Autonomous Reconfiguration in a Self-assembling Multi-robot System

Rehan O'Grady[1], Anders Lyhne Christensen[2], and Marco Dorigo[1]

[1] IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{rogrady,mdorigo}@ulb.ac.be
[2] DCTI-ISCTE, Lisbon, Portugal
anders.christensen@iscte.pt

Abstract. Self-assembling multi-robot systems can, in theory, overcome the physical limitations of individual robots by connecting to each other to form particular physical structures (morphologies) relevant to specific tasks. Here, we show for the first time how robots in a real-world multi-robot system can autonomously self-assemble into and reconfigure between arbitrary morphologies. We use a distributed control paradigm. The robots are individually autonomous and homogeneous - they all independently execute the same control program. Inter-robot communication is visual and strictly local. We demonstrate our technique on real robots.

1 Introduction

In multi-robot systems, the individual robots can carry out different tasks in parallel. When necessary, they can also cooperate. Self-assembly is a mechanism that allows teams of cooperating robots to overcome the physical limitations of the individual team members by connecting to each other to form physically larger composite robotic entities. In order to maximize the utility of self-assembly, the morphology of the resulting robotic entity must be appropriate to the task. In this paper, we demonstrate how a group of robots can autonomously self-assemble into and reconfigure between different specific morphologies.

This paper's contributions are as follows. 1) We demonstrate autonomous distributed reconfiguration in a real world system of self-assembling robots. 2) We build on our previous work to implement a coordination mechanism that allows a group of connected, independently controlled, self-assembled robots to coordinate the distributed reconfiguration process. 3) We show reconfiguration with six real robots from a star morphology to a line morphology to a square morphology and back to a star. 4) We demonstrate more complex reconfiguration in simulation.

2 Related Work

There is a large body of existing research on self-reconfiguring robotic systems. However, successful demonstrations of reconfiguration on real world robotic

M. Dorigo et al. (Eds.): ANTS 2008, LNCS 5217, pp. 259–266, 2008.

platforms have been restricted to systems in which the individual modules were simple and incapable of independently executing meaningful tasks. Other studies have explored the reconfiguration possibilities of systems made up of autonomous self-assembling robots, but only in simulation.

In self-reconfiguring modular systems, morphological flexibility is explored through the use of relatively simple connected units. Examples include CEBOT [1], PolyBot [2], M-TRAN [3], ATRON [4] and SuperBot [5]. For detailed overviews see [6,7]. In a few self-reconfigurable systems such as the Super-Mechano Colony [8] and the Swarm-bot platform (which we use in this study), the individual modules are capable of carrying out meaningful tasks on their own.

Some proposed control algorithms for self-assembly and/or reconfiguration are centralized, see for instance [9]. Other approaches give each unit a unique ID and a predefined position in the final structure, see for instance [10]. More distributed approaches include [11], [12], and [13]. In simulation, Støy and R. Nagpal [14,15] have demonstrated algorithms for self-reconfiguration and directed growth of cubic units based on gradients and cellular automata. Bojinov *et al.* [16] 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.

3 Hardware Platform and Control Methodology

For our experiments we use the Swarm-bot robotic platform [17], see Fig. 1 (left). This platform was designed and built by Francesco Mondada's group at the Laboratoire de Système Robotiques (LSRO) of EPFL. The Swarm-bot platform consists of a number of robots called *s-bots*. Each s-bot is self-propelled using a differential drive system composed of combined tracks and wheels (*treels*). Physical connections between s-bots are established by a gripper-based connection mechanism as shown in Fig. 1. An s-bot is surrounded by a transparent ring that can be grasped by other s-bots. Eight sets of RGB-colored LEDs are distributed around the inside of the transparent ring. Individual LEDs can be independently illuminated. The s-bot camera can perceive the illuminated LEDs of other s-bots at a range of up to approximately 50 cm depending on light conditions. The camera records the panoramic images reflected in a spherical mirror mounted above the s-bot in a perspex turret.

Our control paradigm is distributed. The robots are homogeneous and individually autonomous — each robot independently executes the same control program. Because of the limited sensing range of the robots, no individual robot can perceive the global shape of a morphology composed of physically connected s-bots. This means that an unattached robot cannot deduce where it should connect in order to extend or reconfigure a morphology appropriately. Instead, robots that are already part of a morphology indicate how new robots should connect. This is done using the *directional self-assembly* mechanism, which allows a robot to light up a specific set of LEDs in order to invite a connection at a given point on its body with a corresponding specified orientation for the connecting robot (see Fig. 1 right). This mechanism is described in more detail in [18].

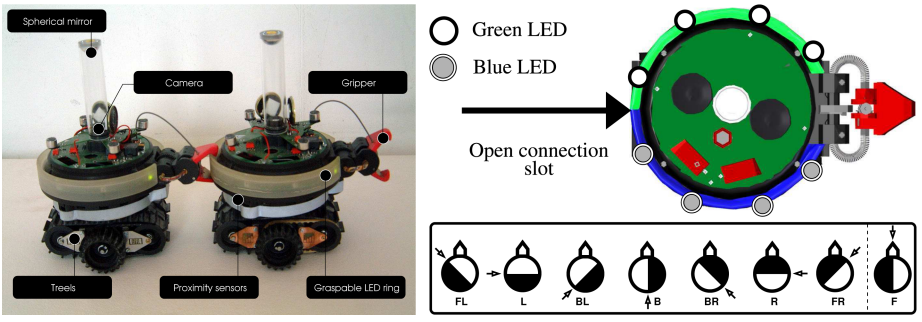

Fig. 1. Left: Two physically connected s-bots. Right top: An s-bot with an open connection slot to its rear (it has illuminated its left green LEDs and its right blue LEDs). Right bottom: Representation of the eight possible connection slots that an s-bot can open. The eighth connection slot (F) is only used for signalling, as it is occupied by the gripper of the s-bot displaying the connection slot.

When a new connection is formed, the two connected robots communicate. Using a low bandwidth visual communication protocol, instructions are typically passed from the robot that is already part of the morphology to the newly connected robot. Similar communication between connected robots occurs during the reconfiguration process. Simple algorithmic rule sets describe when the robots choose to communicate and how to interpret and act upon received communication. Different rule sets allow self-assembly into and reconfiguration between arbitrary morphologies. These rule sets are expressed in a high level descriptive language — SWARMORPH-script. This language is described in more detail in [19].

4 Reconfiguration

In our previous work [19], we showed how homogeneous independently operating robots executing SWARMORPH-script instructions can generate arbitrary morphologies. In this study, we use the same principles to generate autonomous reconfiguration in a distributed self-assembling system.

The key new challenge we have solved in this study is that of coordination between independently controlled robots once they are self-assembled. Coordination is an essential component for any distributed reconfiguration system. Imagine a group of connected robots crossing a hole. If the first robots to cross the hole detect a new obstacle which triggers self-reconfiguration and these robots start trying to reconfigure while some robots are still suspended over the hole, the consequences could be disastrous. In this example, we would want each individual robot to wait before reconfiguring until the whole morphology is ready to reconfigure. What renders this type of coordination difficult is the distributed nature of the system—each robot is controlled independently, and at the same time communication is local and can only occur between directly connected robots.

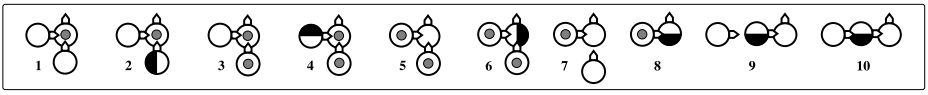

Fig. 2. Coordinated reconfiguration example. A small grey circle in the center of an s-bot indicates that the s-bot is waiting for a signal. For details see text.

On our hardware platform, each robot has a single gripper. This means that any morphology formed, whatever its spatial configuration (shape), has a tree-like connection topology, where a parent node can have many child nodes, but a child node can only have a single parent node. To enable coordination in a connected morphology, we allow for two types of communication. Information can be passed up the tree, and instructions can be passed down the tree.

An example of reconfiguration using these two types of communication is shown in Fig. 2. In this simple example, a three s-bot arrow morphology (one root node, two child nodes) reconfigures into a line morphology. The SWARMORPH-script to generate this behavior is shown in Algorithm 1 (the script includes instructions for the self-assembly of the initial arrow morphology which is not shown in the figure). The control is homogeneous for the three robots—the root node is the first robot to encounter a hole, the other two robots see an open connection slot, attach, and become the child nodes.

In Fig. 2, a small grey circle in the center of an s-bot indicates that the s-bot is waiting for a signal. In Fig. 2 step 1, the arrow morphology has already formed and the seed is waiting for both children to signal that they are ready to reconfigure. In steps 2-4, the child nodes independently wait for a given timeout before they are 'ready'. Once the child nodes are ready to reconfigure, they both independently signal their readiness to the root node.[1] Once they have signaled their readiness to the root node (information passing up the tree), the child nodes wait for a signal from the root node (instructions passing down the tree) before proceeding with any further reconfiguration steps. In step 5, the root node has received signals from both of its children, and thus knows that the whole morphology is ready to reconfigure. In steps 6-8, the root node signals to both of the child nodes in turn that they can proceed with the reconfiguration. In steps 9-10, having received the relevant instruction, each child node carries out its subsequent reconfiguration instructions. The result is the line morphology.

In the more general case, morphologies can be of arbitrary depth and have arbitrary numbers of branches. Information is passed up the tree from child nodes to parent nodes. Information continues to ascend the tree in this manner until one node takes responsibility for collating the information and issuing instructions to nodes beneath it in the tree. Starting from this collating node, instructions to start the reconfiguration process are then passed down the tree from parent nodes to child nodes.

[1] In this example, the child node signals do not overlap temporally. The algorithm would be unaffected, however, if both child nodes signalled their readiness at the same time—the root node would acknowledge receipt of each of the two signals in turn.

Algorithm 1. Reconfiguration script to generate the reconfiguration sequence in Fig. 2. The script is independently executed on each of the robots.

```
RandomWalk();                    // until hole detected OR connection slot detected
if hole-detected then
    // I am the root node since I detected the hole before I saw a connection slot
    OpenConnSlot(back);          // Attract a child robot and...
    SendRuleID(1);               // instruct it to follow rule 1
    OpenConnSlot(left);          // Attract another child robot and...
    SendRuleID(2);               // instruct it to follow rule 2
    WaitForASignal();            // One child is ready
    WaitForASignal();            // Other child is ready
    SendSignal(back);            // Instruct one child to start reconfiguration
    SendSignal(left);            // Instruct other child to start reconfiguration
end
else if connection-slot-detected then
    // I am a child node since I saw a connection slot before I detected the hole
    SearchForAndAttachToConnectionSlot();
    ReceiveRuleID();
    if receivedruleid = 1 then
        Timeout();               // Ready after timeout
        SendSignal(front);       // Inform parent I am ready
        WaitForASignal();        // Wait to receive reconfigure instruction
        Disconnect();
        SearchForConnSlot();
    end
    if receivedruleid = 2 then
        Timeout();               // Ready after timeout
        SendSignal(front);       // Inform parent I am ready
        WaitForASignal();        // Wait to receive reconfigure instruction
        OpenConnSlot(back);
    end
end
StopExecution();
```

In the experiments we perform in this study, the node that takes responsibility for collating information is always the root node (that is, the node that has no parents). However, in other cases, a purely local reconfiguration might be more efficient—this can occur if a node further down the tree takes responsibility for collating information and issuing reconfiguration instructions. Imagine, for example, a self-assembled entity in which a single constituent robot develops a fault. The local structure could reconfigure to eject the faulty robot, or otherwise compensate for the fault. The rest of the assembled robots need not be involved in or even be aware of the local reconfiguration. Note that local configuration still requires coordination—it is just that the coordination is restricted to the subset of assembled robots that are reconfiguring.

5 Results

We performed an experiment with real robots in which six s-bots form the sequence of morphologies: star-line-square-star. We implemented a simple coordination strategy to ensure that each individual morphology is complete before reconfiguration into the next morphology begins: each sub-branch of the

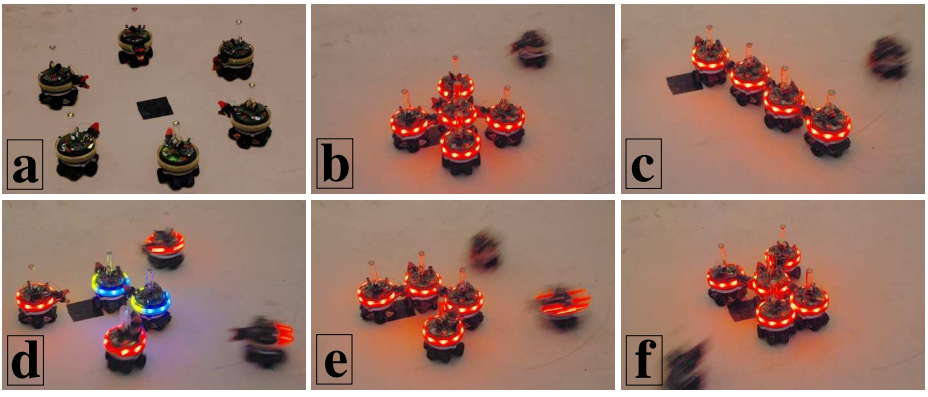

Fig. 3. Photos of different stages in a self-reconfiguration experiment with six real s-bots. a) The start configuration. b) The star morphology. c) The line morphology. d) The line morphology reconfiguring into the square. e) The square morphology. f) The star morphology.

morphology reports local completion up the tree (information passing up the tree). The root node collates the information, and once it is sure that all sub-branches have completed, it sends the instruction to reconfigure down through each branch (instructions passing down the tree). As individual nodes receive the reconfiguration instruction, they execute their subsequent reconfiguration control logic that results in the formation of the next morphology.

Photographic snapshots of this experiment with the real robots are shown in Fig. 3. Videos of the experiments described in this section and other explanatory material, including full SWARMORPH-script reconfiguration algorithms, can be found on the web at http://iridia.ulb.ac.be/supp/IridiaSupp2008-004.

We conducted several more complex experiments in simulation. An example is shown in Fig. 4. Here, by executing the relevant SWARMORPH-script program, the robots first assemble into a 9-robot square formation and then reconfigure into three 3-robot arrow morphologies. During the reconfiguration, two connected pairs of s-bots (four s-bots in total) remain connected. Using a

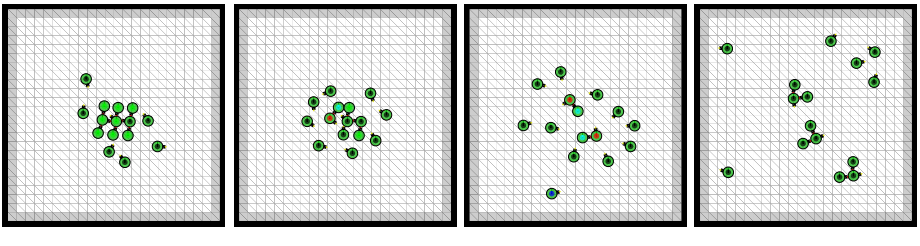

Fig. 4. Snapshots of different stages in a self-reconfiguration experiment with fifteen simulated s-bots (reconfiguration from single 9 s-bot square morphology to three 3 s-bot arrow morphologies)

SWARMORPH-script primitive for coordinated motion, these pairs travel away from the site of the original morphology in opposite directions to give themselves the space required to form subsequent morphologies. These pairs form the basis for two of the three new arrow morphologies. The s-bot that seeded the square morphology does not move, and becomes the seed for the third new arrow morphology. All other s-bots detatch and try to join growing morphologies by attaching to displayed connection slots.

6 Conclusions and Future Work

In this study, we showed how a group of real robots can autonomously reconfigure using self-assembly and local communication between connected robots. Our approach relies on a completely distributed control paradigm — the robots are all independently autonomous and rely only on local communication to cooperate. One advantage of distributed control is scalability. With large numbers of robots, different subsets of robots could perform different tasks in parallel. Since our approach is decentralized, multiple different morphologies can be formed and undergo reconfiguration at the same time.

In our ongoing research we are trying to give the robots the capability to identify different types of obstacle and to reconfigure adaptively into appropriate morphologies based on the environments they encounter.

Acknowledgments. This work was made possible by the innovative robotic hardware developed by Mondada's group at the Laboratoire de Système Robotiques (LSRO) of EPFL. This work was supported by the *SWARMANOID* project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-022888 and by the *VIRTUAL SWARMANOID* project funded by the F.R.S.-FNRS. The information provided is the sole responsibility of the authors and does not reflect the European Commission's opinion. The European Commission is not responsible for any use that might be made of data appearing in this publication. Marco Dorigo acknowledges support from the F.R.S.-FNRS, of which he is a Research Director.

References

1. 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)
2. 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)
3. 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)
4. Østergaard, E.H., Kassow, K., Beck, R., Lund, H.H.: Design of the ATRON lattice-based self-reconfigurable robot. Autonomous Robots 21(2), 165–183 (2006)

5. Shen, W., Krivokon, M., Chiu, H., Everist, J., Rubenstein, M., Venkatesh, J.: Multimode locomotion for reconfigurable robots. Autonomous Robots 20(2), 165–177 (2006)
6. 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)
7. Groß, R., Bonani, M., Mondada, F., Dorigo, M.: Autonomous self-assembly in swarm-bots. IEEE Transactions on Robotics 22(6), 1115–1130 (2006)
8. Damoto, R., Kawakami, A., Hirose, S.: Study of super-mechano colony: concept and basic experimental set-up. Advanced Robotics 15(4), 391–408 (2001)
9. Rus, D., Vona, M.: Crystalline robots: Self-reconfiguration with compressible unit modules. Autonomous Robots 10(1), 107–124 (2001)
10. 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 (2005)
11. 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, ICRAM 2003, vol. 1, pp. 721–726. IEEE Computer Society Press, Los Alamitos (2003)
12. 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)
13. 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)
14. Støy, K., Nagpal, R.: Self-reconfiguration using directed growth. In: Proceedings of the Internation Conference on Distributed Autonomous Robot Systems (DARS-2004), pp. 1–10. Springer, Berlin (2004)
15. Støy, K.: Using cellular automata and gradients to control self-reconfiguration. Robotics and Autonomous Systems 54(2), 135–141 (2006)
16. Bojinov, H., Casal, A., Hogg, T.: Emergent structures in modular self-reconfigurable robots. In: Proceedings of the IEEE International Conference on Robotics & Automation, vol. 2, pp. 1734–1741. IEEE Computer Society Press, Los Alamitos (2000)
17. 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)
18. O'Grady, R., Christensen, A.L., Dorigo, M.: SWARMORPH: Multi-robot morphogenesis using directional self-assembly. Technical Report IRIDIA/2008-001, IRIDIA, Université Libre de Bruxelles (2008)
19. Christensen, A.L., O'Grady, R., Dorigo, M.: SWARMORPH-script: A language for arbitrary morphology generation in self-assembling robots. Swarm Intelligence 2 (in press, 2008)