0, 1]$ the reading value is replaced by a random value, $X(t) \in \{0, 1\}$, which is uniformly chosen at time t. For $s(t) = 0$, the original value is retained.
- *False-positive*: For $s(t) = 0$, with probability $p \in [0, 1]$ the reading value is replaced by a random value, $X(t) \in \{0, 1\}$, which is uniformly chosen at time t. For $s(t) = 1$, the original value is retained.

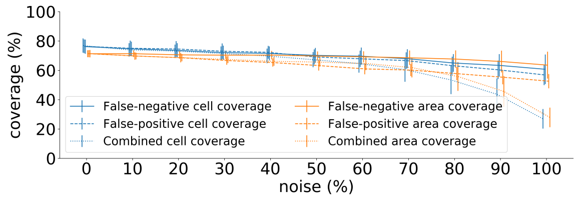


Fig. 4. Cell coverage (blue) and area coverage (orange) of $n = 25$ robots with noisy sensors and the computation-free controller. Three types of noise: (a) false-negative (solid), (b) false-positive (dashed), and (c) combined (dotted). Mean values and standard deviations based on 100 simulation trials per level and type of noise. The environment has dimensions 300 cm × 300 cm.

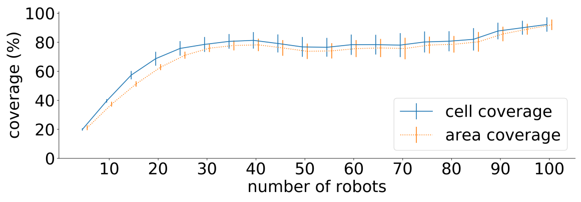


Fig. 5. Cell coverage (blue) and area coverage (orange) of $n = 5, 10, 15, \ldots, 100$ robots in a 300 cm × 300 cm environment using the computation-free controller. Mean values and standard deviations based on 100 simulation trials.

- *Combined*: With probability $p \in [0, 1]$ the reading value is replaced by a random value, $X(t) \in \{0, 1\}$, which is uniformly chosen at time t.

We run 100 simulation trials for each $p \in \{0, 0.1, 0.2, \ldots, 1\}$.

Figure 4 shows the results. The controller is very robust to noise on only one sensor reading (i.e., either false-negative or false-positive), with both performance measures reporting over 60% coverage with 100% noise. The performance degrades faster if noise is present on both sensor readings (i.e., combined), down to around 20% at 100% noise. Note, however, that in this case 100% noise represents a purely random sensor reading.

C. Scalability

The default setup contains no redundancy. If a single robot fails, complete coverage can no longer be achieved. We investigate the performance of swarms of $n = 5, 10, 15, \ldots, 100$ robots in an environment of constant size (the default environment). Irrespective of the group size, the robots are initialized in a circular region of radius 60 cm, which is uniformly randomly placed within the boundaries of the environment. We run 100 simulation trials per group size.

Figure 5 shows the results. The average performance rapidly improves for up to around 40 robots, then plateaus, and finally improves again for 50 and more robots. The presence of the plateau, where the performance may even slightly decrease, is a counter-intuitive result, to be further investigated in the future.

D. Effect of the Environment Shape

We investigate the spatial distribution of robots in more detail, and consider the impact of the environment shape.

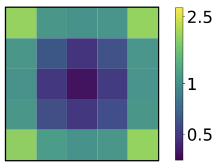


Fig. 6. Spatial distribution of $n = 25$ robots in a $300\,\text{cm} \times 300\,\text{cm}$ environment. Each cell indicates the mean number of robots present at the end of 1000 simulation trials.

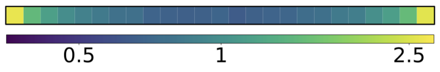


Fig. 7. Spatial distribution of $n = 25$ robots in an elongated, narrow corridor environment, of dimensions $1500\,\text{cm} \times 60\,\text{cm}$. All robots start from the cell in the center and execute the computation-free controller for $T = 600\,\text{s}$. Each cell indicates the mean number of robots present at the end of 1000 simulation trials.

Figure 6 presents a heat map of the mean number of robots that ended up in the 25 cells at the end of 1000 simulation trials. On average, the robots are more likely to be present at the environmental boundary, and in particular, in the four corners. Note that the robots are unable to detect the boundary. They repel from each other in an attempt to cover as much area as possible.

To investigate the swarms' ability to spread through an elongated, narrow corridor, a further 1000 simulation runs are performed in a 25×1 cell environment, of $1500\,\text{cm} \times 60\,\text{cm}$ dimensions. Figure 7 shows a heat map of the spatial distribution. The results are consistent with those obtained in the square environment—the robots are more likely to end up near the corner than in the center. However, every cell is covered, on average, by 0.65 robots or more.

Figure 8 shows heat maps for three $600\,\text{cm} \times 300\,\text{cm}$ environments. We assume a 10×5 cell composition. The first environment is free of obstacles. The second environment is split into two, by a thin vertical wall in the center. The wall contains an orifice in the middle. The third environment contains two vertical walls, creating a z-shaped parkour. As the environment is twice the size of the original environment, we used $n = 50$ and $T = 240\,\text{s}$. In general, the swarm copes well with the restrictions. It can be noted that the distributions are not fully symmetric in Figure 8b.

E. Navigating a Maze

In the following, we test the ability of the computation-free controller to make a swarm navigate a simple but unknown maze. The maze, shown in Figure 9a, contains a number of challenges, including two dead-ends. The robots enter the maze on the left-hand chamber, at a rate of one per $10\,\text{s}$, $12\,\text{s}$, $15\,\text{s}$, or $30\,\text{s}$. They execute the same controller as used in the coverage experiments. Robots are removed as soon as they are fully contained within the right-hand chamber. Figure 9b shows the number of robots within the maze over time. As one can see, the swarm can navigate the maze with a throughput of 1 robot per $15\,\text{s}$. However, as the input rate of new robots increases, the maze becomes increasingly crowded, up to the point that no new robots can be placed.

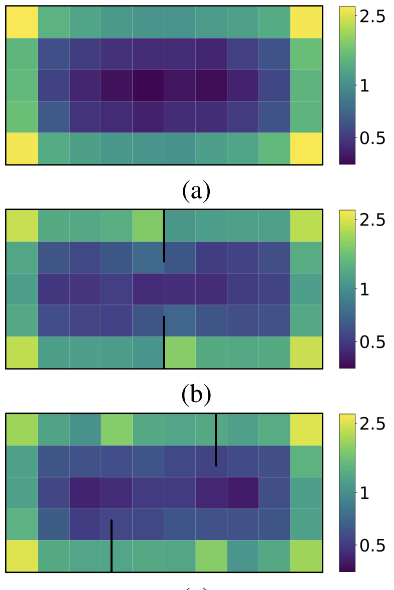


Fig. 8. Spatial distribution of $n = 50$ robots in a $600\,\text{cm} \times 300\,\text{cm}$ environment, with (a) no obstacles, (b) a pair of internal walls, creating an orifice, and (c) a pair of internal walls, creating a z-shaped parkour. Each cell indicates the mean number of robots present at the end of 1000 simulation trials.

V. EXPERIMENTS

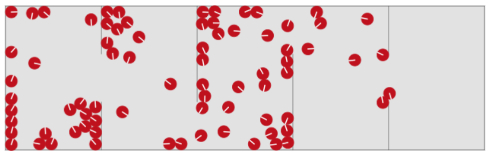
A. Experimental Setup

To validate the feasibility of computation-free coverage in a real environment, we conducted experiments using $n = 25$ physical e-puck robots in a bounded $300\,\text{cm} \times 300\,\text{cm}$ environment. The arena was logically split into a 5×5 grid of cells of $60\,\text{cm} \times 60\,\text{cm}$ dimensions.

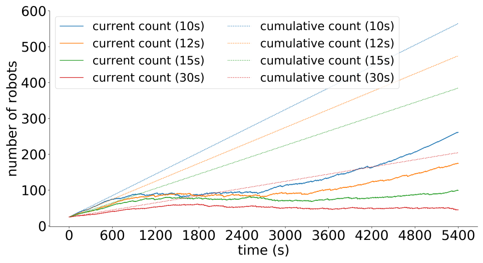
The e-puck has a CMOS-RGB camera that faces in the forward direction, and is used to emulate the line-of-sight sensor. Each e-puck is wrapped in a red 'coat' and operates in a well lit, white environment to improve the reliability of detection. The e-puck is also equipped with a red 'topper' to enable easier detection from the overhead camera for post-analysis. Each e-puck is slightly physically different and, due to only having two wheels, the camera direction, along the pitch axis, may slightly change during a robot's movement. To account for these misalignments, the line-of-sight sensor probes not 1 pixel, but rather a vertical column of 7 pixels, taken symmetrically from the center of the image. The line-of-sight sensor returns a positive reading ($s = 1$) if the color of any of the 7 pixels is not bright[6], and a negative reading ($s = 0$), otherwise.

For each experimental trial, the e-pucks are initialized in a different cell, as was done in the computer simulations. The cells are selected using a uniformly random distribution.

[6]We test $(R, G, B) \prec (180, 180, 140)$, where (R, G, B) is the RGB triplet of the color, and $\prec$ induces the partial order called product (or component-wise) order. Whereas a test if a value is smaller than another requires computation when implemented by digital circuits, an implementation in the analogue world is trivial (single high-gain differential amplifier). Note that irrespective of how the line-of-sight sensor is implemented on the e-puck, the control logic remains free from arithmetic computation.

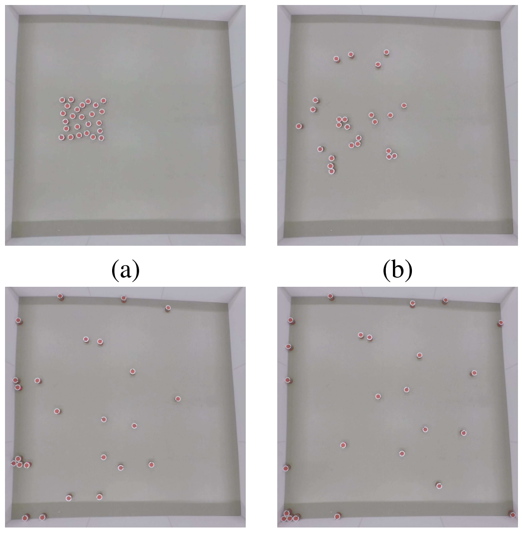


(a)



(b)

Fig. 9. A swarm of robots using the computation-free controller to navigate a maze. New robots enter the left chamber of the maze environment at a constant rate of one per 10 s, 12 s, 15 s, 30 s. Robots are removed as soon as they are fully within the right chamber. (a) Snapshot taken after 3600 s with a rate of one per 15 s. (b) Number of robots that are, or have been, within the maze over time for each of the rates.



(a) (b) (c) (d)

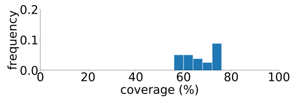
Fig. 10. Sequence of snapshots taken from a typical experimental trial with $n = 25$ physical e-pucks operating in a 300 cm $\times$ 300 cm environment. The snapshots were taken at (a) 0 s, (b) 10 s, (c) 30 s, and (d) 120 s.

Moreover, the robots' assume random orientations during initialization. The robots operate for $T = 120$ s.

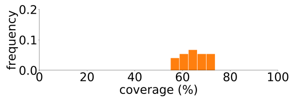
B. Results

We performed 20 experimental trials. All trials were recorded by the overhead camera, and are available in the online supplementary material [22]. From these video recordings, the positions of robots could be tracked. Figure 10 shows an example sequence of snapshots from a typical trial.

Figure 11 summarizes experimental results for all trials. On average, the swarm achieved a cell coverage and area coverage of 65.2% and 64.9%, respectively, which outperforms the previously obtained benchmarks (recall that the corresponding values for the random walk controller were 60.8% and 61.3%, in simulation). The reduction in performance may be attributed to the increased friction between the walls and robots, which could prevent them from continuing to rotate upon collision. Other factors include sensory noise and unknown hardware failures. Out of the 20 trials, 3 robots powered off during a trial, possibly due to low battery, which was rectified for the following trials.



(a)



(b)

Fig. 11. Histogram showing the frequency of cell coverage (left) and area coverage (right) percentages, as observed in 20 experimental trials with $n = 25$ robots, controlled by our optimized controller, which is deterministic and computation-free. The environment has dimensions 300 cm $\times$ 300 cm.

VI. CONCLUSIONS

This paper presented the simplest solution so far to the problem of cooperatively covering an unknown two-dimensional space with a swarm of anonymous mobile robots. The proposed controller is applicable to robots that lack run-time computation or storage. The solution requires only a single bit of information per robot—whether or not another robot is present in its direct line of sight.

A series of computer simulations showed that the controller outperformed a random walk (both solutions being optimized off-line). On average, in situations with no robot redundancy, it covered around 71–76% (depending on the coverage measure being used) of the space, whereas the random walk covered around 61% of the space. The swarm performance was found to degrade gracefully in the presence of noise. In the case of either false-negative or -positive noise, the swarm showed robust performance up to a noise level of 100%. Moreover, the performance was not particularly affected by the robot density in the environment. An analysis of the spatial distribution revealed that on average more robots ended up near the boundary, and in particularly any corners, reducing the efficiency in scenarios without robot redundancy. A further simulation experiment revealed that a constant-rate inflow of robots can navigate a maze up to a critical rate.

Experiments with swarms of 25 physical e-puck robots demonstrated the feasibility of computation-free coverage on a real physical platform. They revealed a moderate decrease in performance, when compared to simulation trials.

In the present study, we focused on the coverage performance and bounded regions that were to be uniformly covered. In the future, one could equip the robots with an additional sensor to detect specific features of the environment, such as the presence of polluting chemicals. Future studies could take the energy consumption of the robots into account, when designing the strategies. In the medium- to long-term, strategies of the simplicity as presented here could inform the design of novel micro-scale or nano-scale robot-like systems. This would enable novel applications of multi-robot coverage, which are not possible with present systems.

References

[1] H. Choset, “Coverage for robotics – a survey of recent results,” *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 113–126, 2001.

[2] S. Rutishauser, N. Correll, and A. Martinoli, “Collaborative coverage using a swarm of networked miniature robots,” *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 517–525, 2009.

[3] D. Portugal and R. Rocha, “A survey on multi-robot patrolling algorithms,” in *Technological Innovation for Sustainability*, L. M. Camarinha-Matos, Ed. Berlin, Germany: Springer, 2011, pp. 139–146.

[4] A. Howard, M. J. Matarić, and G. S. Sukhatme, “Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem,” in *Proceedings of the 6th International Symposium on Distributed Autonomous Robotic Systems (DARS 2002)*. Berlin, Germany: Springer, 2002, pp. 299–308.

[5] J. McLurkin and J. Smith, “Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots,” in *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems (DARS 2004)*. Tokyo, Japan: Springer, 2007, pp. 399–408.

[6] M. Schwager, J. McLurkin, and D. Rus, “Distributed coverage control with sensory feedback for networked robots.” in *Robotics: Science and Systems*, 2006, pp. 49–56.

[7] R. Ramaithitima, M. Whitzer, S. Bhattacharya, and V. Kumar, “Sensor coverage robot swarms using local sensing without metric information,” in *2015 IEEE International Conference on Robotics and Automation (ICRA 2015)*. IEEE, 2015, pp. 3408–3415.

[8] M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß, “Self-organized aggregation without computation,” *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1145–1161, 2014.

[9] M. Gauci, J. Chen, T. J. Dodd, and R. Groß, “Evolving aggregation behaviors in multi-robot systems with binary sensors,” in *Proceedings of the 11th International Symposium on Distributed Autonomous Robotic Systems (DARS 2012)*, vol. 104. Berlin, Germany: Springer, 2014, pp. 355–367.

[10] D. S. Brown, R. Turner, O. Hennigh, and S. Loscalzo, “Discovery and exploration of novel swarm behaviors given limited robot capabilities,” in *Proceedings of the 13th International Symposium on Distributed Autonomous Robotic Systems (DARS 2016)*. Berlin, Germany: Springer, 2018, pp. 447–460.

[11] M. Gauci, J. Chen, W. Li, T. J. Dodd, and R. Groß, “Clustering objects with robots that do not compute,” in *Proceedings of the 2014 International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*. IFAAMAS, 2014, pp. 421–428.

[12] A. Özdemir, M. Gauci, S. Bonnet, and R. Groß, “Finding consensus without computation,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1346–1353, 2018.

[13] T. Wareham and A. Vardy, “Viable algorithmic options for designing reactive robot swarms,” *ACM Transactions on Autonomous and Adaptive Systems*, vol. 13, no. 1, pp. 5:1–5:23, 2018.

[14] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology, Intelligence, and Technology*. Cambridge, MA, USA: MIT Press, 2000.

[15] V. Trianni, *Evolutionary Swarm Robotics: Evolving Self-Organising Behaviours in Groups of Autonomous Robots*, 1st ed. Berlin, Germany: Springer, 2008.

[16] S. Magnenat, M. Waibel, and A. Beyeler, “Enki: An open source fast 2D robot simulator,” https://github.com/enki-community/enki, 2009.

[17] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, “The e-puck, a robot designed for education in engineering,” in *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, vol. 1, 2009, pp. 59–65.

[18] N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[19] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*. Cambridge, UK: Cambridge University Press, 2010.

[20] R. Kershner, “The number of circles covering a set,” *American Journal of Mathematics*, vol. 61, no. 3, pp. 665–671, 1939.

[21] C. Dimidov, G. Oriolo, and V. Trianni, “Random walks in swarm robotics: An experiment with Kilobots,” in *Proceedings of the 10th International Conference on Swarm Intelligence (ANTS 2016)*. Cham, Switzerland: Springer, 2016, pp. 185–196.

[22] A. Özdemir, M. Gauci, A. Kolling, M. D. Hall, and R. Groß, “Online supplementary material,” http://naturalrobotics.group.shef.ac.uk/supp/2019-001, 2019.