COLLECTIVE BEHAVIOR

Morphogenesis in robot swarms

I. Slavkov[1,2]*, D. Carrillo-Zapata[3,4,5]*, N. Carranza[1,2], X. Diego[1,2,6], F. Jansson[7,8], J. Kaandorp[8], S. Hauert[3,5], J. Sharpe[1,2,6,9]†

Morphogenesis allows millions of cells to self-organize into intricate structures with a wide variety of functional shapes during embryonic development. This process emerges from local interactions of cells under the control of gene circuits that are identical in every cell, robust to intrinsic noise, and adaptable to changing environments. Constructing human technology with these properties presents an important opportunity in swarm robotic applications ranging from construction to exploration. Morphogenesis in nature may use two different approaches: hierarchical, top-down control or spontaneously self-organizing dynamics such as reaction-diffusion Turing patterns. Here, we provide a demonstration of purely self-organizing behaviors to create emergent morphologies in large swarms of real robots. The robots achieve this collective organization without any self-localization and instead rely entirely on local interactions with neighbors. Results show swarms of 300 robots that self-construct organic and adaptable shapes that are robust to damage. This is a step toward the emergence of functional shape formation in robot swarms following principles of self-organized morphogenetic engineering.

INTRODUCTION

Whereas human technology is typically constructed by an external builder (humans or robots), most spatially organized biological systems dynamically create their own physical shapes. This process of shape formation is called morphogenesis, and it occurs in a distributed, self-organized, and emergent manner. For example, collectives of insects, such as ants, construct bridges to traverse terrains (Fig. 1A, i). Organisms such as slime mold and bacteria create colonies with regular spatial geometries to optimize nutrient transport and consumption (Fig. 1A, ii and iii). Multicellular organisms provide the most impressive example of morphogenesis, where massive collections of cells combine and actively collaborate during embryo development to build complex tissues and organs (Fig. 1A, iv). Having a functional shape and organization is important for survival because it allows organisms to inhabit certain ecological niches and thrive in given environments.

Two broad principles of spatial patterning exist in biological morphogenetic systems (*1*). Segmentation of the *Drosophila* embryo is a paradigmatic example of the first principle, in which each segment is genetically controlled individually (*2*), whereas the patterning of mouse digits is an example of the second principle, where each digit is a repetition of the same local process (*3*).

1) Top-down control. In some tissues, cells first access information about their location and then make cell fate choices according to this positional information (*4*). The control system is distributed—all cells have the same regulatory circuits or genetic program—but the positional information is achieved by means of an effective coordinate system, which may be created by molecular gradients or other mechanisms (*5*, *6*).

2) Local self-organization. As an alternative to positional information, spatial patterning may be controlled by purely local self-organization—spontaneous symmetry breaking processes, such as chemical reaction-diffusion (RD) systems (*7*). The mathematics of such processes [such as Turing patterns (*8*)] has been extensively studied over the past half century (*9*). These processes can only produce relatively simple periodic patterns, but they do so without the cells requiring access to any positional information and, because of their reliance on feedback mechanisms, are very robust to noise.

The recent new field of morphogenetic engineering introduces the principles of natural morphogenesis into human-engineered systems (*10*). More specifically, it uses the distributed control paradigm of developing tissues to program the generation of structures that both are robust and show predictable behavior. Ideally, such systems should display a high degree of autonomy, self-regulation, and some degree of active self-repair or regeneration in the case of damage. In the area of swarm robotics, where the swarm consists of simple identical robots, a key challenge is to design control algorithms for achieving complex behaviors and shapes based on robots interacting only with their local environment and their neighbors. Swarm morphogenesis might be achieved by the above principle 1 or 2 or a combination of the two. Top-down approaches (principle 1) could have the advantage of creating any arbitrary shape, whereas the self-organized approaches (principle 2), although more limited in the patterns they can create, would have the advantage of being emergent, naturally scalable, robust to failure of individual agents, and flexible, i.e., exhibiting the type of swarm intelligence seen in natural swarms (*11*).

Potential applications of such morphogenetic approaches in swarm engineering are numerous: self-constructing buildings that naturally adjust their structure to the geometry of their location, reconfigurable robots that adapt their shape for different tasks, and self-organized swarms (*12*) for mapping or search and environmental monitoring. Nanomedicine could also benefit from having self-organizing swarms of nanoparticles for more efficient drug targeting and delivery (*13*). Ultimately, machines functioning in this way could achieve dynamically changing physical structures—programmable matter (*14*, *15*)—and hence would open up a whole new world of machinery.

The problem of controlling the configuration of a group of robots has been receiving increasing interest, although mostly in theoretical studies based on simulations rather than real robot swarms. Target behaviors for the swarm have included collective navigation and flocking, trail formation, seizing and enclosing a target, and gap crossing but rarely focused on controlling the shape of the swarm per se. Control systems have included rule-based schemes (*16*–*18*), density-based schemes (*19*, *20*), or attraction/repulsion based on simple signals or

[1]Centre for Genomic Regulation (CRG), Barcelona Institute of Science and Technology, Barcelona, Spain. [2]Universitat Pompeu Fabra (UPF), Barcelona, Spain. [3]University of Bristol, Bristol, UK. [4]University of the West of England, Bristol, UK. [5]Bristol Robotics Laboratory, Bristol, UK. [6]EMBL Barcelona, Barcelona, Spain. [7]Centrum Wiskunde & Informatica (CWI), Amsterdam, Netherlands. [8]University of Amsterdam, Amsterdam, Netherlands. [9]Institució Catalana de Recerca i Estudis Avançats (ICREA), Barcelona, Spain.
*These authors contributed equally to this work.
†Corresponding author. Email: james.sharpe@embl.es

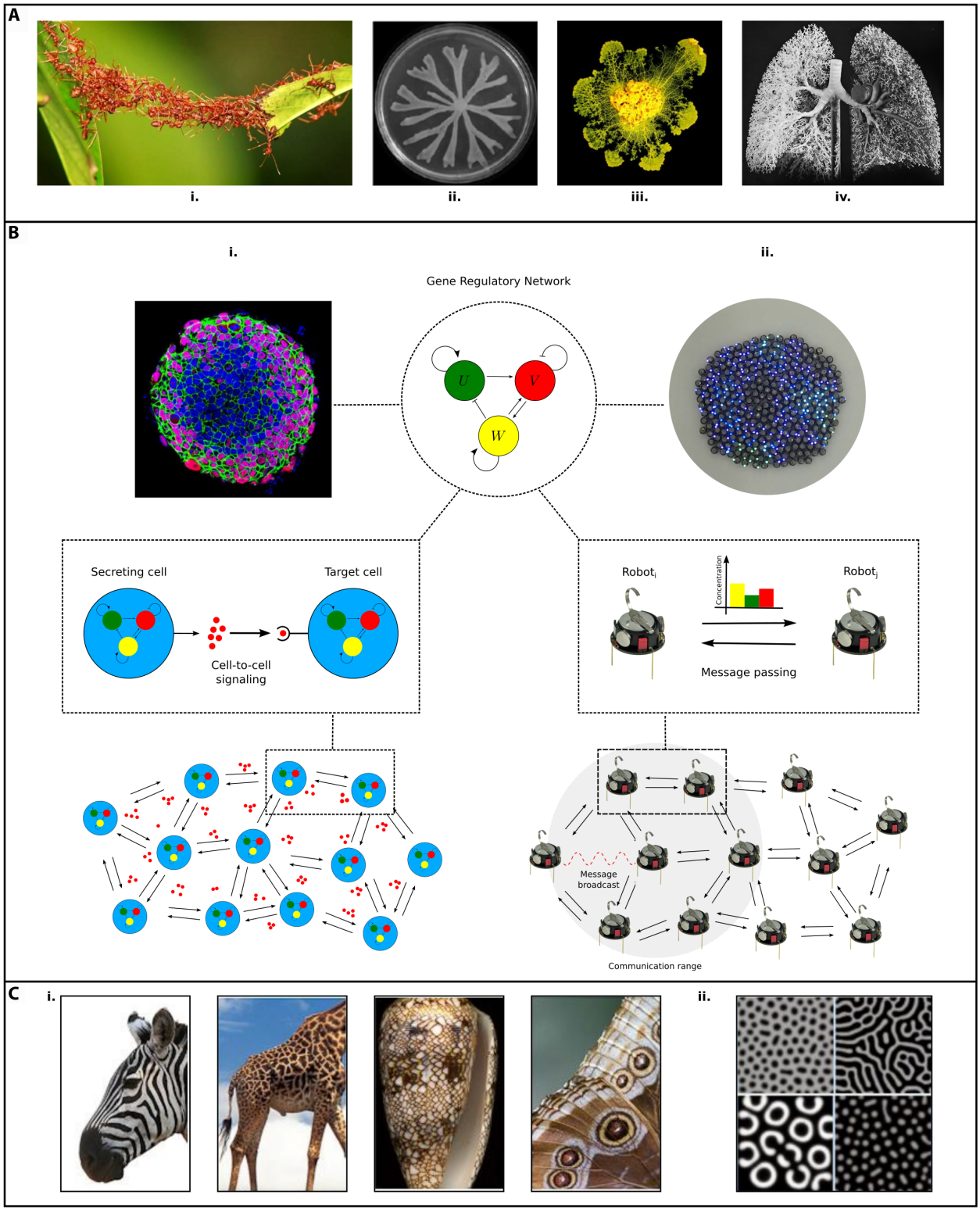


Fig. 1. Morphogenesis in natural systems. (**A**) (i) Fire ants constructing bridges. (ii) Bacterial colony structures formed by swarming. (iii) Slime mold network for optimal nutrient transport. (iv) Lungs consisting of a large network of alveoli for respiration. (**B**) GRN as the underlying mechanism behind patterning and morphogenesis processes in real tissues (e.g., heart valve) or robot swarms. (i) Each individual cell has an identical GRN, and cells communicate by secreting morphogens or direct cell-to-cell communication. A multicellular tissue consists of many cells that are interconnected and communicate with each other, thus allowing for coordinated tissue behavior. (ii) Robots emulate this behavior by running the same GRN and communicate with each other by sending messages about their GRN state. (**C**) (i) Turing patterns in different biological organisms: zebra, giraffe, seashell, and butterfly. (ii) Different types of Turing patterns on fish skin.

gradients (*21*–*29*) or have used more complicated interactions between signals, such as RD systems (*30*, *31*), gene regulatory networks (GRNs) (*32*–*35*), and swarm chemistry (*36*) [for a more extensive review, see (*37*)].

However, most previous work required precise motion or sensing abilities such as measuring angles to neighbors. Furthermore, when adaptability to different scenarios was tested with real robots (*16*–*18*, *22*, *28*, *30*, *32*, *33*), no more than 30 agents were used. In particular, properties such as self-healing of swarm morphologies have mostly been tested in simulations (*20*, *24*, *31*–*33*, *36*), with some exceptions (*16*). Because validation has been mainly simulation based, or using few real robots, it is unclear whether self-organized morphogenesis algorithms proposed so far would cross the reality gap and scale up in a large swarm of simple, noisy robots.

A significant breakthrough in the field of swarm robotics was made by Rubenstein *et al.* (*38*) when they created the kilobot, a minimal, low-cost robot designed to enable swarm experiments in large numbers. Two years later, they specifically demonstrated the shape formation capabilities of this robotic platform (*39*), in which a swarm of 1024 robots successfully arranged itself into predefined morphologies, such as a starfish shape, in a decentralized manner. This impressive result was achieved with only local communication between neighboring robots. However, it depended on the hierarchical control principle mentioned above (principle 1)—each robot had an explicit image of the final shape that should be created, and every robot had access to a coordinate system constructed by the kilobots themselves, such that each robot knew its relative position within the swarm. The shapes were thus not fully emergent (principle 2 above), placing limitations on their ability to be adaptable, scalable, and robust. A more recent study from the same group extended the approach to create swarm shapes by "disassembly" of the swarm using a light attraction/repulsion system (*40*), but it still relied on the same top-down approach.

Here, we chose a specific biological inspiration to address the problem of shape formation, namely, spontaneous self-organized patterning (principle 2) that occurs in some examples of multicellular tissue development. Although the many cells in a tissue do different things (e.g., becoming different cell types or migrating in different directions), they all contain the same GRN—the same genomic "program" (Fig. 1B, i and ii). Biological cells continually sense their neighborhood and communicate with other cells with the help of signaling molecules, thus creating an interconnected network. The design of the GRN leads to spatially nonuniform patterns of gene activity in which different genes are activated in different cells in a coordinated manner. These molecular patterns are then responsible for directing secondary processes—coordinated cell movement (migration), tissue proliferation (cell replication), or apoptosis (cell death)—that physically shape the tissue. Because our individual robots cannot replicate, our goal here was to implement swarm morphogenesis based only on cell movements (migration), which is known to drive a number of well-studied developmental cases, for example, gastrulation.

The key goal was to achieve simple biologically inspired morphologies by purely emergent self-organized morphogenesis (approach 2). This would allow morphogenesis without the robots needing to determine their locations, thus showing a higher degree of adaptability and robustness. Drawing directly on inspiration from developmental systems, we chose to use RD circuits and, in particular, Turing systems, because they have recently been shown to underlie a number of developmental models (Fig. 1C, i) (*41*–*43*). RD circuits describe a system of interacting molecular species (such as diffusible proteins encoded by the gene circuit) that can encode a genuinely symmetry-breaking reaction that produces spatial patterns of spots or stripes (Fig. 1C, ii). Although such patterns are limited to periodic arrangements, in nature, they have led to morphologies with a variety of useful functions. Their lack of dependence on positional information removes the potential errors that such self-localization mechanisms would experience. Furthermore, we explored whether feedback could be observed between patterning and tissue movement. In other words, could examples be found in which robot movements would be driven by the "molecular" pattern, but where the consequent alterations in swarm shape would also feed back to alter the pattern? This is a studied phenomenon in development, known as morphodynamic patterning (*44*, *45*), and believed to provide intrinsic adaptability and self-repairing behavior. These results were achieved in a real but very simple swarm technology, in which the shape-forming behavior of the collective swarm was reliable, even when the behavior of the individual robots was relatively unreliable, thus requiring the collective whole to be greater than the sum of its parts.

RESULTS

The swarm robotic platform that we used for our work is the kilobot (*38*). Kilobots are minimal robots designed to enable swarm experiments in large numbers. Three main functionalities were used: movement, robot-to-robot communication, and a multicolor light-emitting diode (LED) for experimental monitoring (Fig. 2A). In kilobots, locomotion is achieved by two motors that generate vibrations for the legs, resulting in nonholonomic movement with a large amount of noise. Communication is performed by passing infrared messages and is limited in range (see Materials and Methods for more details). Noisy distance measurements to neighbors can directly be extracted through communication but not angles, i.e., when receiving a message, a robot cannot detect from which direction the message arrived. The lack of directional sensing and the degree of noise in motion and communication were ideal to demonstrate robust self-organized collective behavior (morphogenesis) even with robots that are simple and unreliable—a key goal for robust but economical technologies.

The morphogenetic mechanism explored was the interaction of the two activities described above (Fig. 2B): Pattern formation driven by GRNs and migration or "tissue movement," the collective motion of the individual kilobots responsible for reshaping the swarm morphology. These two processes happen simultaneously (not sequentially), thus directly interacting with each other to achieve dynamic morphogenesis.

For the first part, pattern formation, the Turing system was used as the underlying mechanism, comprising a GRN of two virtual molecules, U and V, conceptually represented in Fig. 2C (i). Each robot kept track of its own concentrations of U and V, and the interactions between the two molecules were configured as an activator-inhibitor network, where molecule U acted as an activator and molecule V acts as an inhibitor. The change in concentration of each molecule was given by the RD equations

$$\frac{\partial u}{\partial t} = Rf(u,v) + D_u\nabla^2 u$$
$$\frac{\partial v}{\partial t} = Rg(u,v) + D_v\nabla^2 v$$
$$f(u,v) = (Au + Bv + C) - \gamma_u u$$
$$g(u,v) = (Eu - F) - \gamma_v v$$

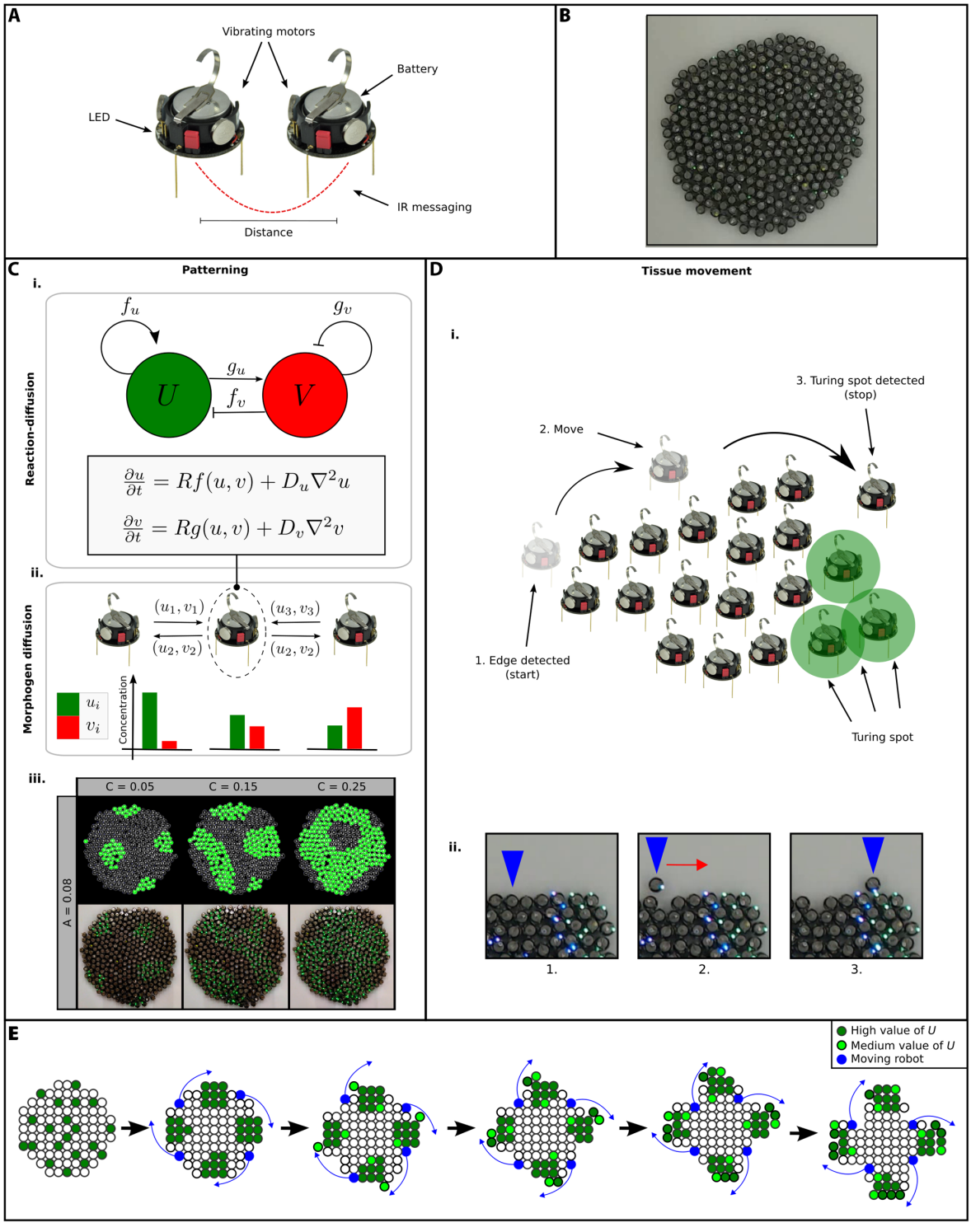


Fig. 2. Swarm morphogenesis approach description. (**A**) Kilobots are small robots, each containing a microprocessor, IR receiver/transmitter, a battery, a multi-colored LED, and two vibration motors. (**B**) Top view of a kilobot swarm consisting of ~300 robots. (**C**) A Turing patterning system consisting of two diffusing molecules *U* (green) and *V* (red) that act as an activator and an inhibitor, respectively. Each individual robot calculates the values of *U* and *V* by using RD equations (i) and transmits them to neighboring robots (ii). (iii) Turing patterns in simulated and real robot swarms for varying values of parameter *C*. (**D**) Kilobots move along the edge of the swarm and aggregate around Turing spots (i). A robot detects that it is on the edge of the swarm (1) and starts moving along the outer edge of the swarm (2). It stops (3) when it gets close enough to the Turing spot. (ii) An example of kilobot movement in a real robot swarm. (**E**) Conceptual execution of the swarm morphogenesis algorithm. The Turing pattern is formed (dark green), and several robots on the edge of the swarm (in blue) move and stop in the proximity of the Turing spots. Ideally, the Turing pattern should adjust to the new morphology (light green) by changing its configuration, while other robots continue moving and surround the Turing spots to build up the protrusions of the swarm.

The reaction part of the equation could be solved directly on each kilobot just by using the current values of U and V. For the diffusion part, the values of both the activator and the inhibitor of each neighbor in range were necessary, which can be obtained via message passing (Fig. 2C, ii; details given in Materials and Methods). In Fig. 2C (iii), we show examples of different Turing patterns produced by varying a parameter of the RD equations. Green activation of the LEDs was programmed to appear where the concentration of the activator U was high (i.e., above a threshold value). By varying parameter C, we could control the type of pattern (spots, stripes, or inverted spots). For the purpose of this paper, we term the green spots Turing spots.

The second process—migration or tissue movement—allowed robots to reposition themselves from areas with low activator U to areas with high activator (Turing spots), given in Fig. 2D. This mimicked the flow of cells seen in natural morphogenesis or could alternatively be seen as equivalent to localized tissue growth in the region of the green spots and localized cell death in between [as seen during digit formation in tetrapods (*46*)]. There were two conditions for a kilobot to start moving: (i) It must detect that it is on the outer edge of the swarm ("edge detection"), which is based on the estimate of the relative change of the local swarm density, and (ii) it must detect a local concentration of U lower than a certain threshold value. To relocate robots, we used an edge-following algorithm, where each individual robot moved along the outer edge of a group of static robots while attempting to maintain constant distance to its current nearest neighbor. Last, the robots were programmed to stop when in proximity to a Turing spot, i.e., when they detected a neighbor with a high concentration of U, thus creating an accumulation of kilobots around the Turing spots. The details of the movement algorithm are given in Materials and Methods. An example of kilobot movement, following the previously described movement algorithm, is given in Fig. 2D (ii).

A conceptual example of the execution of the morphogenesis algorithm is given in Fig. 2E. Swarms started from an initial shape with arbitrary low-morphogen concentrations. As soon as the pattern was formed and the Turing spots were established, individual robots (in blue) started moving and settling on locations adjacent to the Turing spots. In parallel with the movement, the Turing pattern adjusted to the new shape by changing the morphogen concentrations in both the newly positioned robots and the robots that had already become a part of the Turing spots (denoted in light green). This process of tissue movement and pattern adaptation continued, resulting in the emergence of a shape.

Computer simulations were used to explore how the combination of Turing patterns with movement can spontaneously and reliably give rise to morphogenesis (Fig. 3A). To conveniently monitor the states of the kilobots and the local concentrations of the virtual molecules, we used the color of the LEDs. As shown in Fig. 3A, the LED color depended on the level of the activator U, ranging from green (as the highest level) through teal, blue, and purple at decreasingly lower levels, until the LED is turned off for very low values. We explored the parameter space of the Turing pattern around the values provided in Miyazawa *et al.* (*47*). As shown in fig. S1, replication of the type of patterns obtained by varying parameters A and C was successful, hence confirming results shown in that work. We also explored what types of patterns would result in better morphogenesis based on our approach, i.e., spots, stripes, or inverted spots. Neither stripes nor inverted spots were useful because these patterns resulted in many of the edge robots experiencing high concentrations of U, thus being restricted from moving. By contrast, normal spots worked well because they tended to appear on the edge of the swarm but left significant numbers of edge robots with low concentrations of U. They provided a good compromise between number of robots that could move (in areas of low concentration) and areas of "growth" where moving robots would accumulate (the spots themselves). Parameter values that maximized the number of the spots on the edge without becoming stripes were considered as a good starting point for a complex morphology to develop (values given in Materials and Methods).

The morphogenesis approach was then validated on a real swarm of 300 kilobots. The objective was to transform an initially disc-shaped swarm into a more interesting morphology displaying an array of "tentacles" or protrusions. An interesting question was whether regular morphologies could be generated (e.g., with a fourfold symmetry of protrusions) and/or more dynamic organic shapes reminiscent of simple organisms. Ideally, the shape formation process should be emergent, adaptive, and robust. The program that had been tested previously in simulation (emulating the Turing GRN) was run on the robots and, starting from the initial disc shape (Fig. 3B), the swarms were able to reorganize themselves into new coherent shapes. In the case shown in Fig. 3B, five Turing spots emerged—four on the surface of the swarm, roughly equidistant to each other, and one in the center. Subsequently, when the edge robots started to move, they reliably relocated from regions with low concentrations of the Turing morphogens (no LED illumination) to the vicinity of the Turing spots (green LED). Over a short time, robots built up around the four spots to create four protrusions, showing a fourfold symmetry. Occasionally, individual kilobots were "lost" from the swarm, but this was a relatively rare event, and the vast majority remained in the evolving morphology. Figure 3C shows the results from three more experiments, in which the same basic shape, with fourfold symmetry, was created. A close-up of the stages in the growth of a single protrusion can be seen in more detail in Fig. 3D. Reliability of this emergent morphogenesis was high. The successful parameter values were able to create robust organic shapes every time (nine runs performed in total with an initial circular morphology), and five of these produced the fourfold symmetry highlighted by the white dashed boxes in Fig. 3 (B and C). Of the nine runs, six of them were replicates with the same Turing parameters, motion rules, and experimental conditions. The approximate running time of each experiment was around 3 hours. The regular shape shown in Fig. 3 (B and C) was highly reproducible, but it was transient—further evolution of the swarm led to more dynamic morphologies, which are studied next. The complete summary of all these experiments are given in fig. S2.

An important question is whether morphogenesis really emerged from patterning or simply from robots moving around the swarm. To test this, we performed three experiments in which robots had initial random concentrations and patterning was switched off (meaning that they would keep the same concentration throughout the experiment and no reaction diffusion was taking place). Motion rules, number of robots, and initial circular shape were kept the same as in the previous experiments. We then compared both sets of experiments and demonstrated that Turing patterning is essential for the emergent morphogenesis process and that shapes cannot grow from random patterning (fig. S3A, i). Shape index was used as the metric to compare experiments (details in Materials and Methods).

The process of swarm morphogenesis, besides producing shapes in an emergent manner, also proved to be a dynamic and adaptive process. If we changed the initial configuration of the swarm to a

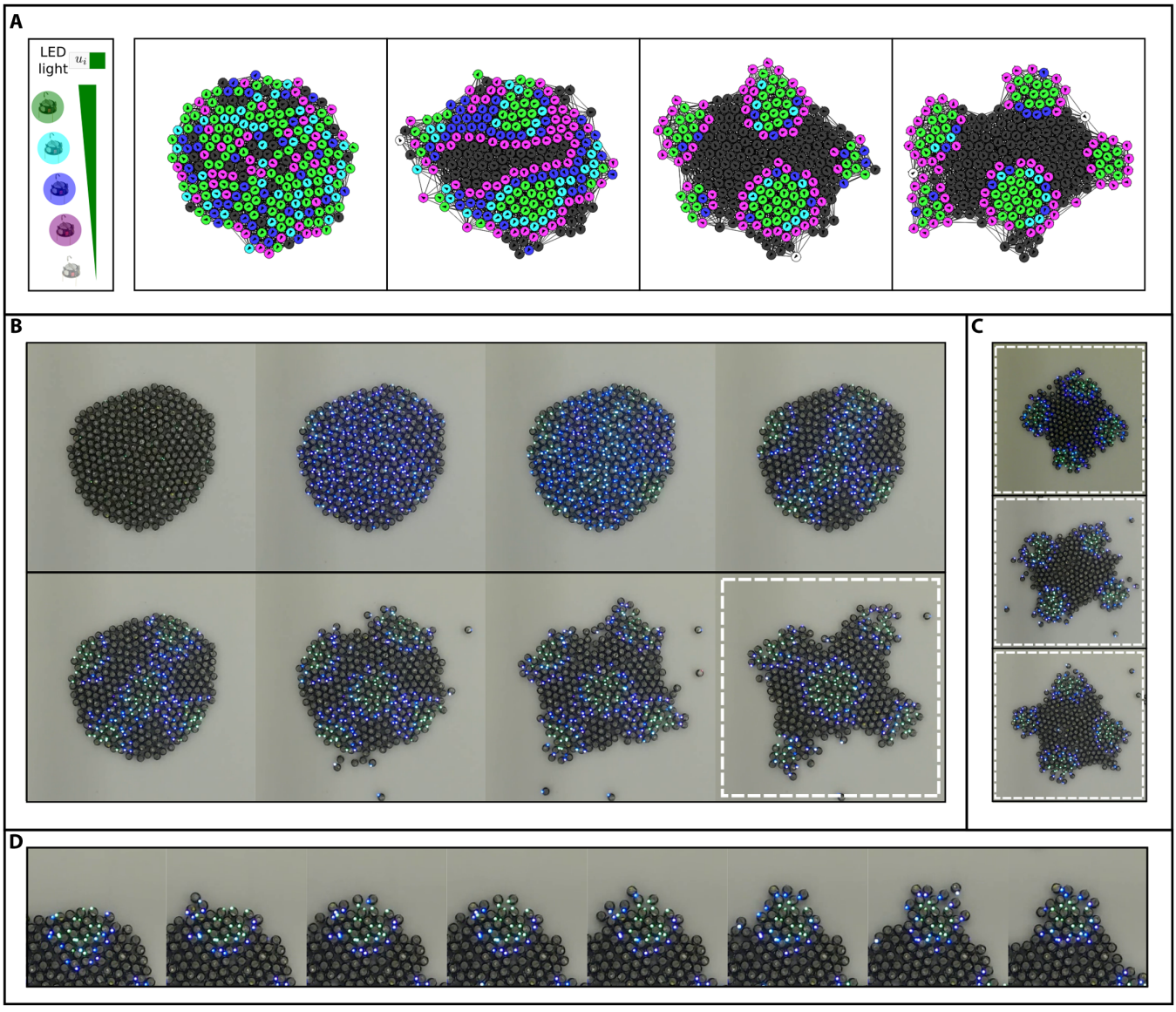


Fig. 3. Emergence of swarm morphologies. (**A**) Morphogenesis in a simulated swarm. Different LED colors indicate different concentrations of the activator *U*. Initially, a swarm of kilobots is visible with arbitrary concentrations of *U*. Next, five Turing spots emerge, around which noticeable protrusions appear (far right). (**B**) A temporal sequence of morphogenesis of a kilobot robot swarm (~300 robots). The initial swarm configuration was roughly circular with five distinct Turing spots, including four spots on the edge of the swarm and one spot in the center of the swarm (top four images). The robots rearranged around the Turing spots (bottom four images), forming initial protrusions. Last, a distinct cross-like shape was formed, which consisted of four tentacles with Turing spots on their tips. (**C**) Three replicates of morphogenesis with ~300 robots, which show similar cross-like, four tentacles morphologies. (**D**) Close-up of a growing tentacle during the swarm morphogenesis process. Starting from a Turing spot in the initial image (left), there is a progressive build-up of kilobots around it, resulting in tentacle growth in the final image.

rectangle, the patterning process adapted to this, and the shape again evolved Turing-driven protrusions (at each corner) as shown in Fig. 4A. To quantitatively test that, we compared the set of replication experiments with initial circular shape with a set of five experiments starting from a rectangle and having the same code as the circular experiments. Results showed that shapes grow indistinguishably, although they started from different initial configurations and therefore had different initial shape indices (fig. S3A, ii). In addition, we ran the same program on a smaller swarm (110 kilobots) to explore the impact of swarm size. These tests produced a similar pattern but with three protrusions instead of four, resembling the letter T (Fig. 4B). This is consistent with Turing patterning systems, where, for a given set of parameters, the frequency of the spots remains the same irrespective of the surface size, i.e., smaller surfaces do not have smaller Turing spots but fewer.

Another important question, related to swarm adaptability, was whether morphodynamic processes occur. This describes the scenario in which a large-scale feedback loop is observed—i.e., pattern drives morphology, but the change in swarm shape also feeds back to alter the molecular pattern (*44*, *45*). Figure 4C shows two examples of molecular pattern shifting through the swarm while the shape changes. In this first case (Fig. 4C, i), this was a necessary part of

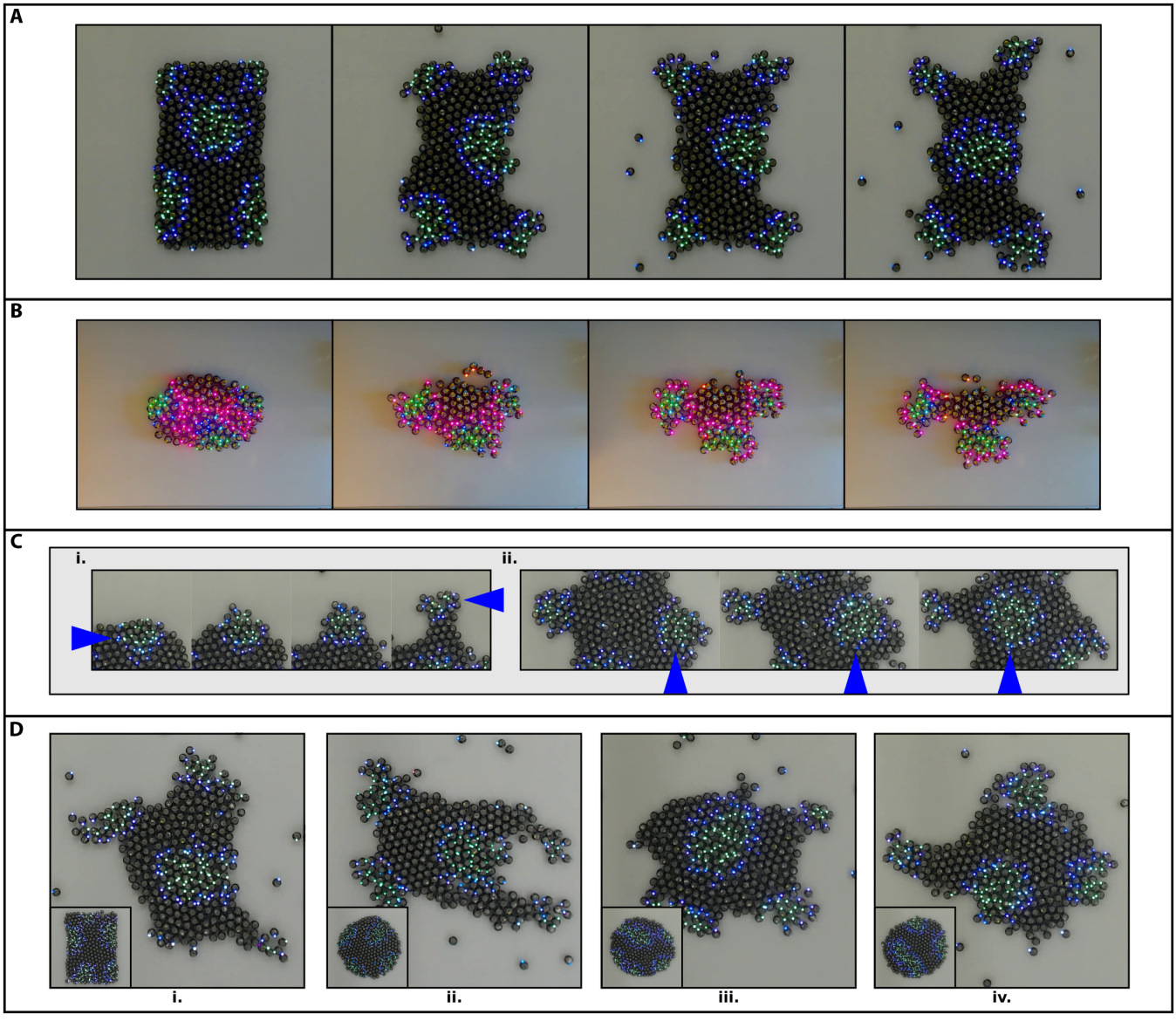


Fig. 4. Adaptability of swarm morphologies. (**A**) Shape formation of a large swarm (~300 robots) starting from a rectangular shape. Four spots emerged on the edges where four tentacles grew, whereas the central spot moved, adapting to the changing morphology. (**B**) A temporal sequence of morphogenesis in a smaller kilobot swarm (~110 robots), with the same settings for the Turing parameters as in (A). Three Turing spots drove the formation of a T-like tentacled morphology. (**C**) Two examples of Turing pattern (spot) adaptation in response to the swarm changing morphology. (i) Adaptation during tentacle growth, where the Turing spot visibly changed shape, size, and location during the growth of a single tentacle, always tending to stay on the tip of the outgrowth. (ii) Starting from the initial spot location on the edge of the swarm, the spot slightly shifted toward the center of the swarm. After a while, it completely moved to the center, while another spot appeared close to its initial location. (**D**) Four variable swarm morphologies with irregular, organic shapes obtained from different runs. Inset images show the initial configuration of the swarm. (i) Started from an initial square morphology, whereas the swarms in (ii to iv) started from circular ones.

the growth of a protrusion. The spot always maintained its position at the distal tip of the protrusion, although the tip did not consist of the same robots over time (growth occurred by new robots arriving and adding to the existing protrusion). This is an important mechanism for tip extension. If the spot were static (remaining in the same group of robots throughout), then it would gradually become enclosed and "hidden" by nongreen robots. Any subsequently arriving robots would continue to edge-follow their way right past the spot, to a different part of the swarm, and growth of the protrusion would be frozen. In the second case (Fig. 4C, ii), a Turing spot could be seen to shift through the tissue. This type of pattern adjustment endowed the swarm with a powerful form of adaptability. If the evolving shape was not compatible with an optimal periodic arrangement (due to either random noise or constraints in the environment), then the pattern could adjust and thus self-correct its own morphology. The adjustment of this molecular pattern could occur either as a gradual shift or as a more abrupt reorganization—in both cases, settling to a new, more stable configuration. Driven by this morphodynamic processes, our swarm produced a variety of other morphologies, whose main feature was that they are very organic, or organism-like, shapes. Figure 4D

shows a variety of swarm shapes, with tentacle-like protrusions growing out of them. Individual quantification of morphologies is shown in fig. S3B (i).

The swarm also showed robustness to direct swarm damage. We explored a couple of damage scenarios: cutting off the protrusions and cutting the whole swarm in half. In the first case, either the original protrusion regrew (Fig. 5A) or the loss of one protrusion promoted the growth of others (on the other side of the swarm in the case of Fig. 5B). Again, this demonstrated the value of the self-organizing behavior. From any given state of the pattern and morphology, the Turing mechanism always pushed the pattern toward an even-spaced periodic one, whether from the initial unpatterned configuration or from a perturbed pattern due to damage. For supplementary quantitative analysis of this first case, see fig. S3B (ii). In the second case, when the swarm was cut into two roughly equal parts, the two halves re-fused to create a single swarm relatively fast (Fig. 5C). The self-organizing dynamic of the Turing spots actually facilitated the merging of the two swarms, contributing to the robustness of the system.

Last, we quantified the dynamic morphologies created by our robot swarms (Fig. 6). We considered the outer contour of the swarm shapes as the main feature to analyze (because this represents the pure morphology of the system) and chose two shape analysis metrics to quantify it: the shape index and the minimum number of characterizing points (details in Materials and Methods) (*48*, *49*). The first measure gave an indication of how much the shape was different from a circle, and the second measured the roughness of the contour. We tracked the change of the shapes over time from nine experiments, starting from the initial, roughly circular shape, finishing with the final evolved shape. By plotting the values of the two measurements for each experiment over time, we produced a trajectory of the swarm through morphospace (Fig. 6A) (*50*). Three clear features of these trajectories could be seen—the starting position, a transient region of regular

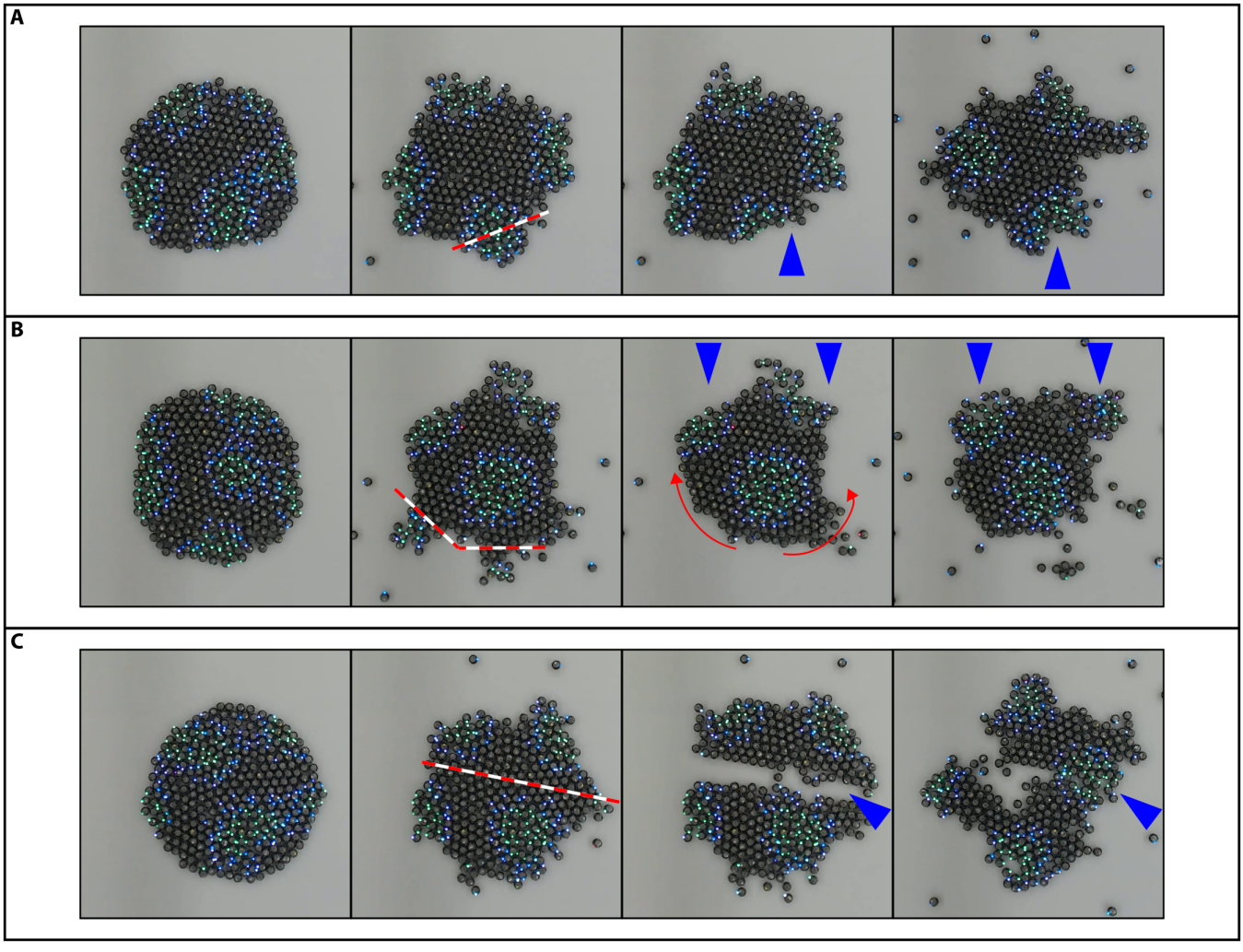


Fig. 5. Robustness of swarm morphologies. (**A**) Regeneration response to minor damage. The starting point was the "standard" four-spot pattern, from which four tentacles developed, as visible in the second panel. The robots from one tentacle were removed, as indicated by the red dashed line, and left behind a Turing spot with a dent in the middle. After a while, the tentacle regenerated with a small dent in the middle. (**B**) Redirected tentacle growth response. Two tentacles were completely cut off, and the whole Turing spots were removed. Unlike the example in (A), where there was some of the original Turing spot remaining, here, these tentacles could not grow back. Instead, by cutting them off, we effectively freed up a large surface of edge robots that could freely move and aided the growth of the remaining two tentacles in the swarm, as visible in the third and fourth panel of the figure. (**C**) Regeneration response to major damage. The developing swarm was cut in two, approximately equal parts, along the red dashed line. We left the two swarms in close proximity to each other, and after a while, they managed to merge into one entity again.

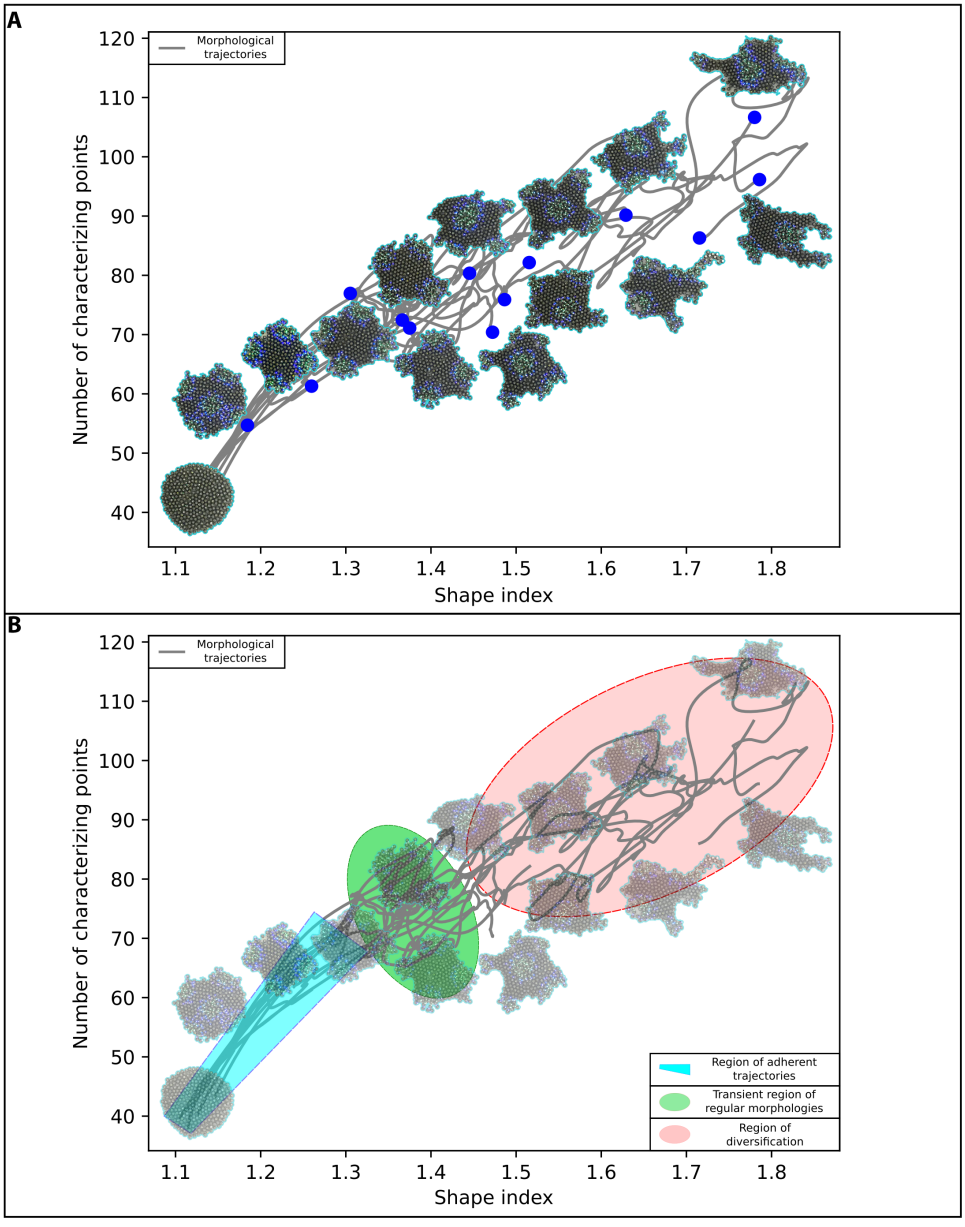


Fig. 6. Quantitative analysis of swarm morphologies. (**A**) Morphospace of nine runs of the morphogenesis algorithm, all starting from an initial circular configuration. Gray lines trace the change of swarm morphologies, and the morphospace is populated by both regular (transient) shapes and more organic shapes. The corresponding morphologies are plotted alongside each point. (**B**) Three distinct regions of morphospace trajectories. The first one is the thin "adherent" region (blue), where all the individual trajectories are similar. Next is the transient region (green), where most of the regular morphologies reside. The third region (red) is where the more organic morphologies reside. Their trajectories in this region are more spread out than in the blue region. However, they still remain in a constrained space.

morphologies, and a dynamic region of variable shapes—which will be discussed in the next section (Fig. 6B).

DISCUSSION

In summary, we have successfully endowed a large swarm of 300 robots with a self-organized morphogenetic behavior (described as approach 2 in the introduction; Movie 1). It is directly based on the principles of developmental biology and results in emergent, adaptive, and robust shapes. Rather than using predefined patterns and explicit information about where each robot is, we used a GRN that implemented a self-organizing Turing process as the basis for the pattern formation. This was translated into physical shape change by using the concept of robot migration (in analogy with natural developmental biology). Analysis of swarm shape over time revealed the overarching control process involved. Starting from the initial circular conditions, all swarms moved in a consistent direction through morphospace (Fig. 6B). This movement represented the emergent but reliable process of protrusions forming on the outer edge of each swarm (blue region). The second feature was the transient region resulting in regular fourfold morphologies (green region), which are highlighted in white dashed boxes in Fig. 3. The third feature was the diversification region of morphospace in which the swarms accumulated. They drifted around this region in a dynamic, adaptable way but reliably stayed within this particular shape space (red region). Thus, both the dynamic adaptability and the shape predictability could be understood within this plot of the morphospace.

A key question was whether the phenomenon of morphodynamic patterning could be found within our robot swarms, a dynamical process documented in the biological literature (*44*, *45*). We demonstrated the existence of this large-scale feedback loop between the patterning process and the changes of the shape morphology (Fig. 4C). On the one hand, the molecular patterning process drives the physical shape change (of the swarm); on the other hand, changes to the shape (which act as boundary conditions for the Turing system) feedback cause the molecular pattern to change. As in biology, this feature makes the swarm more dynamic and adaptable. In particular, the same types of morphology were created when starting from different initial configurations and also upon damage to the swarm from external perturbations (when the researcher "cut" protrusions off the swarm or cut the swarm in half).

Another important feature of this system is scalability. In human technology, the reliability of the whole machine often depends on the reliability of the components ("the strength of the chain is in the weakest link"). The decentralized and collective nature of our system avoids this problem. Although a few robots were lost during morphogenesis, this did not impede the remaining swarm to fulfill its function

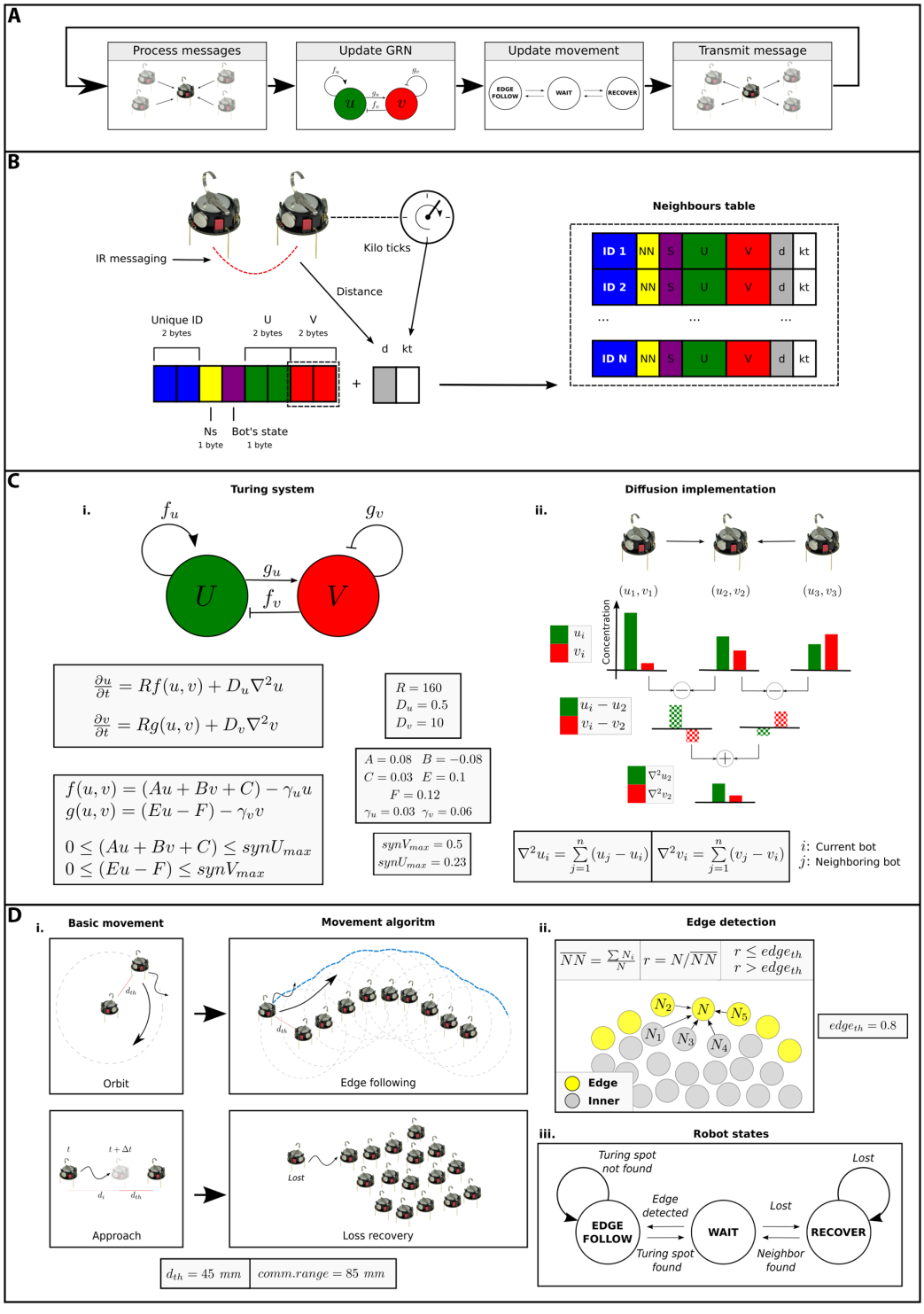
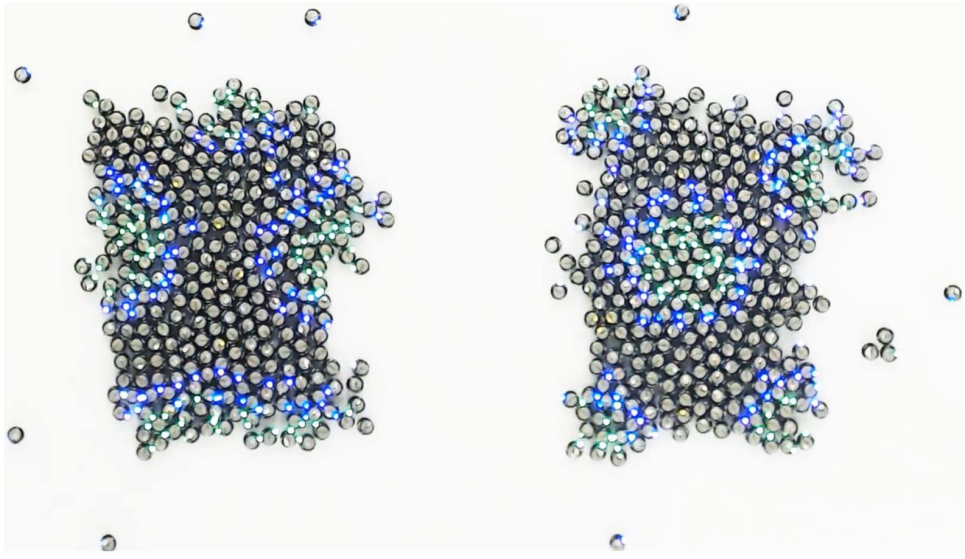


Fig. 7. Morphogenesis approach implementation details. (**A**) Morphogenesis algorithm execution loop. (**B**) Each kilobot broadcasts an 8-byte message containing its ID, number of neighbors, state, and the values of the two morphogens of the Turing system. Received messages are stored in a neighbors' table, together with the distance to the transmitting robot and the kilotick time stamp. (**C**) A linear Turing system (i), consisting of two morphogens, whose concentration is determined by solving the RD equation on each robot. Morphogen diffusion (ii) is calculated by first comparing (substracting) the morphogen values of each neighbor, to a robot's own morphogen concentrations and then these differences are summed up, yielding the net diffusion of each morphogen. (**D**) Orbiting and approaching movements are the basis for edge following and loss recovery movement (i). When an orbiting robot is moving around its current nearest neighbor, the resulting movement is edge following. A lost robot will switch its state away from "recover" if it detects a robot that has other neighbors (iii). (ii) A robot determines whether it is an edge robot (yellow) or not (gray), by calculating the ratio *r* of the average local density of kilobots (*NN*) to its own number of neighbors (*N*), and comparing this to a threshold r_{th}. (iii) Three robot states: WAIT, EDGE FOLLOW, and RECOVER. The WAIT state is a static, nonblocking state. In the other two states, a robot performs the corresponding movement algorithms described in (i).



Movie 1. Morphogenesis in robot swarms.

Table 1. Transition rules for switching between kilobots states.

	WAIT	EDGE FOLLOW	RECOVER
WAIT	*default*	• edge_detected() • check_wait_state(ALL) • dist_to_Turing () > dist_th	• dist(NN) > dist_far
EDGE FOLLOW	• dist_to_Turing () < dist_th • !edge_detected() • !check_wait_state(NN) • dist(NN) < dist_far	*default*	**N/A**
RECOVER	• !check_wait_state(NN) • dist(NN) < dist_far	**N/A**	*default*

(and it is reminiscent of real embryo development, in which not all cells survive). Loss of robots can occur for a variety of reasons—loss of digital infrared (IR) messages, inaccuracy in distance estimation, and unreliable movements of robots—all of which relate to the general lack of reliability of these relatively simple, cheap kilobots. Despite this, the swarm as a whole continued to control its global shape, even when reduced to less than one-third of its original size or when physically "damaged."

This work is an important step in the direction of human-designed hardware showing the dynamic and organic adaptability of living organisms. Although the shapes were not formed with a particular task in mind, this proof of concept should be extended to create functional shapes. The conditions that lead to adaptability and regeneration are understood at a general level (the self-organizing of Turing systems and the feedback of morphodynamics). However, a more detailed theoretical understanding will help to engineer these systems for specific tasks, improving our control of the shapes that emerge. An important idea is to explore greater interactions with the outside world, such as getting the whole swarm to collectively navigate toward an external light source, surrounding and herding foreign objects, searching an environment for areas of interest, or building environment-driven structures (e.g., dynamic bridges to "flow" over a river). In these cases, external cues would influence the GRN state (e.g., gene production), thus directing the growth of the swarm and possibly resulting in whole swarm movement. Overall, we believe that these results provide a glimpse into the future of "programmable matter" in which the limits to the design and flexibility of useful machines are restricted only by our imagination.

MATERIALS AND METHODS

Study design

The objective of our study is to demonstrate that the morphogenesis algorithm produces emergent morphologies in large swarms of real robots. A total of 122 computer simulations of 1000 agents were performed to find the most suitable parameters for the RD equations. Experiments with swarms of real robots were conducted to validate our algorithm. Concretely, 1 experiment with a swarm of 110 robots, 1 experiment with a swarm of 250 robots, and 13 experiments with a

swarm of ~300 robots were conducted. Of the 13 experiments, 11 corresponded to experimental replicates with the same code and experimental conditions, divided in two sets with different initial configuration. Six of them were initialized with a circular shape, whereas the other five were initialized with a rectangular shape. Three additional experiments were conducted on a swarm of 300 robots to test the rehabilitation properties of the morphogenesis algorithm. Last, another three extra experiments with a random morphogenesis algorithm on a swarm of 300 robots with the same code and experimental conditions were conducted as control. Therefore, 21 experiments with large swarms of real robots were conducted.

Morphogenesis algorithm

The morphogenesis algorithm is executed in a loop on each individual robot. It consists of the same sequence of updates, which run in parallel and asynchronously on all kilobots of the swarm. The algorithm can be summarized in three basic steps as (i) communication, (ii) patterning, and (iii) motion.

In the first step, all the inputs, i.e., messages received from neighboring robots, are processed. Next, the values of the two morphogens (U and V) of the GRN are calculated, which underlies the swarm patterning process. Depending on these values and some additional conditions (such as edge detection), a robot updates its movement state and halts or activates its motors accordingly in the last step. The robot also sets the values of the message variables that it broadcasts and the color of the LED light, which depends on the value of the activator U. We summarize the algorithm in Fig. 7A and give the details of the implementation of each of these steps in the following text.

Communication

The primary kilobot inputs are the received messages from other surrounding kilobots (Fig. 7B). For passing a message, an IR broadcast is used, and the message itself has a payload of 9 bytes. It can be received by any kilobot within communication range (~10 cm), and the frequency of the message broadcast is "hardwired" in the kilobots and happens at regular intervals. At any point in the program, the content of the message values can be updated, and they will be transmitted at the next broadcast interval. On the receiving side, when a kilobot detects an incoming message, a program interrupt is generated, and a user-defined handler function processes the received message. In our program, we used a function that stores each incoming message in a circular buffer. The messages from this buffer are emptied and processed at the beginning of each of the main loop executions.

We used 8 of 9 bytes for our message payload, and we transmitted the values of five variables. First is the (locally) unique ID value of a kilobot that took up 2 bytes of the message. The second is the number of neighbors that a robot has and used 1 byte. The robots can be in one of several states, which is also transmitted in 1 byte. The remaining 4 bytes are used for the Turing patterning process, namely, to transmit the value of the U and V morphogens of a robot, each one taking 2 bytes.

Because the morphogen concentrations of U and V are stored as single-precision floating point numbers (4 bytes each) inside the kilobots, they have to be converted to half precision on the transmitter side and then decoded to single precision on the receiving robot side (with a small percentage of error incurred during conversion). The reason is that the 9-byte message that kilobots can send needs to include the other information described above (ID, state, and number of neighbors). Of the 16 bits available in the 2-byte, half-precision floating point numbers, 10 bits are used for the fraction of the number, the next 5 bits for the exponent of the number, and 1 bit for the sign, as described in the Institute of Electrical and Electronics Engineers Standard for Floating-Point Arithmetic (IEEE 754).

After a message from the receiving buffer is processed, it is stored in a so-called neighbors' table, which contains the neighbor's ID, its number of neighbors, and its state. The received bytes of the morphogens U and V are converted to single-precision floating point numbers and stored. In addition, a time stamp is added to each message, which is the current kilo tick number of the robot. The neighbors' table keeps the most recent messages received from the neighbors in range for a certain amount of time, and older messages are discarded (set at 2 s old).

Patterning

The state of the GRN of each robot is responsible for the patterning process of the whole swarm. The equations for the Turing patterning system are given in Fig. 7C (i). These are RD equations that give the rate of change of the morphogens U and V, where R is the reaction parameter and D_U and D_V are the diffusion parameters. The production of U and V is given by functions f and g, and it depends on the synthesis and on the degradation of U and V. The two functions (f and g) are linear functions given in Fig. 7C (i). The parameter values of the linear equations used in our experiments are the following: $A = 0.08$, $B = -0.08$, $C = 0.03$, $\gamma_u = 0.03$, $E = 0.1$, $F = 0.12$, and $\gamma_v = 0.06$. The synthesis terms are limited between zero and user-defined maximum values, which, for our experiments, are given as $\mathrm{syn}U_{\max} = 0.23$ for U and $\mathrm{syn}V_{\max} = 0.5$ for V. The reaction parameter is $R = 160$, and the diffusion parameters are $D_U = 0.5$ and $D_V = 10$ for U and VI, correspondingly. In our implementation of these equations, we used a discrete version with a time step $dt = 0.00005$.

The morphogen diffusion (Fig. 7C, ii) is emulated by passing of morphogen concentrations through the messages between robots. At a given moment, each robot has information about its neighbors' morphogen (U and V) concentrations. By subtracting these neighboring morphogen values from its own and then summing up these differences, each robot determines the net morphogen diffusion that takes place. This allows the robots to edit their own morphogen values, by either increasing or decreasing them due to "diffusion." With the help of the kilobots' LEDs, the Turing pattern can be visible on the swarm. We used different LED colors for different ranges of values of the morphogen U, defined as follows: green LED for $U > 4.0$, teal for $3.0 < U \leq 4.0$, blue for $2.0 < U \leq 3.0$, purple for $1.0 < U \leq 2.0$, and off for $U \leq 1.0$.

Motion

The process of edge detection is relevant for updating the robot state and is the main process responsible for determining whether a robot moves or not. It is based on estimation of the relative difference of local swarm density, as illustrated in Fig. 7D (ii). Each robot transmits the number of its own neighbors N to the robots in range, and in return, it receives the same information N_i from them. If an individual robot's own number of neighbors N is smaller than the average number of its neighbor's neighbors NN, then the robot is on the outer edge of the swarm. The intuition is that if robots are on the inner part of the swarm, then they would have an approximately similar number of neighbors. The threshold for detecting an edge robot that was used, $edge_{\mathrm{th}}$, had a value of 0.8.

Because of the inherent noise in the kilobot swarm, some messages from robots are not received or dropped. This can potentially affect the edge detection and make it unstable. To compensate for this, instead of taking the current number of neighbors, or the current average of neighbor's

neighbors, we used an approximate running average, given with

$$\text{avg}NNs = \alpha \times \text{curr.avg}NNs + (1 - \alpha) \times \text{old.avg}NNs$$

for the average number neighbor's neighbors and

$$N_s = \alpha \times \text{curr.}Ns + (1 - \alpha) \times \text{old.}Ns$$

for the average number of neighbors, where $\alpha = 0.0001$.

After a kilobot detects itself on the edge, motion might start (as described below). The default movement in our approach is the edge-following movement. It is based on orbiting movements, where the aim of a moving robot is to maintain a constant distance to a static one (Fig. 7D, i). The orbiting robot is moving in one of the preferred directions (clockwise or counterclockwise) while estimating the distance to the static robot based on the messages it receives. If the distance is larger than the predefined threshold (d_{th}), then the robot starts rotating in a direction that brings it closer to the static robot. As soon as this movement brings the robot to a distance less than d_{th}, it switches to the opposite direction of rotation. This constant switching of rotation direction produces a forward motion, thus moving the robot in a circle around the static one. The edge-following algorithm is a simple extension of this orbiting algorithm. Namely, there is a swarm of static robots, instead of just one, and a moving robot goes around the edge of this static swarm by always orbiting its nearest neighbor.

The recover movement (approaching) moves a robot toward a static one, bringing it to a predefined distance of d_{th}, the threshold distance. It is used when a robot is too far from a swarm and needs to "recover." The movement starts by the robot rotating in an arbitrary direction and measuring the change of distance from the static robot. If the change is negative, then this means that the robot is approaching the static robot, and the current direction of rotation is maintained. If this change is positive, i.e., the distance to the static robot is increasing, then the robot switches the direction of movement. It does so by selecting the nearest neighbor from the swarm as a static robot and approaching it until it is at a distance lower than d_{th}.

There are three motion-related states in which a robot can be: WAIT, EDGE FOLLOW, and RECOVER, as shown in Fig. 7D, iii. There is one central stationary state, the WAIT state, from which a robot can transit into one of the two moving states—EDGE FOLLOW or RECOVER. The EDGE FOLLOW and RECOVER states are both blocking states, meaning that if a robot is in this state, then all neighbors that are receiving messages from it cannot initialize movement. This constraint is due to the design of the movement algorithms, related to the EDGE FOLLOW and RECOVER state, coming from the kilobots lack of directionality sensing.

The properties of each state are the following:

- The WAIT state is the default kilobot state, in which the robot is not moving; it is just updating its GRN and checking whether the conditions for transitioning into the EDGE FOLLOW or RECOVER state are fulfilled.
- In the EDGE FOLLOW state, the kilobot is performing the edge-following movement.
- In the RECOVER state, the kilobot's goal is to approach the swarm, in case it drifted too far away.

The main rules for transition between the states are summarized in Table 1. Unless a rule allows a state transition to be triggered, the kilobot will by default stay in its current state. There are a few common functions and constants that are used for the state transition rules:

- edge_detected(): *TRUE* if robot on the edge of the swarm.
- check_wait_state(): *TRUE* if neighboring robot(s) are in the WAIT state.
- dist(): returns distance to a specified neighbor robot.
- dist_to_Turing(): returns distance to the nearest neighbor with a Turing spot (if applicable).
- dist_far: upper distance limit beyond which a robot is considered too far from the swarm.
- dist_th: distance threshold used for the edge-following movement.
- ALL: all neighbors.
- NN: nearest neighbors.

A robot that is in the WAIT state can transit to EDGE FOLLOW if it is on the edge, no other neighbors are moving, and it is not a part of, or in near proximity to a Turing spot. The transition in the opposite direction (from EDGE FOLLOW to WAIT) occurs in any of the following scenarios: (i) the robot is no longer on the edge, (ii) it is close (or became part of) a Turing spot, (iii) the kilobot that it is currently trying to orbit around is also moving, or (iv) it moves too far away form the swarm.

If a robot is too far away from the swarm, then it instantly switches to the RECOVER state via the WAIT state. The robot remains in this state until satisfactory distance to the swarm is achieved, after which it switches back to the WAIT state.

Summary of the morphogenesis algorithm

Below is a serial outline of the program that the kilobots execute (patterning and motion). It is worth noticing that the kilobots run several processes such as sending and receiving messages following a timer-interrupt approach. Only the main functions and variables are presented here. A link to the source code can be found at the end of this article.

```
program main:
  u, v, id, state ← initialize_variables() // Random initial concentrations for molecules U and V, random
  // initial id, robot in WAIT state, message with random
  // concentrations starts to be sent
  A ← 0.08, B ← -0.08, C ← 0.03, γu ← 0.03, E ← 0.1, F ← 0.12, γv ← 0.06, Du ← 0.5, Dv ← 10,
  R ← 160, Δt ← 0.00005, synUmax ← 0.23, synVmax ← 0.5
  MAX_DIST_NEIGH ← 85 // Maximum distance in millimeters to neighbors for the diffusion term
  neigh_table ← create_empty_list() // Initializes neighbors table to store their messages
  while TRUE:
  // Messages from neighbors are processed and neighbors table is updated. It also
  // calculates running averages for number of neighbors and number of neighbors' neighbors
  n_neighbors, neigh_table ← process_inputs (neigh_table)
  if state = WAIT then
  u, v ← update_GRN(u, v, A, B, C, γu, E, F, γv, Du, Dv, R, Δt, synUmax, synVmax,
  MAX_DIST_NEIGH, neigh_table)
  end if
  if kilo_ticks ≥ 20000 then // Movement starts after about 10 minutes, when pattern is stable
  state ← update_movement(state, neigh_table)
```

```
  end if
  show_concentration(u) // The color of the LED depends on concentration of molecule u
  id ← local_unique_id(id) // If a neighbor has the same ID, another will be chosen at random
  update_message(id, n_neighbors, state, u, v) // It updates the message that will be sent
  end while
  end program
  algorithm update_GNR:
  input: internal concentrations u and v of molecules U and V,
  parameters A, B, C, γu, E, F, γv, Du, Dv, R of the linear model,
  incremental step Δt for the discretization,
  maximum production rates synUmax, synVmax for molecules U and V
  maximum distance MAX_DIST_NEIGH in millimeters to neighbors for the diffusion term
  neighbors table neigh_table with all the information from neighbors
  output: new concentrations u and v of molecules U and V
  laplaceu ← 0, laplacev ← 0
  // The Laplace operator is calculated
  for i ← 1 to length(neigh_table) do
  neighbor ← neigh_table[i]
  if distance(neighbor) ≤ MAX_DIST_NEIGH then
  laplaceu ← laplaceu + concentration_u(neighbor) − u
  laplacev ← laplacev + concentration_v(neighbor) − v
  end if
  end for
  creationu ← A*u + B*v + C
  creationv ← E*u − F
  if creationu < 0 then
  creationu ← 0
  else if creationu > synUmax then
  creationu ← synUmax
  end if
  if creationv < 0 then
  creationv ← 0
  else if creationv > synVmax then
  creationv ← synVmax
  end if
  creationu ← creationu − γu*u
  creationv ← creationv − γv*v
  u ← u + Δt*(R*creationu + Du*laplaceu)
  v ← v + Δt*(R*creationv + Dv*laplacev)
  return u, v
  end algorithm
  algorithm update_movement:
  input: state state of the robot,
  neighbors table neigh_table with all the information from neighbors
  output: new state state of the robot
  if state = EDGE_FOLLOW then
  if edge_follow_to_wait() then
  state ← WAIT
  stop_motors()
  else // Stays in EDGE_FOLLOW state
  nearest_neigh ← find_nearest_neighbor(neigh_table)
  // By moving around the nearest robot, an edge-following movement is achieved
  move_around(nearest_neigh)
  end if
  else if state = WAIT then
  if wait_to_edge_follow() then
  state ← EDGE_FOLLOW
  start_motor_right()
  else if wait_to_recover() then
  state ← RECOVER
  start_motor_right()
  else // Stays in WAIT state
  do_nothing()
  end if
  else if state = RECOVER then
  if recover_to_wait() then
  state ← WAIT
  stop_motors()
  else // Stays in RECOVER state
  nearest_neigh ← find_nearest_neighbor(neigh_table)
  // By moving towards the nearest neighbor, the bot tries to get back to the swarm
  move_towards(nearest_neigh)
  end if
  end if
  return state
  end algorithm
```

Quantifying swarm morphologies

Shape characterizing points

In the field of landscape ecology, it is of great interest to quantify the heterogeneity of landscapes by identifying and analyzing spatial homogeneous patches (*48*) to counteract the effects of human-induced biodiversity, as Moser *et al.* point out in (*49*). In their work, they analyzed the relation between the shape complexity of the patches and the richness of plant species. In particular, they proposed a metric of geometric complexity based on the contour of the patches, which can be useful here. This metric is called number of shape characterizing points (NSCPs), and it is defined as the minimum number of points required to define the shape. The idea is that the greater the NSCP, the more complex the shape. As a result, this metric can potentially describe the morphology of the protrusions by means of the spikiness of shapes. In their work, they propose to calculate the polygon defining the patch and take only the vertices forming an angle of less than 160°.

In our work, all the points on the contour of each shape were obtained by using the findContours function included in OpenCV 3.2.0 using CHAIN_APPROX_NONE. To calculate the number of characterizing points of the shape, the length of the array resulting from applying the following algorithm to the array of all contour points with a threshold of 160° was obtained.

```
  algorithm shape_characterizing_points:
  input: array P containing all points of the shape contour,
  angles threshold t between 0 and 180 degrees
```

output: array Q containing points in the same order from P with angles $< t$ for all three consecutive points
$Q \leftarrow P$
if Q contains at least 3 points **then**
$base \leftarrow 1$
for $k \leftarrow 2$ **to** length(Q) **do**
$angle \leftarrow$ compute internal angle from points Q_{base}, Q_k and Q_{k+1}
// Q is circular. When k = length(Q), then $Q_{k+1} = Q_1$
if $angle \geq t$ **then**
$Q \leftarrow$ remove point Q_k from Q
else
$base \leftarrow k$
end if
end for
end if
return Q
end algorithm

Perimeter/area ratios

The simplest metrics to measure shape complexity are those using the perimeter and area of the shapes in question. In our scenario, the area of the swarm is practically constant because the number of robots remains the same throughout the experiment, with the exception of the few ones that get lost and the gaps between robots. The perimeter then describes how the contour of the swarm grows/shrinks over time. The longer the protrusions, the bigger the perimeter. Therefore, a metric involving perimeter can be a good estimate of the development of the shape.

Among all perimeter/area ratios, we decided to use a dimensionless one to allow for comparison across experiments, even with different swarm sizes. Moreover, we were interested in comparing the shapes during the morphogenesis process with the initial circular configuration of the swarm. The metric with all these features was shape index, which is a measure of the circularity of a shape. Its formula is

$$\text{Shape index} = \frac{\text{perimeter}}{2\sqrt{\pi \text{area}}}$$

As can be seen, the shape index of a circle is 1. This metric can be useful to quantify how the swarm develops morphological features and how different it becomes from a circle (shape index greater than 1).

To calculate the shape index of the shapes in our work, we used the built-in functions contourArea and arcLength of OpenCV 3.2.0 to calculate the area and perimeter of the shape, respectively. The contour used for this metric was the result of applying the algorithm shape_characterizing_points to the contour with all the points with a threshold of 160°, as described in the subsection above.

SUPPLEMENTARY MATERIALS

robotics.sciencemag.org/cgi/content/full/3/25/eaau9178/DC1
Fig. S1. Parameters exploration in simulation.
Fig. S2. Summary of 15 different runs of the morphogenesis algorithm.
Fig. S3. Quantitative analysis of emergence, adaptability, and robustness.

REFERENCES AND NOTES

1. J. B. A. Green, J. Sharpe, Positional information and reaction-diffusion: Two big ideas in developmental biology combine. *Development* **142**, 1203–1211 (2015).
2. J. Jaeger, Manu, J. Reinitz, Drosophila blastoderm patterning. *Curr. Opin. Genet. Dev.* **22**, 533–541 (2012).
3. J. Raspopovic, L. Marcon, L. Russo, J. Sharpe, Digit patterning is controlled by a Bmp-Sox9-Wnt Turing network modulated by morphogen gradients. *Science* **345**, 566–570 (2014).
4. L. Wolpert, Positional information and the spatial pattern of cellular differentiation. *J. Theor. Biol.* **25**, 1–47 (1969).
5. J. B. Gurdon, P.-Y. Bourillot, Morphogen gradient interpretation. *Nature* **413**, 797–803 (2001).
6. D. Summerbell, J. H. Lewis, L. Wolpert, Positional information in chick limb morphogenesis. *Nature* **244**, 492–496 (1973).
7. A. Gierer, H. Meinhardt, A theory of biological pattern formation. *Kybernetik* **12**, 30–39 (1972).
8. A. M. Turing, The chemical basis of morphogenesis. *Phil. Trans. R. Soc. Lond. B* **237**, 37–72 (1952).
9. J. D. Murray, *Mathematical Biology, II Spatial Models and Biomedical Applications* (Springer-Verlag, 2001).
10. R. Doursat, H. Sayama, O. Michel, Morphogenetic engineering: Reconciling self-organization and architecture, in *Morphogenetic Engineering: Toward Programmable Complex Systems*, R. Doursat, H. Sayama, O. Michel, Eds. (Springer, 2012), chap. 1, pp. 1–24.
11. M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: A review from the swarm engineering perspective. *Swarm Intell.* **7**, 1–41 (2013).
12. C. J. M. Verhoeven, M. J. Bentum, G. L. E. Monna, J. Rotteveel, J. Guo, On the origin of satellite swarms. *Acta Astronaut.* **68**, 1392–1395 (2011).
13. S. Hauert, S. N. Bhatia, Mechanisms of cooperation in cancer nanomedicine: Towards systems nanotechnology. *Trends Biotechnol.* **32**, 448–455 (2014).
14. E. Hawkes, B. An, N. M. Benbernou, H. Tanaka, S. Kim, E. D. Demaine, D. Rus, R. J. Wood, Programmable matter by folding. *Proc. Natl. Acad. Sci. U.S.A.* **107**, 12441–12445 (2010).
15. S. C. Goldstein, J. D. Campbell, T. C. Mowry, Programmable matter. *Computer* **38**, 99–101 (2005).
16. N. Mathews, A. L. Christensen, R. O'Grady, F. Mondada, M. Dorigo, Mergeable nervous systems for robots. *Nat. Commun.* **8**, 439 (2017).
17. R. O'Grady, A. L. Christensen, M. Dorigo, SWARMORPH: Morphogenesis with self-assembling robots, in *Morphogenetic Engineering: Toward Programmable Complex Systems*, R. Doursat, H. Sayama, O. Michel, Eds. (Springer, 2012), chap. 2, pp. 27–60.
18. K. Gilpin, D. Rus, Modular robot systems. *IEEE Robot. Autom. Magazine* **17**, 38–55 (2010).
19. A. F. T. Winfield, J. Nembrini, Emergent swarm morphology control of wireless networked mobile robots, in *Morphogenetic Engineering: Toward Programmable Complex Systems*, R. Doursat, H. Sayama, O. Michel, Eds. (Springer, 2012), chap. 10, pp.239–271.
20. J. Cheng, W. Cheng, R. Nagpal, Robust and self-repairing formation control for swarms of mobile agents, in *Proceedings of the 20th National Conference on Artificial Intelligence* (AAAI Press, 2005), pp. 59–64.
21. Y. Liu, C. Gao, Z. Zhang, Y. Wu, M. Liang, L. Tao, Y. Lu, A new multi-agent system to simulate the foraging behaviors of *Physarum*. *Nat. Comput.* **16**, 15–29 (2017).
22. H. Oh, A. R. Shiraz, Y. Jin, Morphogen diffusion algorithms for tracking and herding using a swarm of kilobots. *Soft Comput.* **22**, 1833–1844 (2016).
23. J. Jones, Influences on the formation and evolution of *Physarum polycephalum* inspired emergent transport networks. *Nat. Comput.* **10**, 1345–1369 (2011).
24. R. Thenius, M. Dauschan, T. Schmickl, K. Crailsheim, Regenerative abilities in modular robots using virtual embryogenesis, in *Adaptive and Intelligent Systems*, A. Bouchachia, Ed. (Springer, 2011), pp. 227–237.
25. L. Bai, M. Eyiyurekli, D. E. Breen, An emergent system for self-aligning and self-organizing shape primitives, in *Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems* (IEEE, 2008), pp. 445–454.
26. T. Schmickl, K. Crailsheim, A navigation algorithm for swarm robotics inspired by slime mold aggregation, in *Swarm Robotics*, E. Şahin, W. M. Spears, A. F. T. Winfield, Eds. (Springer, 2007), pp. 1–13.
27. T. Schmickl, K. Crailsheim, Trophallaxis among swarm-robots: A biologically inspired strategy for swarm robotics, in *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics* (IEEE, 2006), pp. 377–382.
28. R. Groß, M. Bonani, F. Mondada, M. Dorigo, Autonomous self-assembly in a swarm-bot, in *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment*, K. Murase, K. Sekiyama, T. Naniwa, N. Kubota, J. Sitte, Eds. (Springer, 2006), pp. 314–322.
29. K. Støy, Controlling self-reconfiguration using cellular automata and gradients, in *Proceedings of the 8th International Conference on Intelligent Autonomous Systems* (IAS, 2004), pp. 693–702.
30. Y. Ikemoto, Y. Hasegawa, T. Fukuda, K. Matsuda, Gradual spatial pattern formation of homogeneous robot group. *Inf. Sci.* **171**, 431–445 (2005).
31. W.-M. Shen, P. Will, A. Galstyan, C.-M. Chuong, Hormone-inspired self-organization and distributed control of robotic swarms. *Auton. Robot.* **17**, 93–105 (2004).
32. Y. Meng, H. Guo, Y. Jin, A morphogenetic approach to flexible and robust shape formation for swarm robotic systems. *Robot. Auton. Syst.* **61**, 25–38 (2013).

33. Y. Jin, H. Guo, Y. Meng, A hierarchical gene regulatory network for adaptive multirobot pattern formation. *IEEE Trans. Syst. Man Cybern. B Cybern.* **42**, 805–816 (2012).
34. R. Doursat, Organically grown architectures: Creating decentralized, autonomous systems by embryomorphic engineering, in *Organic Computing*, R. P. Würtz, Ed. (Springer, 2008), pp. 167–199.
35. T. Taylor, P. Ottery, J. Hallam, "Pattern formation for multi-robot applications: Robust, self-repairing systems inspired by genetic regulatory networks and cellular self-organisation" (Technical Report EDI-INFRR-0971, School of Informatics, University of Edinburgh, 2007).
36. H. Sayama, Robust morphogenesis of robotic swarms [application notes]. *IEEE Comput. Intell. Mag.* **5**, 43–49 (2010).
37. H. Oh, A. R. Shirazi, C. Sun, Y. Jin, Bio-inspired self-organising multi-robot pattern formation: A review. *Robot. Auton. Syst.* **91**, 83–100 (2017).
38. M. Rubenstein, C. Ahler, R. Nagpal, Kilobot: A low cost scalable robot system for collective behaviors, in *Proceedings of 2012 IEEE International Conference on Robotics and Automation* (ICRA, 2012), pp. 3293–3298.
39. M. Rubenstein, A. Cornejo, R. Nagpal, Programmable self-assembly in a thousand-robot swarm. *Science* **15**, 795–799 (2014).
40. M. Gauci, R. Nagpal, M. Rubenstein, Programmable self-disassembly for shape formation in large-scale robot collectives, in *Distributed Autonomous Robotic Systems*, R. Groß, A. Kolling, S. Berman, E. Frazzoli, A. Martinoli, F. Matsuno, M. Gauci, Eds. (Springer, 2018), pp. 573–586.
41. L. Marcon, J. Sharpe, Turing patterns in development: What about the horse part? *Curr. Opin. Genet. Dev.* **22**, 578–584 (2012).
42. A. D. Economou, A. Ohazama, T. Porntaveetus, P. T. Sharpe, S. Kondo, M. A. Basson, A. Gritli-Linde, M. T. Cobourne, J. B. A. Green, Periodic stripe formation by a Turing mechanism operating at growth zones in the mammalian palate. *Nat. Genet.* **44**, 348–351 (2012).
43. J. D. Murray, *Mathematical Biology* (Springer-Verlag, 1989).
44. I. Salazar-Ciudad, J. Jernvall, S. A. Newman, Mechanisms of pattern formation in development and evolution. *Development* **130**, 2027–2037 (2003).
45. I. Salazar-Ciudad, J. Jernvall, How different types of pattern formation mechanisms affect the evolution of form and development. *Evol. Dev.* **6**, 6–16 (2004).
46. L. Wolpert, C. Tickle, A. M. Arias, *Principles of Development* (Oxford University Press, ed. 5, 2015).
47. S. Miyazawa, M. Okamoto, S. Kondo, Blending of animal colour patterns by hybridization. *Nat. Commun.* **1**, 66 (2010).
48. E. J. Gustafson, Quantifying landscape spatial pattern: What is the state of the art? *Ecosystems* **1**, 143–156 (1998).
49. D. Moser, H. G. Zechmeister, C. Plutzar, N. Sauberer, T. Wrbka, G. Grabherr, Landscape patch shape complexity as an effective measure for plant species richness in rural landscapes. *Landsc. Ecol.* **17**, 657–669 (2002).
50. P. Mitteroecker, S. M. Huttegger, The concept of morphospaces in evolutionary and developmental biology: Mathematics and metaphors. *Biol. Theory* **4**, 54–67 (2009).

Acknowledgments: D.C.-Z. and S.H. thank A. Winfield and L. Giuggioli for useful discussions about the work presented. D.C.-Z. and S.H. also thank J. Wright for the high-quality photos of the robots. **Funding:** I.S., N.C., X.D., F.J., J.S., and J.K. were supported by the Swarm-Organ project, project number 601062 in the European Commission Seventh Framework Program. I.S., N.C., X.D., and J.S. were additionally supported by the Spanish Ministry of Economy and Competitiveness, through "Centro de Excelencia Severo Ochoa 2013-2017," SEV-2012-0208. D.C.-Z. was supported by the EPSRC Centre for Doctoral Training in Future Autonomous and Robotic Systems (FARSCOPE) at the Bristol Robotics Laboratory. **Author contributions:** J.S. conceived the project, obtained the primary funding, created the team, and designed the theoretical framework. I.S. developed, created and tested the morphogenetic approach (both pattern formation and the robot movement control), parameter optimization, and troubleshooting, first in the simulation software and then in the kilobot platform. D.C.-Z. performed the large-scale kilobot runs, simulations, and the morphometric analyses. S.H. supervised the large-scale kilobot runs, simulations, and morphometric analyses. X.D. developed and tailored the Turing model for the swarm. N.C. helped with simulations. F.J. contributed to implementation on the kilobots, which was supervised by J.K. The manuscript was written by I.S., D.C.-Z., S.H., and J.S. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper or the Supplementary Materials. The source code of the morphogenesis algorithm described in this paper has been released under MIT license and can be accessed via https://github.com/Danixk/Turing_morphogenesis.

Submitted 27 July 2018
Accepted 14 November 2018
Published 19 December 2018
10.1126/scirobotics.aau9178

Citation: I. Slavkov, D. Carrillo-Zapata, N. Carranza, X. Diego, F. Jansson, J. Kaandorp, S. Hauert, J. Sharpe, Morphogenesis in robot swarms. *Sci. Robot.* **3**, eaau9178 (2018).

Morphogenesis in robot swarms

I. Slavkov, D. Carrillo-Zapata, N. Carranza, X. Diego, F. Jansson, J. Kaandorp, S. Hauert and J. Sharpe

Sci. Robotics **3**, eaau9178.
DOI: 10.1126/scirobotics.aau9178