Chapter 1
SWARMORPH: Morphogenesis with Self-Assembling Robots

Rehan O'Grady, Anders Lyhne Christensen, Marco Dorigo

1.1 Introduction

Distributed robotic systems are often considered particularly appropriate for inhospitable and rapidly changing environments. Future teams of distributed robots operating in space, in search and rescue conditions, or even inside the human body, will need to work together to overcome the physical limitations of individual robots. For these systems to display efficient physical cooperation, it is a prerequisite that they are able to form larger composite robotic entities with morphologies appropriate to the tasks and environmental conditions encountered.

In this chapter, we describe work done towards arriving at a distributed robotic system whose constituent robots can connect to one another when necessary to form morphologies of sizes and shapes appropriate for solving the tasks they encounter. Our work is based around the concept of a *self-assembling* robotic system. Multi-robot self-assembling systems are made up of independent autonomous mobile agents. These agents are capable of forming physical connections with each other without external direction. Robots in such systems can operate individually to carry out simple tasks in parallel. However, they can also work together to achieve more complex goals through cooperation. Physical cooperation through self-assembly allows the system to overcome the fundamental physical limitations of individual robots such as power, size or reach. Figure 1.1 shows how a multi-robot system can use self-assembly to form appropriate composite robotic entities to solve two different tasks: pushing a heavy object and crossing a gap. In Fig. 1.1(left), 5 robots have assembled into a line in order to cross a gap, while in Fig. 1.1(right), 8 robots have assembled into a shovel shape in order to push a heavy bucket.

To date, self-assembling systems have displayed little active control over the morphology of the composite robotic entity formed through the self-assembly process.

Rehan O'Grady (rogrady@ulb.ac.be) and Marco Dorigo (mdorigo@ulb.ac.be) are with IRIDIA-CoDE, Université Libre de Bruxelles, Brussels, Belgium. Anders Lyhne Christensen (anders.christensen@iscte.pt) is with Instituto de Telecomunicações and Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal.

Fig. 1.1 Two examples of robotic entities self-assembled into morphologies appropriate for the task. Left: A connected robotic entity crosses a gap. A line formation is well-suited to this task, since it allows the entity to stretch further and minimizes the number of robots suspended over the gap. Right: Robots physically connected in a shovel shape to push a heavy object.

In the majority of existing multi-robot self-assembling systems, the geometry of the self-assembled entity is either predefined by the physical characteristics of the connection mechanism or is stochastic. In Sect. 1.2, we discuss related work and the state of the art in more detail.

In this chapter, we detail progress towards giving robots the capacity to assemble into appropriate morphologies and to operate as a single entity when physically connected to one another. Our approach relies on directed morphology growth: a new morphology is started by a single robot, the seed. The seed can invite physical connections from other robots. As new robots connect to the seed, they can continue to extend the morphology in particular directions by inviting other robots to physically connect to them at specified locations. In this way, a morphology with a specific global structure is grown. In order to coordinate when physically connected, the robots use visual point-to-point notifications. The control we use is completely distributed — all the robots remain individually autonomous even when physically connected to other robots. There is no central coordination mechanism or global sharing of information.

The structure of the remainder of this chapter is as follows: In Sect. 1.3, we present the robotic hardware that we use — the swarm-bot platform. In Sect. 1.4, we go on to describe the low-level control logic that allows us to specify the connection angle between two robots, thus enabling the higher level morphology generation mechanisms. In Sect. 1.5, we describe a simple distributed morphology generation mechanism that is capable of generating a wide range of simple repeating morphologies without the use of any robotic IDs, blueprints or symbolic communication. In Sect. 1.6, we show how the addition of symbolic communication can increase the range of possible morphologies, and present a dedicated scripting language in which we embed our morphology generation logic. In Sect. 1.7, we show how our system can use morphology generation to solve real-world tasks by forming dedicated morphologies when the robots encounter particular tasks. Finally, in Sect. 1.8, we

discuss ongoing work towards creating a genuinely adaptive system, which could adaptively form new morphologies appropriate to previously unseen tasks.

1.2 Related Work

There is a large body of scientific literature on the distributed creation and control of robotic morphologies using inter-connectible components. Some research directly investigates morphology control techniques. However, this type of research has often been conducted on simulated robotic platforms with no physical counterpart. By contrast, considerations of morphological flexibility have resulted in the development of two types of real-world robotic system: self-reconfigurable systems and self-assembling systems. However, in both of these types of real-world system, the existing research has focused almost exclusively on the hardware, rather than on the autonomous control required to generate specific morphologies. We describe morphology control research in Sect. 1.2.1, and related hardware research in Sect. 1.2.2.

The research presented in this chapter represents an effort to bridge the gap between the abstract morphology control research that is hard to apply to today's real world robotic systems that have the potential for morphological flexibility, but lack the behavioural control required to generate morphologies autonomously.

1.2.1 Morphology Control Algorithms

The SWARMORPH approach that we present in this chapter is unique because it can autonomously create arbitrary morphologies, while still operating within the restrictions imposed by a physically embodied robotic system. Indeed, we demonstrate all of our algorithms on real robots.

Much of the research in morphology control investigates high-level abstract properties of morphology control mechanisms, without considering the physical constraints inherent in embodied robotic systems. Klavins *et al.* [24], for example, tackled the problem of defining a class of graph grammars that can be used to model and direct distributed robotic self-assembly. They show how a grammar can be synthesized that generates a desired, pre-specified target morphology.

Other studies use simulations of idealised robots that could not be directly realised in the real world. Jones and Matarić's *Transition Rule Set* compiler [21], for example, takes as input the desired morphology and outputs a set of rules. The approach proved scalable and capable of producing a large class of structures in a simulated 2D lattice world populated by simulated unit square agents. Butler *et al.* [7] have studied generic decentralised algorithms for lattice-based self-reconfigurable robots. The algorithms are inspired by cellular automata and control is based on geometric rules. Støy and Nagpal [39, 38] demonstrated algorithms for self-reconfiguration and directed growth of simulated cubic units based on gradi-

ents and cellular automata. Shen *et al.* [37] have demonstrated a bio-inspired control method based on virtual hormone for controlling swarms of robots. The virtual hormone can be propagated through the swarm causing the robots to generate patterns and/or to reconfigure. The authors show results from experiments in simulation and discuss how virtual hormone could be propagated in real-world scenarios either through radio or through infrared communication.

Some simulation based studies have tried to take into account physical constraints. Bojinov *et al.* [3] have shown how a simulated modular robot (Proteo) can self-reconfigure into useful and emergent morphologies when the individual modules use local sensing and local control rules. Mytilinaios *et al.* [28] used the Molecube platform to investigate the application of evolutionary algorithms to design self-replicating morphologies in a 2-D simulation environment. Rus and Vona [33, 34] proposed a centralised control algorithm to allow a robotic system composed of *Crystalline Atom* units to reconfigure its shape. Although a physical prototype of a *Crystalline Atom* unit has been built, the results with the proposed control algorithm were obtained analytically and in simulation.

Direct real-world morphology control research is scarce. Examples include Zykov *et al.*'s use of a physical instantiation of the Molecube system [52] to demonstrate the self-replication of a 4-module entity provided with an ordered supply of additional modules. The system executed a predetermined sequence of actions, successful connections being confirmed through explicit communication. Although autonomous, the demonstration consisted of a specific sequence of actions — the authors were not trying to generate a generic morphology control mechanism. White *et al.* [40] used a scripting language to specify structures for an externally propelled self-assembling system. However, unlike SWARMORPH, their system relied on unique IDs and predefined locations for individual components. Research by Yu and Nagpal [50, 49] is a rare example of a modular system that exhibits a morphological response to different environmental contingencies, and whose control has been tested on a real-world platform. However, their work is based on a self-reconfigurable platform made of pre-assembled modules, and therefore does not consider the problem of morphology control through self-assembly.

1.2.2 Real-world self-reconfigurable and self-assembling systems

Self-reconfigurable systems are made up of interlocking modules that can adjust their configuration by changing the angle of connection between modules, or by changing the global topology of connections [45]. Individual modules in such systems tend to be incapable of independent motion. Notable examples include CEBOT [16, 23], PolyBot [42, 43, 44, 46, 47, 51], CONRO [9, 8, 31], Millibot Train system [6], M-TRAN [27, 48, 22], SuperBot [36, 35] and Molecubes [28].

Self-assembling systems are made up of independent robotic components that can autonomously form physical connections with one another [18]. The individual robots can be either externally propelled or self-propelled. Externally pro-

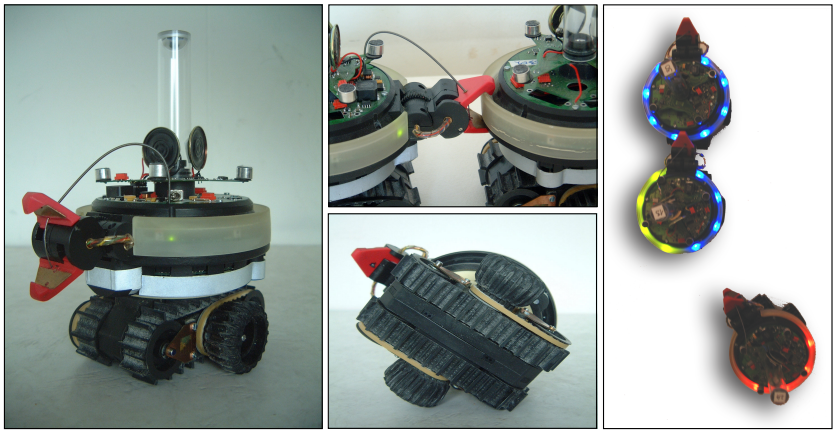


Fig. 1.2 Left: The s-bot. Centre Top: The s-bot connection mechanism. Centre bottom: The s-bot traction system. Right: Two s-bots connected and a third s-bot approaching to connect to the morphology.

pelled systems rely on energy from the environment in order to move. Such systems include Penrose and Penrose's wooden mechanical analogue for self-replication [32], Hosokawa *et al.*'s self-assembling hexagons [20], Breivik *et al.*'s template-replicating polymers [4], White *et al.*'s 2D/3D systems with passively moving modules [41, 40], Bishop *et al.*'s system of air-table suspended triangular modules [2].

Self-propelled self-assembling robotic systems are made up of independent autonomous mobile components that are capable of forming physical connections with each other without external direction. Several architectures have been proposed, which have been implemented with varying degrees of success [5, 14, 16, 17, 19]. However, none of the existing systems display any meaningful control over the morphology of the connected entity formed through the self-assembly process.

1.3 The Swarm-Bot Robotic Platform

For our experiments, we use the swarm-bot robotic platform [26]. This platform is made up of multiple mobile autonomous robots called s-bots (see Fig. 1.2) that can form physical connections with each other. The entity formed by two or more connected s-bots is called a swarm-bot. The s-bot is 12 cm high without its camera turret, and has a diameter of 12 cm without its connection mechanism. Thanks to its traction system that combines tracks and wheels, the s-bot has good mobility on uneven terrain whilst still retaining the ability to rotate on the spot efficiently. The main s-bot body houses most of its sensory and processing systems and can rotate with respect to the chassis by means of a motorised axis.

Physical connections between s-bots are established by a gripper-based connection mechanism. Each s-bot is surrounded by a transparent ring that can be grasped by other s-bots. An optical light barrier inside the s-bot gripper indicates when another s-bot's ring (or another object) is between the jaws of the gripper. S-bots advertise their location by means of eight sets of RGB coloured LEDs (Light Emitting Diodes) distributed around the inside of their transparent ring. These LEDs can also provide indications of the s-bot's internal state to other nearby s-bots.

The s-bot has an omni-directional camera that, depending on light conditions, can detect other s-bots' LEDs up to about 50 cm away or an external light source up to about 400 cm away depending on ambient light conditions. The s-bot has 15 infrared proximity sensors distributed around its body that allow for the detection of obstacles. Ground facing proximity sensors under the tracks allow the s-bot to detect whether or not it is over a hole.

1.4 Low-Level Robotic Control — Directional Self-Assembly

In this section, we present a low-level robotic control algorithm we developed to control the orientation of inter-robot connections formed during autonomous self-assembly. We term this mechanism *directional self-assembly*. We go on to present experiments conducted to analyse the precision and timing characteristics of the directional self-assembly mechanism. All of the higher level morphology creation logic that we present in subsequent sections relies on directional self-assembly.

Directional self-assembly between two s-bots is initiated by a stationary s-bot signalling that it is ready to be grasped at a particular point on its body. The signal used is a particular configuration of illuminated LEDs that we term a *connection slot*. Any other s-bot in the arena may see the signal and attempt to grasp the signalling s-bot.

The left-hand side and right-hand side of a connection slot are indicated by four illuminated green LEDs and four illuminated blue LEDs, respectively (see Fig. 1.3). Each s-bot has 8 LED locations. A green LED, a blue LED and a red LED are located at each LED location. A connection slot can be opened between any two neighbouring LED locations, except between the two front LED locations, where the gripper is mounted. Thus, an s-bot that is connected to a morphology can extend the local structure using one of seven different connection slots. An s-bot can only open one connection slot at a time as a connection slot requires the use of all eight LED locations.

We refer to an s-bot that is displaying an open connection slot as an *extending s-bot*. We refer to an s-bot that is not yet part of the connected structure as a *free s-bot* when it is searching for a connection slot and as an *attaching s-bot* when it is attempting to grip an *extending s-bot*. We refer to an s-bot that has formed a successful connection and thus become part of the morphology as a *connected s-bot*.

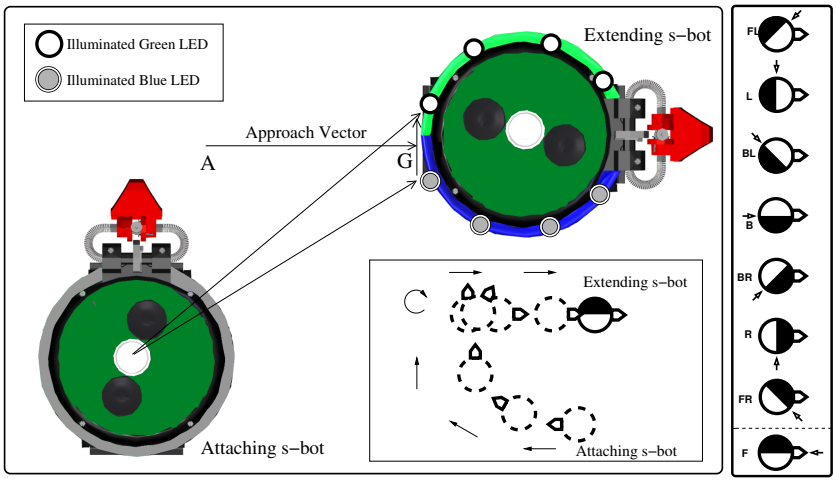


Fig. 1.3 Left: Based on an extending s-bot's illuminated connection slot LEDs, an attaching s-bot calculates the approach point (A), grip point (G) and approach vector $\overrightarrow{AG}$. Inset: The motion of an attaching s-bot as it navigates to and grips a connection slot. Right: The 8 possible connection slots that an s-bot can open. Only seven connection slots can be used for extending the morphology. The eighth connection slot (F) indicates a grip point that is already occupied by the gripper of the s-bot displaying the connection slot. This connection slot is used for signalling instead (see Sect. 1.5).

Free s-bots approach connection slots with a particular trajectory to ensure a consistent angle of connection. To achieve this, free s-bots use the illuminated LEDs of an extending s-bot's open connection slot to calculate an *approach vector* (see Fig. 1.3). The head of the approach vector is called the *grip point* and indicates the point on the extending s-bot's body which the attaching s-bot should grip. The tail of the approach vector is called the *approach point*, and is 13 cm away from the body of the extending s-bot.

Based on the approach vector calculation, we use a five phase strategy to let free s-bots find and grip a connection slot. Phase 1: the free s-bot directly approaches the extending s-bot until it is within 35 cm of the grip point. Phase 2: the s-bot circles around the extending s-bot until it is within +/- 20° of the approach vector. Phase 3: the s-bot navigates to the *approach point* at the start of the approach vector. Phase 4: the s-bot rotates to face the connection slot. Phase 5: the s-bot approaches the connection slot and attempts to grip the extending s-bot. The trajectory of a free s-bot approaching and gripping a connection slot is shown in Fig. 1.3 (inset).

1.4.1 Experimental Results: Precision

We conducted experiments to measure the precision of the directional self-assembly mechanism. In each of our experimental trials, a stationary extending s-bot dis-

played an open connection slot, to which a single free s-bot attached. We conducted 96 trials, varying the starting position and orientation of the free s-bot, while keeping that of the extending s-bot fixed. The extending s-bot also always opened the same connection slot — connection slot B directly behind it (see Fig. 1.3). We used 12 possible starting positions and eight possible starting orientations for the free s-bot. The starting positions for the free s-bot were evenly distributed around a circle of radius 35 cm centred on the extending s-bot.

In all 96 trials, the free s-bot successfully connected to the extending s-bot. In 2 of the 96 trials, the grip failed on the first attempt, and the free s-bot retreated to try another angle. The angular precision of the free s-bot's connection in these trials is shown in Fig. 1.4 (left). Note that the mean angular misalignment is very close to zero. A misalignment of zero indicates the ideal angle of connection which is normal to the curve of the extending s-bots's body. The positional precision of the free s-bot's connection is shown in Fig. 1.4 (right). The positional precision of a connection is measured by the lateral displacement between the grip point specified by the connection slot and the point at which the attaching s-bot grips. The visible bias towards the right of the centre line is due to a hardware asymmetry in the distribution of the s-bot LEDs.

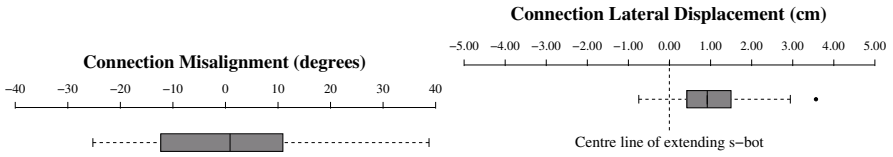


Fig. 1.4 Precision of the connection slot mechanism. Left: Angular precision, measured by the mismatch between the alignment of the attaching s-bot and the alignment of the extending s-bot. Right: Positional precision, measured by the lateral displacement of attaching s-bot's grip from extending s-bot's centre line. (Reprinted from [29].)

1.5 Periodically Repeating Morphologies

In this section, we present our first approach to self-organised morphological growth. We use a fully decentralised system in which robots that join a growing morphology use local morphology extension rules to determine how to extend the morphology appropriately. The rules we use are based only on local communication of internal state. This communication is restricted to the detection of connection slots (or absence thereof) being displayed by nearby robots within range of the camera. Thus the amount of information communicated is highly limited. In addition, the rules are purely reactive, i.e., they do not rely on any form of stored historical state information. As a result, the class of morphologies we can form using this technique is limited to periodically repeating morphologies.[1]Nonetheless, we show that

we can form a wide range of morphologies using a limited set of morphology extension rules. By using decision making based on local sensing, we can also increase the period of repetition to generate more complex morphologies.

We first present the morphology extension rules that we developed, and show how they can be combined to make different morphologies. We present a series of experiments with real robots in which we grow four example morphologies. Finally, we present some simulation-based experiments to test the scalability of the approach.

1.5.1 Methodology

Morphologies are started by a single robot, the seed. When an s-bot attaches to the seed or to another already attached robot, it can itself invite connections from other non-attached robots. The robots choose a particular connection slot to open based on a simple set of rules that depend only on locally sensed data. Each morphology is defined by a single set of rules that is shared by all of the robots except for the seed robot that has its own set of rules.

During morphology growth, all free robots search for connection slots. When a robot connects to a connection slot, the newly attached robot uses its LEDs to signal its successful attachment to the robot displaying the connection slot. Both robots are then free to open new connection slots based on the local morphology extension rules specific to the morphology being formed.

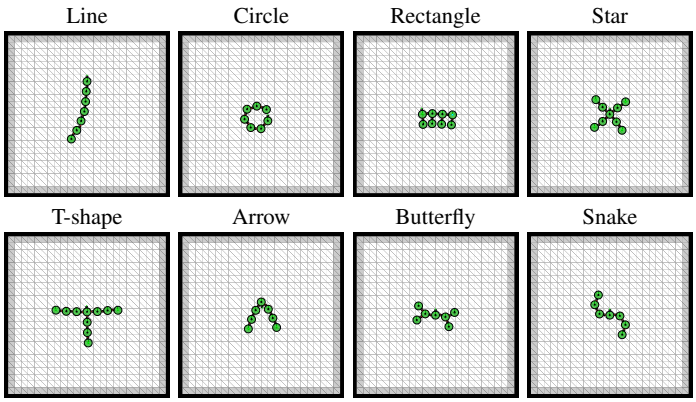


Fig. 1.5 Examples of different morphologies that can be made using SWARMORPH. These morphologies have been generated in simulation.

[1] By contrast, the approach we present in Sec. 1.6 incorporates the possibility of the exchange of symbolic information between connected s-bots. The resulting system can generate arbitrary morphologies.

Different morphology extension rule sequences result in different morphologies. By creating an appropriate rule set, and then by defining different sequences of rules, we can generate a large number of different morphologies. Figure 1.5 shows some of the morphologies we can generate using SWARMORPH.

1.5.2 Morphology Extension Rules

In this section, we present the morphology extension rules that we have developed. We show how the rules can be combined to create different classes of periodically repeating morphologies.

Linear Morphologies — Rule: `Extend`

The simplest morphologies are those in which every newly attached robot opens the same predefined connection slot. Examples of such morphologies are the line and circle morphologies in Fig. 1.5. Such morphologies are based on a single rule, that controls the opening of a particular connection slot: the `Extend` rule. This rule uses directional self-assembly to extend the structure by opening a given connection slot. The rule takes as a parameter the connection slot that should be opened.

The morphology extension rules for line morphology growth and the corresponding morphology growth pattern are shown in Fig. 1.6.

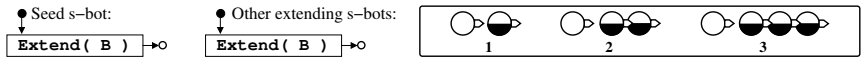


Fig. 1.6 Left and middle: *Line Morphology* extension rules. Right: Line morphology growth. (1): An attaching s-bot approaches the seed's open connection slot. (2): The newly connected robot opens connection slot B, and another attaching s-bot approaches. (3): The newly connected s-bot again opens connection slot B and the growth pattern repeats itself.

Branching Morphologies — Rules: `Send HS Sig, Wait HS Sig`

More complex morphologies require a single s-bot to extend the local structure in more than one direction. To do this, the extending s-bot must know when a connection slot has been filled, so that it can then open the next connection slot. The s-bot hardware does not include any dedicated sensors to detect when it has been gripped by another s-bot. To compensate, we designed a signalling mechanism — the *handshake*. The handshake signal is composed of two morphology extension rules, one executed by the sender of the handshake, and one executed by the receiver of the handshake (see Fig. 1.7). The `Send HS Sig` rule allows an attaching s-bot to communicate to an extending s-bot that it has successfully attached to the extending s-bot's open connection slot. The signal takes the form of opening connection slot F. The `Wait HS Sig` rule tells the extending s-bot to wait until it detects the

handshake signal before executing subsequent morphology-specific extension rules. The handshake is shown in Fig. 1.7.

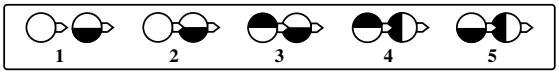


Fig. 1.7 Handshake rules: `Send HS Sig, Wait HS Sig`. (1): The extending s-bot is executing the `Wait HS Sig` rule. (2): The attaching s-bot successfully grips the extending s-bot. (3): The newly connected s-bot executes the `Send HS Sig` rule (4): The extending s-bot recognises the signal and executes its next rule, to open connection slot L. (5): The `Send HS Sig` rule times out and the connected s-bot executes its next rule, to open connection slot B.

The `Send HS Sig` and `Wait HS Sig` rules provide the flexibility to create many more morphologies, such as the star, arrow and T-shape morphologies in Fig. 1.5. The rule sequences required for a simple arrow morphology and the corresponding morphology growth pattern are shown in Fig. 1.8.

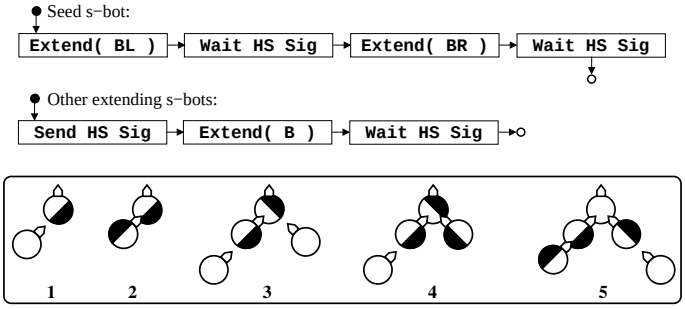


Fig. 1.8 Top: *Arrow Morphology* extension rules. Bottom: Corresponding morphology growth pattern

Symmetrically Growing Morphologies — Rule: `Balance`

When we use the `Send HS Sig` and `Wait HS Sig` rules to generate morphologies such as the star, arrow or T-shape, we run the risk that morphology growth will be asymmetrical. In the arrow morphology, for example, it is stochastically possible that one arm of the arrow will continue to develop while the other arm does not develop at all. Such imbalances are particularly problematic when the number of s-bots is limited, as is the case in our real robot experimentation.

To solve this problem, we introduce the `Balance` rule. When executing this rule, a connected robot waits with its LEDs unilluminated until it cannot see any connection slots around it.[2] In practice, this means that the s-bot waits until it can-

[2] An s-bot cannot distinguish between LEDs that belong to the morphology to which it is currently connected and LEDs on robots in other nearby morphologies. It could therefore happen that a

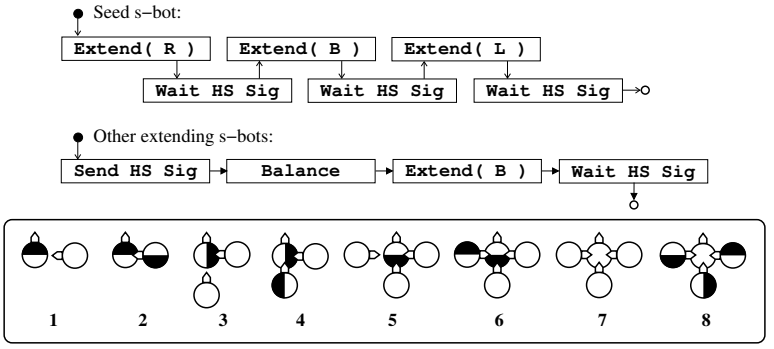


Fig. 1.9 *Balanced T-shape Morphology*. Top: Morphology extension rules. Bottom: Morphology growth. The seed s-bot opens connection slots R, B, L in turn. The connected s-bots each execute the `Balance` rule, and therefore wait until they can see no green or blue LEDs before executing subsequent extension rules. The result is that once the seed s-bot is no longer displaying an open connection slot, all three connected s-bots open connection slot B at the same time. (Reprinted from [29].)

not see any green or blue LEDs. As soon as this is the case, the connected s-bot continues executing subsequent morphology extension rules.

The `Balance` rule allows parts of the morphology that are in visual range of each other to grow at the same rate. In the arrow morphology, for example, as long as the s-bots at the ends of the two arms of the arrow can still see each other, the two arms will differ in length by at most one s-bot. An example use of the `Balance` rule can be seen in Fig. 1.9, which shows the growth of a balanced T-shape morphology.

Morphology extension rules for the seed robots of the *Balanced Arrow Morphology* and for the *Balanced Star Morphology*, respectively, are given in Fig. 1.10. Growth of both of these morphologies is similar to that of the *Balanced T-shape Morphology* shown in Fig. 1.9 above and all other extending s-bots follow the same rules as in the *Balanced T-shape Morphology*.

Experiments using all three balanced morphologies are presented in Sect. 1.5.3.

Morphologies with higher repetition period — Rule: `Decide`

More complex morphologies can be achieved if we allow the robots to make conditional decisions about how to extend the local structure based on local sensing. The `Decide` rule allows an attaching s-bot to modify its behaviour based on the post handshake actions of the extending s-bot to which it is attaching. In Fig. 1.11, the extending s-bot in scenario A (left) is executing a different rule sequence from the extending s-bot in scenario B (right). In both scenarios, the attaching s-bot is

robot executing the `Balance` rule would wait for a connection slot opened by a nearby robot connected to a different morphology. In this section, we do not address this problem—we consider the formation of only one morphology at a time. However, this issue disappears with the symbolic communication based approach we present in Sect. 1.6

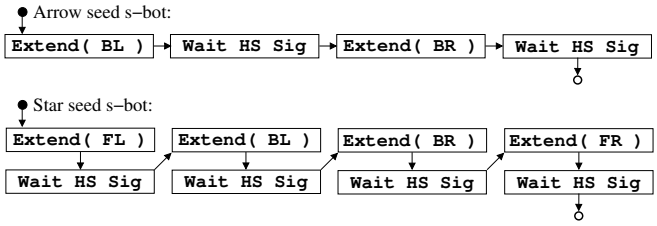


Fig. 1.10 Morphology extension rules for the seed robots of the *Balanced Arrow Morphology* (top) and for the *Balanced Star Morphology* (bottom).

executing the same rule sequence. The attaching s-bot uses the `Decide` rule to determine the appropriate post-handshake behaviour based on the post-handshake actions of the extending s-bot.

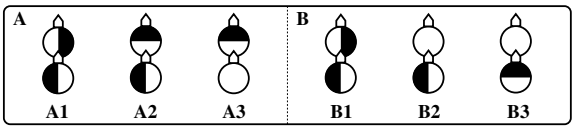


Fig. 1.11 `Decide` rule. (A1,B1): Attaching s-bot successfully grips extending s-bot. (A2): Extending s-bot recognises handshake signal and executes subsequent rule which is to open connection slot R. (B2): Extending s-bot recognises handshake signal and has no subsequent rules. (A3): Attaching s-bot executes decision rule — sees green ahead (i.e., sees an open connection slot ahead) and therefore 'decides' not to execute any subsequent rules. (B3): Attaching s-bot executes decision rule — sees no green ahead (i.e., does not see an open connection slot ahead) and therefore 'decides' to open connection slot R.

The *Rectangle Morphology* is one of the morphologies that can be formed using the `Decide` rule. Figure 1.12 shows the morphology extension rules and the morphology growth for this morphology.

1.5.3 Results

We grew four morphologies (line, balanced arrow, balanced star, rectangle) 10 times with 7 real s-bots. Completed examples of the four morphologies are shown in Fig. 1.13. Photographs and videos of experiments described in this chapter can be found in [1].

We conducted our experiments in a walled arena of 220 cm x 220 cm. In each experiment, we grew a single morphology. The free s-bots started each experiment at one of 12 points (randomly sampled without replacement) evenly distributed around a circle of radius 50 cm centred on the seed s-bot. The free s-bots were placed in

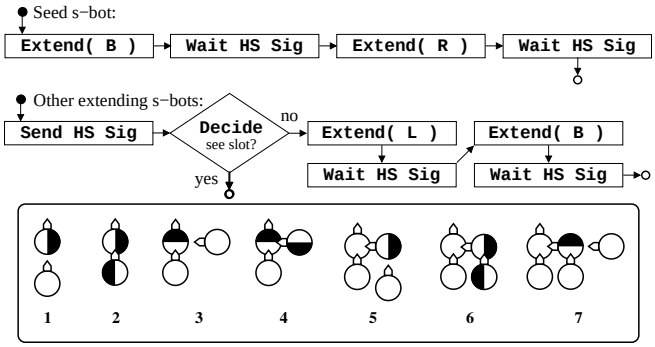
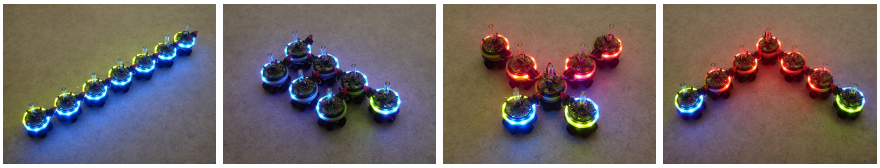


Fig. 1.12 The *Rectangle Morphology*. Above: Morphology extension rules. Below: Morphology growth. (1): Seed opens connection slot B (2): Connected s-bot sends handshake signal. (3): Seed sees handshake signal, then opens connection slot R. First connected s-bot executes `Decide` rule — sees connection slot and therefore executes no subsequent extension rules. (4): Another s-bot connects and handshakes. (5): Seed sees handshake signal but does not open another connection slot. Connected s-bot executes `Decide` rule — does not see a connection slot, and therefore opens connection slot L itself. (6): Another s-bot connects and handshakes. (7): Morphology growth pattern repeats. (Reprinted from [29].)



Line Rectangle Star Arrow

Fig. 1.13 Four different morphologies constructed with 7 real robots: line, rectangle, star and arrow. Note that the rectangle and star morphologies would become symmetric with the addition of more robots. (Reprinted from [11].)

one of four possible starting orientations (randomly sampled with replacement). An experiment was considered finished once all 6 free s-bots had successfully attached or after 15 minutes had expired.

All 6 free robots successfully connected to the morphology in 38 out of 40 experiments. In a single rectangle morphology experiment, one robot failed to connect, and in another of the rectangle morphology experiments two robots failed to connect. Figure 1.14 shows morphology growth over time. Each line in the figure represents the mean growth times of a single morphology over ten experiments. All four morphologies grow at a similar rate until they reach a size of three s-bots. Subsequent growth rates for the star, arrow and line morphologies remain relatively close, and the three morphologies have comparable mean completion times of 335 s, 347 s and 366 s, respectively. The rectangle morphology had a slower growth rate, with a mean completion time of 634 s.

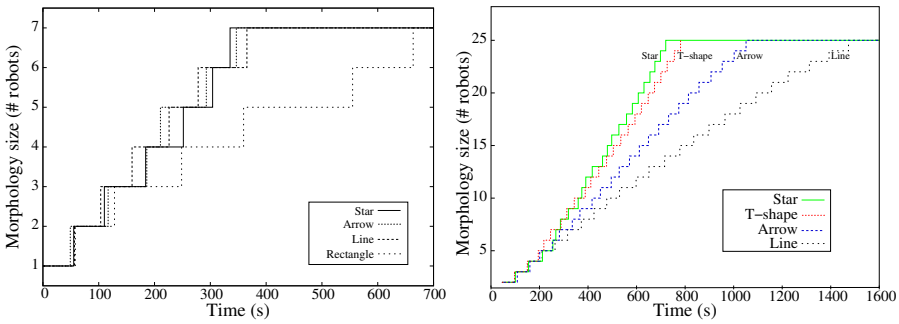


Fig. 1.14 Mean morphology growth times of morphologies. Each horizontal line segments represents a time interval during which the morphology size remains constant. Vertical line segments correspond to moments at which a free s-bot connects to a morphology. Left: Real robot experiments — each morphology grown 10 times with seven s-bots. Right (reprinted from [29]): Experiments in simulation. Each morphology grown 100 times with 25 s-bots. We fix the number of free s-bots in the system at 10, by feeding a new s-bot into the simulation after every connection.

To investigate the scalability properties of our morphology growth scheme, we conducted experiments with larger numbers of s-bots in simulation. We first verified the verisimilitude of our simulation environment [10] by growing the same four morphologies in simulation (see Tab. 1.1). For the star, arrow and line morphologies, simulated morphology growth time is comparable to real-world growth time. The rectangle morphology takes 31% less time in simulation. We believe that this discrepancy is due to the fact that when the robots are densely packed, the camera is more accurate in simulation than in reality, as we do not simulate occlusions and reflections. We chose, therefore, not to use the rectangle morphology or any other densely packed morphologies in our simulation based scalability testing.

Table 1.1 Mean completion times for different morphologies on real s-bots and in simulation.

	Real s-bots	Simulated s-bots	Difference
Star	335.32 s	325.31 s	−3.0%
Arrow	346.56 s	331.37 s	−4.2%
Line	365.53 s	339.76 s	−7.0%
Rectangle	663.46 s	456.71 s	−31.2%

For our simulation tests, we chose the morphologies line, arrow, T-shape and star, because they respectively can have one, two, three and four connection slots open simultaneously (see Fig. 1.5). We kept the number of free robots constant at 10. Each time a free robot connected to the morphology a new free robot was added to the simulation. For each morphology, we conducted 100 trials. A trial was considered finished when the morphology reached a size of 25 connected robots. Table 1.1 shows the mean connection times for the different morphologies. The

results show that all of the morphologies scale linearly. Furthermore, we can see that morphologies with larger numbers of simultaneously open connection slots grow faster.

1.5.4 Summary

In this section, we demonstrated growth of specific morphologies on a real-world self-assembling robotic platform. We achieved a high success rate in building four different morphologies using seven real robots and we showed that the approach scales well. For more details see [11, 29].

The approach presented relies on morphology extension rules. Although this approach has the benefit of being fully distributed, it has a number of limitations with respect to our goal of giving the robots the capacity to adaptively self-assemble into task-specific morphologies:

- The seed robot is predetermined and is given a seed-specific set of rules.
- Only one pre-determined morphology can be built in each experiment.
- Only periodically repeating morphologies are possible.
- There is no way to regulate the size of the morphology, i.e., to stop morphology growth — growth continues until there are no more free robots.

In the next section, we show how the introduction of symbolic communication between connected s-bots can help to overcome the above limitations and allows swarmorph to generate arbitrary morphologies.[3]

1.6 Symbolic Communication and a Morphology Generation Language – SWARMORPH-script

In this section, we present a more sophisticated morphology control paradigm capable of forming arbitrary morphologies. We abstract morphology extension rules into instructions that can be combined into short morphology generation programs or scripts. The resulting scripting language — SWARMORPH-script — also includes instructions that allow physically connected s-bots to communicate symbolically. Using the SWARMORPH-script paradigm, we show how arbitrary morphologies can be formed in a distributed fashion, how morphology size can be regulated, and how multiple separate morphologies can be assembled.

[3] Instead of introducing symbolic communication to generate arbitrary morphologies (as we do in the next section), an alternative avenue (that we have not pursued) might be to create more morphology extension rules of the type presented in this section, and thus generate richer morphologies whilst avoiding symbolic communication. Such a research avenue could also encompass formal analysis of the morphologically expressive power of a given set of extension rules, perhaps using a grammar based approach akin to that pursued by Klavins et al. [24].

1.6.1 Methodology

Using SWARMORPH-script, morphology growth still occurs through local morphology extensions in particular directions. However, when a robot connects to a morphology, we now allow it to communicate strings of symbols with the robot to which it is connected. The strings can be interpreted as indices of instructions describing how to extend the structure locally.

Arbitrary morphologies can be created by communicating information about the local state of the morphology (usually the communication of a single digit suffices) to each new robot that connects to the structure. The newly connected robot can extend the local structure appropriately based on this information, then pass on updated state information to the next robot to attach. A similar mechanism allows for morphology size regulation — a counter can be incremented and passed on by each attaching robot. Once the counter has reached a certain threshold, the next robot can stop extending the morphology.

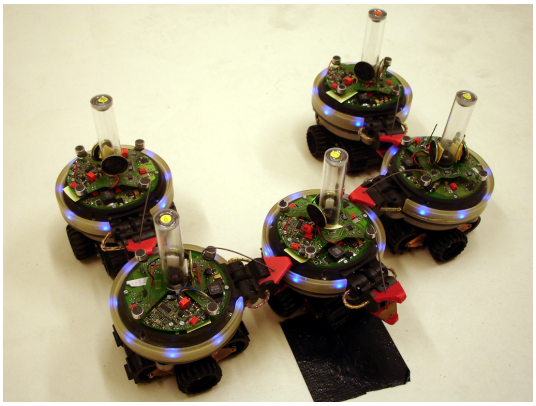
S-bots have no dedicated hardware that allows connected s-bots to communicate with one another. To compensate, we implemented a simple visual communication protocol based on camera and LEDs. Each time a bit is transmitted, the sending robot changes the illumination of its LEDs. The colour green represents a '0' bit, blue represents a '1' and red represents a repeat bit. The receiver acknowledges receipt of each bit by lighting up its LEDs to match the colour of the sender's LEDs. Once the receipt of a bit has been acknowledged, the sender transmits the next bit. In experimental tests, this visual communication mechanism proved slow but reliable. We conducted 10 trials of an experiment that involved transmitting 100 bits between two connected robots. In all 10 trials, the string was successfully communicated and on average the transmission took 121 seconds resulting in an average bandwidth of 0.8 bits/second.

1.6.2 Example — The Horseshoe Morphology

We demonstrate the SWARMORPH-script approach with the horseshoe morphology (see Fig. 1.15). This simple morphology would be impossible using our previous system of local extension rules (presented in Sect. 1.5), as morphology growth is limited. The size of the morphology is determined in advance by script-based logic. The script used to generate the horseshoe morphology is shown in Fig. 1.15.

The control is homogeneous — all of the robots start by executing the first instruction `RandomWalkUntil`. They perform a random walk until they either detect a dark patch of ground or an open connection slot. The first robot that detects a patch of dark ground becomes the seed robot (as dictated by the conditional statement in line 2) and opens the first connection slot (connection slot *left*). Note that the `InviteConnection` instruction causes an s-bot to open a particular connection slot, and then wait until it has been successfully gripped before continuing to execute subsequent instructions. Other robots that see this or subsequent open

connection slots try and attach to a connection slot (as dictated by the conditional statement in line 9). Once attached to a connection slot, a connecting robot receives an instruction sequence identifier from the robot to which it connected. Any of the free robots can assume any position in the morphology. The first robot to connect to the seed receives instruction sequence id 1, which tells it to execute lines $12-14$ in the script (this is the robot to the right of the seed robot in Fig. 1.15). Following instructions in lines $12-14$, this robot then opens a connection slot to its right. The robot that connects to this new open connection slot then receives instruction sequence id 0 and does not open any further connection slots, thus terminating morphology growth for this branch of the morphology. A similar process results in a mirror branch of the morphology being created on the other side of the seed robot, but with the first robot connecting to the seed on that side opening connection slot *left* instead of connection slot *right*.



```
1  RandomWalkUntil( dark-ground-detected
2                   or connection-slot-detected )
3  if dark-ground-detected then
4      InviteConnection( left )
5      SendInstrSeqId( 1 )
6      InviteConnection( right )
7      SendInstrSeqId( 2 )
8      StopExecution()
9  end
10 if connection-slot-detected then
11     FindSlotThenConnect()
12     ReceiveInstrSeqId()
13     if received-seq-id = 0 then
14         StopExecution()
15     end
16     if received-seq-id = 1 then
17         InviteConnection( right )
18         SendInstrSeqId( 0 )
19         StopExecution()
20     end
21     if received-seq-id = 2 then
22         InviteConnection( left )
23         SendInstrSeqId( 0 )
24         StopExecution()
25     end
26 end
```

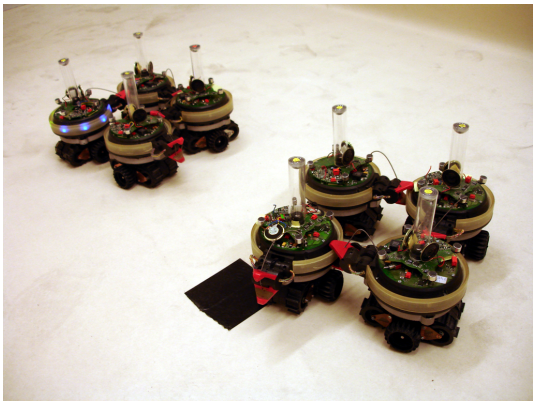
Fig. 1.15 Horseshoe morphology script and result. (Photo reprinted from [12].)

1.6.3 Multiple Morphologies

Using SWARMORPH-script multiple morphologies can be generated by instructing a connected robot to split off from an existing morphology and to start a new mor-

phology. In practice, this occurs by passing an instruction sequence id to a newly connected s-bot that causes it to disconnect and start a new morphology.

We demonstrate the generation of multiple morphologies by generating multiple small squares (Fig. 1.16). The experimental setup used is the same as in Sect. 1.6.2. In the mini-squares morphology, a mini-square of four s-bots is initially constructed. The fourth and final robot in the morphology opens a temporary connection slot. When the fifth robot connects to the morphology, it receives instruction sequence id '4' that causes it to disconnect, retreat for 5 s and then start a new mini-square morphology. The formation process continues until no more robots are available.



```
1  RandomWalkUntil( dark-ground-detected
2                   or connection-slot-detected )
3  if dark-ground-detected then
4      InviteConnection( left )
5      SendInstrSeqId( 1 )
6      StopExecution()
7  end
8  if connection-slot-detected then
9      FindSlotThenConnect()
10     ReceiveInstrSeqId()
11     if received-seq-id = 1 then
12         InviteConnection( right )
13         SendInstrSeqId( 2 )
14         StopExecution()
15     end
16     if received-seq-id = 2 then
17         InviteConnection( right )
18         SendInstrSeqId( 3 )
19         StopExecution()
20     end
21     if received-seq-id = 3 then
22         InviteConnection( back )
23         SendInstrSeqId( 4 )
24         StopExecution()
25     end
26     if received-seq-id = 4 then
27         Disconnect()
28         Retreat( 5 s )
29         InviteConnection( left )
30         SendInstrSeqId( 1 )
31         StopExecution()
32     end
33 end
```

Fig. 1.16 Mini-square morphologies script and result. (Photo reprinted from [12].)

We chose to generate several copies of a relatively simple morphology in order to demonstrate the principle. However, more complex SWARMORPH-script programs could generate a series of different morphologies of arbitrary complexity. Furthermore, the script we presented generated one complete morphology before starting to generate the next morphology. However, morphologies could also be generated in parallel — it would suffice to pass the instruction sequence id for new morphology creation during, rather than at the end, of morphology construction.

Square Shovel Multiple Lines Multiple Triangles

Fig. 1.17 Four morphologies generated using SWARMORPH-script. (Photos reprinted from [12].)

1.6.4 Summary

In this section, we have demonstrated the formation of arbitrary morphologies by self-assembling robots. We abstracted basic control primitives as commands in a simple language, and used this high-level descriptive language to express logic for growing specific morphologies. For more details see [12].

The abstraction of the morphology extension rules into SWARMORPH-script allowed us to overcome most of the limitations of the initial approach detailed in Sect. 1.5. However, two essential features necessary for the system to be capable of solving tasks by self-assembling have still not been demonstrated: the ability to respond to different obstacles and the ability to carry out tasks when assembled into a larger robotic entity. We address those features in the next section.

1.7 Using SWARMORPH to solve tasks

In this section, we show how the SWARMORPH approach can be applied to specific tasks. The tasks we consider are navigation-based tasks. Each task consists of an obstacle that the s-bots must overcome while performing phototaxis. We present a SWARMORPH-script program that allows a group of s-bots to navigate from one end of the environment to the other, forming appropriate morphologies to overcome any obstacles they encounter en route.

The environment we use can contain up to two obstacles placed in any order. Each obstacle is insurmountable by a single robot — to succeed, the robots must form physical connections to each other and navigate as a larger connected entity. However, the obstacles have been chosen so that each morphology requires a dedicated morphology that is incapable of overcoming the other obstacle.

The robots respond to the presence of an obstacle by initiating a self-assembly process and forming the connected morphology appropriate to the obstacle encountered. When an appropriate morphology has been formed, the robots cross the obstacle collectively, then disassemble and continue individually.

1.7.1 Experimental Setup

We conduct our experiments in the arena shown in Fig. 1.18. A strong light source is located at the far end of the arena. The s-bots start each experiment disconnected from each other and their task is to navigate from the start zone to the light source. The arena contains two obstacles: a gap and a bridge that consists of two pipes. The order of the obstacles in the arena is variable. In all experiments, the robots do not know a priori which obstacle they will encounter first. Neither the gap nor the bridge can be overcome by a single robot operating alone.

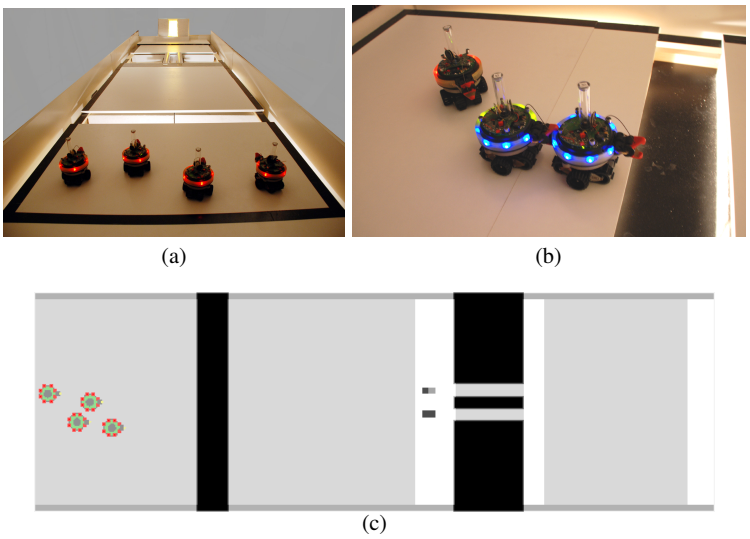


Fig. 1.18 (a) The real arena. (b) A line morphology in the process of being formed after the gap has been encountered. (c) The arena in simulation.

1.7.1.1 The Gap Obstacle

To overcome this obstacle, the robots must cross a 22 cm wide rectangular hole that runs the width of the arena. An s-bot can detect the gap based on readings from its infrared ground sensors. A gap of 22 cm was chosen because it is reliably passable by four real s-bots connected in a line morphology (a three s-bot line morphology will fail unless it is perfectly aligned, and any smaller morphology always fails). The gap and a four s-bot linear swarm-bot crossing the gap can be seen in Fig. 1.19 (left).

Although the grippers on the s-bots are powerful enough for one s-bot to support two other s-bots suspended over the gap, it puts significant strain on the hardware. When we perform replications of our experiment, we therefore replace the gap by

an equal-sized stripe of black arena surface. For the s-bots, the black surface is indistinguishable from the gap. A video of the s-bots crossing a 22 cm real gap can be seen in the supplementary material [1].

1.7.1.2 The Bridge Obstacle

To overcome this obstacle, the robots must use a bridge to cross a 50 cm wide rectangular hole that runs the width of the arena. The bridge is made of two pipes spaced 17.5 cm apart, each with a diameter of 8 cm. The curvature of the pipes is such that a moving s-bot cannot balance on a single pipe. The two pipes are also sufficiently far apart that the wheels of a single s-bot cannot make contact with both pipes at the same time. Thus, a single s-bot cannot traverse a bridge alone. However, a so called 2-s-bot *support* morphology is capable of successfully traversing the bridge. The support morphology consists of two appropriately oriented, physically connected s-bots. In this configuration, the s-bots can traverse a bridge, since the connected morphology makes contact with both pipes at the same time (each constituent s-bot touches one of the pipes). The curvature of the pipes does not cause the morphology to topple, as the two s-bots mutually support each other, see Fig. 1.19 (right).

The on-board computer vision software does not enable the robots to estimate the width of a gap or to see the bridge. We have therefore placed a special reflective material before the bridged 50 cm gap to distinguish it from the 22 cm gap. The reflective material can be detected by an s-bot using its infrared ground sensors: readings are higher than for the normal arena floor. In order to determine the position of the bridge, we have put a distinct simple barcode in front of each pipe, see Fig. 1.19 (right). The barcode is made up of different materials that can be detected by an s-bot's ground sensors. Whenever a robot detects a barcode, it can use the barcode information to determine which pipe it is facing (left pipe or right pipe) and build the morphology to cross the bridge accordingly. We have also added reflective material on the far side of the bridge to allow the robots to detect when they have successfully crossed the bridge.

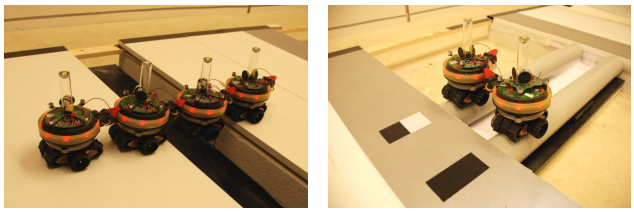
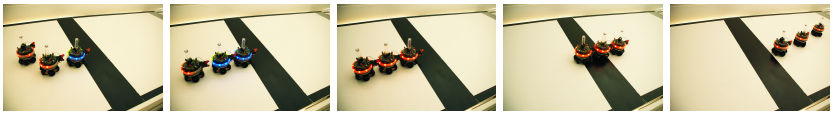


Fig. 1.19 The two obstacles and the appropriate morphology for each tasks. Left: The gap crossing task (line morphology). Right: The bridge traversal task (support morphology). (Photos reprinted from [13].)

1.7.2 Example Script to Respond to Hole with Three S-bot Line Morphology

The gap crossing script (Fig. 1.21 left) is an example of a complete SWARMORPH-script that solves a task using morphology control. When executed on three or more s-bots, the s-bots perform phototaxis until a gap is detected. When a gap is detected, a three s-bot swarm-bot will self-assemble into a linear morphology and attempt to cross the gap. Once the gap has been crossed, the swarm-bot disassembles and the s-bots continue individually.



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

Fig. 1.20 An example of a line of three s-bots forming a line to cross a gap. See text.

An s-bot executing the gap crossing script (Fig. 1.21 left) will first perform phototaxis until either a gap is detected or until another s-bot displaying a connection slot is seen. If an s-bot detects a gap using its infrared ground sensors, it becomes the seed of a new morphology: it first opens a new connection slot to its rear in order to invite a free s-bot to physically connect (Fig. 1.20a). When an s-bot has connected, the seed has to wait for a notification from the newly connected s-bot that indicates that the morphology is complete before starting to cross the gap. When an s-bot connects to the seed, it receives instruction sequence id 0. This tells the newly connected s-bot that it is the middle s-bot in the linear morphology and that it has to receive a connection from another s-bot in order to complete the morphology. The middle s-bot therefore opens a connection slot to its rear (Fig. 1.20b). When a free s-bot connects to the middle s-bot, the middle s-bot sends it id 1 to tell the newly connected s-bot that it is the last s-bot in the morphology. The middle s-bot then notifies the seed that the morphology is complete and starts to move across the gap (Fig. 1.20c-d). Upon receiving the notification from the middle s-bot, the seed begins to move across the gap. The s-bot that connected to the morphology last starts to move across the gap immediately after the middle s-bot has sent it instruction sequence id 1.

The s-bot that connected to the morphology last is also the last s-bot to cross the gap. When it has crossed the gap (i.e., when its infrared ground sensors detect that it is no longer over the gap), it sends a notification to the middle s-bot and disconnects from the morphology by opening its gripper. When the middle s-bot receives the notification, it forwards the notification to the seed and disconnects. When the seed receives the notification, it knows that the swarm-bot has made it across the gap and that the two other s-bots have disconnected (Fig. 1.20e). After having crossed the

gap, all the s-bots restart their script and continue individually until a new gap is encountered.

Gap crossing script:

```
 1 Label: 'start'
 2 IndependentPhototaxisUntil( gap detected or
 3                             connection-slot-detected )
 4 if gap detected then
 5     // Instructions for seed s-bot:
 6     InviteConnection( back )
 7     SendInstrSeqId( 0 )
 8     PauseUntilSignal( signal-from-back )
 9     ConnectedPhototaxisUntil(
10                         signal-from-back )
11     Jump( 'start' )
12 end
13 if connection-slot-detected then
14     FindSlotThenConnect()
15     ReceiveInstrSeqId()
16     if received-seq-id = 0 then
17         // Instructions for middle s-bot:
18         InviteConnection( back )
19         SendInstrSeqId( 1 )
20         SendSignal( front )
21         ConnectedPhototaxisUntil(
22                         signal-from-back )
23         SendSignal( front )
24         Disconnect()
25         Jump( 'start' )
26     end
27     if received-seq-id = 1 then
28         // Instructions for last s-bot:
29         ConnectedPhototaxisUntil(
30                                 gap-crossed )
31         SendSignal( front )
32         Disconnect()
33         Jump( 'start' )
34     end
35 end
```

Multitask script:

```
 1 Label: 'PhototaxisAndLookForTasks'
 2 IndependentPhototaxisUntil( gap detected or
 3     bridge-detected or connection-slot-detected )
 4 if gap detected then
 5     // Instructions for line morphology seed s-bot:
 6     InviteConnection( back )
 7     SendInstrSeqId( 0 )
 8     PauseUntilSignal( signal-from-back )
 9     ConnectedPhototaxisUntil(
10                         signal-from-back )
11     Jump( 'PhototaxisAndLookForTasks' )
12 end
13 else if bridge-detected then
14     // Instructions for support morphology
15     // seed s-bot:
16     InviteConnection( left )
17     SendInstrSeqId( 3 )
18     ConnectedPhototaxisUntil( bridge-crossed )
19     Jump( 'PhototaxisAndLookForTasks' )
20 end
21 else if connection-slot-detected then
22     FindSlotThenConnect()
23     ReceiveInstrSeqId()
24     if received-seq-id = 0..2 then
25         // Code similar to gap crossing
26         // script (left) for s-bots
27         // connecting to a line morphology.
28     end
29     if received-seq-id = 3 then
30         // Instructions for second s-bot in
31         // bridge morphology:
32         ConnectedPhototaxisUntil(
33                                 bridge-crossed )
34         Disconnect()
35         Jump( 'PhototaxisAndLookForTasks' )
36     end
37 end
```

Fig. 1.21 Left: A gap crossing script - when a gap is encountered, a three s-bot line morphology is created, the resulting swarm-bot moves across the gap, the s-bots disassemble, and the script is restarted. Right: High-level structure of the SWARMORPH-script used in our experiments in which four s-bots face multiple tasks.

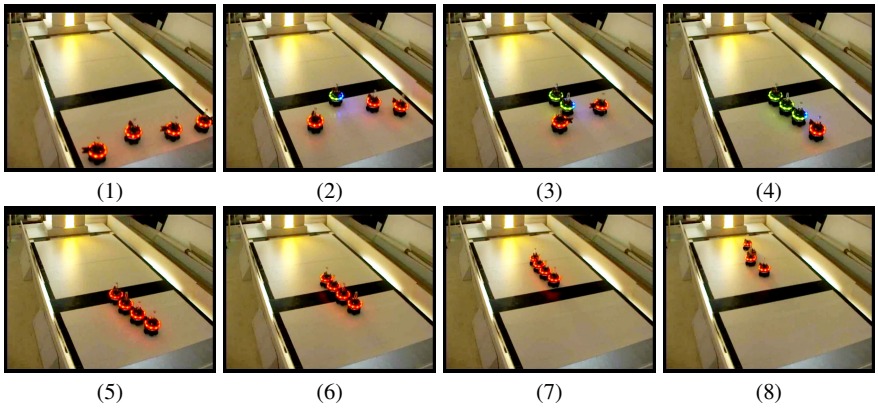


Fig. 1.22 Video frames from an experiment in which four robots form a line morphology and cross a gap.

1.7.3 Overview of Script to Solve Both Tasks

We developed a SWARMORPH-script that enables s-bots to self-assemble into different morphologies depending on which obstacles they encounter. We used this script in both our simulation-based experiments and in our real robot experiments. The high-level structure of this so called *multitask* script is shown in Fig. 1.21 (right). All the s-bots initially perform phototaxis until either an obstacle is encounter or until one of the s-bots opens a connection slot (because it has detected the presence of an obstacle). At this point, logic is executed that depends on the type of obstacle (gap or bridge). The part of the multitask script (Fig. 1.21 right) related to the self-assembly of the line morphology when the gap is detected is similar to the gap crossing script (Fig. 1.21 left)—but in this case a four s-bot line morphology is formed instead of a three s-bot line morphology.

The logic for assembling the support morphology is simpler than the logic for the line morphology. Since there are only two s-bots in the support morphology (whereas the line morphology consists of four s-bots) there is no need to use notifications: both robots know immediately when the morphology has been assembled. Similarly, it is not necessary for the second s-bot in the support morphology to notify the seed that it has crossed the bridge: both robots cross the bridge at the same time and the second s-bot in the support morphology can simply let go as soon as it detects that it has made it safely across the bridge.

1.7.4 Simulation-Based Experiments

We conducted experiments in a physically embodied simulation environment in which the robots were required to assemble, overcome one of the obstacles, disassemble, reassemble to overcome the second obstacle and disassemble again. The simulation environment can be seen in Fig. 1.18 (c). The robots do not know a priori the order of the obstacles they will encounter. We ran 100 experiments in which the robots encountered the gap obstacle before the bridge obstacle, and 100 experiments in which the robots encountered the bridge obstacle before the gap obstacle.

In the bridge-then-gap experiments, the success rate was 96% (success being defined as all robots successfully overcoming both obstacles). The average completion time for the bridge-then-gap experiments was 722 seconds (st.dev. 147 seconds). In the gap-then-bridge experiments, the success rate was 97%. The average completion time was 762 seconds (st.dev. 182 seconds). The difference between the completion times for the bridge-then-gap and the gap-then-bridge experiments was not found to be statistically significant ($p = 0.08$ using a two-tailed Mann-Whitney test).

1.7.5 Real World Experiments

We conducted experiments separately for each of the two potential obstacles. The s-bots were not aware a priori which obstacle they would encounter. We conducted a set of 5 experiments with 4 s-bots encountering the gap obstacle.[4]Figure 1.22 shows a series of frames from a successful experiment. In all five experiments, the system successfully detected the gap and formed the appropriate morphology. In four out of five experiments, the notification mechanism successfully allowed the four s-bot line morphology to start collective phototaxis in a coordinated manner, and to disassemble once the final s-bot had crossed the gap. In a single experiment, a failed notification signal between the third and fourth s-bots in the line morphology resulted in the first two s-bots in the morphology commencing collective phototaxis while the rear two s-bots were still engaged in (unsuccessful) signalling—the experiment was manually aborted.

We conducted 5 trials in which two s-bots encountered the bridge. In all 5 trials, the s-bots successfully detected the environmental cues and formed the correct morphology with which to cross the bridge. In three out of 5 experiments, the s-bots successfully crossed the bridge. In 2 out of 5 experiments, inaccuracies in the detection of the target light source caused the s-bots to veer sideways off the bridge, and they had to be manually placed back on the bridge. In every experiment, the s-bots successfully detected that they had crossed the bridge and disassembled.

Frames from an experiment with 4 s-bots crossing the bridge 2-by-2 with real s-bots are shown in Fig. 1.23. The frames in the figure are taken from a proof-of-concept video in which we ran four s-bots over both obstacles in sequence. The figure shows two sets of two s-bots crossing the bridge, and parallel task execution as the first pair of s-bots over the bridge encounter the gap obstacle, and start forming

the line morphology, while the second set of s-bots forms the support morphology to cross the bridge. These videos can be found in the supplementary material [1].

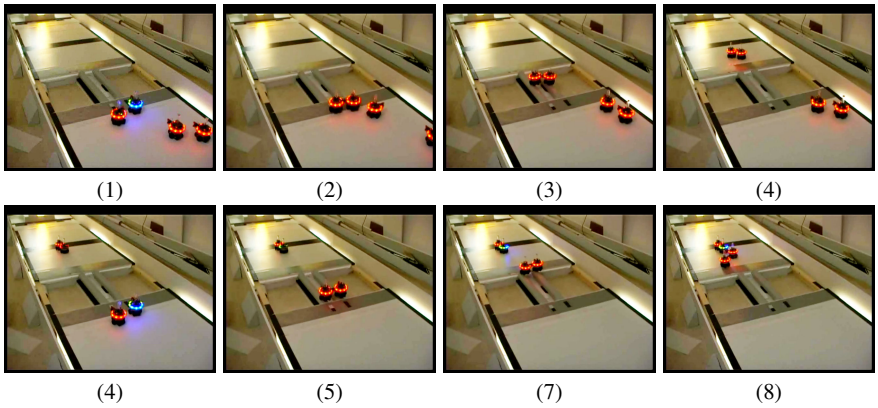


Fig. 1.23 Video frames from an experiment in which four s-bots cross the bridge obstacle by forming two support morphologies.

1.7.6 Summary

In this section, we have demonstrated how, by forming dedicated morphologies, a group of self-assembling robot can overcome obstacles insurmountable for a single robot. We demonstrated 1) how dedicated morphologies are grown from a single seed robot when an obstacle or an environmental clue is encountered, 2) how the local visual notifications are used to coordinate navigation over an obstacle, and 3) how the robots disassemble once the obstacle has been overcome and continue individually until the next obstacle is encountered. In our experiments, the robots were all homogeneous and autonomous.

In our research, the response to the different types of obstacles – namely the formation of dedicated morphologies – was preprogrammed. Whenever a gap is encountered, for instance, a linear morphology of four robots is always self-assembled – regardless of the width of the gap. In order for the robots to be truly adaptive, they should be able to respond to previously unseen obstacles by forming new and appropriate morphologies during task-execution. However, the current sensory equipment does not provide our robots with sufficient information to allow them to, for instance, estimate the width of a gap or to see the location and structure of a bridge.

[4] As noted previously, to prevent damage to the robots we used a black surface instead of a gap. This surface presents sensory information to the s-bots' ground sensors that is almost identical to the sensory information presented by the real gap obstacle.

In our ongoing work, we are therefore investigating the use of heterogeneous robotic swarms in which flying robots with a privileged view of the environment can guide wheeled robots on the ground to form appropriate morphologies when new obstacles are encountered. We discuss this work in more detail in the next section.

1.8 Towards Genuine Adaptivity

In this section, we discuss ongoing and future work that will lead to the SWARMORPH system becoming genuinely adaptive. We first present a control logic transmission mechanism which allows s-bots to transmit whole or partial scripts to each other. This frees SWARMORPH-script from the constraints of having all morphology generation logic pre-coded into a script. Using control logic transmission, robots can formulate an appropriate morphology on the fly, in response to never seen before conditions, and communicate the relevant instructions to create the morphology to other robots. We also discuss future work in the context of the Swarmanoid project [15]. This ongoing project studies the interaction of heteregeneous robotic swarms. In particular, aerial robots might be used in the future to direct the morphology generation activities of ground-based robots.

1.8.1 Control Logic Transmission

Control logic transmission allows robots with no a priori knowledge of a particular morphology to nonetheless take part in morphology growth. The morphologies presented in Sect. 1.6 relied on the communication of a single number between connected robots — an instruction ID that mapped to segments of preprogrammed control logic on the recipient robot. This approach is efficient in terms of restricting communication to a minimum, but has the disadvantage that the set of morphologies that can be generated is limited by the set of rules given to the robots at the start of an experiment.

To overcome these restrictions, we introduced dedicated primitives in the SWARMORPH-script language to allow for the communication of partial or whole morphology generation control programs. To send a script, each SWARMORPH-script command is compiled to a 5-bit string literal before being sent. The robot that receives a script maps these 5-bit string literals back to SWARMORPH-script commands, and then executes the script thus received. The `SendScript` command takes a list of SWARMORPH-script commands as a parameter. Alternatively, a special parameter, `'self'`, can be provided to the `SendScript` command which tells the sending robot to transmit a copy of the whole script it is currently executing.

We demonstrate the communication of morphology control logic with the *transmitted-triangle* morphology. In our experimental setup, we program a single

'master' robot with morphology control logic to create a morphology and to communicate the same control to other robots. All other robots are 'slave' robots that start without any a priori morphology creation logic, except the basic logic required to attach to a connection slot and execute any morphology control logic received through communication. The *morphology slave* script that we execute on these robots is shown in Fig. 1.24.

```
1 FindSlotThenConnect();
2 ReceiveScript();
3 ExecuteReceivedScript();
```

Fig. 1.24 Morphology slave script. This script is executed by robots that do not have any preprogrammed morphology expansion rules.

In the *transmitted-triangle* morphology (Fig. 1.25), the first action of the seed robot is to execute the `Disconnect` and `Retreat` instructions before opening a connection slot. This is done to maintain homogeneity of control, as these are the first instructions that a 'slave' robot will need to execute when it receives instructions to replicate the morphology. The seed robot then opens a connection slot to its rear left and sends a `StopExecution` command to the connecting robot. Thus, the first connected robot takes no further actions. The seed then opens a second connection slot to its rear right and sends a `StopExecution` command to the robot that connects to the slot. When three robots are assembled in this way, a single instance of the triangle morphology has been created. The seed then opens a temporary connection slot to its front left and sends a copy of its own script (`SendScript(self)`) to the robot that connects. This connecting robot is now executing the same script as the seed robot for the first morphology. It therefore disconnects, retreats and starts forming another triangle morphology. This process will continue to generate more triangle morphologies as long as there are still unconnected slave robots available.

1.8.2 Morphology Control in a Heterogeneous Swarm

The sensing capabilities of the s-bot are limited. In the experiments presented in Sect. 1.7, we were forced to place explicit cues in the arena to allow the robots to distinguish between the two types of obstacle. One solution to this problem might be to use a heterogeneous swarm, where supervisory robots with more sophisticated sensors equipment direct the morphogenesis activities of self-assembling robots.

The Swarmanoid project [15] is currently investigating this type of heteregeneous cooperation. In particular, some studies have already addressed the possibility of communication and cooperation between aerial and ground-based robots [25, 30]. Ongoing research effort is being applied to the problem of adapting these heteroge-

```
1  Disconnect();
2  Retreat(10s);
3  InviteConnection(back-left);
4  SendScript(
5    StopExecution();
6  );
7  InviteConnection(back-right);
8  SendScript(
9    StopExecution();
10 );
11 InviteConnection(front-left);
12 SendScript(self);
13 StopExecution();
```

Fig. 1.25 *Transmitted-triangle* morphologies. Script and result. (Photo reprinted from [12].)

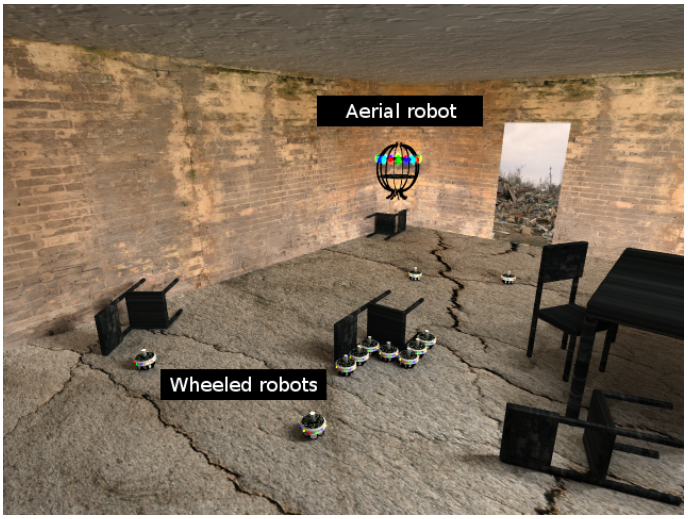


Fig. 1.26 Heterogeneous robot in a disaster zone. A heterogeneous robot swarm clears debris in an afflicted office to facilitate subsequent access for search and rescue workers. An aerial robot locates an object that needs to be moved (the toppled chair) and instructs ground-based robots to self-assemble into an appropriate morphology capable of moving the object.

neous communication and cooperation modalities to enable supervised morphogenesis. Figure 1.26 shows an example of how such supervised morphogenesis might enable a heterogeneous group of robots to react adaptively to its environment in a future search and rescue environment.

1.9 Conclusion

In this chapter, we have demonstrated SWARMORPH — a system that enables the construction of arbitrary two dimensional morphologies with self-propelled autonomous robots. Our approach proved reliable and effective on a real-world robotic platform, and was shown to scale well in a simulated environment. We also showed how our morphology control mechanism could be used to respond appropriately to different tasks. We have already outlined in Sect. 1.8 how SWARMORPH could start to become more of a genuinely adaptive morphology control system.

Although the work presented in this chapter was conducted exclusively on the swarm-bot platform, we do not foresee great difficulty in tranferring SWARMORPH to other robotic platforms. The key ideas underlying SWARMORPH are:

1. Directed local extensions to a morphology enable the creation of a specific global structure.
2. Abstraction of extension rules into script-based instructions.
3. Communication between connected robots to enable the propagation of morphology creation information.

Since SWARMORPH control logic relies on an abstraction of these ideas, it would be straightforward to implement SWARMORPH on any self-assembling system where the robots have an ability to dictate where another robot should attach and where connected robots can exchange symbolic information. In other words, once directional self-assembly has been implemented on a system, SWARMORPH can immediately be applied by using directional self-assembly as a module. By similar logic, SWARMORPH should be capable of creating three dimensional structures. It would suffice to define a directional self-assembly mechanism for a three dimensional self-assembling system (and a corresponding notation for defining connection slots in three dimensions). SWARMORPH-script programs could then be written to build three dimensional robotic entities.

The results presented in this chapter were not always perfect. Actuation and sensing failure of individual components resulted in occasional malformed morphologies and some task execution failures. Self-assembling systems are generally considered to have the potential for robustness in the face of individual agent failure. An important area for future development of the SWARMORPH system would be to explicitly add mechanisms to ensure such robustness. Possible approaches could include verification of shape correctness before a task is performed, time-out and reset mechanisms for malformed or partially formed shapes, and more sophisticated morphology designs that include a degree of physical and/or topological redundancy.

In the longer term, we see SWARMORPH as a layer on which other types of research could be conducted. We hope that ongoing morphology control research will treat SWARMORPH as a tool and focus more generically on how distributed robotic systems can adaptively detect, respond to and communicate information about their tasks and environments in order to generate collective robotic entities with appropriate morphologies.

1.10 Acknowledgments

This work was partially supported by the European Commission via the ERC Advance Grant "E-SWARM: Engineering Swarm Intelligence Systems" (grant 246939). The information provided is the sole responsibility of the authors and does not reflect the European Commission's or the European Union's opinion. The European Commission or the European Union are not responsible for any use that might be made of data appearing in this publication. Marco Dorigo acknowledges support from the Belgian F.R.S.-FNRS, of which he is a research director.

References

1. Swarmorph: Morphogenesis with self-assembling robots — supplementary online material. `http://iridia.ulb.ac.be/supp/IridiaSupp2010-002` (2010)
2. Bishop, J., Burden, S., Klavins, E., Kreisberg, R., W, M., Napp, N., Nguyen, T.: Programmable parts: A demonstration of the grammatical approach to self-organization. In: Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2644–2651. IEEE Computer Society Press, Los Alamitos, CA (2005)
3. Bojinov, H., Casal, A., Hogg, T.: Emergent structures in modular self-reconfigurable robots. In: Proceedings of the 2000 IEEE International Conference on Robotics and Automation, pp. 1734–1741. IEEE Computer Society Press, Los Alamitos, CA (2000)
4. Breivik, J.: Self-organization of template-replicating polymers and the spontaneous rise of genetic information. Entropy **3**, 273–279 (2001)
5. Brown, H.B., Weghe, J.M.V., Bererton, C.A., Khasla, P.K.: Millibot trains for enhanced mobility. IEEE/ASME Transactions on Mechatronics **7**(4), 452–461 (2002)
6. Brown, H.B., Weghe, M.V., Bererton, C., Khosla, P.: Millibot trains for enhanced mobility. IEEE/ASME Transactions on Mechatronics **7**(4), 452–461 (2002)
7. Butler, Z., Kotay, K., Rus, D., Tomita, K.: Generic decentralized control for lattice-based self-reconfigurable robots. International Journal of Robotics Research **23**(9), 919–937 (2004)
8. Castano, A., Behar, A., Will, P.M.: The Conro modules for reconfigurable robots. IEEE/ASME Transactions on Mechatronics **7**(4), 403–409 (2002)
9. Castano, A., Shen, W.M., Will, P.: CONRO: Towards deployable robots with inter-robots metamorphic capabilities. Autonomous Robots **8**(3), 309–324 (2000)
10. Christensen, A.L.: Efficient neuro-evolution of hole-avoidance and phototaxis for a swarm-bot. Mémoire de DEA, technical report TR/IRIDIA/2005-14, Université Libre de Bruxelles, Bruxelles, Belgium (2005)
11. Christensen, A.L., O'Grady, R., Dorigo, M.: Morphology control in multirobot system. IEEE Robotics and Automation Magazine **14**(4), 18–25 (2007)

12. Christensen, A.L., O'Grady, R., Dorigo, M.: SWARMORPH-script: A language for arbitrary morphology generation in self-assembling robots. Swarm Intelligence **2**(2-4), 143–165 (2008)
13. Christensen, A.L., O'Grady, R., Dorigo, M.: Parallel task execution, morphology control and scalability in a swarm of self-assembling robots. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, Robótica 2009, pp. 127–133. IPCB-Instituto Politcnico de Castelo Branco, Castelo Branco, Portugal (2009)
14. Damoto, R., Kawakami, A., Hirose, S.: Study of super-mechano colony: concept and basic experimental set-up. Advanced Robotics **15**(4), 391–408 (2001)
15. Dorigo, M.: The swarmanoid project. `http://www.swarmanoid.org` (2009)
16. Fukuda, T., Buss, M., Hosokai, H., Kawauchi, Y.: Cell structured robotic system CEBOT: control, planning and communication methods. Robotics and Autonomous Systems **7**(2-3), 239–248 (1991)
17. Groß, R., Bonani, M., Mondada, F., Dorigo, M.: Autonomous self-assembly in swarm-bots. IEEE Transactions on Robotics **22**(6), 1115–1130 (2006)
18. Groß, R., Dorigo, M.: Self-assembly at the macroscopic scale. Proceedings of the IEEE **96**(9), 1490–1508 (2008)
19. Hirose, S., Shirasu, T., Fukushima, E.F.: Proposal for cooperative robot "Gunryu" composed of autonomous segments. Robots and Autonomous Systems **17**(1–2), 107–118 (1996)
20. Hosokawa, K., Shimoyama, I., Miura, H.: Dynamics of self-assembling systems: analogy with chemical kinetics. Artificial Life **1**(4), 413–427 (1994)
21. Jones, C., Matarić, M.J.: From local to global behavior in intelligent self-assembly. In: Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA'03, vol. 1, pp. 721–726. IEEE Computer Society Press, Los Alamitos, CA (2003)
22. Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., Kokaji, S.: Automatic locomotion design and experiments for a modular robotic system. IEEE/ASME Transactions on Mechatronics **10**(3), 314–325 (2005)
23. Kawauchi, Y., Inaba, M., T, F.: A principle of distributed decision making of celluar robotic system (CEBOT). In: Proceedings of the 1993 IEEE International Conference on Robotics and Automation, pp. 833–838. IEEE Press, Piscataway, NJ (1993)
24. Klavins, E., Ghrist, R., Lipsky, D.: A grammatical approach to self-organizing robotic systems. IEEE Transactions on Automatic Control **51**(6), 949–962 (2006)
25. Mathews, N., Christensen, A.L., Ferrante, E., O'Grady, R., Dorigo, M.: Establishing spatially targeted communication in a heterogeneous robot swarm. In: Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), pp. 939–946. International Foundation for Autonomous Agents and Multiagent Systems (2010)
26. Mondada, F., Pettinaro, G.C., Guignard, A., Kwee, I.V., Floreano, D., Deneubourg, J.L., Nolfi, S., Gambardella, L.M., Dorigo, M.: SWARM-BOT: A new distributed robotic concept. Autonomous Robots **17**(2–3), 193–221 (2004)
27. Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K., Kokaji, S.: M-TRAN: Self-reconfigurable modular robotic system. IEEE/ASME Transactions on Mechatronics **7**(4), 431–441 (2002)
28. Mytilinaios, E., Desnoyer, M., Marcus, D., Lipson, H.: Designed and evolved blueprints for physical self-replicating machines. In: Proceedings of the 9th International Conference on the Simulation and Synthesis of Living Systems (Artificial Life IX), pp. 15–20. MIT Press (2004)
29. O'Grady, R., Christensen, A.L., Dorigo, M.: SWARMORPH: Multi-robot morphogenesis using directional self-assembly. IEEE Transactions on Robotics **25**(3), 738–743 (2009)
30. O'Grady, R., Pinciroli, C., Christensen, A.L., Dorigo, M.: Supervised group size regulation in a heterogeneous robotic swarm. In: 9th Conference on Autonomous Robot Systems and Competitions, Robótica 2009, pp. 113–119. IPCB-Instituto Politcnico de Castelo Branco, Castelo Branco, Portugal (2009)
31. Payne, K., Salemi, B., Will, P., Shen, W.M.: Sensor-based distributed control for chain-typed self-reconfiguration. In: Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 2074–2080
32. Penrose, L.S., Penrose, R.: A self-reproducing analogue. Nature **179**(4571), 1183 (1957)

33. Rus, D., Vona, M.: Self-reconfiguration planning with compressible unit modules. In: Proceedings of the 1999 IEEE International Conference on Robotics and Automation, ICRA'03, vol. 4, pp. 2513–2520. IEEE Computer Society Press, Los Alamitos, CA (1999)
34. Rus, D., Vona, M.: Crystalline robots: Self-reconfiguration with compressible unit modules. Autonomous Robots **10**(1), 107–124 (2001)
35. Salemi, B., Moll, M., Shen, W.M.: SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system. In: Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3636–3641. IEEE Press, Piscataway, NJ (2006)
36. Shen, W.M., Krivokon, M., Chiu, H., Everist, J., Rubenstein, M., Venkatesh, J.: Multimode locomotion for reconfigurable robots. Autonomous Robots **20**(2), 165–177 (2006)
37. Shen, W.M., Will, P., Galstyan, A., Chuong, C.M.: Hormone-inspired self-organization and distributed control of robotic swarms. Autonomous Robots **17**(1), 93–105 (2004)
38. Støy, K.: Using cellular automata and gradients to control self-reconfiguration. Robotics and Autonomous Systems **54**(2), 135–141 (2006)
39. Støy, K., Nagpal, R.: Self-reconfiguration using directed growth. In: Proceedings of the International Conference on Distributed Autonomous Robot Systems (DARS-04), pp. 1–10. Springer Verlag, Berlin, Germany (2004)
40. White, P., Zykov, V., Bongard, J., Lipson, H.: Three dimensional stochastic reconfiguration of modular robots. In: Proceedings of Robotics Science and Systems, pp. 161–168. MIT Press, Cambridge, MA (2005)
41. White, P.J., Kopanski, K., Lipson, H.: Stochastic self-reconfigurable cellular robotics. In: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, vol. 3, pp. 2888–2893. IEEE Computer Society Press, Los Alamitos, CA (2004)
42. Yim, M., Duff, D., Roufas, K.: Walk on the wild side. IEEE Robotics and Automation Magazine **8**(4) (2002)
43. Yim, M., Duff, D., Roufas, K.D.: Polybot: a modular reconfigurable robot. In: Proceedings of the 2000 IEEE International Conference on Robotics and Automation, vol. 1, pp. 514–520. IEEE Press, Piscataway, NJ (2000)
44. Yim, M., Roufas, K., Duff, D., Zhang, Y., Eldershaw, C., Homans, S.B.: Modular reconfigurable robots in space applications. Autonomous Robots **14**(2-3), 225–237 (2003)
45. Yim, M., Shen, W.M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.S.: Modular self-reconfigurable robot systems. IEEE Robotics & Automation Magazine **14**(1), 43–52 (2007)
46. Yim, M., Zhang, Y., Duff, D.: Modular robots. IEEE Spectrum Magazine **39(2)**, 30–34 (2002)
47. Yim, M., Zhang, Y., Roufas, K., Duff, D., Eldershaw, C.: Connecting and disconnecting for chain self-reconfiguration with PolyBot. IEEE/ASME Transactions on Mechatronics **7**(4), 442–451 (2002)
48. Yoshida, E., Murata, S., Kamimura, A., Tomita, K., Kurokawa, H., , Kokaji, S.: A self-reconfigurable modular robot: Reconfiguration planning and experiments. International Journal of Robotics Research **21(1011)**, 903–915 (2002)
49. Yu, C.H., Nagpal, R.: Sensing-based shape formation on modular multi-robot systems: a theoretical study. In: Proceedings of the 7th Internation Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), pp. 71–78. ACM New York, NY (2008)
50. Yu, C.H., Willems, F.X., Ingber, D., Nagpal, R.: Self-organization of environmentally-adaptive shapes on a modular robot. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 2007 (IROS 2007), pp. 2353–2360. IEEE Press, Piscataway, NJ (2007)
51. Zhang, Y., Roufas, K., Eldershaw, C., Yim, M., Duff, D.: Sensor computations in modular self reconfigurable robots. In: Proceedings of the 8th International Symposium on Experimental Robotics, vol. 5, pp. 276–286. Springer Verlag, Berlin, Germany (2003)
52. Zykov, V., Mytilinaios, E., Adams, B., Lipson, H.: Self-reproducing machines. Nature **435**(7039), 163–164 (2005)