www.sciencemag.org/content/345/6198/795/suppl/DC1

# Supplementary Materials for

## Programmable self-assembly in a thousand-robot swarm

Michael Rubenstein,* Alejandro Cornejo, Radhika Nagpal

*Corresponding author. E-mail: mrubenst@seas.harvard.edu

Published 15 August 2014, *Science* **345**, 795 (2014)
DOI: 10.1126/science.1254295

**This PDF file includes:**



**Other supplementary material for this manuscript includes the following:**

**Movie S1:** The self-assembly of a starfish shape by a thousand-robot swarm. The elapsed time for this experiment was 11.66 hours.

**Movie S2:** The self-assembly of a "K" shape by a thousand-robot swarm. The elapsed time for this experiment was 11.71 hours.

**Movie S3:** The self-assembly of a wrench shape by a five hundred robot swarm. The elapsed time for this experiment was 5.95 hours.

**Movie S4:** An annotated compilation of videos showing the various steps of the self-assembly process.

# Materials and Methods

In the main paper we have presented a thousand-robot swarm capable of large-scale self-assembly of multiple shapes, entirely through distributed and local interactions. Here we provide details of the algorithms, proofs of correctness, hardware implementation, and self-assembly experiments.

Section 1 presents the complete shape self-assembly algorithm, including the primitive collective behaviors it builds on. Section 4 presents a proof of correctness for this algorithm, assuming idealized robots. Section 2 presents algorithm implementation on the Kilobot robots, including methods for dealing with variability and unreliability in robot behavior. Section 3 presents detailed results on 13 self-assembly experiments run with Kilobot robots.

# 1 Shape Self-Assembly Algorithm

The shape self-assembly algorithm presented in the paper is designed to work on any robot swarm where the individual robots have the following capabilities:

1. Robots have the ability to approximate holonomic motion (move straight, turn in place).
2. Robots can communicate with neighboring robots within a fixed radius.
3. Robots can measure distance to communicating neighbors within that radius.
4. Robots have basic computation capabilities and internal memory.

All robots, except the four seed robots, are given an identical program, which includes the self-assembly algorithm and a description of the desired shape. The user specifies the shape as a picture, where each pixel that is part of the shape is colored black, and pixels outside the shape are colored white (see Fig. 2), along with a shape-scale value ($s$) which represents the size in real units (mm) that each pixel in the shape should be formed in the environment. The algorithm uses three primitive collective behaviors (edge-following, gradient formation, localization) which are described next. The four seed robots are given the same program, but are initialized in a different starting state, causing them to remain stationary throughout the assembly.

## 1.1 Edge-Following

In edge-following, a moving robot attempts to follow the boundary of a set of stationary robots in a clockwise direction. The stationary robots broadcast a message indicating that they are part of the stationary set; any moving robot within the communication radius of one or more stationary robots can use this message to determine its distance from those robots using distance-to-neighbor sensing. To edge-follow, the moving robot finds the closest of these neighbors and attempts to orbit (circle around at a fixed desired distance) this neighbor in a clockwise direction while maintaining the desired distance from that neighbor. This is done using a simple feedback controller: if the orbiting robot is further away from the stationary robot than desired, it turns slightly to the right while moving forward; if the robot is too close to the neighbor it turns slightly to the left while moving forward; if it is at the correct distance then it moves straight forward. As the orbiting robot moves, it may also notice a different stationary neighbor; when this happens, the orbiting robot

will switch to orbiting this new neighbor as soon as it appears to be closer. This algorithm leads a moving robot to follow along the outside edge of a group of stationary robots.

**Algorithm 1** Edge follow nearest neighbor at DESIRED_DISTANCE

```
prev ← DISTANCE_MAX// prev will store previously measured distance to nearest neighbor
loop
    current ← DISTANCE_MAX // current will store current distance to nearest neighbor
    for all neighbors n do
        if measured_distance(n) < current then
            current ← measured_distance(n)
    if current < DESIRED_DISTANCE then // desired edge-following distance
        if prev < current then // getting closer to desired distance
            move straight forward
        else
            move forward and counterclockwise
    else
        if prev > current then // getting closer to desired distance
            move straight forward
        else
            move forward and clockwise
    prev ← current
```

## 1.2 Gradient Formation

Individual robots can measure distances between each other; the purpose of gradient formation is to create a long-range sense of distance across a swarm. In many ways our gradient formation algorithm plays a role analogous to that of morphogen gradients during fruit fly embryo development (*3*), which provide a rough measure of distance from one end of the embryo. A single seed robot emits a message with a fixed value of zero indicating it is the source of the gradient. All other robots that can hear this message compute their distance to the seed. If the distance is less than a fixed value, gradient-distance, then the robots assume a gradient value of 1 and broadcast this message. All remaining robots execute the same procedure, by assuming a value of $x + 1$ where $x$ is the lowest value of all neighboring robots within the distance gradient-distance. If the robots are tightly packed, and gradient-distance is roughly equivalent to a robot body length, then the gradient value indicates a discretized geodesic distance between each robot and the source. Similar gradient algorithms have been used in several sensor network and swarm systems (*26, 28*).

**Algorithm 2** Gradient formation. $G$ represents the gradient-distance and GRADIENT_MAX is infinity.

```
loop
  if gradient_seed = TRUE then // check if robot is designated as gradient source
    gradient_value(self)← 0
  else
    gradient_value(self)←GRADIENT_MAX
    for all neighbors n do
      if measured_distance(n) <G then // only consider neighbors closer than G
        if gradient_value(n) <gradient_value(self) then
          gradient_value(self)← gradient_value(n)
    gradient_value(self)← gradient_value(self)+1
    transmit gradient_value(self)
```

## 1.3 Localization

The self-assembly algorithm relies on robots ability to localize in a coordinate system that is generated and shared by robots inside the desired shape. The collective localization process uses only communication and distance sensing with respect to stationary robots to generate such a shared coordinate system. Initially all robots are unlocalized, except for the four seed robots which are pre-programmed to start localized near (0,0), seeding the beginning of the coordinate system. Given a small number of localized robots, nearby robots can use that information to localize, thus propagating the information throughout the swarm.

The method is as follows: robots that are stationary and localized, broadcast messages with their $(X, Y)$ position in the coordinate system. When an unlocalized robot receives these messages, it also measures distances to the transmitting robots. An unlocalized robot that receives messages from 3 or more non-collinear localized robots can compute its own location in this coordinate system. This is done by using a form of distributed trilateration (*20*), where the unlocalized robot computes a position $(X_{self}, Y_{self})$ that best matches the measured distances $D_N$ to its localized neighbors with known locations $(X_N, Y_N)$; this is done by minimizing Eq. 1. In order to accommodate the lower computing power of the robots and the asynchronous nature of communications, we approximate this minimization as a sequential adjustment per neighbor as shown in the pseudocode below. The robot then uses this position as its location in the coordinate system, thus becoming localized. It also can broadcast its localized position, which can aid other new robots to localize as well.

$$\min_{X_{self},Y_{self}} \left( \sum_{N} \left| D_N - \sqrt{(X_N - X_{self})^2 + (Y_N - Y_{self})^2} \right| \right) \tag{1}$$

**Algorithm 3** Localization using greedy search to minimize trilateration equation

**if** not a seed robot **then**
  position(self)$\leftarrow (0, 0)$
**loop**
  nlist$\leftarrow [\,]$ // clear list of localized and stationary neighbors
  **for** all neighbors $i$ **do**
    **if** $i$ is localized and stationary **then**
      nlist.append($i$)
  **if** nlist contains 3 or more non_collinear robots **then**
    **for** all $l$ in nlist **do**
      $c \leftarrow$ distance_between(position(self),position($l$))
      // calculated distance based on position in coordinates, as opposed to measured distance based on signal strength
      $v \leftarrow$ (position(self)$-$position($l$))/c
      // unit vector pointing from position($l$) towards position(self)
      $n \leftarrow$ position($l$)+measured_distance($l$)$*v$ // compute new position
      position(self)$\leftarrow$position(self)$-$(position(self)$-n)/4$
      // move 1/4 of the way from old calculated position towards new one

## 1.4 Locally Unique IDs

In addition to the collective behaviors above, robots also make use of a locally unique identifier (ID), such that no robots within communication range have the same ID. This ID is not necessarily globally unique, so multiple robots may have the same ID as long as they cannot directly communicate with each other. To choose a locally unique ID, each robot uses sensor data to seed a random value, and sets the random value as its ID. Robots constantly communicate their IDs to all neighbors, and if they ever have a neighbor with the same ID as their own, it will randomly choose a new ID from a re-seeded random value. As long as the number of possible IDs is much greater than the number of possible instantaneous neighbors, all robots will eventually create a locally unique ID.

**Algorithm 4** Generate locally unique ID

ID_generated$\leftarrow$FALSE
**loop**
  **if** ID_generated = FALSE **then** // need to generate new id
    choose new random seed from sensor data
    ID(self)$\leftarrow$ random value
    ID_generated$\leftarrow$TRUE
  **else**
    **for** all neighbors $n$ **do**
      **if** ID(self)=ID($n$) **then**
        ID_generated$\leftarrow$FALSE

## 1.5 Self-Assembly Algorithm

**Shape description:** The user specifies the shape as a picture, where each pixel that is part of the shape is colored black, and pixels outside the shape are colored white (see Fig. 2), along with a shape-scale value $s$, which represents the size in real units (mm) that each pixel in the shape should be formed in the environment. There are some restrictions on the allowed shapes, for example they must be 1-connected and the perimeter must allow a 2-robot width corridor for perimeter following; a more rigorous definition of allowable shapes is given in Section 4. It is assumed that the lower left pixel of the desired shape is centered at the location (0,0) in the coordinate system, and every other pixel $(X_{image}, Y_{image})$ in this image is centered around the coordinate system location $(X_{image} * s, Y_{image} * s)$ where $s$ is the user-specified shape-scale value that controls the size of the shape in the environment. Each robot has this description as part of the initial program it receives.

**Setup:** The self-assembly algorithm starts with all robots tightly packed into an arbitrarily shaped group. The robots are given the desired shape and identical programs, but have no knowledge about their location in the environment or the coordinate system (Fig 2). Then a user places 4 seed robots next to the rest of the robots; these seed robots remain stationary throughout the experiment. The seed robots are pre-programmed to start localized near (0,0), which seeds the beginning of the coordinate system, indicating to the swarm where the shape should be formed; we assume that the seed is placed so that none of the robots initially fall within the expected final shape location. One of the seed robots is also pre-programmed as the gradient source, allowing all other robots in the swarm to compute their gradient distance to this robot.

**Starting motion:** Initially, all robots in the swarm are stationary; the goal is to have robots peel away from the edge of the stationary group and edge-follow until they reach the seed and start to localize, later becoming stationary again when they join the final assembly. To decide when to start moving, each robot compares its gradient value to that of its neighbors, and if strictly greater than all neighbors, it concludes that it lies on the edge and can potentially start moving. If it has no neighbors with greater gradient value, but neighbors with equal value, it then checks if its locally unique ID is greater than any of its neighbors with equal gradient value. If this is true, then the robot can potentially start moving as well. To avoid too many neighboring robots moving at once, the robot then checks to make sure there are no other moving robots within its communication range before it begins to move. This method guarantees that while the shape is not completed, there will always be robots edge-following, and they all will eventually reach the seed robots.

**Motion and stopping:** Once edge-following, a robot continuously updates its gradient value. Eventually, a robot will reach a position where it can receive messages from 3 or more non-collinear, stationary and localized robots, allowing the robot to determine its own location in the coordinate system. Once a robot knows its own location, it can determine if it is inside the desired final shape region by finding the pixel in the shape description that is closest to its own location in the coordinate system; if that pixel is black, then the robot is inside the final shape region. Once a robot determines it has entered the desired shape, then it will continue to edge-follow until one of the following two conditions is met: 1) it is about to exit the shape, or 2) it is about to edge-follow around a stationary robot in the shape that has a gradient value equal or greater than its own. Once one of these two conditions is met, the robot becomes part of the final shape and stops moving for the remainder of the experiment. While stationary, a robot continues to transmit its coordinate location and gradient value. Note that these rules have the effect of assembling the shape in layers,

where each layer has the same gradient value as measured from the seed (Fig 2).

**Ending states:** There are 4 possible cases for the final equilibrium of this self-assembly algorithm. The final equilibrium state reached by the swarm depends on the number of robots and the size of the final desired shape. The possible states are: 1) all the non-seed robots in the swarm will eventually start motion, enter the shape, and stop and join the shape. If the number of robots happens to exactly fill the shape, then the shape is completely formed. 2) All the non-seed robots in the swarm will start motion, enter the shape, and stop and join the shape, but there are too few robots to form the shape, so the shape will be incomplete. 3) All the non-seed robots in the swarm will start motion, but there are more robots than required so that some will be unable to find a point to enter the shape. In this case the shape formation is complete, and the remaining robots will continuously edge-follow the periphery of the shape. 4) After the shape is completely formed, there will be enough remaining robots edge-following the periphery of the shape and the starting group of robots, that they will inhibit some remaining robots from starting motion. In this case the shape is complete, but with some robots edge-following around the shape, and some robots still in their starting position, but inhibited from ever starting motion.

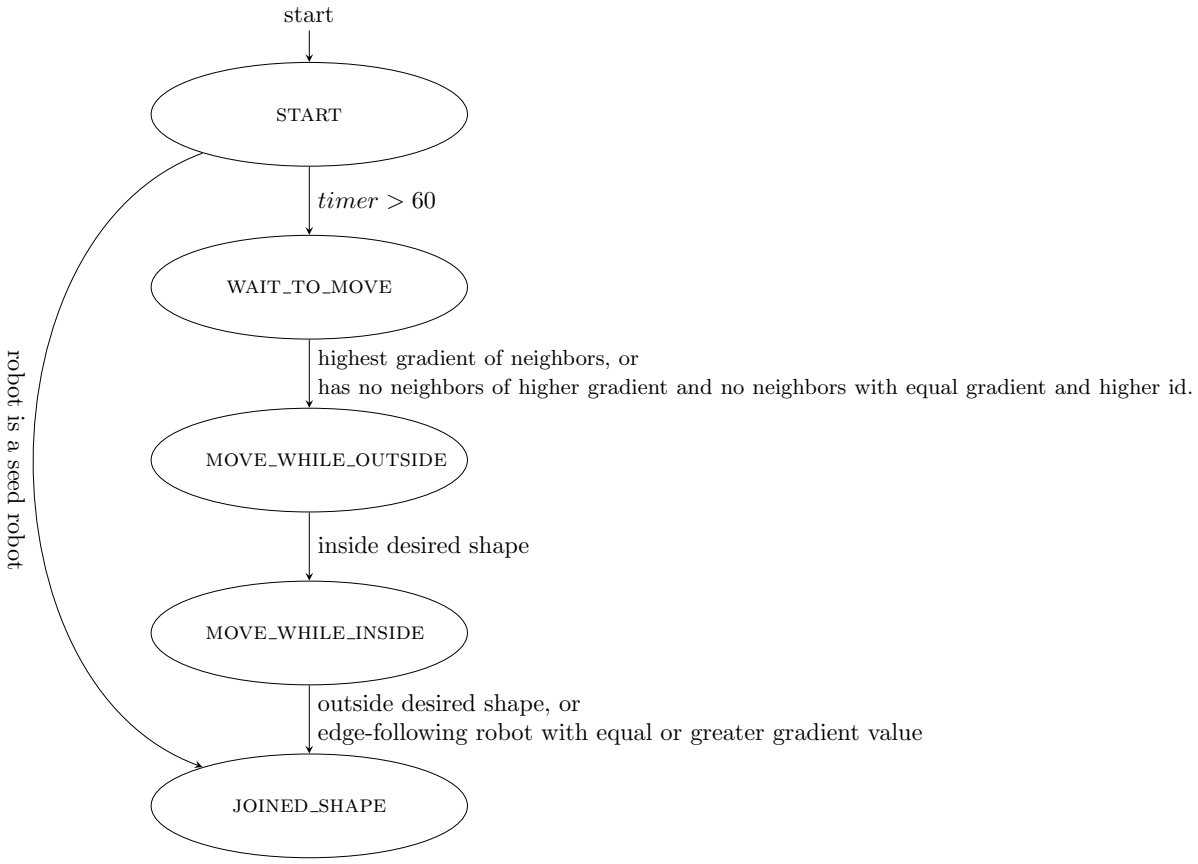


Figure S1: A state diagram of the self-assembly algorithm.

**Algorithm 5** Shape self-assembly

```
1:  motion: stop
2:  state← start
3:  timer← 0
4:  loop
5:    if state=start then // wait for fixed amount of time so all robots are on
6:      if robot is a seed robot then
7:        state← joined_shape
8:      else
9:        begin gradient formation (Alg. 2)
10:       begin localization (Alg. 3)
11:       timer←timer+1
12:       if timer > startup_time then
13:         state← wait_to_move
14:   else if state=wait_to_move then // decide when to start moving
15:     if no moving neighbor seen then
16:       h ← 0        // find highest gradient value among neighbors
17:       for all neighbors n do
18:         if h <gradient_value(n) then
19:           h ←gradient_value(n)
20:       if gradient_value(self)> h then
21:         state← move_while_outside
22:       else if gradient_value(self) = h then
23:         if ID(self) > ID of all neighbors with equal gradient_value then
24:           state← move_while_outside
25:   else if state=move_while_outside then // edge-follow while outside desired shape
26:     if position(self) is inside desired shape then
27:       state← move_while_inside
28:     if distance to front edge-following robot > yield_distance then
29:       motion: edge-follow (Alg. 1)
30:     else // yield to edge-following robot ahead
31:       motion: stop
32:   else if state=move_while_inside then // edge-follow while inside desired shape
33:     if position(self) is outside desired shape then
34:       state← joined_shape
35:     if gradient_value(self) ≤ gradient_value(closest neighbor) then
36:       state← joined_shape
37:     if distance to front edge-following robot > yield_distance then
38:       motion: edge-follow (Alg. 1)
39:     else // yield to edge-following robot ahead
40:       motion: stop
41:   else if state=joined_shape then // become stationary as part of shape
42:     motion: stop
43:     stop localization
44:     stop gradient_value update
```

# 2 Algorithm Implementation on Kilobot Robots

While the algorithm above can provably form a large class of shapes, the proofs assume idealized robots. In contrast, real robots have more complex constraints on movement and exhibit many forms of noise and error. In the case of the Kilobot robot, several factors affect this algorithm including:

1. Non-holonomic movement: A Kilobot robot cannot turn about its center axis. Instead the robot turns about one leg causing it to effectively move forward while turning. This sort of coupled and non-holonomic movement constraint is common in robots.

2. Lossy communication: As with many wireless communication systems, messages are sent over a shared channel which means that if two robots send a message at the same time the message can be corrupted or lost. We use a standard protocol, CSMA-CD to reduce message loss or corruption, but this does not eliminate it, so the algorithm must be able to tolerate messages that are never received. It also introduces a delay between when a message is created in a robot and when it actually gets sent and received by a neighbor. This introduces a significant amount of asynchrony in every robot decision-making process that relies on neighbor information.

3. Error in distance sensing: Kilobots exhibit a variety of types of noise in distance sensing. This includes (1) variation of electronic component sensitivity, which we attempt to reduce with calibration but cannot entirely eliminate, and which causes a difference in accuracy between robots; (2) anisotropic distance sensing: distance sensed can vary based on the orientations of the transmitting and receiving robots, which can result in asymmetric distance sensing, where two robots measure different distances to each other; (3) random noise (fluctuations) in sensing, which can negatively affect behaviors such as edge-following, gradient formation, and collision avoidance; (4) failed distance sensing due to message loss. In the Kilobot system a robot can only sense another robots presence through the second robot transmitting messages. Because messages can be lost, robots may not always detect nearby robots.

4. Variation: Across large numbers of robots there will be persistent variations in behavior where some robots are better than others at certain tasks. In the case of Kilobots, some robots are faster at movement than others and this can lead to many kinds of emergent effects due to persistently fast and slow robots. This variability in robot locomotion and sensing can be manually detected and easily corrected in small groups but are too cumbersome and time consuming to correct in larger groups, resulting in robots with statistically different variability.

5. Rare events: Because of the large number of robots and long time span of the experiment, even rare failures are likely to occur and must be correctly handled. In the case of Kilobots, on rare occasions the motors get stuck causing the robot to move incorrectly; if detected this can be corrected by restarting the motors. Also on rare occasions, stationary or moving robots may get accidentally pushed by other moving robots; robots must be able to recover from sudden unintended position changes.

To demonstrate the algorithm works even with these challenges, we implemented the algorithm on the Kilobots. The following describes modifications of the ideal algorithm primitives (edge-follow, create gradient, localize) used for the implementation on Kilobots.

## 2.1 Edge-following

With ideal holonomic motion, a moving robot could move with zero distance to its closest neighbor, resulting in a tightly packed final shape. However, non-holonomic movement and noisy and imprecise distance sensing prevent Kilobots from edge-following in this ideal manner. Since robots cannot turn in place, they will collide with stationary robots whenever the desired direction of movement changes abruptly. Errors in distance sensing can cause a robot to collide with the neighboring robots. To compensate, robots instead attempt to edge-follow at 20mm away from their nearest neighbor. By maintaining a safe distance, edge-following robots have enough room to turn to edge-follow new neighbors, and not collide with neighbors even with distance sensing errors. However this also means that the final shape assembly has a more complex and variable packing of robots.

In the idealized algorithm, it is assumed robots travel at the same speed; this implies that robots would maintain a fixed distance (along the path of travel) between themselves and other edge-following robots, which prevents them from bumping into each other. With Kilobots, individual motor variances cause some robots to travel faster than others; without any additional programming, distance between moving robots is not maintained, and eventually fast-moving robots will bump into slow robots ahead of them. This may push one or both robots away from the stationary robots, causing a loss of communication with the stationary robots, which will disrupt edge-following. To prevent this from happening, we implement a form of collision avoidance between moving robots. Robots identify other robots that are edge-following in front of them, and yield to them if they get too close. Robots in front are identified based on the ID of stationary robots they have edge-followed or the position of the edge-following robot in the coordinate system. This allows edge-following in an orderly manner without errors caused by collisions between edge-following robots.

In rare cases a robot may command its motors to move, but due to transitory high internal friction they do not. Without correction, this would cause the robot to stop moving, and cause all robots edge-following behind it to stop as well, causing self-assembly to stop. To prevent this from occurring, each robot maintains a list of distances to stationary robots it has recently seen; if none of these distances has changed for long enough of a time, then the robot assumes its motors are stuck. This is an example of cooperative monitoring. To unstick the motors, the robot applies maximum current to both motors until it detects changes in neighbor distances, and then resumes normal motion.

## 2.2 Gradient Formation

The ideal gradient formation algorithm uses ideal distance sensing to determine if a robots state should be updated based on the messages it hears from its neighbors. However since distance sensing is noisy, a stationary robot close to the distance cutoff (G from Alg. 2) may sometimes be considered for the gradient update, and sometimes not. This can have a negative effect on the gradient computation, causing a robots gradient value to fluctuate. These fluctuations are additive, so

they increases in prevalence and magnitude as the distance of the destination robot to the gradient source robot increases. This affects the behavior of the collective algorithm; for example, it can cause robots to incorrectly start moving and may cause disconnection between some robots and the starting seed, preventing them from reaching the desired shape. To correct this, we introduce a hysteresis in the gradient value used to compute the hop count. The hysteresis value is chosen to be large enough to correct for distance sensing noise, stabilizing the hop count computation.

## 2.3 Localization

To build an accurate coordinate system, the idealized localization algorithm assumes perfect distance sensing among robots. In Kilobots, the distance sensors are noisy, and anisotropic (i.e., can vary based on the orientations of the transmitting and receiving robots). These imperfections can cause a robot to incorrectly localize in the coordinate system based on inaccurate information; this is further amplified as the badly localized robot then affects the future localization of other robots.

To counteract the effects of the random sensor noise, a robot first computes the average of a large number of distance samples between itself and its neighbor, and then uses this average as the distance used in trilateration. This minimizes the effect that the zero-mean distance sensor noise has on robot localization. However, another major problem is asymmetric distance sensing caused by the anisotropy in distance. For example, robot A measures a distance of $X1$ to robot B, while robot B measures a distance of $X2$ to robot A, where $X1 \neq X2$. This asymmetric distance sensing can cause robot positions in the coordinate system to drift over time, as the distributed implementation of trilateration is not guaranteed to compute a stable solution for asymmetric distance sensing. To correct this problem, a robot will cease further localization once it has stopped to join the shape. This does not affect the self-assembly algorithm progress and prevents the coordinate system from drifting.

In rare cases, a localized stationary robot may be pushed by an improperly edge-following robot. If the localized robot does not update its location in the coordinate system, it is possible that it would cause large errors in the coordinate system, possibly creating a large error in the shape being formed. To prevent physical effects such as pushing from causing the self-assembly algorithm to fail or have large errors, each localized robot maintains a list of distances to other neighboring localized robots. If many of these distances change significantly at once, the robot assumes it has been pushed, and re-localizes once. This is an example where cooperative monitoring is used to correct for rare errors. In this way the robot only adjusts when an error happens.

## 2.4 Self-Assembly Algorithm

In addition to robust primitives, the overall shape specification is also designed to tolerate real robot sensing and locomotion, which is imprecise and noisy. The shape specification describes the desired shape as a boundary but does not dictate the exact location of robots within that shape. This is in contrast to specifying the shape with lattice positions for each robot (*22, 25, 29*); in such systems small alignment errors can easily cause a whole structure to fail to form. Our self-assembly algorithm tolerates a variety of packing patterns of robots within the shape, without allowing small position errors by the robots to propagate into large-scale self-assembly failures. In fact, as section 3 shows, even for small assemblies there can be considerable variation in the final packing, and

in larger-scale self-assemblies small defects do occur but do not cause the system to fail. The self-assembly absorbs the imprecision inherent in mobile robots.

### 2.5 Computational Cost

The self-assembly algorithm is fairly complex and has many subparts as well as routines for robustness. However, as we show, very simple swarm robots are capable of fully implementing the algorithm. The Kilobot robot has strict limits on its available memory: 2K RAM and 32K for program memory, which includes the code for the bootloader/wireless programming, robot movement, and communication libraries. The full self-assembly algorithm takes approximately 27K of program memory including 241 bytes for the shape description and scale factor. The Kilobot also has limits on message size; we optimize by combining messages from the various primitives, placing all the data in a single 7 byte message.

### 2.6 Ending State

As described previously, there are four possible cases for the final equilibrium of this self-assembly algorithm, depending on the area of the desired shape and the number of robots available to form the shape. Since our goal was to form the shape completely, for each experiment (described in the next section) we chose a shape scale value that would ensure there would be more than enough robots to complete shape formation given the desired shape and initial number of robots. This conservative choice in shape scale resulted in extra robots after the shape had been formed (ending state 3 as described previously). For all experiments, once a robot circled the desired shape without entering, we considered the assembly complete and removed any remaining robots that were not able to enter the shape. This results in fewer robots in the completed shape than had started, and can be seen in the following section.

## 3 Self-Assembly Experiments

### 3.1 Experimental Setup and Recording

All experiments were run on a 2.4x2.4 meter table, which includes metal charging surfaces on one edge, and recorded from above by a wide angle camera (shown in Fig. S2). The camera is calibrated, allowing for the de-warping of images and the recording of individual robots true positions on the table. Images were taken at regular time intervals of 5 seconds, and videos were made by stitching de-warped and cropped images together. In some cases the image brightness was adjusted to increase contrast. The state of each robot (such as position in the coordinate system, gradient value, etc.) can be queried at the end of experiments by using a specially programmed listening robot (separate from the swarm used in the experiment) connected to a computer.

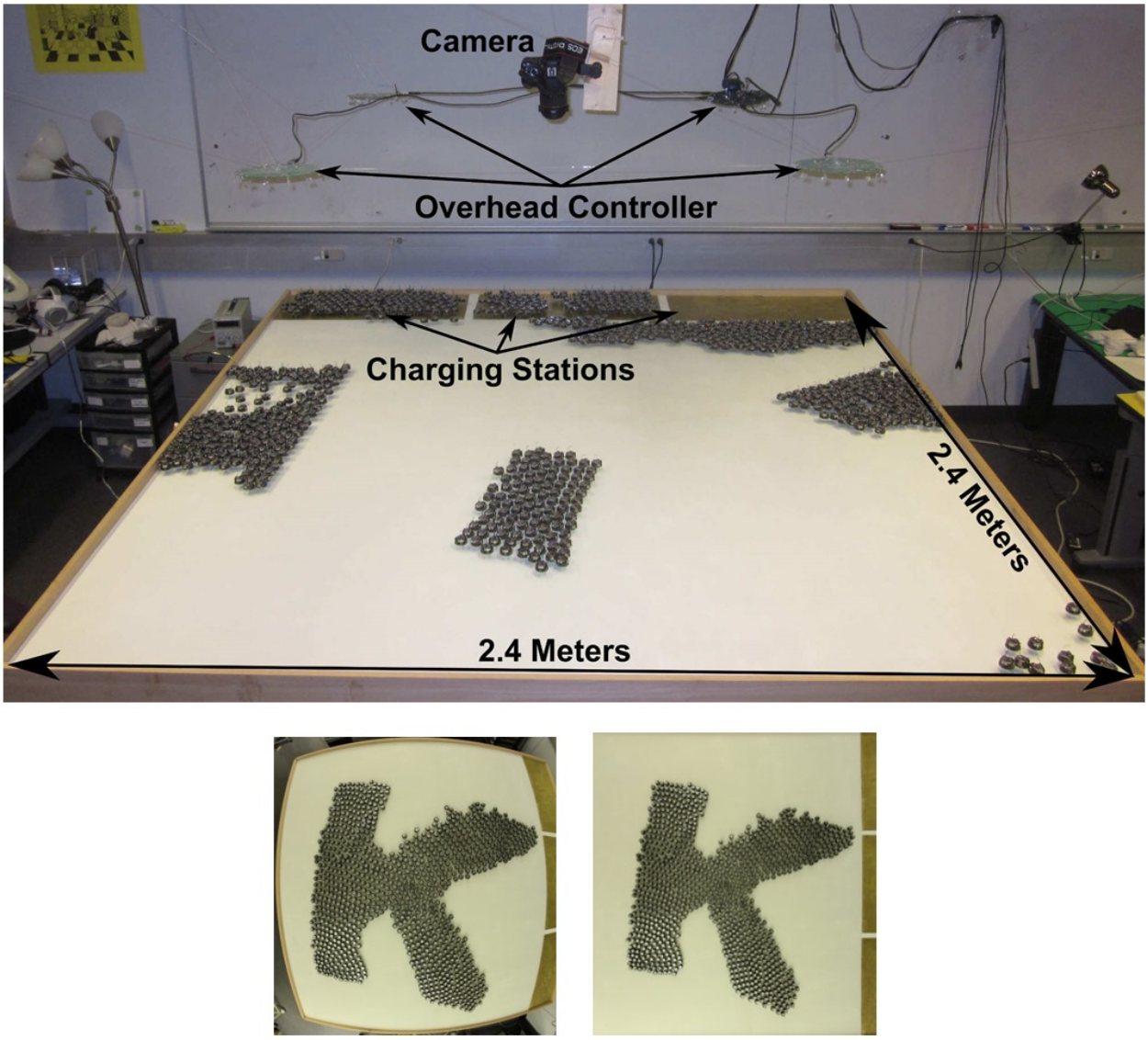


Figure S2: (Top) Experimental setup with robot charging stations, overhead controller (used to turn all robots on/off and send all robots their identical program simultaneously), and camera marked. (Bottom Left) a raw image from camera showing table and metal charging surface. (Bottom Right) de-warped, calibrated, and cropped image.

## 3.2 Experimental Results

Overall, we ran thirteen self-assembly demonstrations total. (E1,E2) The two largest and longest-running experiments, forming the K and starfish shapes, were used to demonstrate that the algorithm can successfully self-assemble shapes with a 1024-robot group. (E3) The third experiment, forming a wrench shape ( 540 robots), was used to look at the detailed accuracy of shape formation, both at the global level and at the level of individual robots. The last group of 10 experiments (E4-E13) were run on approximately 100 robots forming the same shape ten times, to look at repeatability of success and shape accuracy. Table S1 presents a summary of all experiments.

| Desired shape | Number of robots at start | Number of robots in shape at completion | Time to shape completion (hours) | Dimensions of shape formed (approx. Bounding box)(m) |
|---|---|---|---|---|
| K (E1) | 1024 | 1018 | 11.71 | 1.39 x 1.34 |
| Starfish (E2) | 1024 | 946 | 11.66 | 1.47 x 1.47 |
| Wrench (E3) | 543 | 533 | 5.95 | 1.60 x 0.63 |
| Rectangle (E4) | 116 | 104 | 1.18 | 0.35 x 0.61 |
| Rectangle (E5) | 116 | 104 | 1.10 | 0.35 x 0.61 |
| Rectangle (E6) | 116 | 103 | 1.11 | 0.35 x 0.61 |
| Rectangle (E7) | 116 | 104 | 1.15 | 0.35 x 0.61 |
| Rectangle (E8) | 116 | 105 | 1.25 | 0.35 x 0.61 |
| Rectangle (E9) | 116 | 104 | 1.15 | 0.35 x 0.61 |
| Rectangle (E10) | 116 | 100 | 1.06 | 0.35 x 0.61 |
| Rectangle (E11) | 116 | 106 | 1.11 | 0.35 x 0.61 |
| Rectangle (E12) | 116 | 104 | 1.10 | 0.35 x 0.61 |
| Rectangle (E13) | 116 | 104 | 1.16 | 0.35 x 0.61 |

Table S1: A summary of all experiments performed. The difference between the number of robots at start and number of robots in shape at completion is due to the conservative estimate of shape scale, where robots remaining after shape completion were removed, (see ending state description in section 2 of supplement for details).

All thirteen experiments fully assembled the desired shape without human intervention in all trials, showing that the collective behavior is robust. The final shape accuracy is also high but not perfect; the experimental results show how some errors by individual robots during the assembly process can translate into variability in the packing pattern or a slight warping of the shape but without causing the process to fail.

## 3.3 Individual Experiments

(E1,E2): The two largest and longest-running experiments, forming the K and starfish shapes, were used to demonstrate the algorithm can successfully self-assemble large-scale shapes. Both experiments were successful on first try; experiments lasted 11.71 hours and 11.66 hours respectively, and no battery recharging was necessary during the experiments. Edge-following robots travelled path lengths of up to 5.55 meters and 5.60 meters respectively at average speeds of .65 cm/s (or 0.2 body length/sec). Both experiments started with 1024 robots and final shapes contained 1018 and 946 robots respectively.

(E3): The third experiment, forming a wrench shape, Fig. S3, was used to look at the accuracy of shape formation and the accuracy of the localization of individual robots. This experiment involved 533 robots forming a shape (1.6 by .63 meters) over a period of 5.95 hours; as can be seen from the image in Fig. S3(B), the final shape matches the desired shape but shows a slight bend to the right.

To quantitatively measure shape error, we gathered the true position of each robot (as measured from the calibrated camera) and collected each robots internal localized position in the shared shape coordinate system (collected at the end of the experiment by a listening robot). Figure

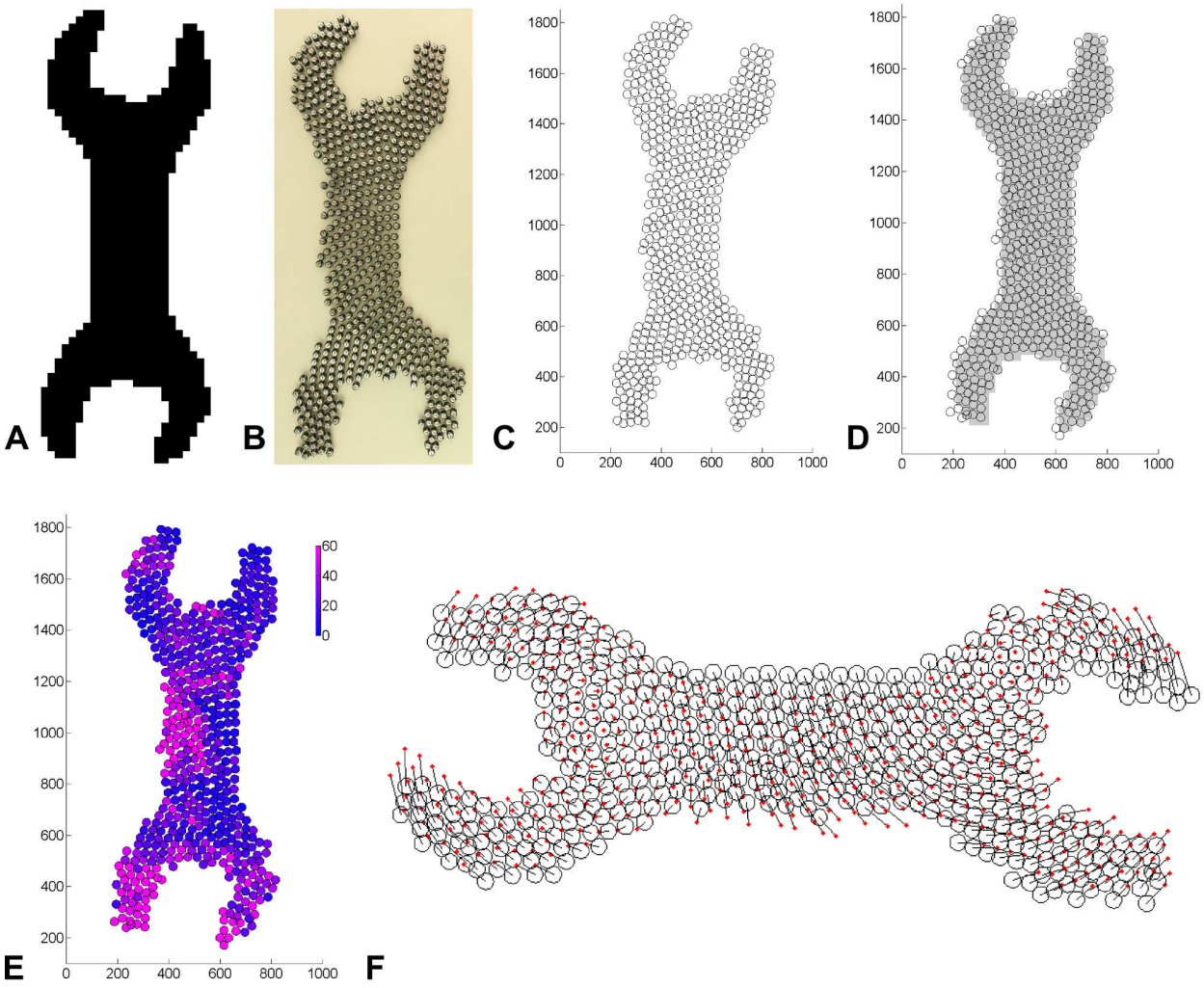


Figure S3: All axis are units of mm. (A) Desired shape given to robots. (B) Image of robots after shape formation. (C) True positions of robots as measured by calibrated camera. (D) Internal localized positions of robots, overlaid on desired shape location in coordinate system (grey). (E) Robots colored according to their localization error (local MSE). (F) A rotated view of the best fit matching between each robots internal localized position (shown as circles) and its true position based on the camera image (shown as red dots). A line is drawn between each robots true position and its position in the internal coordinate system.

S3(B,C) shows the true positions of all the robots; Figure S3(D) shows where each robot thinks it is. As shown, in the robots internal shared coordinate system, the shape is well-formed and is quite straight; robots do not perceive the bend that happened during the formation process.

To make this observation more quantitative, we first computed the rotation and translation between the true positions of the robots and their localized positions in the shape coordinate system that minimized the mean-squared error, Fig. S3(F). This best-fit matching aligns as best as possible the reference frame of the camera image (i.e., physical positions of the robots) with the robots positions in the shape coordinate system. The mean squared error (MSE) value of the matched positions gives the global shape error. Since the majority of robots think they are in the desired shape, this MSE value gives a good estimate on how much the shape formed differs from the desired shape. These best fit MSE values are shown in table S2.

| Desired shape | Best fit MSE($mm^2$) | Average Localization Error ($mm^2$) |
|---|---|---|
| Wrench (E3) | 1613.8 | 35.26 |
| Rectangle (E4) | 32.28 | 15.07 |
| Rectangle (E5) | 133.79 | 28.95 |
| Rectangle (E6) | 39.85 | 14.95 |
| Rectangle (E7) | 44.32 | 15.92 |
| Rectangle (E8) | 45.99 | 17.33 |
| Rectangle (E9) | 45.56 | 18.56 |
| Rectangle (E10) | 84.31 | 19.64 |
| Rectangle (E11) | 36.36 | 24.55 |
| Rectangle (E12) | 95.11 | 20.05 |
| Rectangle (E13) | 48.04 | 28.98 |

Table S2: Analysis of shape warping and localization error.

While the above method focuses on global shape error, we can also look at localization error at the level of each individual robot. The global error is not a good measure of localization error, as it is also dependent on the size of the shape; for example a constant warping of the coordinate system will cause the global error to increase as the size of the shape increases. Measuring individual robot localization error allows us to understand how well the localization process is working; i.e., when a robot localizes, how well does the chosen position match its neighbors positions in the coordinate system, and the true distances (as measured by the camera) to those neighbors. These localization errors are a result of imperfect sensing (asymmetric distance sensing, noisy sensing) and robots being pushed after localization.

Each robots localization error is computed by finding the best fit mean squared error (MSE) between the true positions of all its neighbors (robots within 70mm) and their positions in the coordinate system. The average localization error for all robots in the shape is shown in Table S2. A visualization of individual robots localization error for the wrench shape experiment can be seen in Fig S3(E). As can be seen from the final results, there are a few areas where the error is high, for example on the left side of the wrench middle; the video shows that pushing occurred at that location after robots assembled, most likely causing the coordinate system to be perturbed in that

region and incorrectly formed.

Finally, as we can see from the overlay of the shape image and the robots, Fig. S3(D), not all robots think they are in the shape even though they are stationary. This more minor cause of the mismatch between the desired shape and the shape formed is due to robots pushing other robots. Ideally, a robot edge-following will never bump a neighboring robot, but due to imperfect motion and sensing robots will infrequently bump neighboring robots. This can cause a robot that joined the shape in a correct location to be later pushed outside the shape. Fig. S3(D) shows some robots that have joined the shape are no longer within the desired shape by the end of the experiment; these robots are mostly on the left edge of the shape, where they are exposed to more moving robots, which increases the chance they will be pushed.

(E4-E13) To test the repeatability of this self-assembly algorithm running on Kilobots, we ran 10 experiments where approximately 100 robots created a rectangle, approximately of size .35 x .60 (m). The resulting shape for each experiment can be seen in Fig. S4 and the details of the experiments are recorded in table S1. Additionally, the best-fit shape error and average localization error are reported in table S2. Experiments (E4-E13) all reliably completed shape formation, with 100 to 106 robots forming the desired shape, taking between 1.06 and 1.25 hours to complete. These experiments show an interesting observation: that even for small shapes, the packing pattern for the same shape can show considerable variability, without resulting in defects (large holes) or a failure to complete the shape. The self-assembly process is able to accommodate the inherent variability in individual robot behavior. This robustness is critical in the high success rate for large self-assemblies.

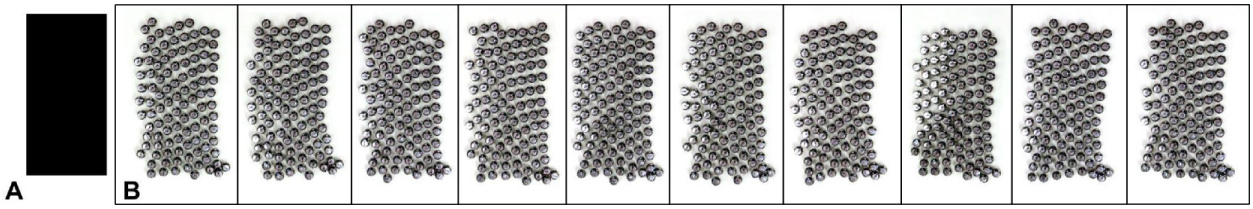


Figure S4: (A) The desired rectangle shape. (B) The resulting shape formed for each of the 10 experiments with approximately 100 robots (remaining edge-following robots after completion of the shape have been removed). In each of these experiments, the four seed robots are located in the lower right corner of the formed shape.

# 4 Proof of Correctness

## 4.1 Problem Definition

We start by defining formally the problem of distributed shape formation in the Euclidean plane.

Specifically let the *desired shape* $\mathcal{S} \subseteq \mathbb{R}^2$ be a closed, bounded and connected subspace of the Euclidean plane, where $\partial\mathcal{S}$ denotes the topological boundary of the shape. We consider only shapes without holes and with a minimum thickness, where the minimum thickness requirement is captured by shapes which are connected after being eroded (as defined in mathematical morphology) with a disk of radius $3r$.

Let $V$ denote a finite set of robots which we model as closed two-dimensional disks of radius $r$. For a robot $v \in V$ we let $p_t(v) \in \mathbb{R}^2$ denote its center at time $t$. Robots cannot occupy the same space at the same time (i.e., the disks do not overlap), and thus the minimum distance between two robots is $2r$. We assume robots are identical and in particular they all move at the same speed.

The configuration of the robots at time $t$ induces an undirected graph $G = (V, E_t)$ where there is an edge $\{u, v\} \in E_t$ between robots $u$ and $v$ iff $\|p_t(u) - p_t(v)\| = 2r$ (i.e., iff the distance between $u$ and $v$ at time $t$ is exactly $2r$). Two robots $u$ and $v$ are *connected* at time $t$ if there is a path between $u$ and $v$ in $G_t$, and we say $G_t$ is connected if every pair of robots in $V$ is connected at time $t$. We use $N_t(v) = \{u \mid \{u, v\} \in E_t\}$ to denote the neighbors of robot $v$ at time $t$.

We assume that a robot can sense the complete state of all of robots closer than distance $d$, and $d > 4r$ ( in practice this is achieved through communication where $d$ is the communication range).

### 4.1.1 Assumptions of Initial Configuration

There is a distinguished subset of four *seed robots* $B \subseteq V$, including a unique *root robot*. The seed robots remain stationary throughout the execution and do not actively participate in the shape formation algorithm; their purpose is to serve as the origin of a virtual coordinate system which is computed during the shape formation algorithm.

Every robot is given a description of the desired shape, including its scale. Moreover, we assume that every robot knows its position with respect to the desired shape. In practice this is achieved through a virtual coordinate system which is computed in tandem with the shape formation algorithm. The virtual coordinate system is described in detail in Section 1 of the supplement.

Initially all non-seed robots are constrained to be in a connected configuration inside a hexagonal lattice with spacing $2r$, where every robot is at distance greater than $3r$ from any point in the desired shape. The empty positions on the lattice are initially two-connected. Once a non-seed robot starts moving it is no longer constrained to be in a lattice position.

We also place constraints on the placement of the seed robots with respect to the shape and the non-seed robots. Informally we require seed robots to be connected to each other, connected to the idle robots, and to be in a "corner" of the shape. In more detail, let $B = \{v_0, v_1, v_2, v_3\}$ be the seed robots, where $v_0$ is the root robot. Robot $v_1$ must be connected to robot $v_0$, and robots $v_2$ and $v_3$ must be connected to robots $v_0$ and $v_1$. Moreover, robot $v_2$ should be connected to at least one robot in the lattice formed by non-seed robots and its center must be outside the convex hull of the desired shape. Finally, the unique point $q$ at distance $(2\sqrt{(3)} + 2)r$ from the center of $v_2$ and at distance $2r$ from the center of $v_3$ should be at distance greater than $3r$ from any point outside the shape (i.e., the point $q$ is well inside the shape). See Figure S5 for an example.

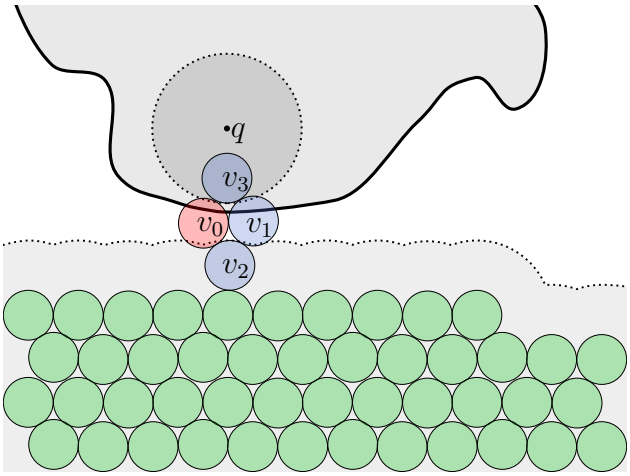


Figure S5: Initial Configuration. Idle robots are green, seed robots are blue and root robot is red. The point $q$ must be at distance greater than $3r$ from any point outside the shape, and the center of every idle robot should be at distance greater than $3r$ from any point inside the shape.

## 4.2 Algorithm Description

We consider a simplified version of the shape formation algorithm where robots can only be in one of 5 states IDLE, MOVE-OUT, MOVE-IN, STOP-IN, STOP-OUT. Initially all robots executing the algorithm start in the IDLE state. We refer collectively to robots in the MOVE-IN and MOVE-OUT state as *mobile* robots, and to robots in the STOP-IN and STOP-OUT state as *stopped* robots.

Robots which are idle or stopped remain stationary, and mobile robots move according to the edge-following algorithm described in pseudo-code in Algorithm 6. Briefly, a robot in the MOVE-IN or MOVE-OUT state performs a clockwise rotation around one of its neighbors, choosing to rotate around a different neighbor only when necessary to continue the clockwise motion without overlapping with another robot. See Figure S6 for a diagram of a sample execution.

**Algorithm 6** Edge Following for Robot $u$

$e_t(u) \leftarrow$ any robot in $N_t(u)$
**loop**
  rotate clockwise around $e_t(u)$ until "hitting"[1] a robot $w$
  $e_t(u) \leftarrow w$

The shape formation algorithm is described formally, together with its state transitions, in Figure S7. The following paragraphs we provide a textual description of the state transitions.

Starting from the IDLE state robots will transition to the MOVE-OUT state in a way in which guarantees the collection of idle robots will not be disconnected, and there is a minimum separation between robots which are in the mobile states. Observe that since IDLE robots start outside the shape and remain stationary while being idle, then when a robot first transitions to the MOVE-OUT it is outside the shape.

A robot in the MOVE-OUT state that enters the shape while executing the edge-following motion

[1] Robot $u$ "hits" a robot $w$ if continuing the rotation around $e_t(u)$ would require that robot $u$ and robot $w$ overlap.

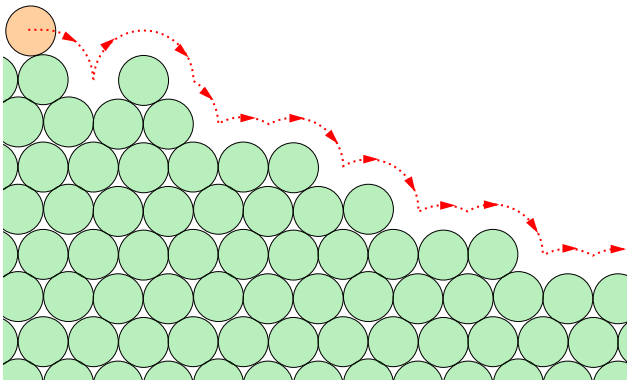

Figure S6: Orange robot is a mobile robot using the edge-following algorithm to navigate around a collection of green idle robots.

— entering the shape means the center of the robot is inside the shape and the closest shape boundary is at least $3r$ from its center— transitions to the MOVE-IN state.

A robot in the MOVE-IN state that leaves the shape while executing the edge-following motion — leaving the shape means the center of the robot is inside the shape and the closest shape boundary is at distance $r$ from its center— transitions to the STOP-OUT state.

A robot in the MOVE-IN state that changes from edge-following a robot with a lower gradient value to a edge-following a robot with a higher gradient value, transitions to the STOP-IN state.

We remark that the conditions used for entering and leaving the shape are not symmetrical, in particular it is "harder" to enter the shape than it is to leave it. The minimum thickness assumption on the desired shape is leveraged to deal with this.

An idle robot $u$ maintains a local id $lid(u) \in \mathbb{N}$ which is unique among all idle robots at distance less than or equal to $d$ (in practice this id is generated through randomization), and a hop count $h(u) \in \mathbb{N}$ which represents the length of the shortest path from $u$ to the root robot in the graph induced by the idle robots and the seed robots.

When a robot $u$ is mobile it keeps track of the robot around which it is currently rotating $e_t(u)$ and remembers the robot around which it was previously rotating while in the present state $e'_t(u)$. When a mobile robot first enters the state we assume $e_t(u) = e'_t(u)$.

When a robots $u$ enters a stopping state it assigns itself a gradient value $g(u)$. Specifically, if a robot $u$ stops while edge-following a robot $v$ then if it entered the STOP-IN state it assigns a gradient value $g(u) = g(v)$ it if entered the STOP-OUT state it assigns a gradient value $g(u) = g(v) + 1$.

Below we list the formal descriptions of the state transition rules.

START-MOVE: If $\nexists v$ where $\|p_t(u) - p_t(v)\| < d$ and where either $v$ is mobile or $v$ is idle and $(h(u), lid(u))$ is lexicographically smaller than $(h(v), lid(v))$.

ENTER-SHAPE: if $p_t(u) \in \mathcal{S}$ and the closest point in the boundary of the shape is at distance greater than $3r$ from $p_t(u)$.

LEAVE-SHAPE: if $p_t(u) \in \mathcal{S}$ and continuing edge-following would imply the closest point in the boundary of the shape is at distance $r$ from $p_t(u)$.

GRADIENT: If $u$ transitions from edge-following a robot $v$ to edge-following a robot $w$ where $g(v) \neq g(w)$.

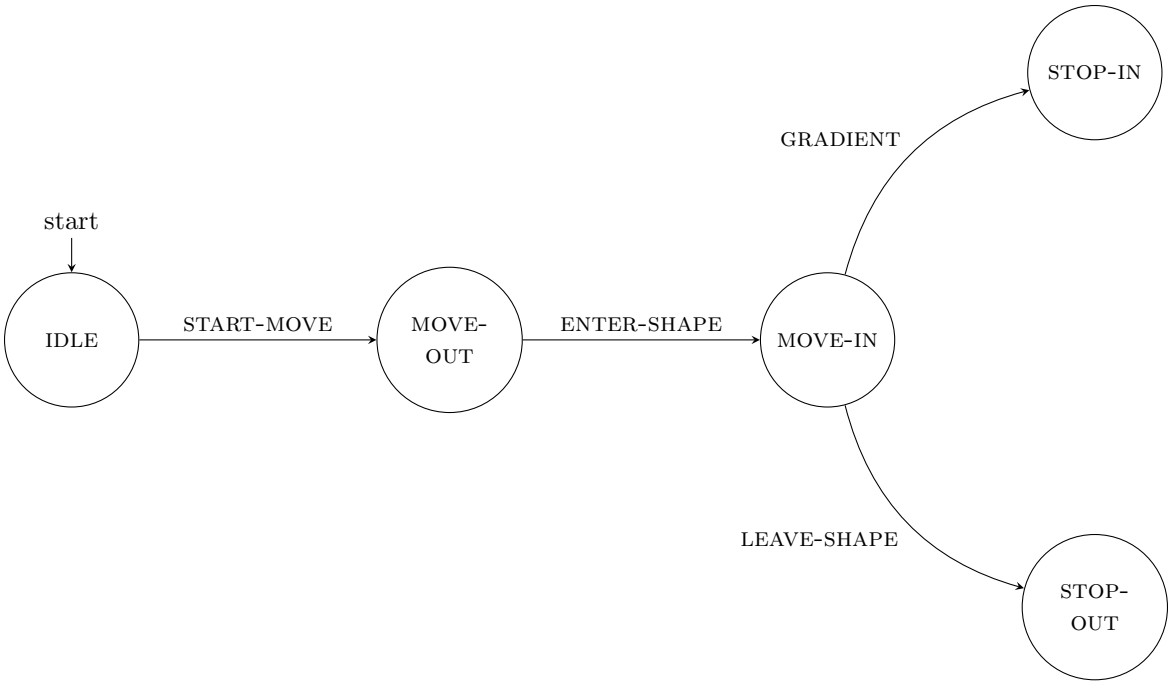


Figure S7: Shape Formation State Machine Representation

## 4.3 Correctness Proof

Broadly speaking the proof of correctness is composed of three parts. The first part reasons about idle robots and how they transition to a mobile state. The second part studies the path followed by mobile robots before stopping and joining the shape. The third and part reasons about the positions occupied by stopped robots, and shows that the stopped robots are packed together tightly (no "holes"). A more detailed outline of each of these parts is described in the paragraphs below.

Initially the idle robots start outside the shape and arranged in a tightly packed and connected lattice configuration adjacent to the seed robots and the desired shape. The rules used to transition from an idle state to a mobile state guarantee that only robots in the outer perimeter of the group of idle robots start moving, and that moving robots are sufficiently spaced apart from each other. This ensures that the remaining idle robots continue to be in a tightly packed connected lattice configuration. Moreover, the rules used to transition out of the idle state also guarantee that mobile robots eventually edge-follow a seed robot. This is shown in Section 4.3.1.

Mobile robots move by edge-following neighboring robots. For simplicity throughout the analysis we make the assumption that only one robot moves at a time. This assumption can be made without loss of generality, since the transition rules guarantees a minimum spacing between mobile robots that ensures two mobile robots will never interact with each other. Our main observation regarding mobile robots, is that as long as there is a point inside the shape which is reachable and can fit an additional robot, the path followed by a mobile robot will end inside the shape and the mobile robot will transition to a stopping state. This is shown in Section 4.3.2.

From the algorithm description it follows that robots will only stop once inside the shape. However, it turns out that the algorithm is such that robots will always stop and join the shape in

small linear sequences that we shall call ribbons. These ribbons always stack on top of each other in a way that ensures no additional robots can be placed in between ribbons. Finally, from this it follows that when no additional robot joins the shape (and thus no new ribbons are created) the shape cannot fit any additional robots. This is shown in Section 4.3.3.

Finally Section 4.3.4 concludes by showing that together, these three parts show that, assuming there are sufficiently many robots to do so, the shape formation algorithm is guaranteed to fill the inside of the desired shape.

### 4.3.1 Idle robots

The main result of this section is to show that as robots transition from the idle state to a mobile, state they are guaranteed to edge-follow a seed robot. To prove this we first show that the START-MOVE condition implies that any robot that starts moving will have two neighboring empty adjacent positions in the lattice of idle robots. Next we show that since every robot starts moving only when it has two neighboring empty adjacent positions in the hexagonal lattice of idle robots, then it must edge-follow the idle robots until reaching a seed robot.

**Definition 1.** A *straight line path* is a path of connected idle robots thats is in a straight line (i.e. lattice axis aligned path).

**Proposition 1.** An idle robot $w$ with hop-count $k$ can only have neighbors with hop-count $k-1, k, k+1$.

*Proof.* First observe that $w$ cannot have a neighbor $v$ with hop-count $k-a$ where $a > 1$ since the hop-count of $w$ would be at most $k-a+1 < k$ (using the path to the root starting from $v$). Similarly $w$ cannot have a neighbor $v$ with hop-count $k+a$ where $a > 1$, since $v$ is neighbors with $w$ with hop $k$, so $v$ can have a hop-count of at most $k+1$. □

**Proposition 2.** An idle robot $w$ with hop-count $k$ has at least one neighbor with hop-count $k-1$.

*Proof.* Robot $w$ must have a minimum path to the root robot of length $k$, and this path must go through one of its neighbors $v$, thus the minimum path from $v$ to the root is of length $k-1$. □

**Proposition 3.** If the graph of idle robots contains no holes (empty space surrounded by connected robots), an idle robot $w$ with hop-count $k$, cannot have any neighbors $u, v$, both with hop-count $k-1$ where $u, v$ are not neighbors.

*Proof.* Robot $w$ has at least one minimum path $p_{wu}$ from $w$ to the root that goes through robot $u$, and one minimum path $p_{wv}$ from $w$ to the root that goes through robot $v$.

Moreover, all lattice positions enclosed between $p_{wu}$ and $p_{uv}$ must be occupied, otherwise there would be a hole in the lattice of idle robots.

If $u$ is not neighboring $v$, then there must exist a straight line path from $w$ that intersects with either $p_{wu}$ or $p_{wv}$, but this would contradict that both $p_{wu}$ and $p_{wv}$ are minimum paths. □

**Proposition 4.** If the graph of idle robots contains no holes, an idle robot with hop-count $k$ can have no more than two neighbors with lower hop-count.

*Proof.* From proposition 1, no robot with hop-count $k$ can have neighbors lower than $k-1$. If it has more than two neighbors with lower hop-count, they must all have hop-count $k-1$, and there must exist two robots $u$,$v$ that have hop count $k-1$ and are not neighbors, violating proposition 3. □

**Proposition 5.** If the graph of idle robots contains no holes, a clique of three idle robots can't have the same hop-count.

*Proof.* Suppose by contradiction that three idle robots in a clique have the same hop-count. It follows that a minimum path from any of these robots to the root cannot go through any other robot in the clique (otherwise they wouldn't have the same hop-count). For two of the three robots $u, v$ there must be a minimum path to the root that surrounds the third robot $w$. The space between these paths must be filled by robots, or a hole would exist.

Consider the straight line path $D$ starting at robot $u$ and in the direction of robot $w$, and the straight line path $E$ starting at robot $v$ and in the direction of robot $w$. If $u$'s shortest path crosses $E$ or $v$'s shortest path crosses $D$, then that path cannot be a shortest path. However since $w$ is inside the shortest paths of $u, v$ then $u$'s shortest crosses $E$ and/or $v$'s shortest path crosses $D$.

□

**Proposition 6.** If the graph of idle robots contains no holes, an idle robot $w$ has hop-count $k$ and the graph induced by $w$'s neighbors has three connected neighbors $u, v, z$, where $u$ has hop-count $k-1$ and $v$ does not have a hop-count of $k-1$, then $v$ must have a hop-count of $k$ and $z$ must have a hop-count of $k+1$.

*Proof.* From proposition 1, $v$ must have hop-count of $k$ or $k+1$. $v$ cannot have hop-count $k+1$, as it is touching a robot with hop-count $k-1$, therefore $v$ has hop-count $k$. $z$ cannot have hop-count $k-1$, as proposition 3 would be violated, and $z$ cannot have a hop-count $k$, otherwise $z, v, u$ are in a clique of three robots touching each other and all 3 have the same hop-count, violating proposition 5, so $z$ must be $k+1$. □

**Proposition 7.** If the graph of idle robots contains no holes, and an idle robot $u$ with hop-count $k$ has 6 or 5 neighbors, it has have a neighbor with hop-count $k+1$.

*Proof.* From proposition 2 and proposition 4, robot $u$ must have one or two neighbors of hop-count $k-1$. If it has two neighbors of hop-count $k-1$ then proposition 3 implies they will be adjacent to each other. Therefore if $u$ has 5 or 6 neighbors, it must have three connected neighbors $v, w, z$ where $v$ has hop-count $k-1$ and $w$ does not have hop-count $k-1$, so by proposition 6, $z$ must have a hop-count $k+1$. □

**Proposition 8.** The graph of idle robots contains no holes.

*Proof.* For a hole to be created, a robot with 6 neighbors must start moving. From section 4.1.1 the locations without robots must belong to a single 2-connected graph, so the starting group of idle robots contains no holes. From proposition 7, if no holes exist, no robot with 6 neighbors will ever start moving, therefore a hole is never created. □

**Proposition 9.** Given a idle robot $z$ with hop-count $k$, if the graph induced by $z$'s neighbors is not connected, it must have a neighbor with hop-count $k+1$.

*Proof.* From the graph induced by $z$'s neighbors, exactly one of the connected components, must have a $k-1$ robot, or proposition 3 is violated. Any other connected component, $H$ must have an empty location on both sides of it. All shortest paths of robots in $H$ must go through robot $z$, otherwise the shortest paths of these robots and the path from robot $z$ create a chain of robots surrounding an empty lattice position, violating proposition 8. Since the robots part of $H$ have a shortest path goes through robot $z$, robots in this group have hop-count $k+1$. □

**Proposition 10.** If an idle robot $z$ with hop-count $k$ has 3 or 4 neighbors and no two adjacent empty lattice positions, then it must have a neighbor with hop-count $k+1$.

*Proof.* If robot $z$ has 3 or 4 neighbors with no two adjacent empty neighbors, then it follows that the graph induced by its neighbors is disconnected. Therefore proposition 9 implies $z$ has a neighbor with hop-count $k+1$. □

**Proposition 11.** If an idle robot does not have two empty adjacent lattice positions, then it has a neighbor with hop-count $k+1$.

*Proof.* If the robot has 6,5,4,or 3 neighbors, and no two empty adjacent lattice positions, it will have a neighbor with hop-count $k+1$ (proposition 7, proposition 8, proposition 10). If a robot has 2 or 1 neighbors, it must have 2 empty adjacent positions, and a robot cannot have 0 neighbors, or proposition 2 is violated. □

Since a robot will not move when it has a neighbor of higher hop-count, proposition 11 shows that any robot that starts moving will have two neighboring empty adjacent lattice positions in the lattice. Next we will show that if every robot starts moving only when it has two neighboring empty adjacent positions in the lattice, it will always edge follow to the root robot.

**Proposition 12.** The graph $G_{empty}$ of locations on the hexagonal lattice not-occupied by idle robots and outside the desired shape form a single bi-connected graph.

*Proof.* The graph of non-occupied locations is initially biconnected (from starting assumptions). From proposition 11 any empty lattice location added to $G_{empty}$ (i.e. the robot in that position started moving) has at least 2 edges, then the resulting graph after the addition is also biconnected. □

**Proposition 13.** Two idle robots $u$,$v$ where $\|p_t(u) - p_t(v)\| = r\sqrt{3}$ must share a neighbor.

*Proof.* Since idle robots are connected to the root robot by assumption, then there must be a path using only idle or seed robots between $u$ and $v$.

If $\|p_t(u) - p_t(v)\| = r\sqrt{3}$ then there are exactly 2 lattice positions $a, b$ where $u, v$ could share a neighbor. By contradiction, if both $a$ and $b$ are empty, then a chain of connected robots from $u$ to $v$ prevents $a$ from having more than one path in $G_{empty}$ to reach $b$, which would contradict proposition 12. □

**Definition 2.** An *idle edge-following cycle* is the cycle of robots another robot would edge-follow when executing the edge-following algorithm using only idle robots.

An idle edge-following cycle is *empty* if the cycle of robots surrounds an empty lattice position.

**Proposition 14.** Two robots which are adjacent in the idle edge-following cycle must be at distance less than $4r$ from each other.

*Proof.* This follows directly from the definition of edge-following, since if robot $u$ is edge-following a robot $v$ it cannot switch to edge-following a robot $w$ unles $\|p_t(w) - p_t(v)\| < 4r$. □

**Proposition 15.** There are no empty idle edge-following cycles.

*Proof.* From proposition 14, an empty idle edge-following cycle is composed of a cycle of robots at distance less than $4r$ from each other. In a hexagonal lattice, distances between lattice positions that are $< 4r$ are limited to $2r$ or $r\sqrt{3}$. For any pair of adjacent robots $u$ and $v$ in the edge-following cycle which are at distance $r\sqrt{3}$ from each other, proposition 13 implies there is a $w$ such that the distance between $u$ and $w$ and $w$ and $v$ is $2r$. Therefore there is a cycle of robots, where each pair of robots adjacent in the cycle are at distance $2r$ from each other. However, if this cycle contains an empty lattice position there would be a hole in the graph of idle robots, which is impossible due to proposition 8. □

**Proposition 16.** Every idle robot is connected to the seed robots using a path of neighboring idle robots.

*Proof.* From the assumptions in the initial configuration it follows that initially all idle robots are connected to the root robot and therefore they all have a finite hop count.

To prove the statement it suffices to show that as robots transition from being idle to being mobile, the hop-count of the remaining robots remains finite. Next we show the hop-count of the idle robots remains unchanged, which proves the statement.

From the definition of START-MOVE for a robot $v$ to transition from being idle to mobile then robot $v$ must not have any neighbors in the idle state with a greater hop-count. However, this implies that when a robot transitions from being idle to being mobile it cannot increase the hop-count of any other robot in the idle state. □

**Proposition 17.** If there are any idle robots, there exists exactly one non-empty idle edge-following cycle that surrounds all idle robots.

*Proof.* From proposition 15 there are no empty idle edge-following cycles therefore we need only to show that there cannot be zero or more than one idle edge following cycles. The assumption that there are idle robots implies there is at least one non-empty idle edge following cycle.

Moreover, if there were two or more non-empty idle edge following cycle, then robots surrounded by different non-empty idle edge following cycle would be disconnected in the lattice, which contradicts proposition 16. Therefore there is a single idle edge-following cycle, and since the set of idle robots are connected this cycle contains all idle robots. □

**Theorem 18.** *Any robot that transitions out of the idle state must eventually edge-follow a seed robot.*

*Proof.* From proposition 17 a robot that transitions out of the idle state will edge-follow an idle cycle that encloses all idle robots until it encounters a non-idle robot. From the initial assumptions any point in the shape is at distance greater than $3r$ from any of the idle robots that are edge-followed during the idle cycle. Therefore a robot following the idle cycle cannot encounter any

stopped robots. Moreover, since by the initial assumptions the seed robots are connected to the idle robots, and the idle robots have no holes, it follows that a robot following the idle cycle must encounter a seed robot, which concludes the proof. □

So we have shown that whenever a robot transitions from being idle to mobile it doesn't create holes in the lattice of idle robots and will always reach the seed robots.

### 4.3.2 Mobile Robots

This section analyzes the behavior of mobile robots, and in particular the topological path followed by a moving robot. Informally speaking, we will show that as long as there is a path which can lead a robot to a point inside the desired shape, then mobile robots are guaranteed to continue entering and stopping inside the desired shape.

Before analyzing the path followed by mobile robots, in this section and the next one it will be useful to make the simplifying assumption that at any point during the execution there is at most one mobile robot. Next we will show that this assumption can be made without loss of generality.

**Proposition 19.** An execution where mobile robots are always at distance greater than $4r$ from each other, is indistinguishable from an execution where there mobile robots move one at a time.

*Proof.* Let $(u_1, u_2, \ldots)$ be the sequence of mobile robots in the order in which they neighbor the root robot for the first time; by theorem 18 all mobile robots must neighbor the root robot, and by the assumption that mobile remain at distance greater than $4r$ it follows they can't neighbor the root robot at the same time.

For the base case consider robot $u_1$, and observe that before it stops it will never see another mobile robot, and therefore cannot distinguish the execution from one where it was the only mobile robot.

For the inductive step consider robot $u_i$, and observe that it cannot stop before neighboring robot $u_{i-1}$, since up to that point it observes exactly the same environment and therefore must take the same decisions as robot $u_{i-1}$. However, since robot $u_i$ and $u_{i-1}$ cannot be neighbors while mobile (since by assumption they remain at distance at least $4r$ when mobile), it follows that before robot $u_i$ stops, it cannot distinguish the execution from one where it was the only mobile robot and $u_1, \ldots, u_{i-1}$ were already stopped. □

The rules used for transitioning from being idle to being mobile guarantee a minimum separation between mobile robots (in terms of the edge-following path), which is summarized by the next proposition.

**Proposition 20.** When measured clockwise in the edge-following path, two mobile robots are never closer than $d$ from each other.

*Proof.* From START-MOVE it follows that a robot will only transition to a mobile state when the euclidean distance to any other mobile robot is at least $d$. Since the distance in the edge-following path between two points can be no less than the euclidean distance between them, this implies initially the distance along the edge-following path between two robots is at least $d$. Since robots move at the same speed, then it follows that the distance measured clockwise in the edge-following path between two moving robots will remain at least $d$. □

By making the communication distance $d$ large enough with respect to the curvature of the desired shape, we can ensure that mobile robots never get closer than $4r$ from each other. This together with the previous propositions implies that without loss of generality we can assume that at any point during the execution there is at most one mobile robot.

In the rest of the proof we reason about the path followed by a mobile robot.

**Definition 3.** For a mobile robot $v$ let $\text{path}_v$ be the topological path it follows during its motion.

Recall that theorem 18 implies that for every mobile robot $v$ while edge-following through the path $\text{path}_v$, it must edge-follow a seed robot.

Next we describe two different partitions of the Euclidean plane based on the positions of stopped robots.

**Definition 4.** A point $x \in \mathbb{R}^2$ is *occupied* at time $t$ iff there is an idle or stopped robot such that $\|x - p_t(v)\| \leq 2r$, otherwise $x$ is *unoccupied*.

**Definition 5.** A point $x \in \mathbb{R}^2$ is *reachable* at time $t$ iff there is a path (in the topological sense) from a point in the idle cycle at time $t$ to $x$ that doesn't contain occupied points at time $t$, otherwise $x$ is *unreachable*.

From the first definition it follows that if there is an unoccupied point inside the desired shape then the shape can fit an additional robot. Conversely if there are no unoccupied points inside the desired shape then it means that the shape cannot fit an additional robot.

Moreover, from the second definition we have that in order for a mobile robot to reach an unreachable point, it must at some point overlap with stopped robots, which is impossible by assumption.

The next three propositions examine the properties of the path followed by mobile robots.

**Proposition 21.** For a mobile robot $v$ either $\text{path}_v$ is itself a loop or it contains no loops.

*Proof.* Suppose by way of contradiction that $\text{path}_v$ is not a loop but contains a loop.

Let $w_0$ be the last robot that robot $v$ edge-follows in $\text{path}_v$ before entering the loop, let $w_1$ the robot that $v$ edge-follows after $w_0$ (and therefore $w_1$ is part of the loop), and let $w_2$ be a robot inside loop which is edge-followed by $v$ right before returning to $w_1$.

Since robot $v$ transitioned between $w_0$ and $w_1$ when entering the loop, then the distance between the centers of those two robots must be strictly less than $4r$. Similarly, robot $v$ transitions between $w_2$ and $w_1$ when inside the loop, then the distance between the centers of those two robots must be strictly less than $4r$.

However, it has to be the case that either robot $v$ passes through $w_2$ and $w_1$ when it first enters the loop (which would contradict the fact that the distance between the center of $w_1$ and $w_2$ is less than $4r$), or robot $v$ passes through $w_0$ and $w_1$ when edge-following $w_2$ in the loop (which would contradict the fact that the distance between the center of $w_0$ and $w_1$ is less than $4r$). □

To simplify the remaining discussion, we introduce an additional definition.

**Definition 6.** A point is *well inside* the desired shape, if it is inside the desired shape and the closet point in the shape boundary is more than $3r$ away.

**Proposition 22.** For a mobile robot $v$ either $\text{path}_v$ is itself a loop with no points well inside the desired shape, or it contains a point well inside the shape and has no loops.

*Proof.* We need to consider only two cases.

First suppose that $\text{path}_v$ is a loop. In that case if there is a point of $\text{path}_v$ which is well inside the shape, then by continuing traversing the loop defined by $\text{path}_v$, robot $v$ must eventually leave the shape (since when robot $v$ first starts edge-following it was outside the shape.) However, that is impossible since the transition LEAVE-SHAPE prevents robot $v$ from leaving the shape.

Next suppose that $\text{path}_v$ is not a loop, then proposition 21 implies that $\text{path}_v$ contains no loops. In this case, $\text{path}_v$ has both a beginning and an end, and since a mobile robot $v$ will only stop edge-following after being well inside the shape and then reaching a stopping condition, then it follows that $\text{path}_v$ must have a point well inside the shape. □

**Proposition 23.** If robot $v$ became mobile at a time $t$ where there is a reachable unoccupied point well inside the desired shape then $\text{path}_v$ contains a point well inside the desired shape and has no loops.

*Proof.* Suppose robot $v$ became mobile at a time $t$ when there is an reachable unoccupied point $x$ well inside the shape. From proposition 22 we need only to show that $\text{path}_v$ cannot be a loop that does not contain a point well inside the shape.

Suppose by contradiction that $\text{path}_v$ is a loop and does not contain a point well inside the shape. Theorem 18 implies at some point during $\text{path}_v$ robot $v$ edge-follows a seed robot, and since the seed robots are connected this implies all seed robots are enclosed by the loop defined by $\text{path}_v$. Moreover, by the initial assumptions on the configuration there is a point $q$ which is at well inside the shape and at distance $2r$ from one of the seed robots. Since by assumption this point cannot be contained in $\text{path}_v$, then it must be enclosed by it.

From this it follows that the point $x$ must also be enclosed by $\text{path}_v$ since otherwise we have point $q$ is well inside the shape and enclosed by $\text{path}_v$ and point $x$ is well inside the shape and outside the area enclosed by $\text{path}_v$, which together with the fact that $\text{path}_v$ has no points well inside the shape would violate the minimum thickness assumption on the shape. On the other hand if $x$ is enclosed by $\text{path}_v$ it contradicts that $x$ is reachable, which concludes the proof. □

We are finally ready to prove one of the main results in this section. The following theorem captures the fact that additional robots will stop inside the desired shape as long as there is a place inside the desired shape which can fit a robot and which can be reached without having to overlap with other robots on the way.

**Theorem 24.** *If there are mobile or idle robots and there is an reachable unoccupied point well inside the desired shape, then an additional robot stops.*

*Proof.* Suppose there are idle or mobile robots and there is a reachable unoccupied point well inside the desired shape. First we show that we need only to consider the case where there is at least one mobile robot.

If there are no mobile robots and there are idle robots, then the idle robot $u$ with the lexicographically greater tuple $(h(u), lid(u))$ would have immediately transitioned to a MOVE-OUT state.

So given a mobile robot $v$ then proposition 23 implies that $\text{path}_v$ has no loops and contains a point well inside the desired shape. Finally since paths end only a robot enters the stopping state, then the theorem follows. □

### 4.3.3 Stopped Robots

This section reasons about the way in which robots stop and join the desired shape. We will show that stopped robots join the desired shape in structured sequences that we will call "ribbons". Moreover, we will also show that these ribbons satisfy certain geometrical constraints and form on top of a previous ribbon. Finally, we will show that the properties satisfied by these ribbons imply that no additional robots can be placed between ribbons. This, together with the results in the previous section show that when the algorithm terminates the desired shape cannot fit any additional robots.

Using the assumption that robots move one at a time, we can easily induce an ordering on the stopped robots. Specifically we label the stopped robots $u_1, u_2, \ldots$ where robot $u_{i+1}$ stops after robot $u_i$ (or equivalently robot $u_{i+1}$ becomes mobile after robot $u_i$).

For the remainder of the analysis it will be useful to consider contiguous subsequences of stopped robots. For that purpose we formalize the notion of a ribbon of robots.

**Definition 7.** A ribbon $(w_1, \ldots, w_k)$ is a maximum contiguous subsequence of the stopped robots such that robot $w_1$ is in the STOP-OUT state, and robot $w_i$ for $i > 1$ is in the STOP-IN state.

We will later show that all robots in the same ribbon have the same gradient value and are connected in a line graph (i.e. robot $w_i$ neighbors robot $w_{i-1}$ for $i > 1$). For a ribbon $R_i$ we denote with $g(R_i)$ the gradient value of the robots contained in that ribbon.

To include the seed robots into the same analysis technique we define the special cases of the root ribbon $R_0 = (v_0)$ and the seed ribbon $R_1 = B \setminus \{v_0\}$. The definition of ribbons induces a partition of the sequence of stopped robots into a sequence of ribbons $(R_0, R_1, R_2, \ldots)$ such that for any $i < j$ when a robot of ribbon $R_j$ entered the STOP state, all robots of ribbon $R_i$ had already entered the STOP state. See Figure S8 for an example of a sequence of ribbons.

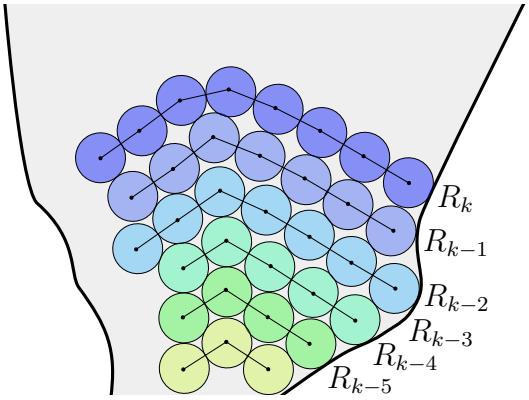


Figure S8: Stopped robots in the shape are grouped into ribbons (robots with the same color). The right most robot in each ribbon is the first robot of that ribbon, and it is in a STOP-OUT state, all other robots in the ribbon are in a STOP-IN state.

The next definition formalizes the concept of a parent ribbon, where informally a child ribbon forms "on top of" a parent ribbon (we will later show ribbons always form in this way).

**Definition 8.** Consider ribbon $R_i = (u_1, \ldots, u_n)$ and ribbon $R_j = (v_1, \ldots, v_m)$. $R_j$ is *parent* of $R_i$ iff $g(R_i) = g(R_j) + 1$ and every robot $u_k$ in $R_i$ neighbors a robot in $v_l$ in $R_j$ and if $k < n$ then $u_{k+1}$ neighbors $v_l$ or $v_{l+1}$.

We use the notion of $R_j$ being parent of $R_i$ and $R_i$ being child of $R_j$ to mean the same thing. Moreover when referring to individual robots $u \in R_j$ and $v \in R_i$, we will say that $u$ is the parent of $v$ (or equivalently $v$ is the child of $u$) if $u$ and $v$ are neighbors and $R_j$ is the parent of $R_i$.

To prove the different properties satisfied by ribbons we must study the path followed by robots before stopping. For this purpose it is useful understand the path that "would" be followed by a robot if it didn't enter a stopping state. For the sake of the proof, we introduce the perpetual edge-following algorithm, which is essentially the shape formation algorithm without stopping states.

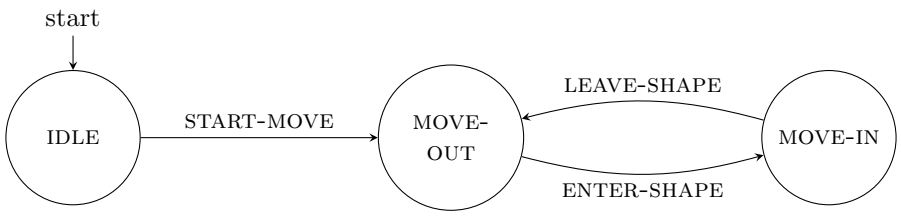


Figure S9: Perpetual Edge-Following Algorithm

We let extended_path$_v$ refer to the topological path followed by a mobile robot $v$ if it were executing the perpetual edge-following algorithm until it returns to its starting position.

Observe that by definition it follows that path$_v$, which is the path followed by robot $v$ if executing the shape formation algorithm, is a prefix of extended_path$_v$, which is the path followed by $v$ if it executed the perpetual edge-following algorithm.

The next proposition captures a useful observation about the path followed by two consecutive stopped robots.

**Proposition 25.** Given two consecutive stopped robots $u_i$ and $u_{i+1}$, robot $u_{i+1}$ edge-follows the same robots as $u_i$ up until the point where it edge-follows $u_i$ itself.

*Proof.* This follows from the fact until robot $u_{i+1}$ neighbors and starts edge-following robot $u_i$, it will have executed the same algorithm and observed exactly the same stopped robots observed by robot $u_i$ before it stopped. □

Proposition 25 guarantees that robot $u_{i+1}$ will edge-follow robot $u_i$ before stopping. We leverage this fact to examine the extended edge-following path of a robot $u_{i+1}$, before and after it starts edge-following robot $u_i$. For this purpose we introduce the following definition (exemplified in Figure S10).

**Definition 9.** Consider a stopped robot $u_i$.

- $\mathrm{prefix}_{u_i}$ is the path segment from extended_path$_{u_{i+1}}$ while robot $u_{i+1}$ is in the MOVE-IN state, and before and excluding the point where it starts edge-following robot $u_i$.
- $\mathrm{suffix}_{u_i}$ is the path segment(s) from extended_path$_{u_{i+1}}$ while robot $u_{i+1}$ is in the MOVE-IN state, and after and including the point where robot $u_{i+1}$ starts edge-following robot $u_i$.

Observe that Proposition 25 implies that a robot $u_{i+1}$ will only stop when in $\mathrm{prefix}_{u_i}$ or the in the first segment of $\mathrm{suffix}_{u_i}$.

Now we define a proper sequence of robots.

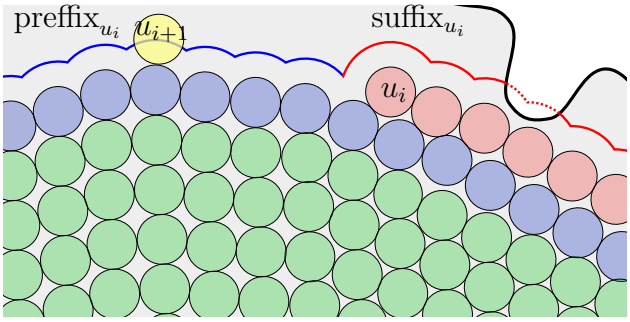


Figure S10: Ribbon of red robots is being formed on top of ribbon of blue robots. Robot $u_i$ is the last robot to join the red ribbon. $\text{preffix}_{u_i}$ is the path segment of $extended\_path_{u_{i+1}}$ while inside the shape and before robot $u_{i+1}$ neighbors robot $u_i$. $\text{suffix}_{u_i}$ is a series of path segments of $extended\_path_{u_{i+1}}$ while inside the shape and after robot $u_{i+1}$ neighbors robot $u_i$.

**Definition 10.** A segment of an edge-following path is *proper* if the sequence of robots that form the path: a) are a contiguous subset of the same ribbon and ordered such that the times at which they stopped are non-increasing, b) and —except for the first and last robots— all robots contribute an arc of at least $\pi/3$ to the path.

With these definitions in place we can state and prove the next lemma which captures a series of useful properties about ribbons, as well as additional properties about the path edge-followed by robots while in the MOVE-IN state.

**Lemma 26.** The following invariants hold:

1. Robots in a ribbon have the same gradient value and are connected in a line graph.
2. Except for the root ribbon $R_0$, every ribbon has a parent.
3. For every stopped robot $u_i$:
   (a) the segment in $\text{preffix}_{u_i}$ is proper.
   (b) if $w \in \text{preffix}_{u_{i-1}}$ either $w \in \text{preffix}_{u_i}$ or $w$ is parent of $u_i$.
   (c) the segment(s) in $\text{suffix}_{u_i}$ are proper.
   (d) if $w \in \text{suffix}_{u_{i-1}}$ either $w \in \text{preffix}_{u_i} \cup \text{suffix}_{u_i}$ or $w$ is parent of $u_i$.
   (e) if both $\text{preffix}_{u_i}$ and $\text{suffix}_{u_i}$ are non-empty, the ribbon of the segment $\text{preffix}_{u_i}$ is the parent of the ribbon in the first segment of $\text{suffix}_{u_i}$.

*Proof.* We prove the statement by induction over the stopped robots, where the base case can be verified by inspection. For the inductive step we consider a stopped robot $u_{i+1}$ and assume that by inductive hypothesis the statement holds for robot $u_i$ and for all robots that stopped before $u_i$. We consider separately the case where $u_{i+1}$ is in the STOP-IN state or STOP-OUT state (following a similar outline for each case).

1. If $u_{i+1}$ is in the STOP-IN state, then it stopped exactly at the point where $u_{i+1}$ becomes neighbor of robot $u_i$ and transitions from edge-following a robot in $\text{preffix}_{u_i}$ to edge-following a robot in $\text{suffix}_{u_i}$.

   In this case robot $u_{i+1}$ joins the same ribbon as that of robot $u_i$, adopting the gradient value $g(u_{i+1}) = g(u_i)$; this already shows 1 and 2, it remains to show 3.

Moreover, observe that the resulting $\text{preffix}_{u_{i+1}}$ is identical to $\text{preffix}_{u_i}$ with the exception that the last robot of $\text{preffix}_{u_i}$ (which $u_{i+1}$ is neighboring when it stopped) could be excluded from $\text{preffix}_{u_{i+1}}$. Since by inductive hypothesis $\text{preffix}_{u_i}$ is proper then so is $\text{preffix}_{u_{i+1}}$, which proves 3a-b.

Next to analyze $\text{suffix}_{u_{i+1}}$ we must consider the extended edge-following path of $u_{i+2}$. Concretely consider the extended edge-following path of robot $u_{i+2}$ at the point where it starts edge-following $u_{i+1}$.

If robot $u_{i+2}$ never enters the MOVE-IN state while edge-following $u_{i+1}$, or if after entering the MOVE-IN state while edge-following $u_{i+1}$ it transitions to edge-following robot $u_i$, then $\text{suffix}_{u_{i+1}}$ is identical to $\text{suffix}_{u_i}$ with the exception that the first segment of $\text{suffix}_{u_{i+1}}$ could start with a contribution of $u_{i+1}$. Since by inductive hypothesis each segment in $\text{preffix}_{u_i}$ and $\text{suffix}_{u_i}$ is proper then it follows that each segment in $\text{suffix}_{u_{i+1}}$ is also proper (showing 3c). Moreover, the fact that $\text{suffix}_{u_{i+1}}$ is the same as $\text{suffix}_{u_i}$ but with the possible exception of an additional robot $u_{i+1}$ at the start of the first segment which is a child of $\text{preffix}_{u_i}$ (as was $u_i$) implies the remaining invariants 3d and 3e.

Otherwise, assume robot $u_{i+2}$ enters the MOVE-IN state while edge-following $u_{i+1}$ and let $w \neq u_i$ be the first robot it edge-follows after that; in the remainder of the proof we will show that such a robot $w$ cannot exist.

First observe that $w$ cannot be in $\text{preffix}_{u_{i+1}}$ since that would require robot $u_{i+2}$ to pass between two robots which are at distance less than $4r$ while edge-following. Therefore $w$ must be in $\text{suffix}_{u_{i+1}}$. If $w$ belongs to a different segment than $u_{i+1}$ this implies that the extended edge-following path of $u_{i+1}$ goes through a point $x$ which is outside the shape before reaching $w$. However observe that $x$ is enclosed by a loop of points inside the shape created by the edge-following path of $u_{i+2}$ from $u_{i+1}$ to $w$, and the robots used in the edge-following path of $u_{i+1}$ from $u_i$ to $w$, which contradicts the assumption that the shape has no holes.

Finally if $w$ belongs to the same segment as $u_{i+1}$, then the assumption that segments are proper —and in particular the fact that in proper segments all robots are connected in a line and each contribute at least $\pi/3$ to the edge-following path— would imply that the seed robots are contained inside the convex hull of the shape, which contradicts the assumptions on the initial placement.

2. If $u_{i+1}$ is in the STOP-OUT state, then $\text{preffix}_{u_i}$ was empty and $u_{i+1}$ stopped when edge-following the last robot of the first segment of $\text{suffix}_{u_i}$ exactly when it touches the boundary of the shape, but before leaving the shape.

   In this case $u_{i+1}$ creates a new ribbon, adopting gradient value $g(u_{i+1}) = g(R) + 1$ where $R$ is the parent ribbon that belongs to the first segment of $\text{suffix}_{u_i}$; this trivially shows 1 and 2, it remains to show 3.

   Moreover, observe that the resulting $\text{preffix}_{u_{i+1}}$ is identical to the first segment of $\text{suffix}_{u_i}$ with the exception that the last robot of $\text{suffix}_{u_i}$ (which $u_{i+1}$ is neighboring when it stopped) could be excluded from $\text{preffix}_{u_{i+1}}$. Since by inductive hypothesis $\text{suffix}_{u_i}$ is proper, then so is $\text{preffix}_{u_{i+1}}$, which proves 3.a, and since $\text{preffix}_{u_i}$ was empty we have 3.b. (This also shows 3.d for the case where $w$ is in the first segment of $\text{suffix}_{u_i}$.)

Next to analyze $\text{suffix}_{u_{i+1}}$ we must consider the extended edge-following path of $u_{i+2}$. Concretely consider the extended edge-following path of robot $u_{i+2}$ at the point where it starts edge-following $u_{i+1}$.

If robot $u_{i+2}$ never enters the MOVE-IN state while edge-following $u_{i+1}$, or if after entering the MOVE-IN state while edge-following $u_{i+1}$ it transitions to edge-following robot $u_i$, then $\text{suffix}_{u_{i+1}}$ is identical to $\text{suffix}_{u_i}$ with the exception that the first segment of $\text{suffix}_{u_{i+1}}$ could start with a contribution of $u_{i+1}$. Since by inductive hypothesis each segment in $\text{preffix}_{u_i}$ and $\text{suffix}_{u_i}$ is proper then it follows that each segment in $\text{suffix}_{u_{i+1}}$ is also proper (showing 3c). Moreover, the fact that $\text{suffix}_{u_{i+1}}$ is the same as $\text{suffix}_{u_i}$ but with the possible exception of an additional robot $u_{i+1}$ at the start of the first segment which is a child of $\text{preffix}_{u_i}$ (as was $u_i$) implies the remaining invariants 3d and 3e.

Otherwise, assume robot $u_{i+2}$ enters the MOVE-IN state while edge-following $u_{i+1}$ and let $w \neq u_i$ be the first robot it edge-follows after that; in the remainder of the proof we will show that such a robot $w$ cannot exist.

First observe that $w$ cannot be in $\text{preffix}_{u_{i+1}}$ since that would require robot $u_{i+2}$ to pass between two robots which are at distance less than $4r$ while edge-following. Therefore $w$ must be in $\text{suffix}_{u_{i+1}}$. If $w$ belongs to a different segment than $u_{i+1}$ this implies that the extended edge-following path of $u_{i+1}$ goes through a point $x$ which is outside the shape before reaching $w$. However observe that $x$ is enclosed by a loop of points inside the shape created by the edge-following path of $u_{i+2}$ from $u_{i+1}$ to $w$, and the robots used in the edge-following path of $u_{i+1}$ from $u_i$ to $w$, which contradicts the assumption that the shape has no holes.

Finally if $w$ belongs to the same segment as $u_{i+1}$, then the assumption that segments are proper —and in particular the fact that in proper segments all robots are connected in a line and each contribute at least $\pi/3$ to the edge-following path— would imply that the seed robots are contained inside the convex hull of the shape, which contradicts the assumptions on the initial placement.

□

Consider a ribbon $R_i = (u_1, \ldots, u_n)$ with a parent ribbon $R_j = (v_1, \ldots, v_m)$, where the robots in $R_i$ touch robots $v_a$ to $v_{a+k}$ in ribbon $R_j$. The space between ribbons $R_i$ and $R_j$ is described by the polygon $P_{ij}$ bounded by the edges between robots $u_1$ to $u_n$, the edges between robots $v_a$ to $v_{a+k}$, and the edges between $u_1$ and $v_a$ and $u_n$ to $v_{a+k}$.

**Proposition 27.** Every point inside the polygon $P_{ij}$ has a robot at a distance closer than $2r$.

*Proof.* From the fact that ribbon $R_j$ is the parent of $R_i$, and the fact that ribbons are connected in a line we have the following invariants.

1. For $n > 1$ robot $v_n \in R_j$ neighbors robot $v_{n-1} \in R_j$.
2. For $n > 1$ robot $u_n \in R_i$ neighbors robot $u_{n-1} \in R_i$.
3. For $n > 1$ if robot $u_n \in R_i$ neighbors robot $v_m \in R_j$ then $u_{n-1}$ neighbors $v_m$ or $v_{m-1}$.

Let $u_d$ and $v_e$ be the highest numbered robots in $P_{ij}$ that belong to ribbon $R_i$ and $R_j$ respectively.

It is not possible for $u_d$ to have an edge to robot in $R_j$ other than $v_e$ and for $v_e$ to have an edge to a robot in $R_i$ other than $u_d$, as this would require edges to cross, which would require robots to be closer than $2r$ from neighbors. Therefore, we need to consider only three cases.

If robot $u_d$ has no edges to robots in $Rj$ other than $v_e$, but $v_e$ has an edge to another robot $u_{d-1}$ in $R_i$, then $u_d$ and all edges to $u_d$ can be removed from $P_{ij}$ while all remaining robots maintain the polygon properties. The space removed from $P_{ij}$, is a equilateral triangle with side lengths of $2r$ and vertices $v_e$,$u_d$,$u_{d-1}$, so there is no point inside this triangle which is at distance more than $2r$ from a robot.

If robot $v_e$ has no edges to robots in $Ri$ other than $u_d$, but $u_d$ has an edge to another robot $v_{e-1}$ in $R_j$, then $v_e$ and all edges to $v_e$ can be removed from $P_{ij}$ while all remaining robots maintain the polygon properties. The space removed from $P_{ij}$, is a equilateral triangle with side lengths of $2r$ and vertices $v_e$,$u_d$,$v_{e-1}$, so there is no point inside this triangle which is at distance more than $2r$ from a robot.

If robot $u_d$ has no edges to robots in $Rj$ other than $v_e$, and robot $v_e$ has no edges to robots in $Ri$ other than $u_d$, then $v_e$ and $u_d$ and all edges to $v_e$ and $u_d$ can be removed from the $P_{ij}$ while all remaining robots maintain the polygon properties. From invariant 1 and 2, $v_e$ must touch $v_{e-1}$ and $u_d$ must touch $u_{d-1}$, and from invariant 3 since $u_d$ touches $v_e$ in $R_j$ then $u_{d-1}$ touches $v_{e-1}$. The space removed from $P_{ij}$ is a parallelogram with sid lengths of $2r$ and vertices $v_e$,$v_{e-1}$,$u_d$,$u_{d-1}$, and no point inside this parallelogram is at distance more than $2r$ from a robot.

This shows that there is always at least on vertex that can be removed from $P_{ij}$, where the resulting polygon satisfies the same initial invariants, and any point in the part of the polygon that is removed has a robot closer than $2r$. Since this can be repeated until we are left with an empty polygon, the statement follows. □

**Proposition 28.** If there are no reachable unoccupied points well inside the desired shape, a robot without children is at distance $5r$ or less from the boundary of the desired shape.

*Proof.* Consider a robot $u_i$, and observe that at the time $u_i$ stopped if it was at distance greater than $5r$ from the desired shape boundary, then it would contribute something to the edge-following path of $\mathrm{suffix}_{u_i}$. If there are no reachable unoccupied points well inside the desired shape, then it means the contribution of $u_i$ was removed by a robot that stopped later.

In the rest of the proof we show by induction that the contribution of robot $u_i$ from $\mathrm{suffix}_{u_i}$ cannot be removed by any robot except its child. Consider a robot $u_j$ for $j > i$ that stopped after $u_i$, and suppose by inductive hypothesis that the contribution of $u_i$ is present in $\mathrm{preffix}_{u_j}$ or $\mathrm{suffix}_{u_j}$. Observe that item 3e, and in particular 3b and 3d, imply that when a robot $u_{j+1}$ stops, then either $u_i$ is present in $\mathrm{preffix}_{u_{j+1}}$ or $\mathrm{suffix}_{u_{j+1}}$, or robot $u_{j+1}$ must be a child of $u_i$, which concludes the proof. □

**Theorem 29.** *If there is no reachable unoccupied point well inside the desired shape, then there is no unreachable unoccupied point inside the desired shape that is at distance* $5r$ *or more from the boundary.*

*Proof.* Consider a tree formed by the parent-child relationship between ribbons. Let $P$ be the simple polygon formed by the union of all the polygons $P_{ij}$ for every ribbon $R_i$ and its parent $R_j$ in this tree of ribbons; by definition all stopped robots are contained in the polygon defined by $P$.

Observe that the vertices of $P$ correspond to the robots that do not have any children or are the first or last robot of a ribbon.

If there is no reachable unoccupied point well inside the desired shape, then proposition 28 implies that all the vertices of $P$ must be at distance $5r$ or less from the boundary of the desired shape.

Suppose by contradiction that there is an unoccupied unreachable point $x$ inside the desired shape at distance $5r$ or greater from the desired shape boundary. Clearly $x$ cannot be in $P$, since proposition 27 implies every point in $P$ is at distance $2r$ from a robot; therefore $x$ must be outside $P$. For $x$ to be unreachable it has to be surrounded by a cycle of robots, all of which are vertices of $P$. However, this contradicts that $x$ is at distance $5r$ or more from the boundary of the desired shape, since all vertices of $P$ are within distance $5r$ from the boundary of the desired shape. □

### 4.3.4 Termination

Together, the previous theorems imply that given sufficiently many robots, the shape formation algorithm guarantee that once no additional robots transition to a stopped state, there are no unoccupied spaces inside the desired shape. This is captured formally by the next theorem.

**Theorem 30.** *Given sufficiently many robots then, once no additional robots stops, there is no unoccupied point inside the desired shape that is at distance* $5r$ *or more from the boundary.*

*Proof.* First observe that given sufficiently many robots, then the fact that the desired shape is finite and robots only stop while inside the desired shape, implies there will always be idle or mobile robots. In that case theorem 24 implies that once no additional robots stop, there is no reachable unoccupied point well inside the desired shape. Finally theorem 29 implies there is no unreachable unoccupied points inside the desired shape that are at distance $5r$ or more from the boundary. □

**References**


1. G. M. Whitesides, B. Grzybowski, Self-assembly at all scales. *Science* **295**, 2418–2421 (2002). Medline doi:10.1126/science.1070821

2. H. C. Berg, The rotary motor of bacterial flagella. *Annu. Rev. Biochem.* **72**, 19–54 (2003). Medline doi:10.1146/annurev.biochem.72.121801.161737

3. P. Lawrence, *The Making of a Fly: The Genetics of Animal Design* (Blackwell, Oxford, 1992).

4. S. Camazine *et al*., *Self-Organization in Biological Systems* (Princeton Univ. Press, Princeton, NJ, 2003).

5. C. Anderson, G. Theraulaz, J. Deneubourg, Self-assemblages in insect societies. *Insectes Soc.* **49**, 99–110 (2002). doi:10.1007/s00040-002-8286-y

6. N. J. Mlot, C. A. Tovey, D. L. Hu, Fire ants self-assemble into waterproof rafts to survive floods. *Proc. Natl. Acad. Sci. U.S.A.* **108**, 7669–7673 (2011). Medline doi:10.1073/pnas.1016658108

7. M. Kirschner, J. Gerhart, *The Plausibility of Life: Resolving Darwin's Dilemma* (Yale Univ. Press, New Haven, CT, 2005).

8. R. Wood, R. Nagpal, G. Y. Wei, Flight of the robobees. *Sci. Am.* **308**, 60–65 (March 2013). Medline doi:10.1038/scientificamerican0313-60

9. R. Groß, M. Bonani, F. Mondada, M. Dorigo, Autonomous self-assembly in swarm-bots. *IEEE Trans. Robot.* **22**, 1115–1130 (2006). doi:10.1109/TRO.2006.882919

10. E. William, G, Mermoud, A. Martinoli, Comparing and modeling distributed control strategies for miniature self-assembling robots. In *Proceedings of the 2010 International Conference on Robotics and Automation* (Anchorage, AK, May 2010), pp. 1438–1445.

11. M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, G. Chirikjian, Modular self-reconfigurable robot systems. *Robot. Automat. Mag.* **14**, 43–52 (2007). doi:10.1109/MRA.2007.339623

12. S. Goldstein, J. Campbell, T. Mowry, Programmable matter. *Computer* **38**, 99–101 (2005). doi:10.1109/MC.2005.198

13. M. Tolley, H. Lipson, On-line assembly planning for stochastically reconfigurable systems. *Int. J. Robot. Res.* **30**, 1566–1584 (2011). doi:10.1177/0278364911398160

14. P. Wurman, R. D'Andrea, M. Mountz, Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Mag.* **29**, 9–19 (2008).

15. K. Konolige *et al*., Centibots: Very large scale distributed robotic teams. *Exp. Robot.* **IX**, 131–140 (2006).

16. J. McLurkin *et al*., Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before* (Stanford, CA, March 2006), pp. 72–75.

17. M. Rubenstein, C. Ahler, R. Nagpal, Kilobot: A low cost scalable robot system for collective behaviors. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation* (St. Paul, MN, May 2012), pp. 3293–3298.

18. P. Vartholomeos, E. Papadopoulos, Analysis, design and control of a planar micro-robot driven by two centripetal-force actuators. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation* (Orlando, FL, May 2006), pp. 649–654.

19. See supplementary materials on *Science* Online.

20. D. Moore, J. Leonard, D. Rus, S. Teller, Robust distributed network localization with noisy range measurements. In *The 2nd International Conference on Embedded Networked Sensor Systems* (Baltimore, November 2004), pp. 50–61.

21. J. Werfel, Y. Bar-Yam, D. Rus, R. Nagpal, Distributed construction by mobile robots with enhanced building blocks. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation* (Orlando, FL, May 2006), pp. 2787–2794.

22. D. Arbuckle, A. Requicha, Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: Algorithms and simulations. *Auton. Robots* **28**, 197–211 (2010). doi:10.1007/s10514-009-9162-7

23. K. Støy, Using cellular automata and gradients to control self-reconfiguration. *Robot. Auton. Syst.* **54**, 135–141 (2006). doi:10.1016/j.robot.2005.09.017

24. K. Tomita, S. Murata, H. Kurokawa, E. Yoshida, S. Kokaji, Self-assembly and self-repair method for a distributed mechanical system. *IEEE Trans. Robot. Autom.* **15**, 1035–1045 (1999). doi:10.1109/70.817668

25. M. De Rosa, S. Goldstein, P. Lee, J. Campbell, P. Pillai, Scalable shape sculpting via hole motion: Motion planning in lattice-constrained modular robots. *Proceedings of the 2006 IEEE International Conference on Robotics and Automation* (Orlando, FL, May 2006), pp. 1462–1468.

26. R. Nagpal, H. Shrobe, J. Bachrach, Organizing a global coordinate system from local information on an ad hoc sensor network. In *Information Processing in Sensor Networks*, F. Zhao, L. Guibas, Eds. (Springer, Berlin, 2003), pp. 333–348.

27. P. W. Rothemund, Folding DNA to create nanoscale shapes and patterns. *Nature* **440**, 297–302 (2006). Medline doi:10.1038/nature04586

28. T. He, C. Huang, B. Blum, J. Stankovic, T. Abdelzaher, Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking* (San Diego, CA, September 2003), pp. 81–95.

29. M. Yim, Y. Zhang, J. Lamping, E. Mao, Distributed control for 3D metamorphosis. *Auton. Robots* **10**, 41–56 (2001). doi:10.1023/A:1026544419097