

# Hearing the shape of an arena with spectral swarm robotics

Leo Cazenille<sup>1</sup> , Nicolas Lobato-Dauzier<sup>2,3</sup> , Alessia Loi<sup>4</sup> , Mika Ito<sup>5</sup>, Olivier Marchal<sup>6</sup> , Nathanael Aubert-Kato<sup>5</sup> , Nicolas Bredeche<sup>4,\*</sup> , Anthony J. Genot<sup>2,\*</sup>

<sup>1</sup>Université Paris Cité, CNRS, LIED UMR 8236, F-75006 Paris, France – leo.cazenille@gmail.com

<sup>2</sup>LIMMS (IRL2820)/CNRS-IIS, University of Tokyo, Tokyo, Japan

<sup>3</sup>Sorbonne Université, CNRS, Institut de Biologie Paris-Seine, Laboratoire Jean Perrin, F-75005, Paris, France

<sup>4</sup>Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France

<sup>5</sup>Ochanomizu University, Department of Information Sciences, Tokyo, Japan

<sup>6</sup>Université Jean Monnet Saint-Étienne, CNRS UMR 5208, Institut Camille Jordan, Institut Universitaire de France, F-42023 Saint-Etienne, France

## Abstract

Swarm robotics promises adaptability to unknown situations and robustness against failures. However, it still struggles with global tasks that require understanding the broader context in which the robots operate, such as identifying the shape of the arena in which the robots are embedded. Biological swarms, such as shoals of fish, flocks of birds, and colonies of insects, routinely solve global geometrical problems through the diffusion of local cues. This paradigm can be explicitly described by mathematical models that could be directly computed and exploited by a robotic swarm. Diffusion over a domain is mathematically encapsulated by the Laplacian, a linear operator that measures the local curvature of a function. Crucially the geometry of a domain can generally be reconstructed from the eigenspectrum of its Laplacian. Here we introduce spectral swarm robotics where robots diffuse information to their neighbors to emulate the Laplacian operator -enabling them to “hear” the spectrum of their arena. We reveal a universal scaling that links the optimal number of robots (a global parameter) with their optimal radius of interaction (a local parameter). We validate experimentally spectral swarm robotics under challenging conditions with the one-shot classification of arena shapes using a sparse swarm of Kilobots. Spectral methods can assist with challenging tasks where robots need to build an emergent consensus on their environment, such as adaptation to unknown terrains, division of labor, or quorum sensing. Spectral methods may extend beyond robotics to analyze and coordinate swarms of agents of various natures, such as traffic or crowds, and to better understand the long-range dynamics of natural systems emerging from short-range interactions.

## Main

In swarm robotics [3–5], robots collaborate to solve a given problem. Each robot has a limited capac-

ity for sensing, processing, and actuation, but, collectively, the robots compensate with their massive parallelism, redundancy, and adaptability. Proof-of-concepts in swarm robotics have showcased the construction of nests [6], the formation of shapes [2, 7], flying UAVs [8, 9] and social learning [10]. However, these ad-hoc approaches rely on robots having direct access to all necessary information in their surroundings [11]. It remains unclear how a swarm of robots can solve a global problem like identifying the geometry of its entire environment in a decentralized way. In nature, biological swarms solve geometrical problems with limited perception and mobility [12–15]. Ants find the shortest path between their nest and a food source (foraging) [16, 17] and termites build 3D networks of conduits to ventilate their nest [18]. Schools of fish or flocks of birds move in a synchronized pattern (schooling or flocking) [19], and a colony of bees can split into two colonies to grow (swarming) [20]. Cells depend on the spatial gradients of chemicals (morphogens) to establish a universal positioning system [21]. Although the details vary, these global patterns emerge from the local diffusion of information across the agents of the swarm. These biological phenomena hint at an underlying principle of information diffusion which, though not explicitly calculating mathematical models, effectively solves complex spatial problems.

Mathematically, the diffusion of a physical quantity across space is described by the Laplacian  $\nabla^2$  [22] - a second-order linear operator that correspond to the divergence ( $\nabla \cdot$ ) of the gradient ( $\nabla$ ) and quantifies the local curvature of a function:  $\nabla^2 f = \nabla \cdot (\nabla f)$ , *i.e.*, how quickly does the function vary with space. The Laplacian is ubiquitous in physics [23] and describes the diffusion of heat, matter or momentum (through the heat equation), or the propagation of mechanical or electromagnetic waves (through the wave equation). The Laplacian also controls the energy level of a quantum system (through the Schrödinger equation). It is an intrinsic differential operator, meaning that it operates based on the internal geometry of a space without reference to an external coordinate system, as opposed to the gradient for instance. This property makes it invariant by rotation and translation, and is essential for

\*co-corresponding authors (equal contribution): genot@iis.u-tokyo.ac.jp nicolas.bredeche@sorbonne-universite.fr

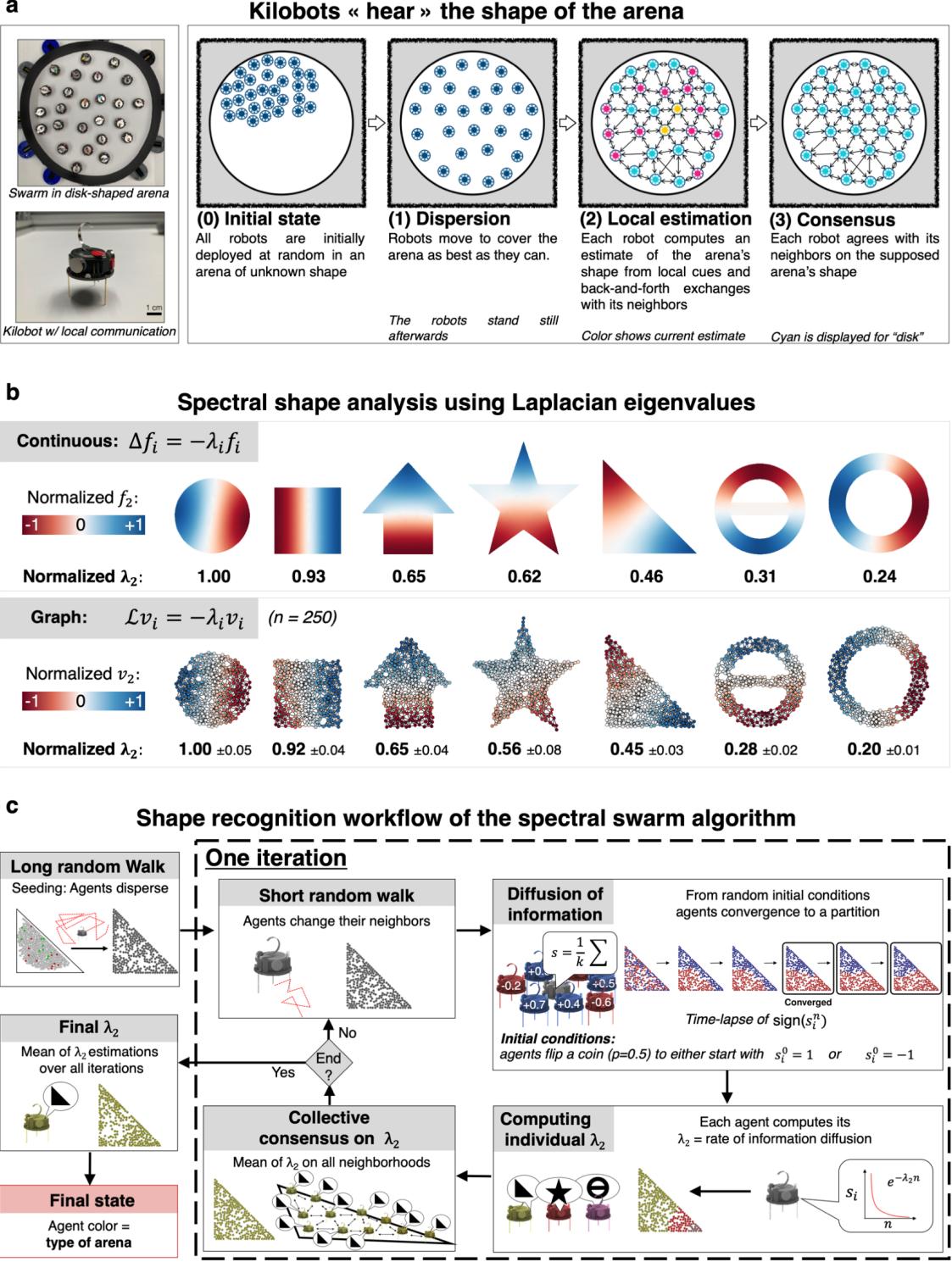


Figure 1: **Framework of spectral swarm robotics.** **a:** Description of our contribution: a swarm of robots computes the spectral fingerprint of its arena in a distributed way, so that each individual robot has its own fingerprint estimate; the robots reach a consensus on this value across the entire swarm; finally, this value is used to classify the shape in which the swarm is located. Our approach is validated using swarms of Kilobots [1, 2]. **b:** Comparison of results from the continuous and graph Laplace operators over 7 geometric shapes. Color codes correspond to the local components of the second eigenfunction (in the continuous case) and eigenvector (in the graph case). The graph cases have 250 nodes randomly distributed in the shapes.  $\lambda_2$  is averaged over 64 runs with different seeds and shown with its standard deviation. To easily compare the continuous and graph cases, values of the second eigenvalue  $\lambda_2$  are normalized to have a value of 1 for disk arenas. Images are re-scaled for better visualization, computations are performed with arenas of the same surface. **c:** Workflow of the spectral swarm algorithm (see Supplementary Sec. 3 for full details) for decentralized shape classification. Kilobots will diffuse their internal state, resulting in a partition. Then the convergence rate of the diffusion process is used to estimate  $\lambda_2$ . The robots reach a consensus on this value across the entire swarm and show a different LED color depending on its value, resulting in a classification decision.

easily formulating physical and geometrical laws that are the same across different frames of references.

The Laplacian is also a central concept in the fields of differential geometry and spectral shape analysis [24,25]. Mark Kac’s 1966 paper [26], “Can One Hear the Shape of a Drum?” introduces a pivotal question in mathematical physics, asking whether the shape of a drum can be deduced from its sound. The vibrational frequencies of a drumhead depend on its shape, and through the Helmholtz equation correspond to the eigenvalues of the Laplacian in the space [27].

The eigenspectrum of the Laplacian can be used as a signature that encode crucial geometrical and topological properties about a shape: eigenvalues inform on the presence of bottlenecks [25] and can serve as a (often but not always unique) spectral fingerprints of a shape [28–30]. Laplacian eigenvectors are natural indicators of how a shape can be partitioned into parts (“nodal domains”) that are internally cohesive but distinct from each other [31, 32]. The Laplacian presents thus a promising yet unexplored tool for swarm robotics to address geometrical problems only solved currently by biological swarms (Supplementary Sec. 1).

Here, we propose to develop spectral methods for swarm robotics settings, bringing together the local information of each robot to reach a collective consensus and precipitate collective decision-making. At a microscopic level, the Laplacian can be fully computed: it can be discretized with arbitrarily precision through a random graph embedded in the space [28]. Specifically, methods of spectral shape analysis extract global details about a shape (*i.e.*, the macroscopic organization of a swarm) by examining the local diffusion of information within the graph of linked nodes (*i.e.*, the graph of communication channels between robots). The spectrum of eigenvalues of the Laplacian is utilized to study and identify the graph shape, corresponding to the spatial distribution of a robot swarm in our case.

## Spectral swarm robotics

We introduce **spectral swarm robotics** - adapting spectral approaches to be computed in a distributed way on robot swarms (Supplementary Sec. 1). The spectral signature of the Laplacian suggests a way for a robotic swarm to sense the geometry of its arena with diffusion (Fig. 1a, Supplementary Sec. 2). The diffusion of information within a certain shape is strongly influenced by its geometry.

Therefore, robots can learn the geometry of their arena by passing information to each other to emulate the Laplacian operator: the robot swarm simulates heat diffusion as each robot adjusts its internal states to the average temperature (internal state) of its neighbors. By looking at **how quickly** the swarm gets to a stable state (*i.e.*, by locally measuring the relaxation time to the equilibrium), we can reliably estimate the eigenvalues locally. As a result, each robot can determine the spatial configuration of the entire

swarm, exploiting the fact that some shapes are more conducive to diffusion than others (*e.g.*, heat can diffuse uniformly across two dimensions in a disk, but it is predominantly confined to one dimension in an elongated ellipse). Robots then move to change their neighbors and repeat these measurements several times to reduce variance and eventually converge to an estimated eigenvalue. While, for a finite swarm of robots, this average value would not necessarily converge to the true value of the continuous case, it can be used as a fingerprint that helps the robot to sense its surroundings.

In particular, the second smallest eigenvalue ( $\lambda_2$ ) of the Laplacian, also known as “algebraic connectivity” or “Fiedler value” [33] influences the swarm’s connectivity and synchronization speed<sup>1</sup>, and is often used as a fingerprint by the spectral shape analysis community [28,34](Supplementary Sec. 2.3).

Figure 1b shows for seven different arenas the second smallest eigenfunction  $f_2$  or eigenvector  $v_2$  of the continuous and graph Laplace operators as well as the corresponding eigenvalue  $\lambda_2$ . The graph eigenvector  $v_2$  approximates well its continuous counterpart  $f_2$  and, once normalized, the values of  $\lambda_2$  are very similar in both cases.  $\lambda_2$  is higher in shapes with higher connectivity, like the disk, and decreases with the presence of nodes with lower connectivity acting as information bottlenecks.

Our method enables decentralized classification of robotic arena shapes, detailed in Fig. 1c, Supplementary Fig. 1, Methods and Supplementary Sec. 2 & 3. The process begins with  $N$  robots randomly walking in the arena until they are evenly distributed. They then identify their neighbors, defined as robots within communication range  $\sigma$ , forming a graph that captures the swarm’s connectivity and the arena’s geometry. Next, the robots engage in information diffusion rounds, updating their states  $s_i^n$  ( $i$ -th individual at the  $n$  time step) based on the Laplacian operator’s diffusion law. This reveals global spectral properties and allows each robot to locally estimate  $\lambda_2$ , corresponding to diffusion rates.

Finally, robots average their  $\lambda_2$  estimates, leading to a consensus on the arena’s shape, indicated by specific LED colors for different shapes (*e.g.*, gold for triangle, violet for annulus, cyan for disk).

We tested this method using physical simulations with up to 550 robots and classified seven geometric shapes. Figure 2 showcases the behavior of the algorithm in simulation. It confirms the evolution of the sign and magnitude of state  $s^n$  during the diffusion stage. The swarm starts from a highly heterogeneous  $s^n$ , where typically half of the neighbors of each robot  $i$  have a different internal state of the opposite sign of  $s_i^0$ . With time, diffusion smooths out local disparities in sign: robots homogenize their individual states with their neighbors and the swarm eventually partitions into two stable nodal domains of opposite signs. The partition is reproducible between runs for shapes

<sup>1</sup>The first smallest eigenvalue  $\lambda_1$  is always equal to 0.

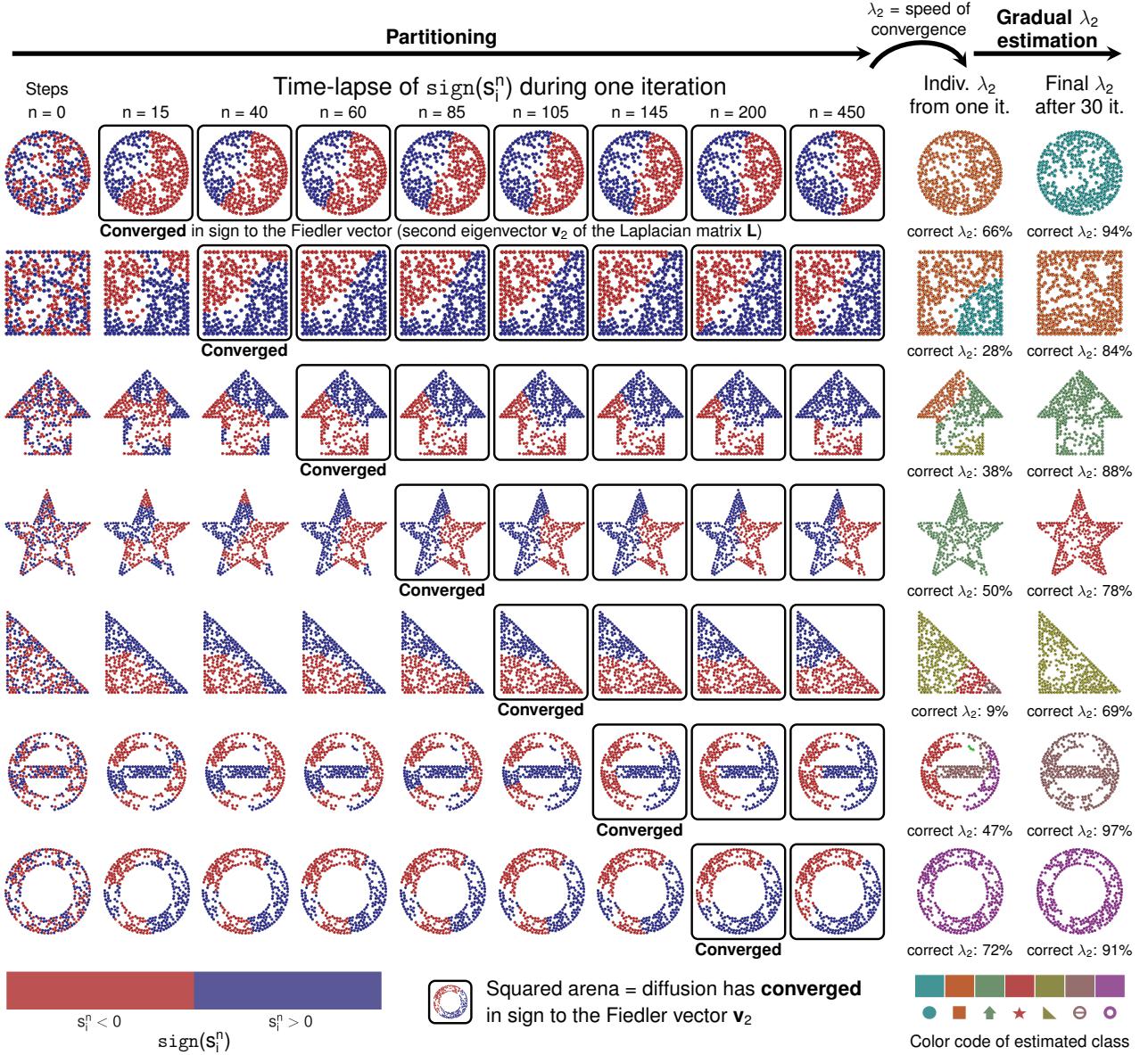
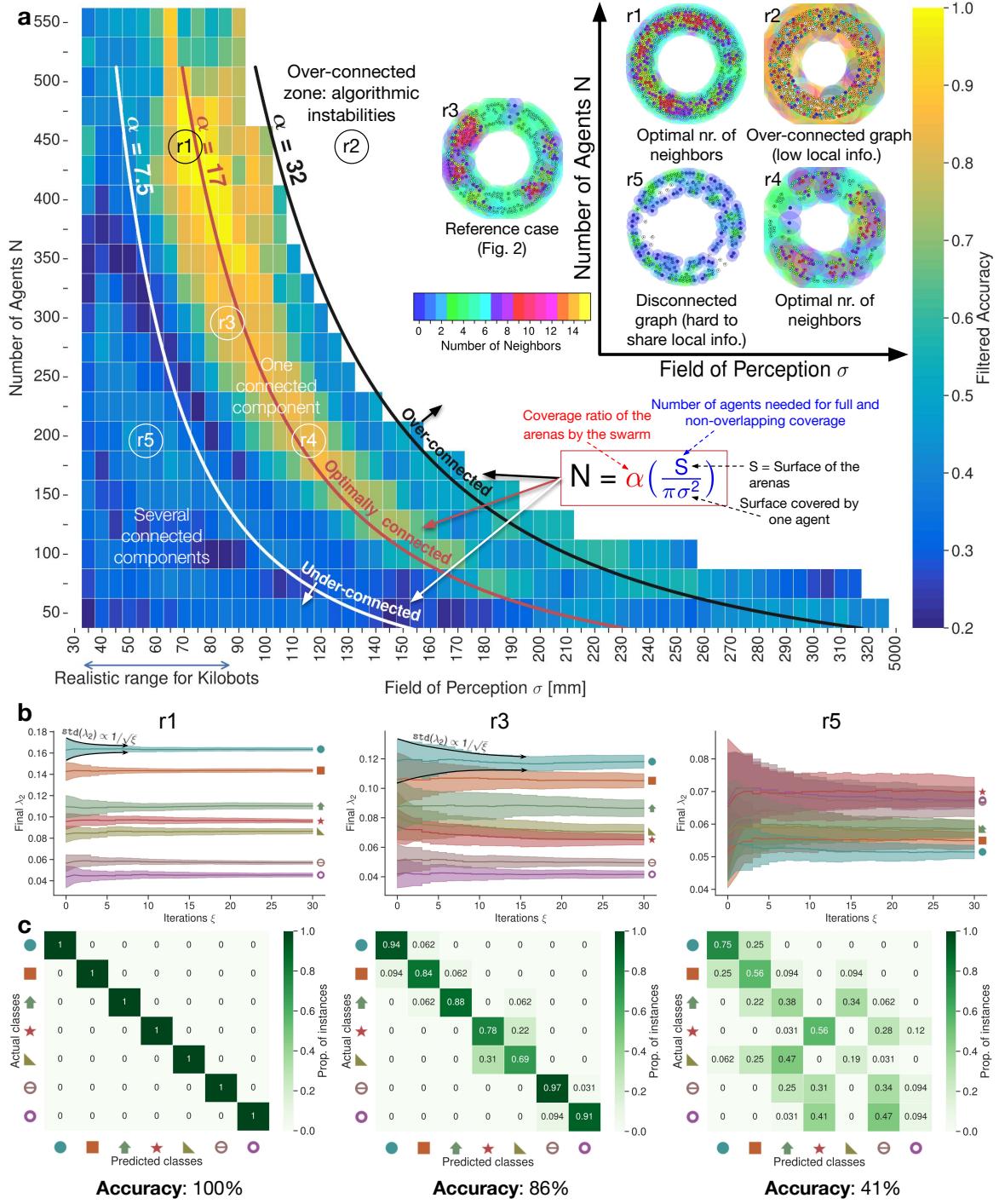


Figure 2: **Representative time-lapses of the behavior of the proposed algorithm applied to 7 arenas in simulations.** **Left:** diffusion stage, resulting in a partitioning of the shapes (color code: sign of internal state  $s_i^n$ ). The second eigenvalue  $\lambda_2$  of the Laplacian matrix of the communication graph between robots corresponds to the convergence rate of diffusion: the information diffusion converges at different rates depending on the topology of the arena. **Right:** in turn, robots use the observed convergence rate to compute a local estimate of this value iteratively refined at each iteration of the algorithm. After 30 iterations, the estimations have converged and formed a consensus. Based on the value of  $\lambda_2$ , the robots display a color code corresponding to the detected arena shape. Images are re-scaled for visualization; during simulations, arenas surface are normalized to be  $500000\text{mm}^2$ , with  $\tau = 1/15$  seconds corresponding to the amount of time between two steps of diffusion. We use the same parameters as regime r3 of Fig. 3, with  $N = 300$  robots and  $\sigma = 85\text{mm}$ . The percentages of correct  $\lambda_2$  are computed over 64 runs.



**Figure 3: Classifying regimes of the system computed in simulations over 64 runs.** **a:** Influence of the number of robots  $N$  and of the field of perception  $\sigma$  on the accuracy. We filter the accuracy score to only show results where the simulated diffusion sessions converge: empty bins correspond to results with more than 50% of cases with algorithmic instabilities from over-connected graphs and numerical errors (*i.e.*, without exponential decay of  $s_i^n$ ). We identify five representative regimes: r1 to r5. Examples of regime r3 behavior are shown in Fig. 2. The best-performing results are found on the hyperbola  $N = 17S/(\pi\sigma^2)$ . Top-right corner: examples of robot distribution in the annulus arena; colored disks around each robot represent their field of perception  $\sigma$ . **b:** Evolution of estimated  $\lambda_2$  values over 30 iterations of the algorithm, for each arena. Line colors correspond to the color code in Fig. 2. Only r1, r3, r5 are shown (r4 is similar to r3, and r2 is an over-connected case). The order of  $\lambda_2$  values for each arena is generally conserved across regimes, except in r5, where the order is inverted compared to r1, r3, r4. **c:** Confusion matrices and accuracy scores for regimes r1, r3, r5, showing the performance of the system to accurately classify the shapes. Each row represents the instances of an actual class (*i.e.*, the shapes the robots are situated in) while each column represents the instances of a predicted class (*i.e.*, the shapes the robots report using LED color). The diagonal elements of the matrix (from top-left to bottom-right) represent the number of points for which the predicted class is equal to the true class, *i.e.*, the correct predictions; perfect accuracy corresponds to the identity matrix. Off-diagonal elements in the matrix describe the misclassifications, *i.e.*, the instances where the model predicted a class that differs from the true class.

without symmetries (*i.e.*, arrow), because the eigenvalues are not expected to be degenerate (*i.e.*,  $\lambda_2 < \lambda_3$ , and the eigenspace associated to  $\lambda_2$  is of dimension 1). As for the magnitude, the absolute value of each  $s_i$  decays exponentially after a sufficient time, although the time needed to fall into this regime depends on the spatial position of the robots. The signs of the  $s_i$  of robots located near the interface are expected to resolve later than that of robots far from the interface, which quickly converges. After the diffusion stage, each robot estimates the diffusion convergence rate of (*i.e.*, slope of  $\log(|s_i^n|)$ ) to compute a local estimation of  $\lambda_2$ . Robots will then predict the shape of the arena they are in depending on the value of  $\lambda_2$ . Robots estimation can vary depending on initial conditions and so will be different at each iteration, considering robots move and change their neighbors at the beginning of each iteration. After 30 iterations, a consensus will emerge and the probability of the swarm to correctly identify the shape is strongly increased (*e.g.*, from 66% to 94% for the disk). The algorithm accuracy measures how often the agents can correctly classify in a distributed way the shape they are situated in. Supplementary Fig. 3 & 5 show in detail the improvement in accuracy when going from one to 30 iterations.

## Universal scaling law

We explored how shape classification accuracy was influenced by two parameters that are important for experimental design: the number of robots  $N$  and the field of perception  $\sigma$ . Associated results are found in Fig. 3a. Scaling laws properties give rise to five different regimes depending on chosen parameters: **r1** to **r5**, that are separated through three hyperbolae  $N = \alpha S/(\pi\sigma^2)$  with  $S$  the surface of the arenas,  $\pi\sigma^2$  the surface occupied by one agent,  $S/(\pi\sigma^2)$  is the number of agents needed for full and non-overlapping coverage of the arenas by the agents, and  $\alpha$  is the coverage ratio of the arena by the swarm. The constant  $\alpha$  is an intrinsic factor different for each hyperbola.

We find that the best pairs of parameters  $(N, \sigma)$  fall along the hyperbola  $\alpha = 17$  (**r1**, **r3**, **r4**) shown in red in Fig. 3a: **r1** has a relatively high number of agents, with low  $\sigma$ , **r4** has a relatively low number of agents but with high  $\sigma$ , and **r3** is a middle-ground between **r1** and **r4**. As suggested by [35], highest values of field of perception is not necessarily correlated with higher accuracy. Typically, each robot is surrounded by an empty area  $S/N$  (where  $S$  is the surface of the arena). This gives a typical lengthscale  $l = \sqrt{S/N}$  for the distance between robots. Now the field of perception  $\sigma$  must be finely tuned to accommodate this distance. If  $\sigma$  is several times larger than  $l$ , then the details and granularity of the shape are lost. If  $\sigma < l$  then the connectivity of the graph drops and information is not spread efficiently across the whole shape.

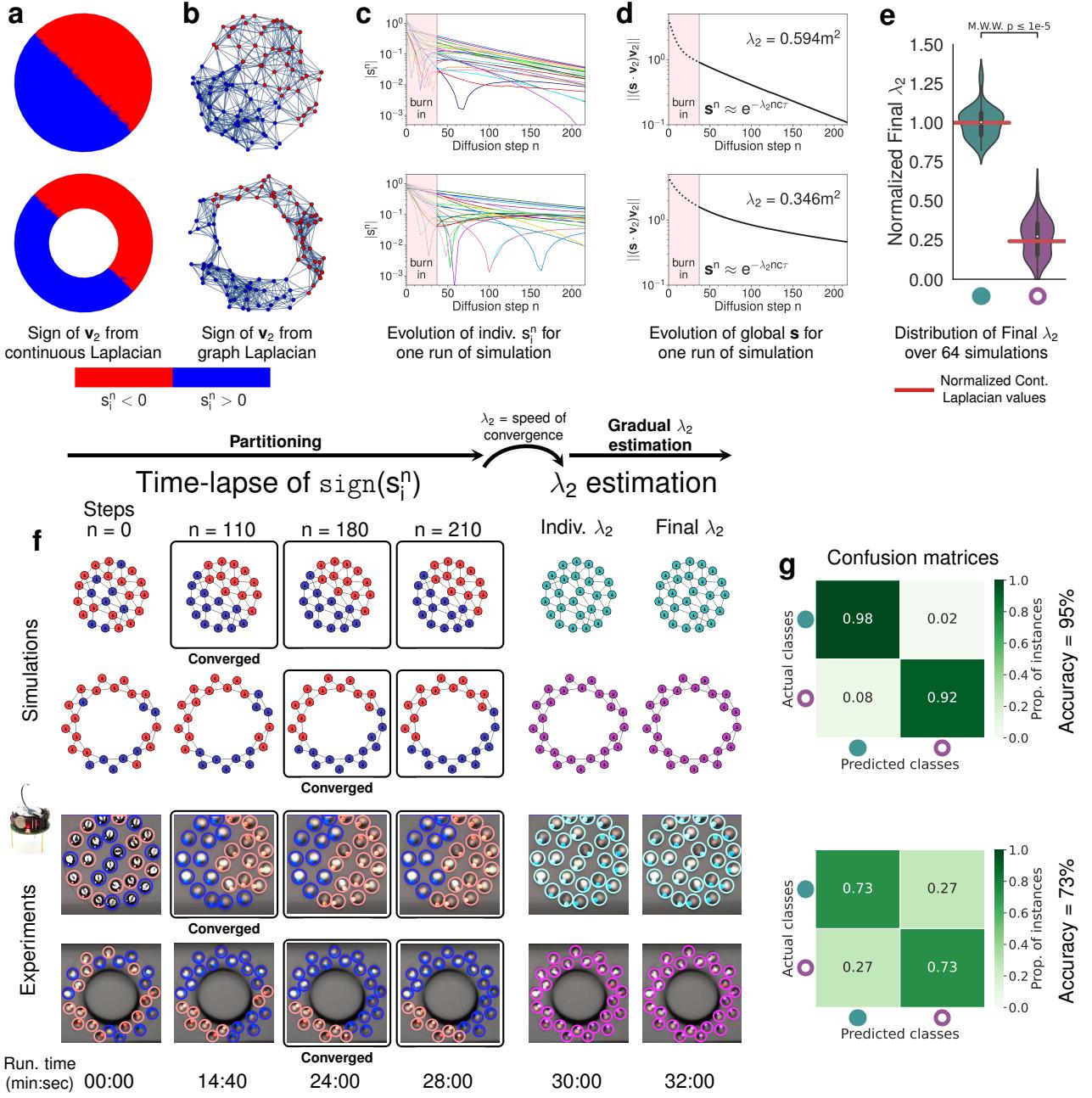
The lower boundary of the optimal conditions region is the hyperbola  $\alpha = 7.5$  (white in Fig. 3a) that marks the transition between graphs with several connected

components (below the curve) to only one component (above the curve, normal behavior of the algorithm). The mean number of connected graph components is presented in Supplementary Fig. 6 (bottom). Below the curve, agents can still propagate local information from components to components when they move at the beginning of each iteration and change their neighbors. The swarm will converge to inexact  $\lambda_2$  estimates but that may still be different depending on the shape of the arena, as seen in regime **r5** where the algorithm can distinguish a disk from an annulus, but is not accurate enough to identify 7 different shapes. Results directly on the curve correspond to a “low-accuracy valley” at the bifurcation between the two dynamics.

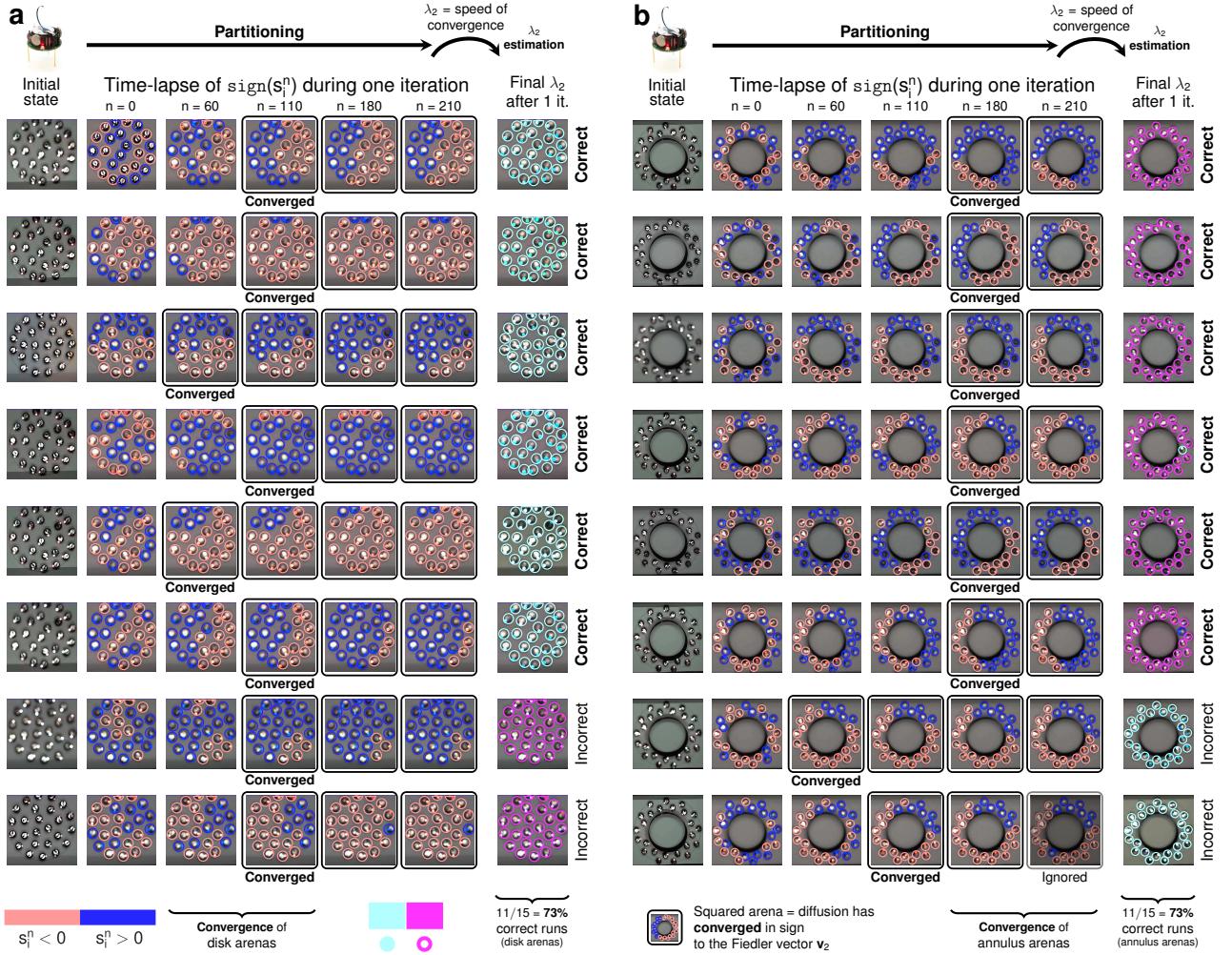
The upper boundary of the optimal conditions region is the hyperbola  $\alpha = 32$  (black in Fig. 3a) that separates convergent results (below the curve) from over-connected results with algorithmic instabilities (regime **r2**). The former are cases where the diffusion process behaves as expected, with an exponential decay of  $|s_i^n|$  on all agents. The latter cases have over-connected graphs that renders  $\lambda_2$  estimation difficult for two reasons: 1) all arenas will correspond to relatively similar communication graph topologies and 2) too much neighbors broadcast information, resulting in numerical instabilities. Supplementary Fig. 6 (Top) shows the average number of neighbors of agents and that having more than 25 neighbors does not allow the algorithm to converge.

We estimated how many runs were necessary for a swarm to discriminate the shapes (Fig. 3b). For regimes **r1** and **r3**, most shapes are readily separated after a few iterations (*e.g.*, the square and the disk are easily separated from the rest). Following the central limit theorem, the standard deviation of  $\lambda_2$  estimates across runs converges as  $x^{-k}$  with  $x$  the amount of iterations, and  $0 < k \leq 1$  a variable that depends on the case. It takes about 20 iterations to separate shapes with similar  $\lambda_2$ , such as the square and the disk. The confusion matrices in Fig. 3c confirm the quality of classification: **r1** and **r3** have respectively 100% (perfect score) and 86% of accurately classified shapes. Interestingly, the magnitude of  $\lambda_2$  is roughly in line with the expected connectivity of the shape. A convex shapes like a disk diffuses information roughly ten times more quickly than a non-convex shape like the annulus.

The way the agents move during their random walk phase influences the diffusion dynamics (Supplementary Sec. 5.2). If the agents are evenly distributed within their arena, and only form one connected graph component they will be able to capture all local features of the shape and communicate it to the entire swarm – in this case, staying immobile and avoiding the random walk phase would not have a detrimental effect on the accuracy of the algorithm. Inversely, if the agents are aggregated only in some parts of the arena, and form several disconnect graph components (*e.g.*, regime **r5**), it is necessary for them to disperse at each iteration to change their neighbors and broadcast their states from graph component to graph component.



**Figure 4: Comparison between theoretical, simulated, and experimental results** on a two classes classification setting with two shapes (disk and annulus) using one iteration of the algorithm. **a,b:** Examples of arena partition in both the continuous and discrete (graph) cases. Colored zones and nodes correspond to the sign of the local component of the Fiedler vector  $\mathbf{v}_2$  of the Laplacian (red: negative, blue: positive). In both cases, global information (*i.e.*, access to the full Laplacian matrix) is used to compute  $\mathbf{v}_2$ . **c:** Evolution of the internal state  $|s_i^n|$  of each robot  $i$  during a diffusion session (*i.e.*, using only local information) in simulations (64 runs). The slope of each respective curve is used to estimate the local value of  $\lambda_2$  on each robot  $i$ . **d:** Computation of  $\lambda_2$  using global information directly from the entire graph in b. **e:** Distribution of  $\lambda_2$  values estimated on each robot (violin plots) in simulations, compared to the value computed using global information from the continuous Laplacian (red lines). Values are normalized so that the mean of disk values is 1.0. **f:** Time-lapses of the algorithm, in simulations and in experiments, with 25 robots. Experimental photos are blurred to ease the visualization of LED colors. Left: diffusion, resulting in partitioning (the color of each robot corresponds to a local component of the Fiedler vector  $\mathbf{v}_2$ ). Right: individual  $\lambda_2$  estimation on each robot from the convergence rate of diffusion, and consensus (collective averaging) over the entire swarm. Robots show a LED color-code according to the value of  $\lambda_2$ : cyan when a disk is detected, violet for an annulus. Images are re-scaled for better visualization; both simulations and experiments have arenas surface approximately equal to  $70000\text{mm}^2$ . **g:** Confusion matrices and accuracy scores of simulations (over 64 runs per arena) and experiments (over 15 runs per arena).



Extended Data Figure 5: **Time-lapses of the experiments**, with 25 robots, respectively for the disk (**a**) and annulus (**b**) arenas. Experiments use the same parameters as simulations and experiments of Fig. 4. Only the results of 16 runs are shown (over 30 considered runs). Time-lapses of all runs are presented in Supplementary movie 1. Experimental photos are blurred to ease the visualization of LED colors (except left column). Images are re-scaled for better visualization; experiments have arenas surface approximately equal to  $70000\text{mm}^2$ . **Left:** Initial state of the robots before the start of experiments. **Middle:** diffusion, resulting in partitioning (the color of each robot corresponds to a local component of the Fiedler vector  $v_2$ ). In the studied configuration, robots typically converge in the domain  $n \in [60, 110]$  for the disk arenas and in the domain  $n \in [180, 210]$  for the annulus arenas. In rare cases, the robots may still change state after convergence, due to computation errors, or message transmission errors – in such case, the partitioning the robots converged into may be broken (*e.g.*, last experiment on the annulus arena). The spectral swarm robotics algorithm will detect and ignore divergent cases after convergence (cf details in Supplementary Sec. 3). **Right:** individual  $\lambda_2$  estimation on each robot from the convergence rate of diffusion, and consensus (collective averaging) over the entire swarm. Robots show a LED color-code according to the value of  $\lambda_2$ : cyan when a disk is detected, violet for an annulus. The robots detected the correct shape on 22 runs over 30 runs (15 runs on each arena), which translates into an accuracy score of  $22/30 = 73\%$ .

## Experimental validation

We successfully validated experimentally our algorithm on a swarm of 25 Kilobots robots (Fig. 4 and Fig. 5) with the experimental setup described in Supplementary Sec. 4 (videos listed in Supplementary Sec. 5.3). We focus only on two arenas: the disk and the annulus. Those shapes were selected because they correspond to the  $\lambda_2$  values that are the furthest apart (Fig. 3b). To increase the difficulty of our experimental validation, and showcase the robustness of our algorithm, we select particularly harsh conditions in term of number of agents and number of iterations. Indeed, simulation results (Supplementary Fig. 8) show that on 2-arenas classification tasks only one iteration of the algorithm was sufficient, even for relatively small number of robots. In the experimental setup, robots start well-distributed in the arena to study the performance of the diffusion process (Supplementary Sec. 5.2). The actual field of perception varies from robot to robot and depends on their current position in the arena (environmental effects), but a rough estimate places it around 85mm.

The cases that use only local information (simulations and experiments in Fig. 4f) show similar behavior compared to theoretical results computed using global information (with access to the full Laplacian matrix, either in the continuous or discrete case: Fig. 4a,b). Notably, the diffusion processes in all four cases results in a graph partition in two equal zones, suggesting that the swarm has correctly converged along the second eigenvector. Figures 4c,d show instances of time-evolution of the internal state of robots during a diffusion process, respectively in a 25-agent simulation (c) and computed directly on the full graph Laplacian (d): both cases show an exponential decay, of which the slope (*i.e.*, converge rate) approximates  $\lambda_2$ . In both simulations and experiments, the diffusion processes tend to converge faster in disk arenas than in annulus arenas (Figs. 4g and 5a,b).

We consider that the diffusion process has converged when two distinct nodal zones emerge. In most cases, there will not be any evolution afterwards. However, the diffusion process can sometimes diverge (*i.e.*, partition no longer encompasses only 2 zones) after reaching convergence, due to the propagation of computation errors when the local states are close to 0. After the diffusion has converged, the algorithm will ignore any possible subsequent divergence in its estimation of  $\lambda_2$ . Communication errors and local asymmetries in the Laplacian matrix (one-way communication between an agent and its neighbor) often render the partition uneven (Fig. 5), despite algorithmic methods to reduce communication errors (Supplementary Secs. 3.2 and 4.2.1).

Our approach is robust to these issues for shape classification purposes: we obtain accuracy scores of 73% over 30 experiments and show that the behavior of the algorithm in experiments matches the one seen in simulations (Fig. 4f,g, 95% of accuracy over 64 runs of simulations) and in theoretical analyses (Fig. 4e).

## Conclusions

We numerically showed that a swarm of robots controlled by the spectral swarm robotics algorithm could “hear” the shape of its arena in spite of the robots having no physical sensors to probe their environment. This was validated both in simulations and experimentally.

Spectral analysis is robust and scalable and, in theory, is not susceptible to error done by a single robot. In the case of the naive gradient algorithm (which computes the distance of each robot to a reference robot), a single and transient error by a single robot could be amplified and propagated in the swarm, eventually making the whole computation unstable [36,37]. These error cascades severely limit the scalability of the algorithm, as punctual errors increase with the size of the swarm. In contrast, we did not observe catastrophic failures of computations with our approach (except for over-connected cases), which we attribute to the inherent regularization brought by diffusion, an operator known to smooth local noise and errors. If a robot incorrectly computes its internal states, the error is quickly corrected by the diffusion of information from the neighbors of the faulty robot [7]. This self-repair process by diffusion allows the algorithm to scale to much larger swarms.

While it is theoretically possible for different shapes to share the same spectrum, only a few such instances have been discovered in the field of spectral shape analysis, and they tend to be anomalies with non-smooth, non-convex boundaries [38]. It is likely that this may not even apply to our approach, as robot motility inside arenas can slightly bias our eigenvalue estimations: two shapes with the same geometric spectrum may not have exactly the same spectrum estimated by the swarm.

Spectral analysis by robotic swarm could be furthered in several directions. We implicitly used a Neumann boundary conditions (diffusion does not cross the wall), but with robots that can sense walls, we could use Dirichlet boundary conditions to estimate the distances of the robots to the nearest walls (to do that one could clamp the state of robots near the walls, and initializing other robots to 0 - the rate at which they converge to 1 being a proxy for their distance to the boundaries).

With more processing power, the robots could also estimate eigenvalues beyond  $\lambda_2$  and get a finer view of the shape’s spectrum, so as to discriminate a larger number of shapes with fewer iterations. The method also naturally extends to the 3rd dimension, which could find use in swarms of flying robots.

Experimental validation, currently classifying only the disk and annulus arenas, could be extended to more shapes: this would require several iterations of the algorithm with robots moving at the start of each iteration to change their neighbors, making actual experimental tests on Kilobots to be difficult. Kilobots in closed settings tend to aggregate towards the walls, rendering the distribution of Kilobots in a disk eventually converge to the same distribution as in an annulus.

lus. This could be alleviated by using a different kind of robots, either embarked with a distinct morphology preventing wall bottlenecks [39], or with a more complex dispersion scheme. For now, the algorithm requires the robots to be immobile during the diffusion stage, and to move during the random-walk stage. Our approach could be extended so that robots are always moving, with no stop. However, this would involve using a different information propagation mechanism than chemical diffusion: possibly taking inspiration from fluid dynamics or on the synchronization of moving oscillators (*e.g.*, in nature: firefly synchronized flashing, while constantly changing their neighbors), or by adapting the emerging field of Graph Fourier Transforms [40] in signal graph processing to dynamical cases.

Spectral swarm robotics could also be extended to other tasks: for instance, estimate the number of robots in a swarm [37], identify the shape of objects in an experimental arena, compute geometrical properties of the environment from spectral analysis (number of objects, curvature of walls, etc).

In summary, we introduced a way for a swarm of robots to compute spectral statistics of their environment in a distributed way, and providing effective methods to close the gap between local interaction and global dynamics to precipitate collective decision making. This problem is critical because it will emerge in any kind of real-world application of swarm robotics: robots deployed in the field will always have to adjust their behavior depending on the geometry of their environment. Our approach addresses this problem allowing the swarm to recognize environmental features and adapt to gain in efficiency.

## Methods

### Swarm robotics and Kilobots

We use Kilobots robots to test our approach (Fig. 1a, Supplementary Fig. 2). In the past ten years, Kilobots [1, 2] have become a standard to prototype robots swarms, and exemplify the characteristics of robots in swarms. They have been used for numerous tasks including self-assembling into predetermined shapes [2, 7], phototaxis [41], creating moving and deforming soft-bodied kilobots aggregates [42], morphological computation and decentralized learning [39], contour detection [43].

On its own, a Kilobot is a centimeter-sized robot that (rudimentarily) moves and communicates (imperfectly) with its neighbors. These robots ignore their position or velocity because they are devoid of position or velocity sensors. As a result, they haphazardly navigate their environments, similarly to a random walk. Kilobots are also “blind” to their environment. They only sense ambient light and lack the panoply of proximity sensors (camera, LIDAR, radar, ultrasound, capacitive or force sensors) embarked on larger robots required to detect obstacles, follow walls, or orient themselves.

### Spectral swarm robotics algorithm

Here is a theoretical description of our proposed algorithm of spectral swarm robotics for shape analysis, as illustrated in Fig. 1c. More details and a table of variables can be found in Supplementary Secs. 2 & 3.

In the seeding phase, an even number  $N$  of robots are seeded in the arena and walk randomly, long enough to spread homogeneously. After 25 minutes (Supplementary Table 5), the robots stop and initiate several iterations of the algorithm. They enter a discovery phase to register their neighbors. Two robots are defined as neighbors if they can communicate with each other, that is, they are within each other’s field of perception  $\sigma$ . For typical robotic experiments,  $\sigma$  is several times the extent of a single robot, which means that a robot only sees a fraction of the swarm at any time (which may comprises tens or hundreds of robots).

The neighborhood relationship immediately defines a graph  $\mathcal{G}$  on the robots which captures the connectivity of the swarm and the geometry of the arena, with an adjacency matrix  $A$  defined such that  $A_{i,j} = 1$  if the robots  $i$  and  $j$  are neighbors and 0 otherwise (by convention a robot is not its own neighbor). The robots then engage in rounds of diffusion of information.

Robots initially set their internal state  $s_i^0$  to  $-1$  or  $1$  (a state chosen to have a zero mean, which is conserved during diffusion) and, at each round of diffusion, they update their internal state according to the law of diffusion on a graph [44, 45] (Supplementary Sec. 2.1):

$$s_i^{n+1} = s_i^n + c\tau \sum_{j=1}^N A_{ij}(s_j^n - s_i^n) \quad (1)$$

with  $n$  the current time step.

which can be rewritten in a matrix form:

$$\mathbf{s}^{n+1} = \mathbf{s}^n - c\tau \mathbf{L}\mathbf{s}^n \quad (2)$$

where  $\mathbf{L}$  is the Laplacian matrix of the graph,  $c$  is the diffusion rate across edges of the graph and  $\tau$  is time between two steps of diffusion. Note that  $c$  includes a notion of distances between nodes; here we assume that all edges have the same weight (distance is 1, *i.e.*,  $\mathcal{G}$  is unweighted). On the diagonal,  $L_{i,i}$  is the number of neighbors detected by the robot  $i$ , and off the diagonal,  $L_{i,j} = -1$  if  $i$  and  $j$  are neighbors, and  $L_{i,j} = 0$  otherwise.

Assuming  $c\tau$  is small, this time evolution can be approximated analytically by taking the exponential of the Laplacian matrix.

$$\mathbf{s}^n = e^{-nc\tau \mathbf{L}} \mathbf{s}^0 \quad (3)$$

Since the Laplacian matrix  $\mathbf{L}$  is symmetric and positive, its eigenvalues  $\lambda_i$  are real and positive, with  $\lambda_1 = 0 \leq \lambda_2 \leq \lambda_3 \leq \dots$ . By the spectral theorem, the time evolution of  $\mathbf{s}^n$  is given by its projection on the eigenvectors  $\mathbf{v}_i$  of  $\mathbf{L}$ :

$$\mathbf{s}^n = \sum_{i=1}^k e^{-\lambda_i nc\tau} (\mathbf{s}^0 \cdot \mathbf{v}_i) \mathbf{v}_i \quad (4)$$

where  $\mathbf{s}^0 \cdot \mathbf{v}_i$  is the projection of the initial state  $\mathbf{s}^0$  on  $\mathbf{v}_i$ . The internal state  $s_i^n$  of each robot converges to the mean of the initial  $\mathbf{s}^0$ , which corresponds to the state where the information is fully mixed and the robots share the same internal state. Since by design the mean was set to be 0, after some time,  $\mathbf{s}^n$  decays exponentially along  $\mathbf{v}_2$ :

$$\mathbf{s}^n = e^{-\lambda_2 n c \tau} (\mathbf{s}^0 \cdot \mathbf{v}_2) \mathbf{v}_2 \quad (5)$$

This equation is the crux of our strategy to identify the shape of the robot arena. The shape controls the rate at which information diffuses in the robot graph through  $\lambda_2$ , which is known in the literature as the algebraic connectivity of the neighborhood graph [33, 46]. The magnitude of  $\lambda_2$  reflects the connectivity of the robot graph: the more paths there are between any two robots, the larger  $\lambda_2$  tends to be, and the faster the robots synchronize their internal state and converge to their collective mean. So a swarm of robots dispersed in a disk is expected to have a larger connectivity than a swarm dispersed in a annulus with the same surface, and so information is expected to spread faster in the former case. Of note, if the robot graph is not connected (there exists at least one pair of robots without a path between them in the graph), then  $\lambda_2 = 0$  and the robotic swarm is partitioned into distinct clusters that synchronize internally, but not globally. This can happen even for connected shapes when the field of perception or the number of robots are too small, which opens holes in the neighborhood graph and prevents information from fully flowing in the swarm. The associated eigenvector  $\mathbf{v}_2$  also plays an important role: it is known as the Fiedler vector and the sign of its coordinates define a partition of the graph, as seen in Fig. 1b where color codes correspond to local components of the second eigenvector.

From Eq. 5, each robot individually estimates  $\lambda_2$  by linearly fitting the exponential decay rate – after a suitable time – of its internal variable  $s_i^n$  to 0. Each robot thus gets a slightly different estimate of  $\lambda_2$ , which is then averaged across the swarm after each iterations to obtain a more accurate value. Still, this estimate remains noisy, because the number of robots is finite. To refine this estimate, after each iteration, the robot randomly walk in the shape, stop, and start again the process of estimating  $\lambda_2$ . By taking the cumulative average of  $\lambda_2$  since the beginning, the swarm refines and converges to an estimate of  $\lambda_2$ . For sets of shapes that are not pathological (*i.e.*, with distinct  $\lambda_2$ ), it is in principle possible to discriminate them provided enough iterations, since our Monte Carlo estimate converges to  $\lambda_2$  as  $1/\sqrt{N}$ .

## Acknowledgements

We thank Yoones Mirhosseini and Jérémie Fersula for their advice on using Kilobots.

## Author contributions

A.J.G, L.C., and N.B. conceptualized, planned, and supervised the project. A.J.G, N.L-D., N.A-K., O.M. provided the theoretical background and design, L.C. and A.J.G. designed the algorithm. L.C. wrote the code for robotic controllers and simulation, and most of analysis and plotting scripts. A.J.G. and N.L-D. performed the theoretical analysis. L.C. computed and analyzed all simulations. N.B, A.L. and M.I. conducted all experiments with robots. L.C., N.L-D. and N.B. assembled the figures. L.C., A.J.G, N.B., N.L-D. and N.A-K. wrote the manuscript. L.C., A.G. and N.B. wrote the supplementary information and designed the supplementary videos. N.A-K. and N.B. acquired the funding.

## Funding

This work was supported by the MSR project funded by the Agence Nationale pour la Recherche under Grant No ANR-18-CE33-0006 and JSPS Fostering Joint International Research (B) Grant No JP19KK0261. NLD acknowledges support from European Research Council (ERC) under the European’s Union Horizon 2020 programme (grant No 770940)

## Competing interests

The authors declare no competing interests.

## Additional information

**Supplementary Information** is available for this paper.

**Correspondence and requests for materials** should be addressed to Leo Cazenille, Nicolas Bredeche or Anthony J. Genot.

# Hearing the shape of an arena with spectral swarm robotics

## Supplementary Information

Leo Cazenille<sup>1</sup> , Nicolas Lobato-Dauzier<sup>2,3</sup> , Alessia Loi<sup>4</sup> , Mika Ito<sup>5</sup>, Olivier Marchal<sup>6</sup> , Nathanael Aubert-Kato<sup>5</sup> , Nicolas Bredeche<sup>4,\*</sup> , Anthony J. Genot<sup>2,\*</sup> 

<sup>1</sup>Université Paris Cité, CNRS, LIED UMR 8236, F-75006 Paris, France – leo.cazenille@gmail.com

<sup>2</sup>LIMMS (IRL2820)/CNRS-IIS, University of Tokyo, Tokyo, Japan

<sup>3</sup>Sorbonne Université, CNRS, Institut de Biologie Paris-Seine, Laboratoire Jean Perrin, F-75005, Paris, France

<sup>4</sup>Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR, F-75005 Paris, France

<sup>5</sup>Ochanomizu University, Department of Information Sciences, Tokyo, Japan

<sup>6</sup>Université Jean Monnet Saint-Étienne, CNRS UMR 5208, Institut Camille Jordan, Institut Universitaire de France, F-42023 Saint-Etienne, France

## 1 Spectral shape analysis

Mark Kac’s famous 1966 paper, “Can One Hear the Shape of a Drum?” [26] presented a novel and fascinating question rooted in the field of mathematical physics: if you only heard the sound a drum makes, could you figure out what shape the drum is? In other terms, is it possible to identify the shape of a drumhead merely from the list of its vibrational frequencies (or eigenvalues), essentially ‘hearing’ its shape? This concept is an important question in the realm of spectral geometry. It challenges us to consider whether different drums (interpreted as 2D manifolds) could produce identical sounds, thereby sharing the same eigenvalue spectrum, or if the sound of a drum is as unique as its shape. While it has been established that instances of differing shapes producing identical spectra are possible, such cases are remarkably rare (and tend to be anomalies with non-smooth, non-convex boundaries) [38], emphasizing the intricate interplay between the auditory perception of an object and its geometric structure.

In the realm of network theory, the process of extracting a comprehensive graph structure has been investigated in temporal networks [47], which focus on whether and when interaction occurs between node pairs, considering the flow of data instead of its nature. It has also been applied to the field of graph signal processing that study the harmonics of a graph using so-called Graph Fourier Transforms [40]. A similar exploration was made in spectral graph theory [24, 25], but here the aim is to pull out global graph traits under the assumption of a static graph and a central observer overseeing all nodes. Specifically, methods of **spectral shape analysis** [26] extract global details about a shape (*i.e.*, the macroscopic organization of a swarm) by examining the local diffusion of information within the graph of linked nodes (*i.e.*, the graph of communication channels between robots). The spectrum of eigenvalues of the Laplacian is utilized to study and identify the graph’s shape (in our case, the spatial distribution of a robot swarm).

The Laplace–Beltrami operator (LBO) is a generalization of the Laplace operator to functions defined on submanifolds in Euclidean space and more generally on Riemannian manifolds. The recent surge of interest in the spectral analysis of the LBO has resulted in a considerable number of spectral shape signatures that have been successfully applied to a broad range of areas, including manifold learning [29], object recognition and deformable shape analysis [30, 48], medical imaging [49], and shape classification [28]. For instance, Shape-DNA [28] extracts a fingerprint of a shape by computing the first eigenvalues of the LBO. More generally, spectral analysis is widely used to study networks of distributed sensors (for example networks of antenna in 5G).

The LBO is ideal for shape analysis for several reasons: (1) it is intrinsic (invariant to isometric deformations [50]); (2) it is not tied to a specific shape representation and can be discretized on 2D manifolds, meshes, point clouds, etc [51]; (3) its eigenbasis is optimal for approximating functions with limited variation, and in many cases, only the first few eigenfunctions are needed for a satisfactory approximation [52].

However, a critical limitation for swarm robotics is that, while diffusion can be enacted on a microscopic scale, graph traits are derived by centrally combining information. This implies that individual nodes do *not* have access to macroscopic data. By using linear and non-linear operators for a distributed application of diffusion, we propose that the diffusion of information can serve as a bridge between the local scale of an individual robotic agent and the global scale of the swarm (Main Text Fig. 1a).

\*co-corresponding authors (equal contribution): genot@iis.u-tokyo.ac.jp nicolas.bredeche@sorbonne-universite.fr

To gain access to global graph traits at the individual node (or robot) level, the Laplacian operator can be deployed to instantiate diffusion [22]. Diffusion bridges the local scale of a robot to the global scale of the environment. Mathematically, diffusion is represented by the Laplacian, an ubiquitous linear operator measuring how much a quantity (like temperature or molecular concentration) changes compared to its surroundings. It governs many aspects of physics, ranging from the energy levels of quantum systems, the propagation of electromagnetic waves, mechanical vibrations, the diffusion of heat, or the viscosity of a fluid.

Even though the Laplacian is a local operator (*i.e.*, a function that only considers what's happening at a specific point and its immediate surroundings), it is fully determined by the geometry of the environment in which it functions: the eigenvalues and eigenfunction of the Laplacian are completely determined by boundary conditions. For instance, the shapes in Main Text Fig. 1b all results with a different value of  $\lambda_2$  (*i.e.*, second eigenvalue of the Laplacian), making this value usable as a fingerprint to identify these shapes.

Spectral analysis is typically only computed using global information, by having the entire Laplacian matrix available [51, 53]. Robots only have access to local information (their own internal state and information communicated by their neighbors). As such, computing global statistics on individual robots is challenging. Due to very slow communications (a few bytes per seconds) and very limited memory capabilities, it may be impossible, or at least extremely difficult, to regroup the entire Laplacian matrix on all robots of the swarm. Instead, we propose to estimate spectral statistics in a distributed way, and embodied in a swarm of robots, allowing the swarm to collectively “hear the shape of their arena”.

## 2 Theoretical analysis

Variable name	Unit	Description
$N$	-	Number of agents.
$H$	-	Mean number of neighbors of agents in a given configuration.
$\sigma$	[m]	Radius of the field of perception of the agents.
$S$	[m <sup>2</sup> ]	Arena surface.
$\mathcal{G}$	-	Graph of communication flows between agents
$\mathbf{A}$	-	Adjacency matrix of $\mathcal{G}$ . $A_{i,j} = 1$ if the agents $i$ and $j$ are neighbor, and 0 otherwise (by convention a robot is not its own neighbor).
$\mathbf{L}$	-	Laplacian matrix of $\mathcal{G}$ . On the diagonal, $L_{i,i}$ is the number of neighbors detected by the agent $i$ , and off the diagonal, $L_{i,j} = -1$ if $i$ and $j$ are neighbors, and $L_{i,j} = 0$ otherwise.
$s_i^n$	-	Internal state of agent $i$ at diffusion step $n$ ( <i>i.e.</i> , time $t = n \times \tau$ ). Used during the diffusion stages of the algorithm. Intended to have a zero mean at each step $n$ .
$\kappa$	[m <sup>2</sup> .s <sup>-1</sup> ]	Coefficient of information diffusion.
$c = 1s^{-1}$	[s <sup>-1</sup> ]	Information diffusion rate across edges of $\mathcal{G}$ .
$\tau$	[s]	Amount of time between two steps of diffusion.
$T$	-	Number of diffusion steps.
$B$	-	Number of diffusion steps to ignore at the beginning of a diffusion session when estimating $\lambda_2$ .
$C$	-	Number of collective averaging rounds.
$\lambda_i$	-	$i$ -th eigenvalue of the Laplacian $\mathbf{L}$ .
$\text{Indiv}\lambda_{ij}$	-	Local estimation of the $i$ -th eigenvalue of the Laplacian $\mathbf{L}$ by agent $j$ during one iteration.
$\text{Consensus}\lambda_{ij}$	-	Consensus estimation of the $i$ -th eigenvalue of the Laplacian $\mathbf{L}$ by agent $j$ during one iteration.
$\text{Final}\lambda_{ij}$	-	Final estimation of the $i$ -th eigenvalue of the Laplacian $\mathbf{L}$ by agent $j$ .
$\mathbf{v}_i$	-	$i$ -th eigenvector of the Laplacian $\mathbf{L}$ .
$l = \sqrt{S/N}$	[m]	Typical lengthscale of the distance between agents.
$P$	-	Number of information diffusion sessions computed in parallel.
$I$	-	Number of iterations.

Supplementary Table 1: Nomenclature of all considered variables

### 2.1 Information diffusion model

The diffusion model refers to a mathematical representation of the physical process of diffusion, in which a large number of small elements spread in the environment through random movement and collisions. Assuming the number of elements is large enough, one can define their distribution at a given spatial position  $(x, y)$  (for the two-dimensional case considered in this article) as a local concentration  $s(x, y)$ , with changes over time according to the following partial differential equation:

$$\frac{\partial s}{\partial t} = \kappa \nabla^2 s \quad (6)$$

where  $\kappa$  is the coefficient of diffusion, a macroscopic value specific to the elements and  $\nabla^2$  is a 2D Laplacian operator defined as:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad (7)$$

That notion of diffusion can be extended to work with graphs, where the concentrations of interest are stored in the nodes and diffuse through the edges. In this case, we need to change to define a diffusion rate  $c$  that works similarly to  $\kappa$  and describes the rate at which the elements will spread through the edges. While  $c$  could be edge specific (the same way that  $\kappa$  can be position specific in a non-homogeneous environment), for the sake of simplicity, we set  $c$  to be identical for all edges.

Then, based on the formulation of diffusion, the flow going through an edge connecting two nodes  $i$  and  $j$ , with respective concentrations  $s_i$  and  $s_j$ , in the  $j$  to  $i$  direction over a period  $dt$ , will be  $c(s_j - s_i)dt$ . Thus, the contribution of that edge to the concentrations  $s_i$  and  $s_j$  will be:

$$\left. \frac{ds_i}{dt} \right|_{(i,j)} = c(s_j - s_i) \quad \left. \frac{ds_j}{dt} \right|_{(i,j)} = c(s_i - s_j) \quad (8)$$

We can then extend that calculation to the whole graph:

$$\frac{ds_i}{dt} = c \sum_{j=1}^N A_{i,j}(s_j - s_i) \quad (9)$$

where  $A$  is the adjacency matrix, thus defining  $A_{i,j}$  if nodes  $i$  and  $j$  are connected by an edge. Note that, since we are using undirected graphs,  $A$  is necessarily symmetric.

When considering discrete time steps of duration  $\tau$ , equation (4) becomes:

$$s_i^{n+1} = s_i^n + \tau \sum_{j=1}^N A_{ij}(s_j^n - s_i^n) \quad (10)$$

which is exactly the equation used by the robots to update their internal state.

## 2.2 Convergence

In this section we discuss the convergence rate of the discrete graph Laplacian toward the continuous Laplacian. We consider the more general context of [54]. Following its terminology, we consider a Riemannian manifold  $M$  of dimension  $d = 2$ , and  $N$  points  $x_i$  which are independently and uniformly distributed over  $M$ . We consider a weight matrix  $W$  defined as:

$$W_{ij} = k \left( \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\epsilon} \right) \quad (11)$$

where  $k$  is the kernel function, and  $\epsilon$  is the kernel bandwidth. In machine learning, a typical choice for the kernel function is  $k(x) = e^{-x}$  (radial basis function kernel). However, in our experiments and simulations we use a step kernel ( $k(x) = 0$  if  $\|x\| > \theta$ , and  $k(x) = 1$  if  $\|x\| \leq \theta$  for some constant  $\theta$ ). The parameter  $\sqrt{\epsilon}$  is homogeneous to a distance and plays the role of field of perception in our setting. It is the distance over which the points can communicate with their neighbors. Thanks to that definition, we set  $\theta = 1$  in the step kernel function. As such, we obtain in our case  $W = A$ , where  $A$  is the adjacency matrix defined earlier.

To build the graph Laplacian, we then consider the diagonal matrix  $D$  given by:

$$D_{ii} = \sum_{j=1}^N W_{ij} \quad (12)$$

For a step kernel,  $D$  is simply the degree matrix counting the number of neighbors of each point  $\mathbf{x}_i$  within a distance  $\theta$ .

We now define the matrix of the negative defined left-normalized discrete Laplacian:

$$\tilde{L} = D^{-1}W - I \quad (13)$$

(Note that in our simulation and experiments, we consider the non-normalized discrete Laplacian  $L = D - A$ ).

We consider a smooth test function  $f$ . Singer established the following uniform estimate of the error of the left-normalized discrete Laplacian compared to the continuous Laplacian [54].

$$\frac{1}{\epsilon} \sum_{j=1}^N \tilde{L}_{ij} f(\mathbf{x}_i) = \frac{1}{2} \nabla f(\mathbf{x}_i) + O\left(\frac{1}{N^{1/2}\epsilon}, \epsilon\right) \quad (14)$$

This error estimate shows that for a fixed  $\epsilon$ , the left-normalized discrete Laplacian converges point-wise to the continuous Laplacian as  $1/\sqrt{N}$ , which is the classical scaling expected from the central law. Similarly to our numerical simulations on robots, it also predicts that for a fixed  $N$ , there is an optimal bandwidth  $\epsilon$  that minimizes error because the error terms contains two sub terms that behaves oppositely with  $\epsilon$ . If  $\epsilon$  is too large, all points are connected and the Laplacian matrix is an almost constant matrix (a clique connectivity) that carries little information over the local geometry. Conversely, if  $\epsilon$  is too small, points are disconnected and the left-normalized Laplacian matrix does not encode information about the geometry either. Balancing the two errors terms, Singer shows that the optimal  $\epsilon$  is:

$$\epsilon = \frac{C(M)}{N^{1/4}} \quad (15)$$

where  $C(M)$  is a function that depends on the manifold  $M$ . It must be noted that this equation predicts an optimal scaling for the field of perception  $\sqrt{\epsilon} N^{1/8}$ , which is sensibly different from the scaling  $N^{1/2}$  that we find in numerical simulations for robots. The discrepancies in the exponents may be explained by various factors: Singer considers Riemannian manifold without boundaries and point, while we consider shapes with boundaries with non-penetrating hard spheres. In addition we consider slightly different problems: Singer estimates the optimal field of perception for approximating the continuous Laplacian [54], while we estimate the optimal field of perception for classifying distinct shapes.

## 2.3 Spectral gap

In this section, we discuss the physical interpretation of  $\lambda_2$  in light of spectral graph theory [33]. In the general case, the spectral gap of a self-adjoint linear operator is the modulus of the smallest non-zero eigenvalue. The spectral gap plays a central role in graph theory and quantum mechanics, as it controls the dynamical properties of the system on which the operator is defined. In our case, if we assume that the graph is connected, and given that the first eigenvalue of the graph Laplacian  $\lambda_1$  is always 0 (because the constant vector is an eigenvector with a null eigenvalue), the spectral gap is simply  $\lambda_2$ . As such,  $\lambda_2$  is expected to provide information about the shape of the graph, which allows us to distinguish between the different arena presented in the main text.

### 2.3.1 Isoperimetric inequality and Cheeger constant

The eigenvalue  $\lambda_2$  of the graph Laplacian measures how easily information diffuses on a graph. A large  $\lambda_2$  implies that information flows easily and smoothly between all parts of the graph, while a small  $\lambda_2$  indicates the presence of bottlenecks in the flow. In the worst case,  $\lambda_2 = 0$  implies that the graph is not connected, with at least two independent parts.

To see this quantitatively, let us first define the boundary  $\partial(\mathcal{S})$  of a subset  $\mathcal{S}$  of vertices as the set of edges that connect a vertex inside  $\mathcal{S}$  to a vertex outside  $\mathcal{S}$ .

$$\partial(\mathcal{S}) = \{(u, v) \in \mathcal{E} \mid u \in \mathcal{S}, v \notin \mathcal{S}\} \quad (16)$$

The isoperimetric ratio of  $\mathcal{S}$  measures the size of the boundary  $\partial(\mathcal{S})$  with respect to the size of  $\mathcal{S}$ :

$$h(\mathcal{S}) = \frac{|\partial(\mathcal{S})|}{|\mathcal{S}|} \quad (17)$$

where  $|\cdot|$  is the sum of degrees  $d$  of vertices in a set:

$$|\mathcal{S}| = \sum_{x \in \mathcal{S}} d_x \quad (18)$$

We can then define the Cheeger constant, or isoperimetric number of a graph  $G$ , as the minimum isoperimetric number taken over all subsets that have less than half of the total number of vertices.

We additionally define  $\Delta(G)$  as the maximum degrees of vertices of the graph  $G$ . We can now state the Cheeger inequalities [25, 46] which provides lower and upper bounds for the eigenvalue  $\lambda_2$  with the isoperimetric number  $h(G)$ :

$$h(G) \geq \frac{\lambda_2}{2} \geq \frac{h^2(G)}{2\Delta(G)} \quad (19)$$

This inequality is central in spectral theory as it directly relates local properties that are controlled by diffusion (*e.g.*, the rate of diffusion of information) with global properties that are set by the shape of a graph. For instance, on graph with large  $\lambda_2$ ,  $h(G)$  is large and information flows easily because all their subsets have large boundaries and are well connected to other subsets. Conversely, if a graph  $G$  is not connected, there exists two subsets that are not connected so  $h(G) = 0$ , and the Cheeger inequality forces  $\lambda_2 = 0$ .

### 2.3.2 Fiedler vector algebraic connectivity

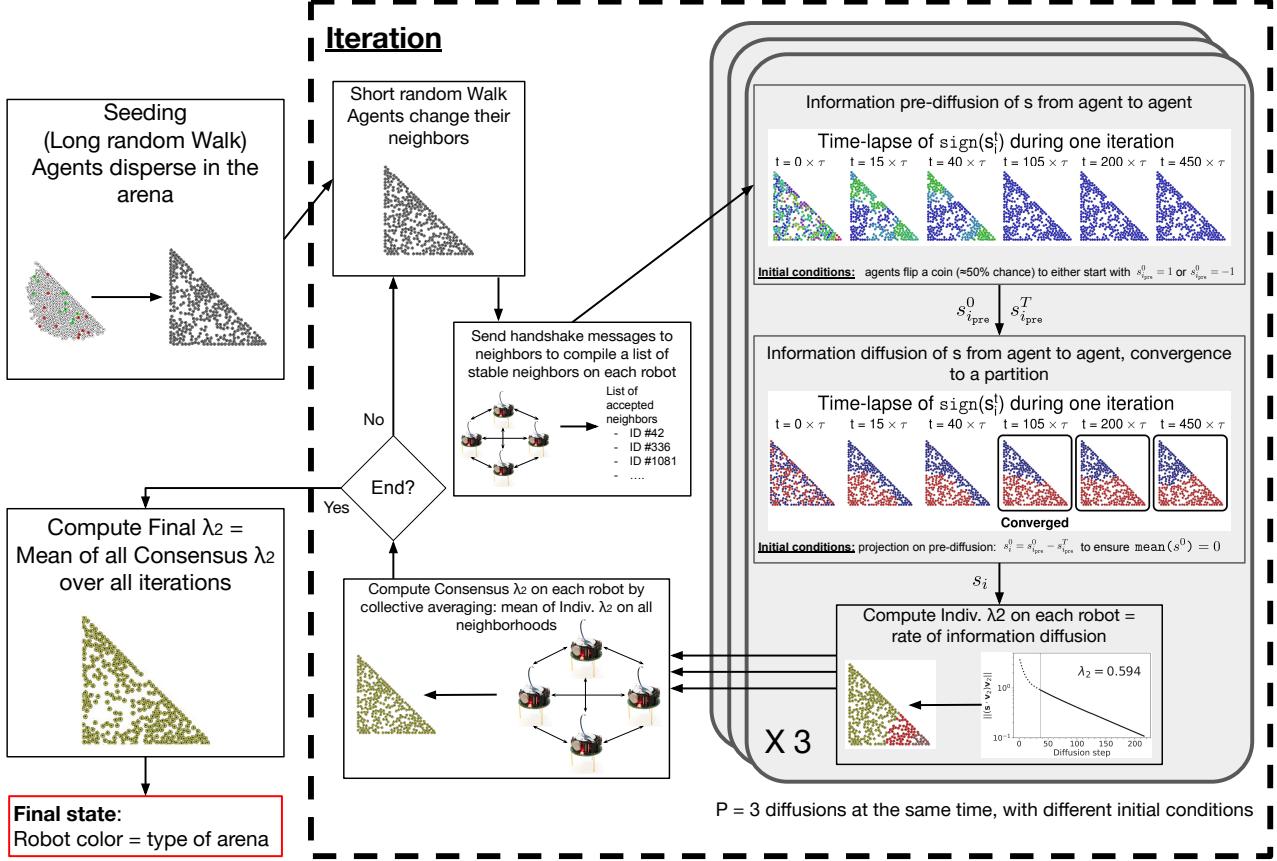
The eigenvalue  $\lambda_2$  and its eigenvector  $\mathbf{v}_2$  are also related to the algebraic connectivity and Fiedler vector of graph theory.

The eigenvector  $\mathbf{v}_2$  can be used to partition a graph into two natural clusters.

### 3 Spectral swarm robotics algorithm

Here we propose to translate the theoretical approach presented earlier into a multi-agents algorithm that can be implemented as robotic controller of robotic swarms in simulations or experiments. The swarm of agents can collectively classify the shape of their arena based on distributed spectral shape analysis. Each agent computes its own estimation of  $\lambda_2$  using only local information (*i.e.*, its own state and that of neighboring agents) – this value serving as a signature of arena shapes.

The spectral swarm robotics algorithm is summarized in Supplementary Fig. 6. It is also formally described in Alg. 1.



Supplementary Figure 6: Workflow of the spectral swarm robotics algorithm.

This algorithm is designed to be easily implementable on Kilobot robots. As such, it makes the following assumptions:

- the agents can all synchronously execute the same stage of the algorithm – *i.e.*, all agent have access to a temporal counter iterated continuously during the experiment and with roughly the same value across all agents. This allows the algorithm to provide a fixed budget of time for each stage of the algorithm.
- the agents can communicate to their neighbors across a given field of perception.
- the agents can display ongoing results (such as their estimated value of  $\lambda_2$ ) by changing the color of an in-board LED.
- the agents can perform floating-point computations, with a library implementing basic math functions – *i.e.* `abs`, `log`, `exp`

Agents are assumed to be initially placed packed at the center of the arena. Our approach requires agents to be distributed in all parts of the arenas, so the spectral swarm robotics algorithm starts by making the agents disperse and move to random positions in the arena (“initial seeding”) by using the “run-and-tumble” algorithm.

After the seeding stage, the algorithm starts its main loop. In each iteration  $a$  of the main loop, the agents will compute their own estimation of  $\lambda_2$ , termed here as  $\text{Indiv}\lambda_{2,i}(a)$  for each agent  $i$ . During one iteration, the agents will sequentially perform the following operations:

1. Agents will move with a short budget of time, in order to change their neighbors. This allows the algorithm to change its initial conditions and to start with different configurations of neighbors from iteration to iteration. After the allotted time has elapsed, agents will stop moving and stay immobile for the rest of the iteration.
2. Agents will use a handshaking system to determine what are their “stable” neighbors, *i.e.*, the list of neighbors  $k$  of a focal agent  $i$  such as, most of the time,  $i$  can broadcast messages to  $k$  and  $k$  can broadcast messages to  $i$ . That process is required as there is a variability in the actual field of perception  $\sigma$  of robots, due to small physical differences (*e.g.*, infra-red captors) or environmental conditions (*e.g.*, slightly higher IR luminescence in parts of the arena). This variability can result in cases where agent  $i$  would be able to reach  $k$  but not the opposite. However, our algorithm relies on the Laplacian of the communication graph which is expected to be symmetrical. As such, it is important to determine which neighbors are stable.
3. Agents compute  $P = 3$  diffusion sessions in parallel (further details in subsequent sections), each with different initial conditions. This process allows each agent to compute  $P$  different local estimates of  $\text{Indiv}\lambda_{2_i}(a, d)$  with  $d$  the index of the diffusion session performed in parallel. Each agent then average their local estimates to obtain a single local estimate  $\text{Indiv}\lambda_{2_i}(a)$ .
4. Local estimates are broadcasted on all neighborhoods so that agents can reach a consensus towards a global estimate  $\text{Consensus}\lambda_{2_i}$ . This is achieved through a collective averaging process: agents iteratively average their local estimate with those of their neighbors.
5. Agents will check if the total budget of time allotted to the experiment is exceeded or not. If the latter is true, they proceed to the next iteration.

The algorithm will sequentially compute  $I$  iterations. The  $\text{Consensus}\lambda_{2_i}$  values obtained at each iteration are averaged to compute the overall  $\lambda_{2_i}$  estimation, termed  $\text{Final}\lambda_{2_i}$ . The agents will set the color of their LED depending on the value of  $\text{Final}\lambda_{2_i}$ : for a given set of arenas, we define a corresponding set of LED colors, respectively associated in turn to a set of centroid values of  $\lambda_2$  values.

Each part of the algorithm will be further developed in the following sections.

---

**Algorithm 1** Main algorithm, executed on each robot  $i$ .

---

```

run_and_tumble(seeding_duration)                                ▷ Seeding, long random walk

for  $a$  in  $1..I$  do                                              ▷ Perform  $I$  iterations
    run_and_tumble(short_rw_duration)                         ▷ Short random walk at the beginning of each iteration

    neighbors ← handshake()                                    ▷ Identify stable neighbors

    ▷ Start diffusion of information with neighboring robots
    parallel-for  $d$  in  $1..P$  do                                ▷ Perform  $P$  diffusion sessions in parallel
         $s_i^0(a, d) \leftarrow \text{random\_choice}(-1, 1)$           ▷ Initialization, with random values: either -1 or 1
         $s_i(a, d), \text{Indiv}\lambda_{2_i}(a, d) \leftarrow \text{diffusion(neighbors}, s_i^0(a, d))$       ▷ Pre-diffusion
         $s_i^0(a, d) \leftarrow s_i^0(a, d) - s_i^T(a, d)$            ▷ Initialization from pre-diffusion values to ensure  $\sum_i s_i^0(a, d) = 0$ 
         $s_i(a, d), \text{Indiv}\lambda_{2_i}(a, d) \leftarrow \text{diffusion(neighbors}, s_i^0(a, d))$       ▷ Diffusion
    end parallel-for
     $\text{Indiv}\lambda_{2_i}(a) \leftarrow \frac{\sum_d^P \text{Indiv}\lambda_{2_i}(a, d)}{P}$           ▷ Average  $\lambda_{2_i}$  values estimated through the  $P$  diffusion sessions

    ▷ Reach consensus across the swarm, through collective averaging of  $\lambda$ 
     $\text{Consensus}\lambda_{2_i}(a) \leftarrow \text{coll\_averaging(neighbors, Indiv}\lambda_{2_i}(a))$ 
end for

     $\text{Final}\lambda_{2_i} \leftarrow \frac{\sum_a^I \text{Consensus}\lambda_{2_i}(a)}{I}$           ▷ Average  $\text{Consensus}\lambda_{2_i}$  across all iterations
    set_robot_LED_color(Final\lambda_{2_i})

```

---

### 3.1 Dispersion of agents in bounded arenas (random walk)

For both dispersion stages (in Supplementary Fig. 6: “initial seeding” and “short random walk”), the agents move according to a run-and-tumble motion. Run-and-tumble is a motion pattern used in nature primarily by bacteria, such as *Escherichia coli* (*E. coli*), to navigate their environment [55, 56].

The run-and-tumble motion can also be implemented in robotic systems (especially in swarm robotics) – and specifically in scenarios where the robot has to explore an unknown environment. The process works as follows:

**Run** During the “run” phase, a robot moves in a generally straight path. This can be achieved by setting the wheels, tracks, or other locomotion systems to move forward in a set direction.

**Tumble** At random or predetermined intervals, the robot will enter the “tumble” phase. This is a state in which the robot changes its direction. This might involve stopping, spinning in place, or otherwise altering its trajectory to set a new course. The new direction can be random or determined by certain rules or sensors.

After tumbling, the robot begins its next “run” in this new direction.

This pattern continues, allowing a robot to explore its environment. The run-and-tumble algorithm is straightforward and does not require complex computations or detailed knowledge of the environment, which makes it suitable for small, relatively simple robots. It also allows for robust exploration and can even be used in groups of robots, where collective behaviors can emerge from individual run-and-tumble movements.

For the spectral swarm robotics algorithm, we assign a budget of time for both dispersion stages. At the end of these budgets, the robots stop dispersing and switch to the next stage of the algorithm. We selected the run-and-tumble motion strategy for the spectral swarm robotics algorithm because it ensures that the robots can disperse in the arena in a roughly uniform manner, and change their neighbors even in crowded scenarios.

### 3.2 Handshake messages

Kilobots communicate by broadcasting a message to their neighborhood through infrared signals, resulting in potential conflicts if two or more robots are broadcasting at the same time. Message integrity is ensured through a CRC mechanism. However, the portion of correctly received messages depend on the number of neighbors. The proportion of correctly received messages can be quite low ( $\approx 10 - 30\%$  when 4-8 Kilobots broadcast messages in the same neighborhood).

Agents will employ a handshaking process to identify their “stable” neighbors, or the collection of neighbors  $k$  for a focal agent  $i$ , such that, in most cases,  $i$  can transmit messages to  $k$ , and  $k$  can reciprocate. Indeed, the actual field of perception  $\sigma$  of robots can vary due to minor physical distinctions (e.g., infrared sensors) or environmental factors (e.g., marginally increased IR luminescence in certain parts of the arena). This may lead to situations where agent  $i$  can communicate with  $k$ , but not the other way around. However, our algorithm relies upon the Laplacian of the communication graph, which is assumed to be symmetrical. Consequently, it is crucial to establish which neighbors are reliable.

In our approach, each focal agent  $i$  will continuously broadcast handshake messages containing its own identifier, and the identifier of agents that it has previously successfully received messages from. As such, if agent  $i$  receives a message from agent  $k$  confirming that  $k$  can receive a message from  $i$ , it means that there is a bidirectional communication channel between  $i$  and  $k$ . This process is repeated several times: at each new round of handshaking, agents will first empty their list of neighbors, then continuously broadcast for some time. After a fixed budget of time, agents stop broadcasting handshake messages, and will assess which agents they have a stable bidirectional communication channel with, on all handshaking rounds. On each agent, the latter are regrouped into a list of stable neighbors.

### 3.3 Diffusion and $\text{Indiv}\lambda_2$ estimation

In this stage of the algorithm, agents aim to compute a local estimate of  $\lambda_2$ , noted  $\text{Indiv}\lambda_{2,i}$  for agent  $i$ . This is achieved by diffusing local information of each agent within the entire swarm, as described in the Method section of the main text. Local information is formalized by agent internal state at diffusion step  $t$ :  $s_i^t$ . The latter is propagated throughout the swarm by following the dynamics of diffusion, as implemented in Alg. 2. Theoretical sections explained that when  $\sum_i s_i^0 = 0$ , the states of each agent decays along  $\mathbf{v}_2$ . At this time,  $\text{Indiv}\lambda_{2,i}$  corresponds to the rate of diffusion at the coordinates of agent  $i$  and can be estimated by computing the slope of the decay of each  $s_i^t$ .

The difficulty with this approach is that we need to ensure condition  $C_{\text{mean}}$  i.e.,  $\langle s^0 \rangle = \frac{1}{N} \sum_{i=1}^N s_i^0 = 0$  (considering that the  $s_i^0$  are not all set to 0), without having any access to global information (in this case, the mean of  $s$ ). Let's consider the following strategy P: each agent randomly starts with an initial state  $s_i^0$  either equal to  $-1$  or to  $1$ . Assuming that the number of agents is even, and sufficient large, this will result in a mean very close to 0 (variability will decrease as the number of agents increase). However, if these conditions are not met, this strategy would not work.

Instead, we propose a strategy that will ensure  $C_{\text{mean}}$  even if the number of agents is small and not even. In order to do this, we first carry out a diffusion session (termed **pre-diffusion**) with initial conditions similar to P:  $s_{i,\text{pre}}^0 = -1$  or  $1$ . As a result of the diffusion dynamics,  $s_{i,\text{pre}}$  converges to the mean of initial conditions  $\langle s_{i,\text{pre}}^0 \rangle$ . This allows us to estimate how to correct the initial conditions to ensure  $C_{\text{mean}}$ . After the pre-diffusion, we thus

execute a new diffusion session, with initial conditions corrected by using a projection from pre-diffusion values:  $s_i^0 = s_{i_{\text{pre}}}^T - s_{i_{\text{pre}}}^0$ . This allows the diffusion session to ensure condition  $C_{\text{mean}}$ , and it can be used to effectively estimate  $\text{Indiv}\lambda_{2_i}$  for any number of agents.

---

**Algorithm 2** Code for diffusion and pre-diffusion sessions.

---

```

function DIFFUSION(neighbors,  $s_i^0$ )
     $\triangleright$  Start diffusion of information with neighboring robots
     $t_i^0 \leftarrow 0$ 
    for  $j$  in  $1..T$  do                                 $\triangleright$  Perform  $T$  diffusion steps, each with a duration of timeout_diff
         $\text{initial\_time\_diff} \leftarrow \text{current\_time\_ticks}()$ 
         $\text{broadcast}(s_i^{j-1})$                        $\triangleright$  Allow neighboring robots to retrieve it
         $\triangleright$  Retrieve all  $s_k$  from neighbors, until timeout
         $c_j \leftarrow 0$ 
        while  $\text{current\_time\_ticks}() - \text{initial\_time\_diff} < \text{timeout\_diff}$  do
             $s_k^j \leftarrow \text{retrieve\_value\_from\_next\_neighbor(neighbors)}$ 
             $c_j \leftarrow c_j + (s_k^j - s_i^j)$ 
        end while
         $s_i^j \leftarrow s_i^{j-1} + \tau c_j$                  $\triangleright$  Update  $s_i$  according to the law of diffusion
         $t_i^j \leftarrow \tau j$ 
    end for

     $\triangleright$  Compute local approximation of  $\lambda_2$  by using ordinary least square (OLS)
    
$$\text{Indiv}\lambda_{2_i} \leftarrow -\frac{(T-B) \sum_{j=B}^T (t_i^j \log(|s_i^j|)) - \sum_{j=B}^T (t_i^j) \sum_{j=B}^T (\log(|s_i^j|))}{(T-B) \sum_{j=B}^T (t_i^j)^2 - (\sum_{j=B}^T t_i^j)^2}$$
       $\triangleright B = \text{Number of diffusion steps to ignore}$ 
    return  $s_i$ ,  $\text{Indiv}\lambda_{2_i}$ 
end function

```

---

Algorithm 2 describes how diffusion and pre-diffusion sessions are computed. We rely on the OLS (ordinary least square) algorithm applied to  $\log(|s_i^t|)$  to estimate the slope of the exponential decay of  $s_i^t$  on each agent.

For the computation of  $\text{Indiv}\lambda_{2_i}$  we only take into account the time steps after  $B$ , which correspond to a burn-in period. Indeed, as seen in Main Text Fig. 4 (c and d), the values of  $\log(|s_i^t|)$  at the start of a diffusion session are not linear.

Here is a description of a number of internal functions used in Alg. 2:

`current_time_ticks()` returns the current ticks of the agent. Ticks are initialized at 0 at the start of a simulated or experimental run, and are iterated at a frequency of  $1/31 \approx 0.322$  Hz.

`broadcast(val)` continuously emit messages containing value `val` in the neighborhood. Neighboring agents will be able to receive the latter message, if they are situated at a distance smaller than  $\sigma$  of the focal agent.

`retrieve_value_from_next_neighbor(neighbors)` Wait for an agent listed in `neighbors` to broadcast a message to the focal agent, and return this message. If the same agent broadcasted several messages, only take the last one into account.

In this stage of the algorithm, agents displays (*i.e.*, LED color) a color code corresponding to the sign of their value of  $s_i^t(a, 0)$ : blue for negative values, red of positive values.

Computing several diffusion sessions in parallel allows the algorithm to have the results over several initial conditions, while pooling transmitted values of  $s_i^t(a, d)$  of all  $P$  diffusion sessions into only one message. This enables the computation of several diffusion dynamics concurrently, for the same amount the messages transmitted (and corresponding budget of time) as would be needed for a single diffusion session. Of course, depending on the hardware implementation, and communication capabilities of the agents, it may be possible to compute a larger number of diffusion sessions in parallel.

### 3.4 Computing Consensus $\lambda_2$ through collective averaging

After the diffusion sessions, each agent of the swarm has a different local estimation of  $\lambda_2$ , *i.e.*,  $\text{Indiv}\lambda_{2_i}$ . There can be a large variability in those estimations, depending on the position of the agents in the arena. As such, we aim to compute the mean of these values so as to reach a global consensus on the estimation. This is achieved by using a collective averaging scheme, as described in Alg. 3. The value of `Consensus` $\lambda_{2_i}$  is initially set to  $\text{Indiv}\lambda_{2_i}$  on each agent. Then, agents will average their value of `Consensus` $\lambda_{2_i}$  with their neighbors. This

---

**Algorithm 3** Code for reaching a consensus on a value through collective averaging.

---

```
function COLL_AVERAGING(neighbors, w)
    ▷ Reach consensus across the swarm, through collective averaging
    Consensusw ← w
    for k in 1..C do
        ▷ Perform C coll. avg. rounds, each with a duration of timeout_cons
        initial_time_cons ← current_time_ticks()
        broadcast(Consensusw)                                ▷ So that neighboring robots can retrieve it
        ▷ Retrieve values from neighbors, until timeout
        Consensusk ← 0
        Nk ← 0                                              ▷ Number of neighbors we received a message from
        while current_time_ticks() - initial_time_cons < timeout_cons do
            Consensusk ← Consensusk + retrieve_value_from_next_neighbor(neighbors)
            Nk ← Nk + 1
        end while
        Consensusw ←  $\frac{\text{Consensus}_k + \text{Consensus}_w}{N_k + 1}$ 
    end for
    return Consensusw
end function
```

---

process is repeated over  $C$  rounds. The fact that all agents operate over the same number of rounds ensures that all agents are represented equally in consensus building, and that the consensus is indeed the global average over all values of  $\text{Indiv}\lambda_{2_i}$ .

In this stage of the algorithm, agents display (*i.e.*, LED color) a color code corresponding to the arena they estimate they are in, depending on the value of  $\text{Consensus}\lambda_2$  (cf Sec. 3.5). This value will be updated during the collective averaging session, and so the associated LED color will iteratively change accordingly.

### 3.5 Identifying centroids for visualizing $\text{Consensus}\lambda_2$ and $\text{Final}\lambda_2$ on each robots

The LED color code displayed by agents during the *collective averaging* and *Final* $\lambda_2$  computation stages corresponds to the arena the agent estimate they are in. Namely, the color codes correspond to the predicted classes (*i.e.*, arena shape) in a classification process involving a given library of possible shapes. Each possible class in the library of possible shapes will be respectively associated with a centroid, *i.e.*, the mean value of  $\text{Final}\lambda_2$  across 64 simulated runs (cf Main Text Fig.3c). Then, the predicted class of agent  $i$  will be decided using a nearest centroid classifier based on its current value of  $\text{Final}\lambda_{2_i}$ , *i.e.*, it will correspond to the class in the library of shapes with which its  $\text{Final}\lambda_{2_i}$  is closest.

## 4 Experimental setup

### 4.1 Robots and Arenas

Supplementary Figure 7 shows the experimental setup used at Sorbonne Université for conducting all the robotic experiments. We use two arenas (disk and annulus), each populated with 25 robots. The robots used are Kilobots [1, 2], originally designed at Harvard University and bought from the K-Team SA company.

All experiments are conducted while the whole setup is isolated from external perturbation using light-occluding black curtains placed on all sides and above the setup. The whole setup is enclosed in a construction of an aluminum section that supports the camera and lightning (see Caption of Supplementary Fig. 7 for an exhaustive description).

Setting up each experiment starts with removing the dust on the arena by using a wipe with a very small quantity of water to ensure maximum ground reflection. The target area (Disk or Annulus) is drawn on the floor to help position the robots before the experiment starts. Kilobots are then initially placed randomly in the target area, broadly respecting a uniform distribution (*i.e.*, clusters are avoided), with random orientation. Initial positions thus vary from one run to another. Each experiment starts with all robots at full charge. The experiment is started using the infrared overhead controller, and the black curtains are closed within 10 seconds.

Monitoring is performed with a dedicated webcam connected to a desktop computer. The image is voluntarily blurred so that each Kilobot’s overhead LED standout as a blurred circle of color that corresponds to the current state of the robot.

An extensive description of the experimental procedure can be found at:

[https://docs.google.com/document/d/1OKSpUdTHtPYe3wSIos1Mmxrl\\_913OnisLJLadvIc4j8/edit?usp=sharing](https://docs.google.com/document/d/1OKSpUdTHtPYe3wSIos1Mmxrl_913OnisLJLadvIc4j8/edit?usp=sharing)

### 4.2 Controller design and implementation of the algorithms on kilobots

We implement the spectral swarm robotics algorithm on Kilobots by using the Kilombo [57] framework: it allows the same C code to be used both in simulations and as controllers of actual Kilobots, without any modification. Supplementary Table 2 lists the colors displayed by the Kilobots LEDs depending on their current behavior.

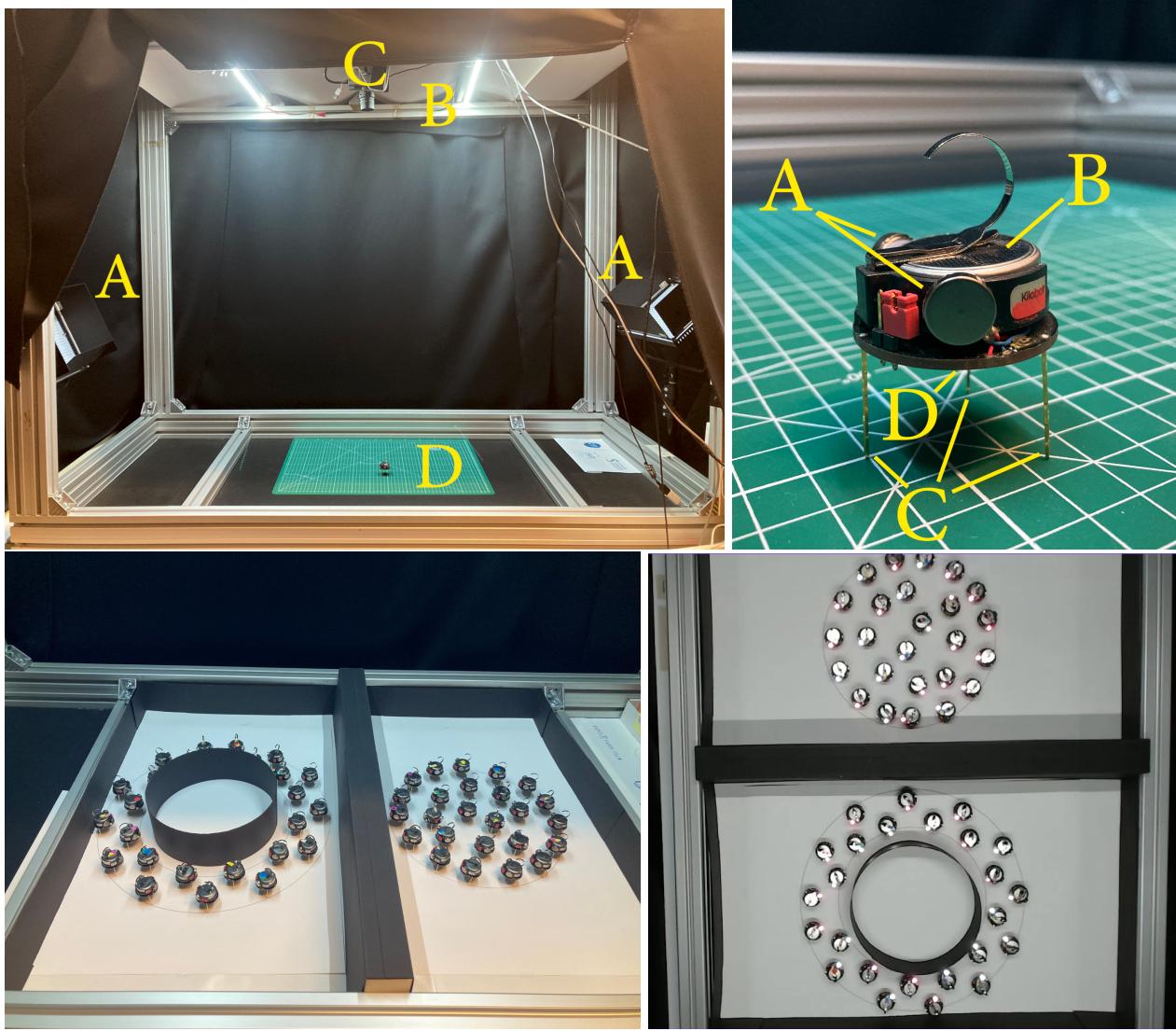
	LED color	Description
Dispersion	Red	Tumble behavior
	Green	Run behavior
Diffusion	White	Pre-diffusion stage
	Blue	$\text{sign}(s_i^t) > 0$
	Red	$\text{sign}(s_i^t) < 0$
	Grey	$\text{sign}(s_i^t) = 0$
$\lambda_2$ estimation	Cyan	Disk detected
	Orange	Square detected
	Green	Arrow detected
	Red	Star detected
	Gold	Triangle detected
	Brown	Stop detected
	Violet	Annulus detected

Supplementary Table 2: Table of LED colors displayed by the robots depending on their current behavior.

#### 4.2.1 Implementation challenges

There is a number of pitfalls to keep in mind when implementing the spectral swarm robotics algorithm on Kilobots.

First of all, robots have problems to communicate due to collisions between messages. The more neighbors there are, the more collisions there will be. Furthermore, there is variability between each robot in their ability to receive and send messages: some robots will have a more powerful IR transmitter than others, or be affected by different environmental artifacts (background light, presence of ground particles, etc). This discrepancy in the ability of the robots to communicate also varies during the experiment, even if the robots are immobile. This leads to cases where a robot can communicate with a neighbor, but that neighbor cannot communicate back – which, in turn, precipitates the emergence of asymmetries in the Laplacian matrix (which are supposed to be symmetrical). If the number of asymmetries is sufficiently large, several robots may have diffusion sessions that become divergent (*i.e.*, they no longer decay exponentially). Such errors will rapidly impact the entire swarm



Supplementary Figure 7: General overview of the experimental setup used for the robotics experiments. **Top-left:** general view of the experimental setup, which is closed using light-occluding black curtains. The robot area is 152 cm per 72 cm, within which a smaller area (88 cm per 72 cm) is physically delimited. The green floor (D) placed on the robot area is a cutting map on which stands a single Kilobot robot. Two Pixel K80 professional LED photography light projectors (A) are placed on either side of the setup, pointing upwards with an angle of 45 deg, and complemented with two rails of LED (White 5700K) placed on the ceiling, for a global illumination of 506-584 lux (B). A Logitech C920s HD Pro Webcam (C) and a Pixelink PL-D734MU camera (not used here) are placed in the center of the ceiling (115 cm above the robot area) and point downwards. The ceiling is covered in white matte paper **Top-right:** a Kilobot robot (*height* = 34mm, *diameter* = 33mm). The Kilobot features two vibrating motors (A), one 3.3V rechargeable battery, three metallic rods (C) and an infrared LED transmitter and photodiode receiver (D) placed underneath the robot and pointing downwards. **Bottom-left:** the annulus (left) and disk (right) configurations, using 25 robots. The disk configuration has a radius of 150mm and a surface of  $\approx 70685.8\text{mm}^2$ . The annulus configuration has an external radius of 200mm, internal radius of 133mm and a surface of  $\approx 70092.1\text{mm}^2$ . **Bottom-right:** same as above, acquired from the overhead webcam. A matte white floor is used, as well as a black disk shape for the annulus configuration used to physically forbid communication between robots from the opposite region of the annulus.

by being propagated by the diffusion process, biasing the results of the algorithm. This problem is partially alleviated through the Handshake mechanism of the spectral swarm robotics algorithm, as described in Sec. 3.2. However, it is often not sufficient to avoid the presence of asymmetries: cases where a robot will not be able to communicate with a given neighbor only for a small number of steps still remain. We have found that working

with low values of  $\tau$  (*e.g.*,  $\tau = 1/50s$ ) decreases the influence of asymmetries on the diffusion process, as it reduces the global rate of the diffusion process. Furthermore, the amount of time robots have to broadcast and receive messages at each step (`timeout_diff` in Alg. 2) needs to be adjusted in order to leave enough time for the robots to communicate with all their neighbors (taking into account message collisions).

Second, the robots need to be temporally synchronized so that they are able to all execute the same stages of the algorithm at the same time. We rely on the internal “kiloticks” counter of the Kilobots: it is initialized at 0 at the start of an experiment, and incremented at a frequency of 31 Hz. It is thus important that all robots start an experiment at the exact same time. There are small variabilities in the clock frequencies of the robots, and they all have a small time lag. During the diffusion, some robots may be 1-4 steps behind the others. Again, this can be partially alleviated by using small values of  $\tau$ , as it will reduce the impact of bias and errors in the diffusion process. Furthermore, variabilities in clock frequency are influenced by the charge of the Kilobots battery – as such, we start each experiment with robots that are all fully charged.

Third, the diffusion can degenerate after a certain time-step (*e.g.*, if time lags or large number of asymmetries in the Laplacian corrupted some robots’ internal state). In this case, the degenerated part of the diffusion should not be taken into account for the  $\text{Indiv}\lambda_{2_i}$  calculation. This is implemented by computing  $\text{Indiv}\lambda_{2_i}$  at each step of the diffusion by using OLS, and computing the mean squared error (MSE) of the fit. At the end of the diffusion, only the fit with the lowest MSE is kept as the value of  $\text{Indiv}\lambda_{2_i}$ .

#### 4.2.2 Communication between robots

Kilobots communication is very slow: they can broadcast at most 9 bytes of payload per messages, twice per second (18 bytes per seconds). This is a theoretical maximum rate that will rarely be reached. When the amount of neighbors increases, so does the amount of message collisions, resulting in a very low number of correctly received messages in such setting:  $\approx 10 - 30\%$  with 8 neighbors *i.e.*,  $\approx 1.8 - 5.4$  bytes per seconds.

Supplementary Table 3 describes the content of the messages broadcasted by the robots for each stage of the algorithm.

In the *handshake* stage, each robot keep a list of known neighbors. The payload of the messages contains the unique identifiers (UID) of 3 selected known neighbors. The selection of known neighbors from the list iterates from message to message, in a round-robin fashion.

In the *diffusion* stage, the messages contain the values of  $s_i^t(a, d)$  for agent  $i$ , at diffusion step  $t$ , during iteration  $a$ , and for all  $P = 3$  diffusion sessions  $d$  computed in parallel. Due to size constraints of the message, the values of  $s_i^t(a, d)$  are transmitted as half-precision float, while they are stored as single-precision float in the robots.

In the *collective averaging* stage, the messages contain the current  $\text{Consensus}\lambda_{2_i}$  value of agent  $i$ .

### 4.3 Analysis and plotting scripts

All analysis and plotting scripts were written in Python 3.10. The following libraries were used to perform analyses: NumPy [58], Pandas [59], NetworkX [60], SciPy [61], Geopandas [62] and shapely [63]. Plotting was realized through the Matplotlib [64] and Seaborn [65] libraries.

	Variable name	Type	Size (bits)	Description
Handshake	kilo_uid	uint16	16	Kilobot unique identifier
	known_uid0	uint16	16	Unique identifier of already known neighbor
	known_uid1	uint16	16	Unique identifier of already known neighbor
	known_uid2	uint16	16	Unique identifier of already known neighbor
	-	-	8	Empty
	type	uint8	8	Type of message ( <b>handshake</b> , diffusion, coll. averaging, etc)
	CRC	uint16	16	Cyclic redundancy checksum
Diffusion	Variable name	Type	Size (bits)	Description
	kilo_uid	uint16	16	Kilobot unique identifier
	$s_i^t(a, 0)$	half-precision float	16	Diffused value, for iteration $a$ and diffusion session 0
	$s_i^t(a, 1)$	half-precision float	16	Diffused value, for iteration $a$ and diffusion session 1
	$s_i^t(a, 2)$	half-precision float	16	Diffused value, for iteration $a$ and diffusion session 2
	-	-	8	Empty
	type	uint8	8	Type of message (handshake, <b>diffusion</b> , coll. averaging, etc)
Coll. avg.	CRC	uint16	16	Cyclic redundancy checksum
	Variable name	Type	Size (bits)	Description
	kilo_uid	uint16	16	Kilobot unique identifier
	$\text{Consensus} \lambda_{2_i}$	single-precision float	32	Broadcasted consensus value
	-	-	24	Empty
Coll. avg.	type	uint8	8	Type of message (handshake, diffusion, <b>coll. averaging</b> , etc)
	CRC	uint16	16	Cyclic redundancy checksum

Supplementary Table 3: Description of the data packets transmitted from kilobots to kilobots during the **handshake**, **diffusion** and **collective averaging** stages of the spectral swarm robotics algorithm. The maximal size of kilobots messages payload (*i.e.*, data, without type and CRC) is 72 bits (9 bytes).

## 5 Supplementary Results

Here, we provide additional results that provide a more detailed analysis compared to those of the main text. Specifically, we investigate in simulations:

- the influence of the number of iterations on the accuracy of the spectral swarm robotics algorithm, first in the case with 7 arenas and then focusing on the 2 most different arenas in terms of  $\text{Final}\lambda_2$ .
- the influence of the simulation parameters (number of agents  $N$ , field of perception  $\sigma$ ) on the order of the  $\text{Final}\lambda_2$  values relative to the shape of arenas.
- how the dispersion scheme used and initial position of the agents can influence the results of the algorithm, and whether having moving agents is necessary.

Supplementary Table 4 lists the simulation and experimental cases conducted in this paper. The SI present the results of the five first cases (Supplementary Figs. 8- 18). The main text presents the results of the first case (Main Text Figs. 2 and 3) as well as the remaining two cases, **sim-2** and **expe-2** (Main Text Fig. 4).

The parameters values used for all cases are listed in Supplementary Table 5.

Case name	Simulations or experiments	Dispersion alg.	Surface of arenas $S$	Arenas	Nr. of runs	Nr. of agents $N$	Lenghtscale $l$	Nr. of iterations $I$
sim-large-7	Simulations	run-and-tumble	$500000\text{mm}^2$	Disk, Square, Arrow, Star, Triangle, Stop, Annulus	64	50 – 550	100.0 – 30.15	1-30
sim-large-2	Simulations	run-and-tumble	$500000\text{mm}^2$	Disk, Annulus	64	50 – 550	100.0 – 30.15	1-30
sim-small-2-dispersion	Simulations	run-and-tumble	$70000\text{mm}^2$	Disk, Annulus	64	10 – 80	83.67 – 29.58	1-10
sim-small-2-random	Simulations	None - immobile robots initialized at random positions	$70000\text{mm}^2$	Disk, Annulus	64	10 – 80	83.67 – 29.58	1-10
sim-small-2-uniform	Simulations	None - immobile robots initialized uniformly in the arenas	$70000\text{mm}^2$	Disk, Annulus	64	10 – 80	83.67 – 29.58	1-10
sim-2	Simulation	None - immobile robots initialized uniformly in the arena	$70000\text{mm}^2$	Disk, Annulus	64	25	52.91	1
expe-2	Experiments	None - immobile robots initialized uniformly in the arena	$70000\text{mm}^2$	Disk, Annulus	15	25	52.91	1

Supplementary Table 4: List of all considered cases in simulation and experiments.

### 5.1 Simulations: arenas with surface $S = 500000\text{mm}^2$

Here, we consider in simulation the cases **sim-large-7** (with 7 arenas) and **sim-large-2** (with only 2 arenas: disk and annulus). Both cases have arenas with a surface of  $S = 500000\text{mm}^2$ . Main Text Figs. 2 and 3 present results from the **sim-large-7** case, after 30 iterations. That number of iterations was selected because, for all parameters considered, the algorithm needs at most 30 iterations to converge when considering arenas of that size.

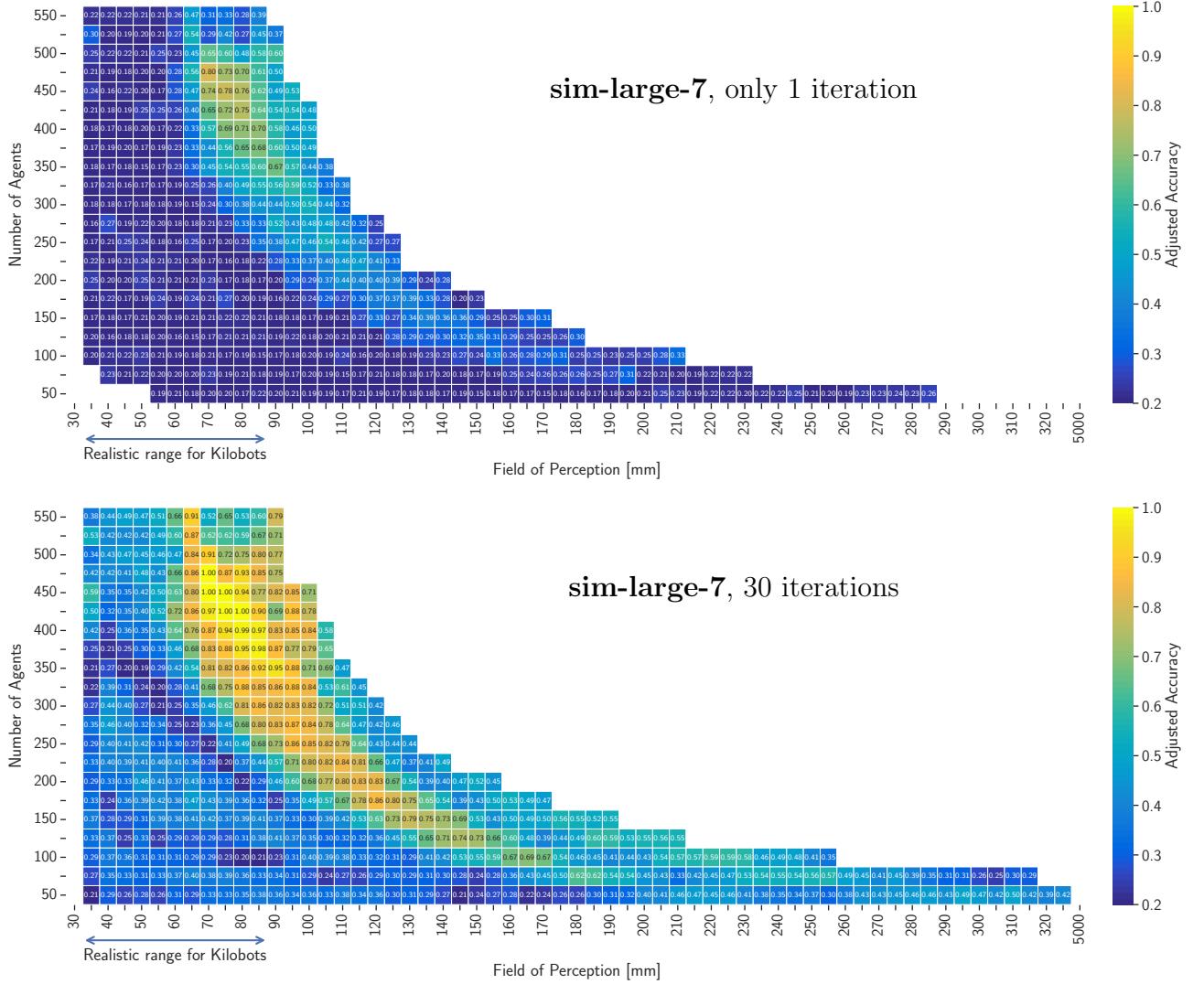
Supplementary Fig. 8 compares the results with respect to accuracy after respectively 1 and 30 iterations. Both configurations show a clear impact of the simulation parameters ( $N$  and  $\sigma$ ) on the accuracy. These results validate the conclusions from Main Text Fig. 3: a higher number of iterations translate into a higher accuracy. For both 1 and 30 iterations, we observe similar regimes as in Main Text Fig. 3, with the optimal solutions found in the hyperbole  $N = \alpha S / (\pi \sigma^2)$  with  $\alpha = 17$ .

Supplementary Figure 9 shows the influence of the parameters  $N$  and  $\sigma$  on the ordering of the values of  $\text{Final}\lambda_2$  corresponding to all 7 arenas. This ordering (“ranking”) corresponds to the order of their associated values: the lowest value is ranked first (0), the second lowest is ranked second, and so on. The highest value is ranked last (6). On the optimal  $\alpha = 17$  hyperbole (including regimes r1, r3 and r4 from Main Text Fig. 3), the ordering is generally consistent – starting from low to high: annulus, stop, triangle, star, arrow, square and disk. In the neighborhood of regime r5, this ordering is roughly inverted – it starts with the disk to finish with the annulus. In most cases, the disk and annulus are the furthest apart in terms of ranking.

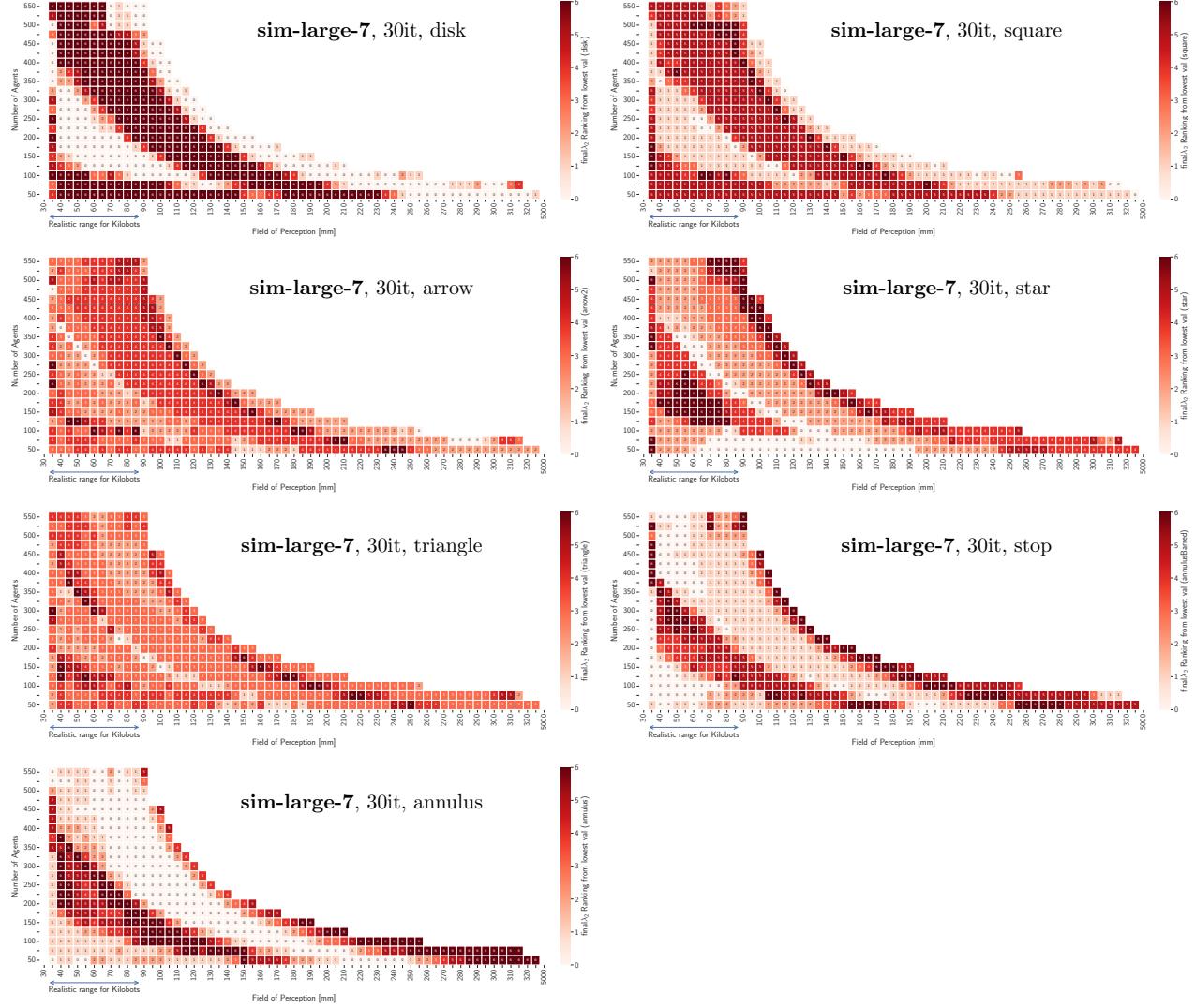
We thus chose to focus on the disk and annulus arenas in subsequent cases, allowing us to investigate a more canonical scenario compared to the **sim-large-7**. Namely, Supplementary Fig. 10 presents accuracy results for the **sim-large-2** case (including rankings). Compared with the **sim-large-7** case, the accuracy results of **sim-large-2** are higher for all considered parameters. After one iteration, the best-performing parameters are only found close to the hyperbole  $p$ . After 30 iterations, most parameters result in high-performing solutions: optimal solutions (accuracy of 100%) are found close to the optimal  $\alpha = 17$  hyperbole, close to the r5 regime, as well as for very low number of agents and short fields of perception ( $N \in [50, 150]$ ,  $\sigma \in [40, 85]$ ). The only solutions with lower accuracy are found on the boundaries between regimes, and corresponds to the boundaries seen in the ranking figure at the bottom.

	Parameter Name	sim-large-7	sim-large-2	sim-small-2-dispersion	sim-small-2-random	sim-small-2-uniform	sim-2	expe-2			
Base	Nr. of runs	64									
	Arena Surface $S$	500000mm <sup>2</sup>		70000mm <sup>2</sup>							
	Agent diameter	33 mm									
	Field of Perception $\sigma$	30mm-320mm					$\approx 85$ mm				
	Nr. of agents $N$	50-550		10-80		25					
	Initial $s^0$ value	-1 or 1, chosen randomly, uniform distribution is *not* enforced									
	$\tau$	1.0/15.0s					1.0/50.0s				
	Number of iterations $I$	1-30		1-10		1					
	Initial position of agents	Packed in the center of the arena			Random positions	Uniformly distributed (equidistant to neighbors)					
	Prop. of correctly received messages between neighbors	100%					$\approx 10\%$				
	Number of diffusion sessions computed in parallel per it.	3									
	Max nr. of neighbors	20									
Time	Simulation/Experiment time	1143900 kt $\approx$ 615min	257300 kt $\approx$ 138min	148800 kt = 80min	146630 kt $\approx$ 79min						
	Initial dispersion duration	46500 kt = 25min		0 kt							
	Iteration total duration	36580 kt $\approx$ 20min	21080 kt = 680s	14880 kt = 480s	146630 kt = 79min						
	Iteration waiting time	930 kt = 30s									
	Iteration handshake	310 kt = 10s					6200 kt = 200s				
	Iteration handshake step dur.	31 kt = 1s					248 kt = 8s				
	Iteration dispersion duration	6200 kt = 200s			0 kt						
	Iteration diffusion/pre-diffusion duration	13950 kt = 450s	6200 kt = 200s			54250 kt = 29min					
	Iteration diffusion/pre-diffusion nr. of steps	450	200			218					
	Iteration diffusion/pre-diffusion step duration	31 kt = 1s					248 kt = 8s				
	Iteration diffusion burn-in	5000 kt $\approx$ 161s	620 kt = 20s			10000 kt $\approx$ 323s					
Dispersion	Iteration collective avg. duration	620 kt = 20s					15500 kt = 500s				
	Iteration collective avg. nr. of steps	20					62				
	Iteration collective avg. step dur.	31 kt = 1s					248 kt = 8s				
	Step duration	15 kt $\approx$ 0.48s		-							
	Tumble duration	93 + $\mathcal{N}(0, 1) * 31$ kt = 3 + $\mathcal{N}(0, 1)$ s		-							
	Tumble duration domain	0 - 124 kt = 0 - 4s		-							
	Run duration	19 kt = 0.6s		-							

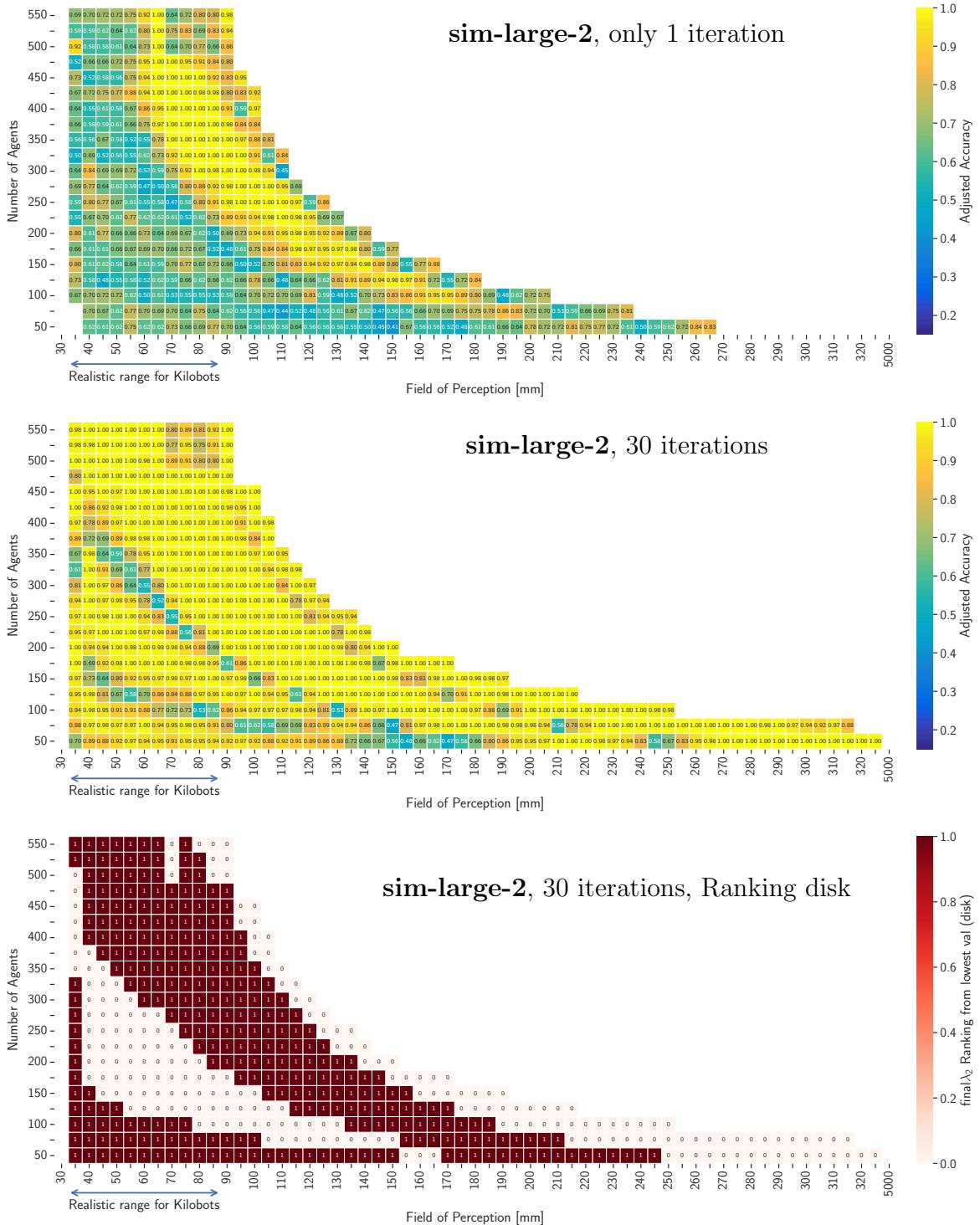
Supplementary Table 5: List of parameter values used with all cases. One kilotick (kt) corresponds to  $1/31 \approx 0.03$  seconds.



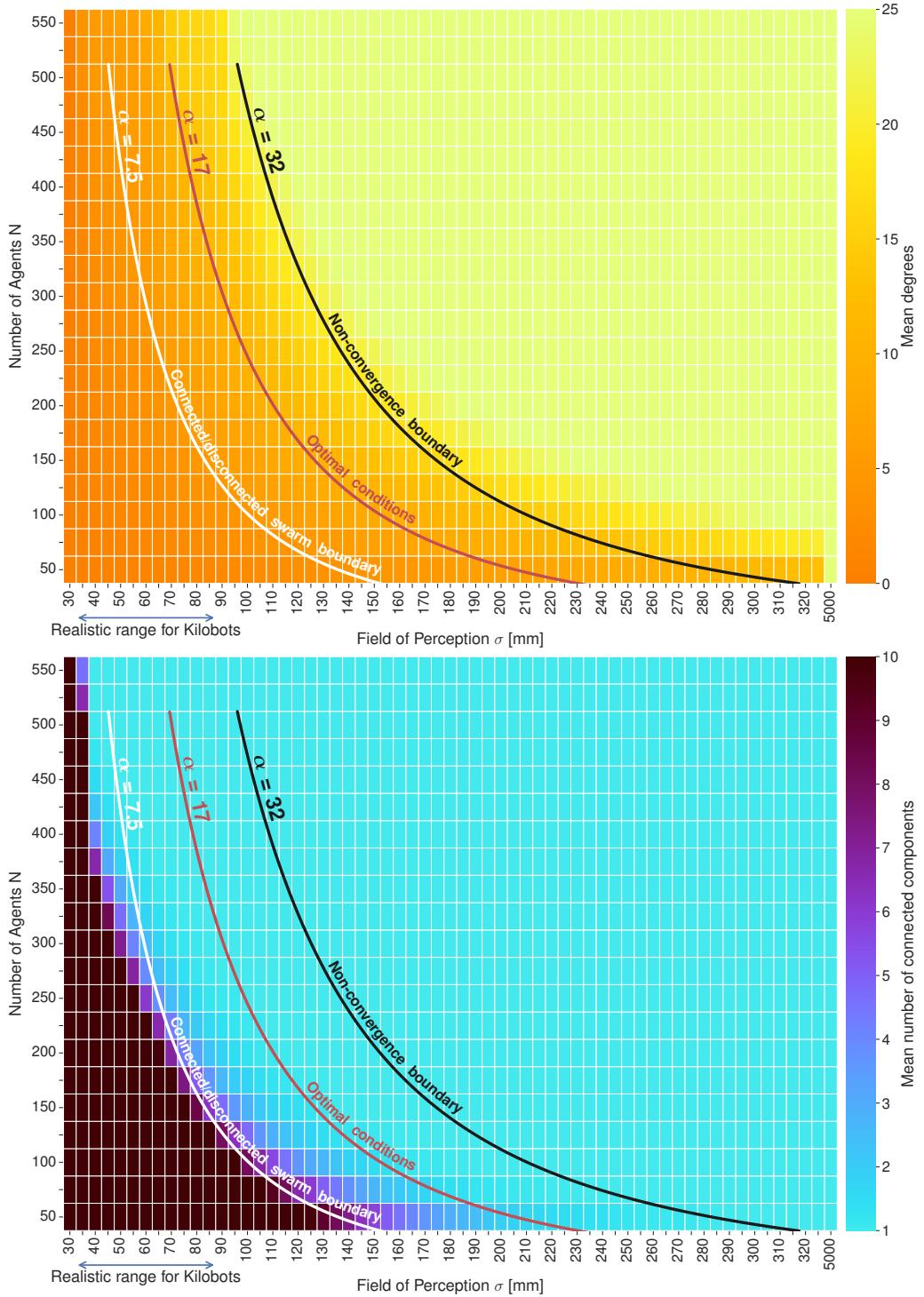
Supplementary Figure 8: Evolution of the accuracy with respect to the field of perception of each agent and the number of agents, for the case **sim-large-7**. Here, we consider 7 arenas with a surface  $S = 500000\text{mm}^2$ . All results are computed over 64 simulation runs. We adjust the accuracy score to only show results where the simulated diffusion sessions in the swarm did not diverge: empty bins correspond to results where there are more than 50% of runs exhibiting divergences. **Top:** results obtained after only 1 iteration. **Bottom:** results obtained after 30 iterations.



Supplementary Figure 9: Ranking of the values of the Final $\lambda_2$  centroids for all arenas in the **sim-large-7** case, after 30 iterations.



Supplementary Figure 10: Evolution of the accuracy with respect to the field of perception of each agent and the number of agents, for the case **sim-large-2**. With this case, we consider 2 arenas (disk vs annulus) of surface  $S = 500000\text{mm}^2$ . All results are computed over 64 simulation runs. We adjust the accuracy score to only show results when the simulated diffusion sessions in the swarm did not diverge: empty bins correspond to results where more than 50% of runs failed to converge. **Top:** results obtained after only 1 iteration. **Middle:** results obtained after 30 iterations. **Bottom:** Ranking of the values of the Final  $\lambda_2$  centroids for the disk arena.



Supplementary Figure 11: **Top:** Mean number of neighbors of agents (*i.e.*, mean degree of the communication graphs), and **Bottom:** mean number of connected graph components in the **sim-large-7** case, after 30 iterations.

## 5.2 Simulations: influence of the dispersion algorithm

We want to study how the dispersion algorithm and the initial position of the agents influence the behavior of the spectral swarm robotics algorithm and associated results in terms of shape classification accuracy. We consider the following three cases:

**sim-small-2-dispersion** where agents are initially packed at the center of the arena and move according to a run-and-tumble motion during both the “seeding” and the “short random walk” stages of the spectral swarm robotics algorithm. As such, agents first disperse throughout the arena and then move at the beginning of each iteration of the algorithm to change their neighbors.

**sim-small-2-random** where agents are initially placed at random positions of the arena and where they remain static: the “seeding” stage of the spectral swarm robotics algorithm is skipped and agent do not move during the “short random walk” stage. As such, they always keep the same neighbors for all iterations.

**sim-small-2-uniform** where agents are initially optimally distributed throughout the arena so that the distance between each pair of neighboring agents is the same across the entire swarm. This ensures that all parts of the arena are identically discretized by the communication graph of the agents. The optimal positions of the agents are found by using K-Means to find  $N$  clusters from a large (100000) number of points randomly placed in a space with the boundaries as the arenas.

All three cases are performed in simulation, with two arenas (disk and annulus) of surface  $S = 70000\text{mm}^2$  – in turn, we will use a smaller number of agents compared to the cases of the previous section (Supplementary Table 4).

While cases **sim-small-2-dispersion** and **sim-small-2-random** do not assume any prior knowledge about the geometry of the arenas, the **sim-small-2-uniform** case assumes that it is possible to distribute uniformly the agent in the arenas before the start of the algorithm. In experiments with Kilobots, this can be achieved by having the human experimenter positioning the robots so that they are evenly distributed before starting the experiment.

Supplementary Figure 12 shows the densities of agents after the seeding stage over 64 runs. In all cases, for both arenas, the agents reach all parts of the arenas, with no visible gap.

Supplementary Figure 13 shows representative examples of agents position after the “initial seeding” stage for the three cases.

Supplementary Figure 14 characterizes the spacial distribution of agents after the seeding stage for the three cases considered here. We consider the following three complementary statistics and assess their distribution using violin plots:

**Degrees** The distribution of degrees in the communication graphs. This metric only takes into account the communication graph  $\mathcal{G}$  itself, without considering the distance between agents.

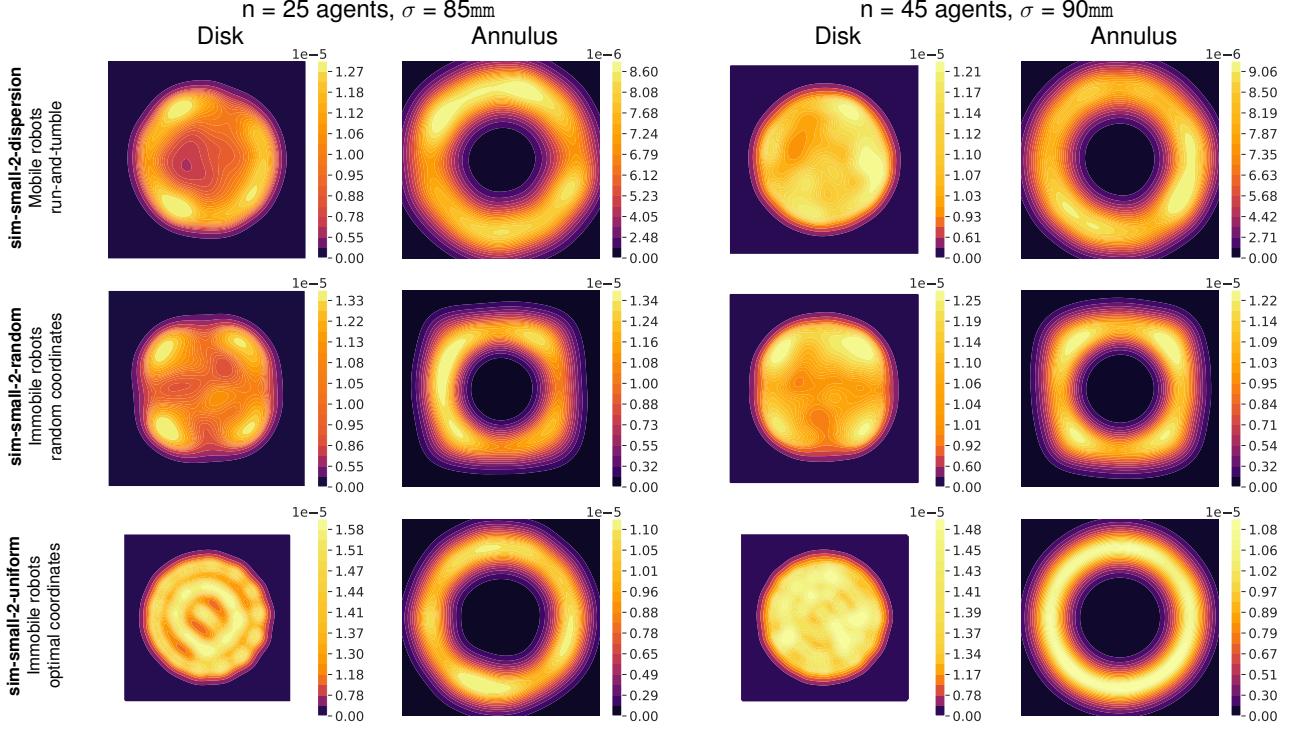
$d_{\min}/\sigma$  The distance between the focal agent and its closest neighbors divided by  $\sigma$ , the field of perception of the agents. This metric only takes into account the distance to one neighbor, without involving the communication graph  $\mathcal{G}$ .

**Cell Surface of Voronoi tessellation** The area of each cells obtained by computing a Voronoi tessellation of the positions of the agents in the arena (constrained by the geometry of the arena). This metric only takes into account the spatial size of neighborhoods, without involving the communication graph  $\mathcal{G}$ . The Voronoi tessellation is computed in our Python scripts using the Geopandas and shapely packages [62, 63].

In both configurations, the results of cases **sim-small-2-dispersion** and **sim-small-2-random** are generally similar for the degrees and cell surface metrics and differ from the results of case **sim-small-2-uniform**. The results of the  $d_{\min}/\sigma$  metric suggest that agents tend to be more packed together in the **sim-small-2-dispersion** and **sim-small-2-random** cases compared to the **sim-small-2-uniform** – this matches the examples of Supplementary Fig. 13. The cell surfaces in the uniform case has generally less variability compared to the other cases (at least for the annulus): indeed all agents are equidistant to their neighbors in this case. In turn, the discretization of the shape is more uniform in the **sim-small-2-uniform** case.

Note however, when the number of agents is sufficiently low, it is very possible to obtain a graph of agents with multiple graph components (cf Supplementary Fig. 13, *i.e.*, a graph that is not fully connected. This will have a different set of consequences depending on the case:

**sim-small-2-dispersion** The agents move and change their neighbors every iteration. Consequently, the graph components do not stay static over iterations: they may sometimes merge together or split; some agents may go from one component to another. In turn, the estimations of  $\lambda_2$  will be periodically transferred from component to component, making the system able to correctly (but imprecisely) estimate  $\lambda_2$  while sometimes having more than one graph component. That means that it is still possible to have a classification



Supplementary Figure 12: Density of agents in the arenas after the initial seeding, for cases with arena surface  $S = 70000\text{mm}^2$  and for two different configurations (either 25 or 45 agents). Results were obtained over 64 runs.

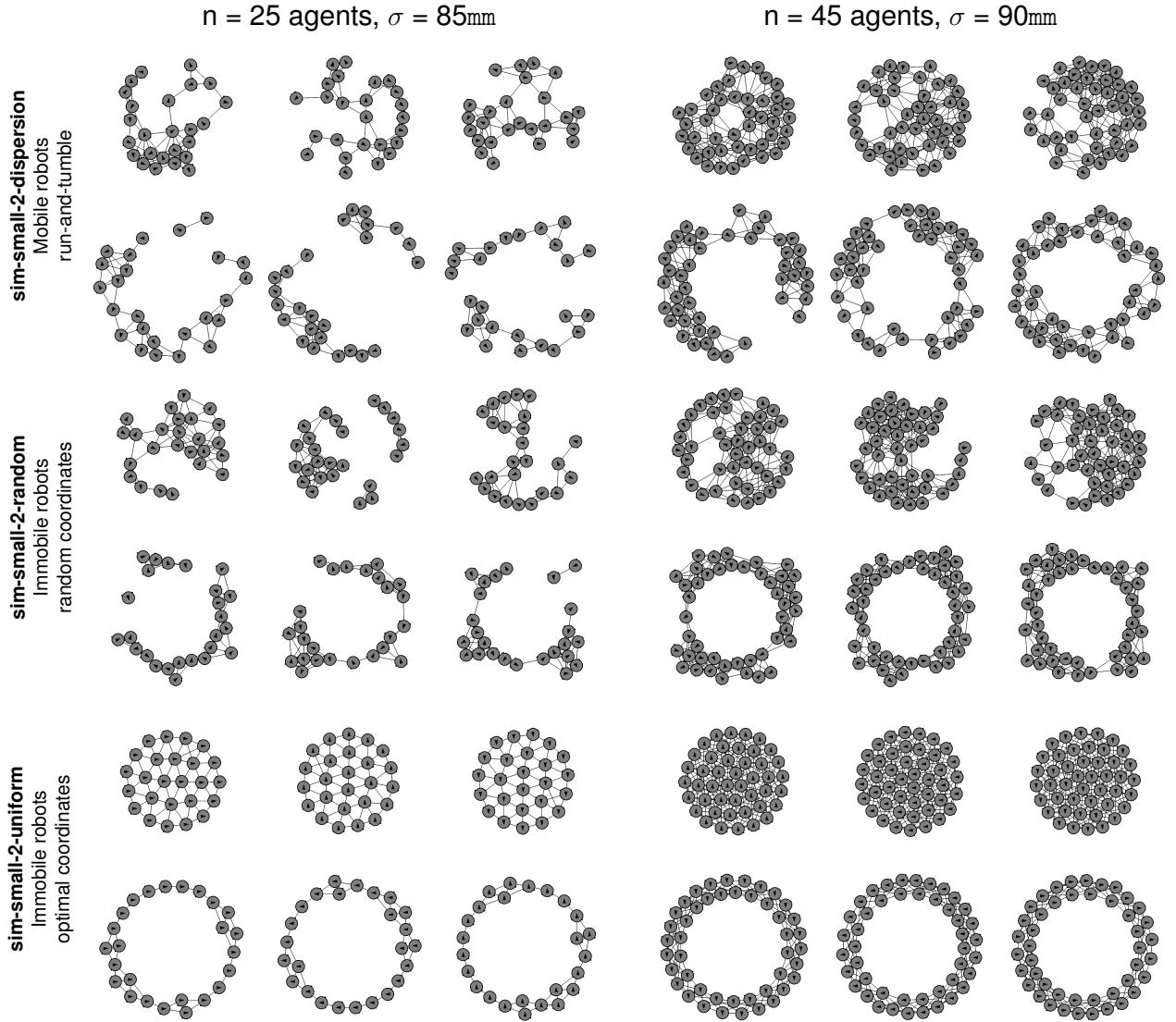
with high-performing accuracy even if there are more than one component. Of course, if the number of components is too high (*i.e.*, the number of agents is too low), such computation will be impossible: the diffusion sessions will diverge, and/or the information from one component to another won't be transferred, resulting in an inaccurate classification.

**sim-small-2-random** The agents are immobile and always keep the same neighbors. Therefore, isolated agents will always remain isolated and graph components will not be able to communicate with each other over multiple iterations. This will result in each component doing its own independent estimation of  $\lambda_2$ . As this estimation assumes that the agents are distributed across the entire arena, this process will almost always result in an incorrect (rather than imprecise) estimation, only taking into account the spaces covered by each component.

**sim-small-2-uniform** The agents are immobile and equidistributed in the arena. As such, if there are several components, it follows that there are actually  $N$  components, that is, the agents are all too distant from each other to communicate. In this case, all agents are isolated and the diffusion stages cannot be computed, resulting in undecided classification results.

These findings are corroborated by Supplementary Fig. 15, showing the difference in accuracy scores between the three cases after 10 iterations (the number of 10 iterations was selected so that `Final $\lambda_2$`  estimations have converged for all settings). The case **sim-small-2-dispersion** has generally higher (or at least equivalent) accuracy than the **sim-small-2-random** case: as explained before, it may be attributed to the capacity of the former to have agents that change neighbors across iterations. In particular, that is the case when  $\sigma$  is small ( $\leq 85\text{mm}$ , *i.e.*, the realistic range for Kilobots), placing the system below the optimal  $\alpha = 17$  hyperbole or to the right of this hyperbole: in these settings, the dispersion case having far higher scores than the random case.

Both the **sim-small-2-dispersion** and **sim-small-2-uniform** cases have similar (100%) accuracy scores close to the optimal  $\alpha = 17$  hyperbole (1). The latter has better scores than the former immediately below p (2), while the dispersion case outperforms the uniform case above and to the right of p (3), and with settings with low values of  $N$  ( $\leq 40$ ) and  $\sigma$  ( $\leq 85\text{mm}$ ) (4). The good results of the uniform case in (1) and (2) can be explained by considering an uniform distribution of agents in the arena: it both ensures that the graph is connected (except for settings where the agents are too far from each other) and that the discretization of the shape is uniform (each point occupies a Voronoi cell that all have similar size). However, the agents in

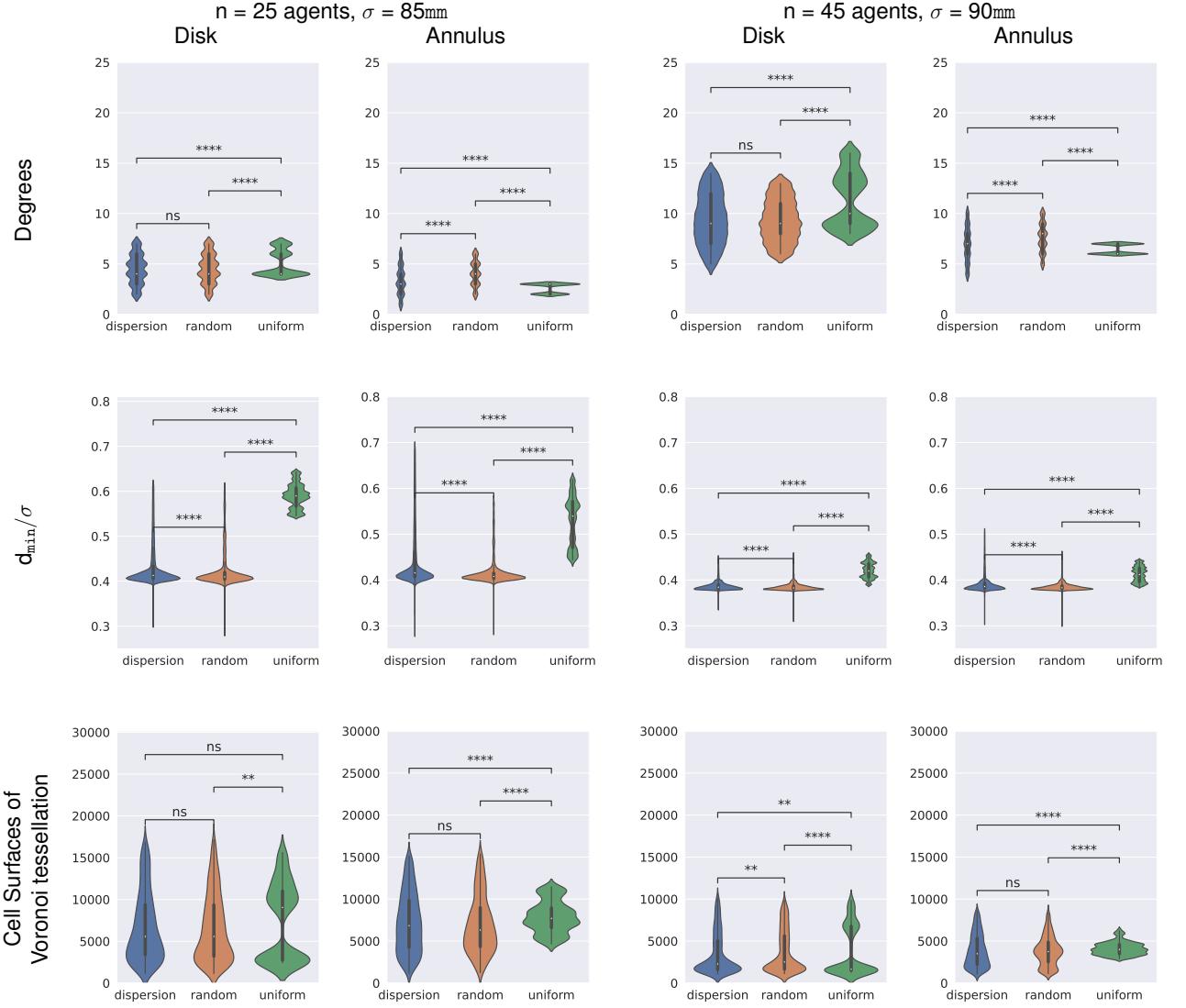


Supplementary Figure 13: Examples of position of the agents after the initial seeding, for cases with arena surface  $S = 70000\text{mm}^2$  and for two different configurations (either 25 or 45 agents).

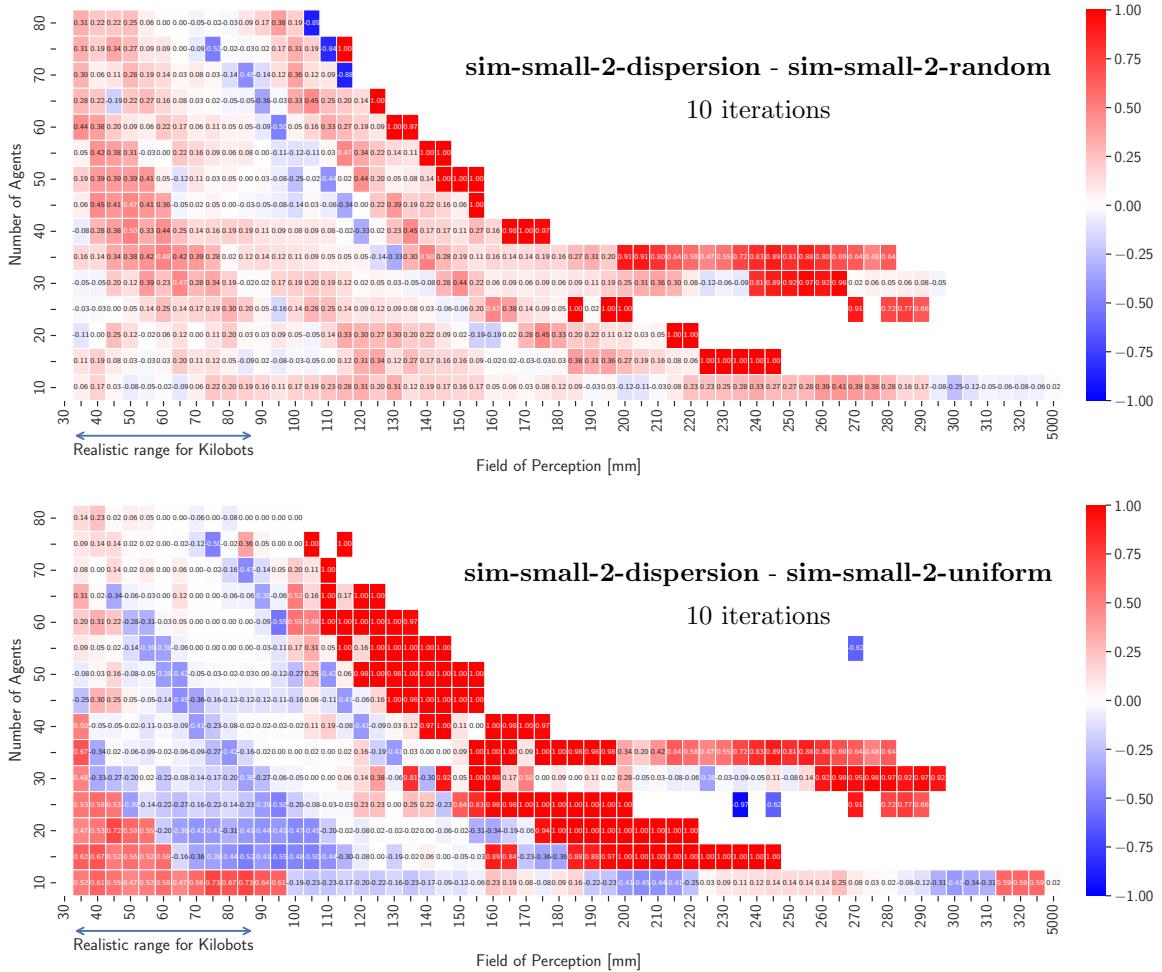
the uniform case do not move and always keep the same neighbors at each iteration; while this lowers the variability in  $\text{Final}\lambda_2$  estimations, it also makes the estimation less resilient to graph artifacts that could bias such estimations (*e.g.*, small number of agents diverging, affecting their entire neighborhood). This may also be the case in the (3) settings where the uniform case may have graphs that are too dense, increasing the probability of divergences and computation anomalies during the diffusion stages; this is alleviated in the dispersion case by changing neighbors at each iteration. In the (4) settings, the uniform distribution will have agents that are too far apart from each other to be able to communicate, while the dispersion case would still have clusters of agents sufficiently close to each other to estimate  $\text{Final}\lambda_2$ , possibly using only the local geometry, as the graph will be too small to match the entire surface of the arenas with those settings.

Finally, Supplementary Figs. 16, 17 and 18 show the results in accuracy after 1 and 10 iterations respectively for cases **sim-small-2-dispersion**, **sim-small-2-random** and **sim-small-2-uniform**. They also show the influence of the parameters  $N$  and  $\sigma$  on the ordering of the values of  $\text{Final}\lambda_2$  corresponding to the two arenas (cf Sec. 5.1 for details). The results are consistent with our previous analysis in Supplementary Fig. 15, with rankings showing demarcation lines between regimes.

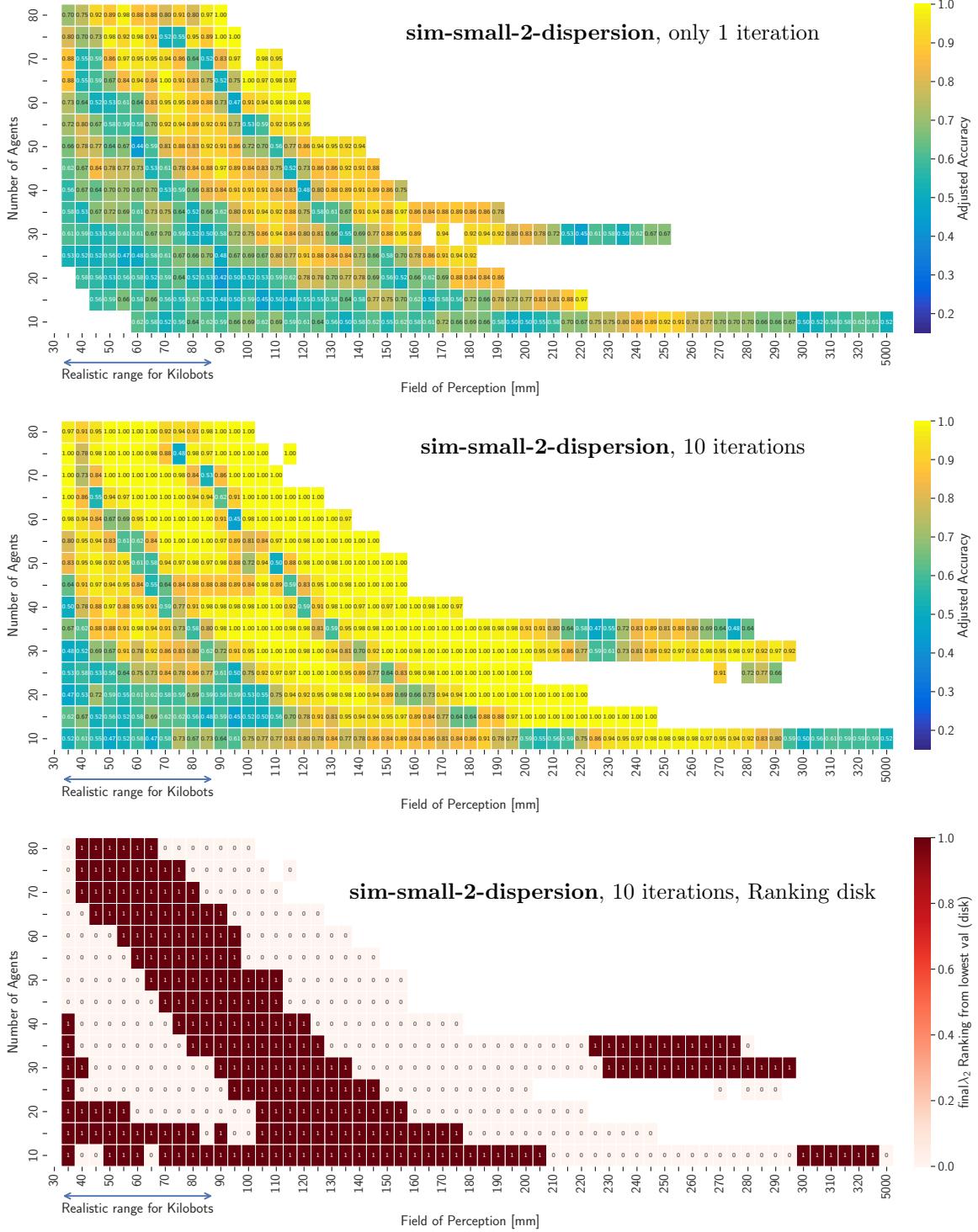
In the end, the **sim-small-2-dispersion** and **sim-small-2-uniform** are complementary and may be selected depending on  $N$  and  $\sigma$ . They exhibit similar results on the optimal  $\alpha = 17$  hyperbole.



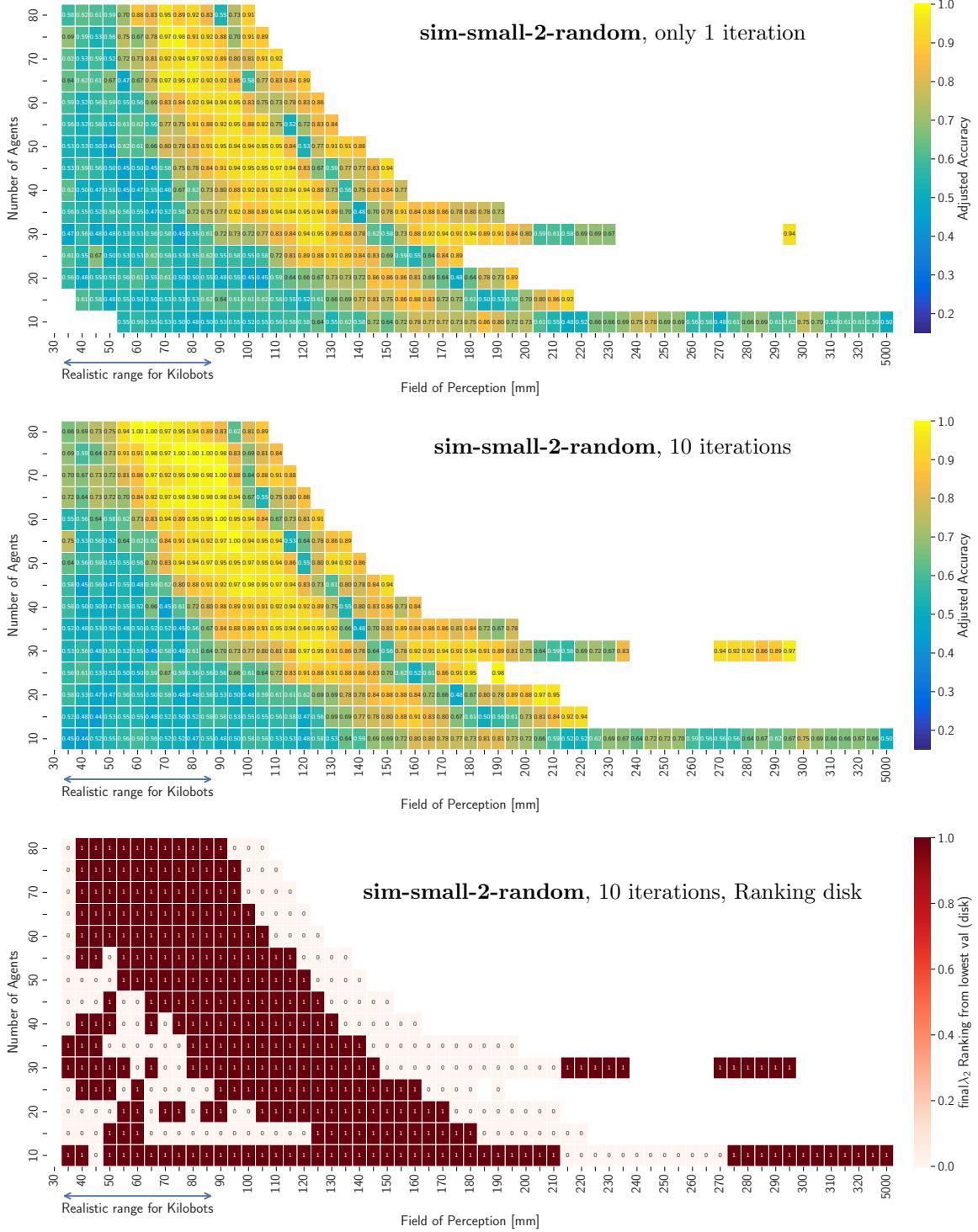
Supplementary Figure 14: Statistical analysis of the different types of agent dispersion in arenas of surface  $S = 70000\text{mm}^2$ . We consider the **sim-small-2-dispersion**, **sim-small-2-random**, and **sim-small-2-uniform** cases – named in the figure “dispersion”, “random” and “uniform” respectively. Two configurations are considered, using 25 and 45 agents respectively. Statistical significance is computed using the two-sided Mann-Whitney-Wilcoxon test with Bonferroni correction. Significant pairs are designated as follows: ns:  $0.05 < p \leq 1.00$ , \*:  $0.01 < p \leq 0.05$ , \*\*:  $0.001 < p \leq 0.01$ , \*\*\*:  $0.0001 < p \leq 0.001$ , \*\*\*\*:  $p \leq 0.0001$ .



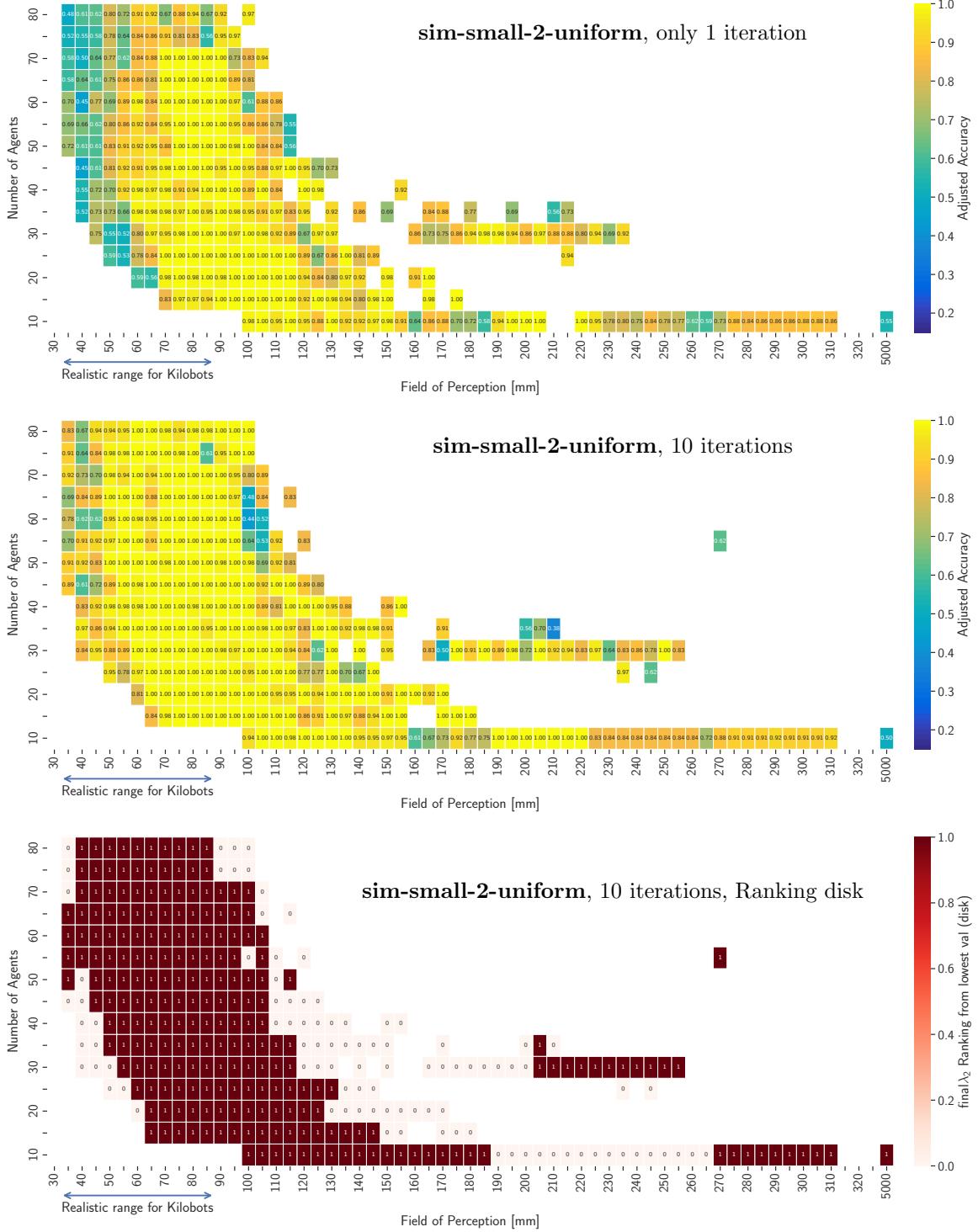
Supplementary Figure 15: Difference in accuracy scores between cases **sim-small-2-dispersion**, **sim-small-2-random** and **sim-small-2-uniform**, depending on the number of agents and their field of perception. All results are obtained in simulations over 10 iterations and over 64 runs. Each bin of the 2D histograms corresponds to the difference between the accuracy score of the **sim-small-2-dispersion** with the scores of either the **sim-small-2-random** (top) or the **sim-small-2-uniform** case (bottom). We adjust the accuracy score to only show results where the simulated diffusion sessions in the swarm do not diverge: empty bins correspond to results where there are more than 50% of the runs exhibiting divergences.



Supplementary Figure 16: Evolution of the accuracy depending on the number of agents and their field of perception, for the case in simulation **sim-small-2-dispersion**. In this case, we consider 2 arenas (disk vs annulus) of surface  $S = 70000\text{mm}^2$  and the robots use the run-and-tumble dispersion algorithm to move throughout the arenas between each iteration. All results are computed over 64 runs of simulation. We adjust the accuracy score to only show results where the simulated diffusion sessions in the swarm do not diverge: empty bins correspond to results where there are more than 50% of runs exhibiting divergences. **Top:** results with only 1 iteration. **Middle:** results obtained after 10 iterations. **Bottom:** Ranking of the order of values of the Final  $\lambda_2$  centroids for the disk arena.



Supplementary Figure 17: Evolution of the accuracy depending on the number of agents and their field of perception, for the case in simulation **sim-small-2-random**. In this case, we consider 2 arenas (disk vs annulus) of surface  $S = 70000mm^2$  and the robots always stay immobile during the simulation. The initial position of the robots inside the arenas are random (2D uniform distribution). All results are computed over 64 runs of simulation. We adjust the accuracy score to only show results where the simulated diffusion sessions in the swarm do not diverge: empty bins correspond to results where there are more than 50% of runs exhibiting divergences. **Top:** results after 1 iteration. **Middle:** results after 10 iterations. **Bottom:** Ranking of the values of the Final  $\lambda_2$  centroids for the disk arena.



Supplementary Figure 18: Evolution of the accuracy with respect to the number of agents and their field of perception, for the case in simulation **sim-small-2-uniform**. In this case, we consider 2 arenas (disk vs annulus) of surface  $S = 70000\text{mm}^2$  and the robots always stay immobile during the simulation. Robots are distributed uniformly in the arenas, so that they are equidistant to their neighbors (obtained through Voronoi tessellation). All results are computed over 64 runs of simulation. We adjust the accuracy score to only show results where the simulated diffusion sessions in the swarm do not diverge: empty bins correspond to results where there are more than 50% of runs exhibiting divergences. **Top:** results after one iteration. **Middle:** results after 10 iterations. **Bottom:** Ranking of the values of the Final  $\lambda_2$  centroids for the disk arena.

### 5.3 Experimental runs with Kilobots & Supplementary Videos

All Kilobots experiments involve the experimental case **expe-2** described in Supplementary Table 4. Individual runs and obtained classification results are listed in Supplementary Table 6. The raw video files of all experiments are made available at the following link: <https://e.pcloud.link/publink/show?code=kZY21SZoia38Rb7tFj0mXK4jwqqmJe18Kmk>.

Additionally, we provide three explanatory videos as supplementary materials, described as follow:

Video **SI1** (<https://e.pcloud.link/publink/show?code=XZdVJjZXeAay878b8ysf33nFmRnP7mJy80X>) presents the results of the **expe-2** case, involving experiments with 25 robots in arenas of surface equals to  $S = 70000\text{mm}^2$ . First, we display the observed behavior of the DDSA algorithm on one experimental run with the disk and annulus arenas. This is illustrated by a time-lapse showing the active LED colors of each robots, both in the “diffusion” and “ $\lambda_2$  estimation” stages of the algorithm. Then, we concurrently present the results obtained through all experimental runs of Supplementary Table 6.

Video **SI2** (<https://e.pcloud.link/publink/show?code=XZIVJjZ3UkDoIU0aKkwM0VJxz00WSFHai77>) compares experimental results of case **expe-2** with simulation results of case **sim-2**. Both experimental and simulated agents are driven by the same controllers implementing the spectral swarm robotics algorithm on a Kilobot architecture. In either cases, we observe similar dynamics, for the “diffusion” (swarm exhibit a correct partitioning) and “ $\lambda_2$  estimation” (swarm correctly classify arenas) stages.

Video **SI3** (<https://e.pcloud.link/publink/show?code=XZGVJjZ7701LvCtiY7URffCtgVCq88zhwXV>) showcases swarm behavior in the **sim-large-7** case, with parameters of regime r3 (shown in Main Text Figs. 2 and 3). It involves 300 agents with a field of perception of  $85\text{mm}$  and 7 arenas of surface equals to  $S = 500000\text{mm}^2$ . This video presents a time lapse similar to the one in Main Text Fig. 2, both for the “diffusion” and “ $\lambda_2$  estimation” stages of the algorithm.

Exp. number	Associated video file	Disk classification	Annulus classification	Additional notes
0	2022-07-21_14-16-04.mp4	<b>Disk</b>	<b>Annulus</b>	Disk LED color is orange instead of cyan (early version of the color set), code, parameters, centroids are the same as other experiments.
1	2022-07-21_15-47-12.mp4	<b>Disk</b>	<b>Annulus</b>	-
2	2022-07-22_10-18-21.mp4	<b>Disk</b>	<b>Annulus</b>	-
3	2022-07-22_11-50-10.mp4	<b>Annulus</b>	<b>Annulus</b>	-
4	2022-07-22_13-33-07.mp4	<b>Disk</b>	Disk	-
5	2022-07-22_15-25-54.mp4	<b>Disk</b>	<b>Annulus</b>	-
6	2022-07-22_17-06-29.mp4	<b>Disk</b>	Disk	-
7	2022-09-15_15-45-26.mp4	<b>Disk</b>	Disk	-
8	2022-09-16_10-38-16.mp4	<b>Disk</b>	<b>Annulus</b>	-
9	2022-09-16_12-42-32.mp4	<b>Annulus</b>	<b>Annulus</b>	-
10	2022-09-19_10-41-44.mp4	<b>Annulus</b>	<b>Annulus</b>	-
11	2022-09-19_12-14-47.mp4	<b>Disk</b>	<b>Annulus</b>	-
12	2022-09-19_13-43-33.mp4	<b>Disk</b>	Disk	-
13	2022-09-19_15-24-21.mp4	<b>Disk</b>	<b>Annulus</b>	-
14	2022-09-19_16-58-24.mp4	<b>Annulus</b>	<b>Annulus</b>	-

Supplementary Table 6: List of experimental runs and names of associated video files. Each experiment involved two arenas: Disk (color code: cyan) and Annulus (color code: violet). Correctly detected arenas are in bold. Note that experiment 0 used the color code “orange” rather than cyan, as we initially intended to use the former to indicate a disk arena. We changed the color code to violet to ease visualization. However, the results of this experiment remain correct, as they used the same code, parameters and centroids as the other 14 experiments. Raw video files can be found at <https://e.pcloud.link/publink/show?code=kZY21SZoia38Rb7tFj0mXK4jwqqmJe18Kmk>

## References

- [1] M Rubenstein, C Ahler, and R Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE international conference on robotics and automation*, pages 3293–3298. IEEE, 2012.
- [2] M Rubenstein, A Cornejo, and R Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 2014.

- [3] H Hamann. *Swarm robotics: A formal approach*, volume 221. Springer, 2018.
- [4] CR Kube and H Zhang. Collective robotics: From social insects to robots. *Adaptive Behavior*, 1993.
- [5] R Pfeifer, M Lungarella, and F Iida. Self-organization, embodiment, and biologically inspired robotics. *Science*, 2007.
- [6] J Werfel, K Petersen, and R Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 2014.
- [7] I Slavkov, D Carrillo-Zapata, N Carranza, X Diego, F Jansson, J Kaandorp, S Hauert, and J Sharpe. Morphogenesis in robot swarms. *Science Robotics*, 2018.
- [8] C Virág, G Vásárhelyi, N Tarcai, T Szörényi, G Somorjai, T Nepusz, and T Vicsek. Flocking algorithm for autonomous flying robots. *Bioinspiration & biomimetics*, 9(2):025012, 2014.
- [9] M Coppola, KN McGuire, C De Wagter, and G De Croon. A survey on swarming with micro air vehicles: Fundamental challenges and constraints. *Frontiers in Robotics and AI*, 7:18, 2020.
- [10] N Bredeche and N Fontbonne. Social learning in swarm robotics. *Philosophical Transactions of the Royal Society B*, 2022.
- [11] JF Boudet, J Lintuvuori, C Lacouture, T Barois, A Deblais, K Xie, S Cassagnere, B Tregon, DB Brückner, JC Baret, et al. From collections of independent, mindless robots to flexible, mobile, and directional superstructures. *Science Robotics*, 6(56):eabd0272, 2021.
- [12] A Reina, E Ferrante, and G Valentini. Collective decision-making in living and artificial systems. *Swarm Intelligence*, 15(1-2):1–6, 2021.
- [13] CW Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH*, 1987.
- [14] T Vicsek, A Czirók, E Ben-Jacob, I Cohen, and O Shochet. Novel type of phase transition in a system of self-driven particles. *Physical review letters*, 1995.
- [15] T Vicsek and A Zafeiris. Collective motion. *Physics Reports*, 2012.
- [16] C Detrain and JL Deneubourg. Collective decision-making and foraging patterns in ants and honeybees. *Advances in insect physiology*, 2008.
- [17] JW Bradbury, SL Vehrencamp, et al. Principles of animal communication. 132, 1998.
- [18] EO Wilson. *Sociobiology: The New Synthesis, Twenty-Fifth Anniversary Edition*. Harvard University Press, 2000.
- [19] A Attanasi, A Cavagna, L Del Castello, I Giardina, TS Grigera, A Jelić, S Melillo, L Parisi, O Pohl, E Shen, et al. Information transfer and behavioural inertia in starling flocks. *Nature Physics*, 2014.
- [20] S Camazine, JL Deneubourg, NR Franks, J Sneyd, G Theraula, and E Bonabeau. Self-organization in biological systems. In *Self-organization in biological systems*. Princeton university press, 2020.
- [21] AM Turing. The chemical basis of morphogenesis. *Bulletin of mathematical biology*, 52(1-2):153–197, 1990.
- [22] LC Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.
- [23] W Arendt, R Nittka, W Peter, and F Steiner. Weyl’s law: Spectral properties of the laplacian in mathematics and physics. *Mathematical analysis of evolution, information, and complexity*, pages 1–71, 2009.
- [24] D Spielman. Spectral graph theory. *Combinatorial scientific computing*, 18, 2012.
- [25] FRK Chung. *Spectral graph theory*, volume 92. American Mathematical Society, 1997.
- [26] M Kac. Can one hear the shape of a drum? *The American Mathematical Monthly*, 1966.
- [27] KF Riley, MP Hobson, and SJ Bence. Mathematical methods for physics and engineering, 1999.
- [28] M Reuter, FE Wolter, and N Peinecke. Laplace–beltrami spectra as ‘Shape-DNA’ of surfaces and solids. *Computer-Aided Design*, 2006.
- [29] M Belkin, P Niyogi, and V Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 2006.

- [30] RM Rustamov et al. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Symposium on geometry processing*, volume 257, pages 225–233, 2007.
- [31] A Ng, M Jordan, and Y Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.
- [32] U Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.
- [33] M Fiedler. Laplacian of graphs and algebraic connectivity. *Banach Center Publications*, 1(25):57–70, 1989.
- [34] Y Wang and J Solomon. Intrinsic and extrinsic operators for shape analysis. In *Handbook of numerical analysis*, volume 20, pages 41–115. Elsevier, 2019.
- [35] MS Talamali, A Saha, JAR Marshall, and A Reina. When less is more: Robot swarms adapt better to changes with constrained communication. *Science Robotics*, 6(56):eabf1416, 2021.
- [36] M Gauci, ME Ortiz, M Rubenstein, and R Nagpal. Error cascades in collective behavior: a case study of the gradient algorithm on 1000 physical agents. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1404–1412, 2017.
- [37] H Wang and M Rubenstein. A fast, accurate, and scalable probabilistic sample-based approach for counting swarm size. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7180–7185. IEEE, 2020.
- [38] C Gordon, DL Webb, and S Wolpert. One cannot hear the shape of a drum. *Bulletin of the American Mathematical Society*, 27(1):134–138, 1992.
- [39] MY Ben Zion, J Fersula, N Bredeche, and O Dauchot. Morphological computation and decentralized learning in a swarm of sterically interacting robots. *Science Robotics*, 8(75):eab06140, 2023.
- [40] DI Shuman, SK Narang, P Frossard, A Ortega, and P Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.
- [41] M Divband Soorati, P Zahadat, J Ghofrani, and H Hamann. Adaptive path formation in self-assembling robot swarms by tree-like vascular morphogenesis. In *Distributed Autonomous Robotic Systems: The 14th International Symposium*, pages 299–311. Springer, 2019.
- [42] F Pratissoli, A Reina, Y Kaszubowski Lopes, C Pincioli, G Miyauchi, L Sabattini, and R Groß. Coherent movement of error-prone individuals through mechanical coupling. *Nature Communications*, 14(1):4063, 2023.
- [43] M Raoufi, P Romanczuk, and H Hamann. Estimation of continuous environments by robot swarms: Correlated networks and decision-making. *arXiv preprint arXiv:2302.13629*, 2023.
- [44] JR Baxter and RV Chacon. The equivalence of diffusions on networks to brownian motion. *Contemporary Mathematics*, 26(33-47):0546–60080, 1984.
- [45] MI Freidlin and AD Wentzell. Diffusion processes on graphs and the averaging principle. *The Annals of probability*, pages 2215–2245, 1993.
- [46] JJ Moliterno. *Applications of combinatorial matrix theory to Laplacian matrices of graphs*. CRC Press, 2016.
- [47] P Holme and J Saramäki. Temporal networks. *Physics reports*, 519(3):97–125, 2012.
- [48] B Lévy. Laplace-beltrami eigenfunctions towards an algorithm that “understands” geometry. In *IEEE SMI*, 2006.
- [49] AJ Chaudhari, RM Leahy, BL Wise, NE Lane, RD Badawi, and AA Joshi. Global point signature for shape analysis of carpal bones. *Phys. Med. Biol.*, 2014.
- [50] H Zhang, O Van Kaick, and R Dyer. Spectral mesh processing. In *Computer graphics forum*, volume 29, pages 1865–1894. Wiley Online Library, 2010.
- [51] S Melzi, E Rodolà, U Castellani, and MM Bronstein. Localized manifold harmonics for spectral shape analysis. In *Computer Graphics Forum*, volume 37, pages 20–34. Wiley Online Library, 2018.

- [52] Y Aflalo, H Brezis, and R Kimmel. On the optimality of shape and data representation in the spectral domain. *SIAM Journal on Imaging Sciences*, 8(2):1141–1160, 2015.
- [53] Y Choukroun, A Shtern, A Bronstein, and R Kimmel. Hamiltonian operator for spectral shape analysis. *IEEE transactions on visualization and computer graphics*, 26(2):1320–1331, 2018.
- [54] A Singer. From graph to manifold laplacian: The convergence rate. *Applied and Computational Harmonic Analysis*, 21(1):128–134, 2006.
- [55] HC Berg and DA Brown. Chemotaxis in escherichia coli analysed by three-dimensional tracking. *Nature*, 239:500–504, 1972.
- [56] SH Larsen, RW Reader, EN Kort, WW Tso, and J Adler. Change in direction of flagellar rotation is the basis of the chemotactic response in escherichia coli. *Nature*, 249(5452):74–77, 1974.
- [57] F Jansson, M Hartley, M Hinsch, I Slavkov, N Carranza, TSG Olsson, RM Dries, JH Grönqvist, AFM Marée, J Sharpe, et al. Kilobot: a kilobot simulator to enable effective research in swarm robotics. *arXiv preprint arXiv:1511.04285*, 2015.
- [58] CR Harris, KJ Millman, SJ van der Walt, R Gommers, P Virtanen, D Cournapeau, E Wieser, J Taylor, S Berg, NJ Smith, R Kern, M Picus, S Hoyer, MH van Kerkwijk, M Brett, A Haldane, J Fernández del Río, M Wiebe, P Peterson, P Gérard-Marchant, K Sheppard, T Reddy, W Weckesser, H Abbasi, C Gohlke, and TE Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.
- [59] W McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.
- [60] A Hagberg and D Conway. Networkx: Network analysis with python. *URL: https://networkx.github.io*, 2020.
- [61] P Virtanen, R Gommers, TE Oliphant, M Haberland, T Reddy, D Cournapeau, E Burovski, P Peterson, W Weckesser, J Bright, SJ van der Walt, M Brett, J Wilson, KJ Millman, N Mayorov, ARJ Nelson, E Jones, R Kern, E Larson, CJ Carey, İ Polat, Y Feng, EW Moore, J VanderPlas, D Laxalde, J Perktold, R Cimrman, I Henriksen, EA Quintero, CR Harris, AM Archibald, AH Ribeiro, F Pedregosa, P van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.
- [62] K Jordahl, J Van den Bossche, M Fleischmann, J Wasserman, J McBride, J Gerard, J Tratner, M Perry, AG Badaracco, C Farmer, GA Hjelle, AD Snow, M Cochran, S Gillies, L Culbertson, M Bartos, N Eubank, A Bilogur, S Rey, C Ren, D Arribas-Bel, L Wasser, LJ Wolf, M Journois, J Wilson, A Greenhall, C Holdgraf, and F Leblanc. Geopandas: v0.8.1, July 2020.
- [63] S Gillies et al. Shapely: Manipulation and analysis of geometric objects, 2007–.
- [64] JD Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.
- [65] ML Waskom. Seaborn: Statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.