Short Papers

SWARMORPH: Multirobot Morphogenesis Using Directional Self-Assembly

Rehan O'Grady, Anders Lyhne Christensen, and Marco Dorigo

***Abstract*—In this paper, we propose SWARMORPH: a distributed morphology generation mechanism for autonomous self-assembling mobile robots. Self-organized growth of global morphological structures emerges through the repeated application of local morphology extension rules. We present details of the directional self-assembly mechanism that provides control over the orientation of interrobot connections. We conduct real-world experiments to validate the low-level directional self-assembly mechanism and the growth of global morphologies. We demonstrate the scalability of the approach with large numbers of robots in simulation-based experiments.**

***Index Terms*—Autonomous agents, cellular and modular robots, distributed robot systems, morphology control, self-assembly.**

I. Introduction

Flexibility is one of the key potential advantages of multirobot systems. 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. With the right type of teamwork, even fundamental physical limitations of individual robots such as power, size, or reach can be overcome. One enabling mechanism for this type of physical cooperation is self-assembly—the process of forming physical connections between robots to generate a larger composite robotic entity [1], [2].

Multirobot 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. To date, however, such systems have displayed little active control over the morphology of the composite robotic entity formed through the self-assembly process. This lack of morphological control is a major limitation, as for any robotic entity to complete a task efficiently, its morphology must be appropriate to the task. In the majority of existing multirobot self-assembling systems, the geometry of the self-assembled entity is either predefined by the connection mechanism of the hardware or is stochastic.

There is a large body of work on morphologically flexible modular robotic systems [3]–[12]. However, the modules of such self-reconfigurable systems tend to be either incapable of autonomous motion or so simple as to be incapable of meaningful independent action. Furthermore, these systems are often incapable of autonomously assimilating additional modules. For detailed overviews of self-reconfigurable systems and self-assembling systems, see [1] and [13]. Various approaches to controlling and assembling modular robots have been proposed, including [14]–[18]. Existing research, however, tends to be either quite abstract and simulation-based or relies on specific robot IDs to control positions in the generated structures.

In this paper, we propose a technique for controlling the self-organized growth of morphological structures on a real-world self-propelled self-assembling multirobot system. Using our system—SWARMORPH—we can specify global morphologies using local morphology extension rules that govern the self-assembly process. Our rule definition paradigm avoids the use of central control, broadcast communication, and symbolic communication. The rules are executed in turn by each new robot that connects to the morphology. SWARMORPH shares some stochastic features of many natural and artificial self-assembling systems; the self-assembling components (robots in our case) are interchangeable, and the movement of the components in the environment is random. In contrast with most previously studied self-assembling systems, however, the random movement occurs through the use of a self-propelled random walk. The main contribution of this paper is to show the systematic self-assembly of global structures with self-propelled self-assembling robots.

In a previous work [19], we showed the growth of four global morphologies. Here, we extend that work into a generic morphology growth platform. We abstract our local rules into a set of primitives and show how these primitives can be combined in different ways to generate a large number of different global morphologies. We detail the low-level control mechanism—*directional self-assembly*—that these rules use to control the self-assembly process and present experiments validating this mechanism.

A key benefit of distributed control is scalability. We present results in simulation to demonstrate the scalability of our system. We validate our simulated environment by comparison with our earlier results on real robots.

II. Robotic Platform

This study was conducted on the *swarm-bot* robotic platform [20], which was designed and built by Mondada's group at the École Polytechnique Fédérale de Lausanne (EPFL) in Switzerland. The swarm-bot system consists of a number of mobile autonomous robots called *s-bots* (see Fig. 1) that are capable of forming physical connections with each other.

Each s-bot is surrounded by a semitransparent LED ring that contains eight sets of red, green, blue (RGB) colored LEDs. The LEDs are

Manuscript received July 14, 2008; revised December 1, 2008. First published April 7, 2009; current version published June 5, 2009. This paper was recommended for publication by Associate Editor I-M. Chen and Editor L. Parker upon evaluation of the reviewers' comments. This work was supported by the *SWARMANOID* project, funded by the Future and Emerging Technologies Programme of the European Commission, under Grant IST-022888 and by the VIRTUAL SWARMANOID project, funded by the Fonds de la Recherche Scientifique (FNRS).

R. O'Grady and M. Dorigo are with the Computer and Decision Engineering Department, Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, Université Libre de Bruxelles, Brussels 1050, Belgium (e-mail: rogrady@ulb.ac.be; mdorigo@ulb.ac.be).

A. L. Christensen is with the Departamento de Ciências e Tecnologias da Informação, Lisbon University Institute, Lisbon 1649-026, Portugal (e-mail: anders.christensen@iscte.pt).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. This material includes videos (arrow_morphology_7_real_robots_h264.avi, line_morphology_7_real_robots_h264.avi, rectangle_morphology_7_real_robots_h264.avi, star_morphology_29_simulated_robots_h264.avi, star_morphology_7_real_robots_h264.avi, and t_shape_morphology_10_simulated_robots_h264.avi) that show morphology generation experiments using SWARMORPH, both on real robots and in simulation. The videos are encoded using h264 compression. VLC media player 0.8.6 and above will play these videos. VLC media player is available on almost every platform and can be downloaded from http://www.videolan.org/vlc/. The total size of the supplementary material is 42 MB. Contact rogrady@ulb.ac.be for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2008.2012341

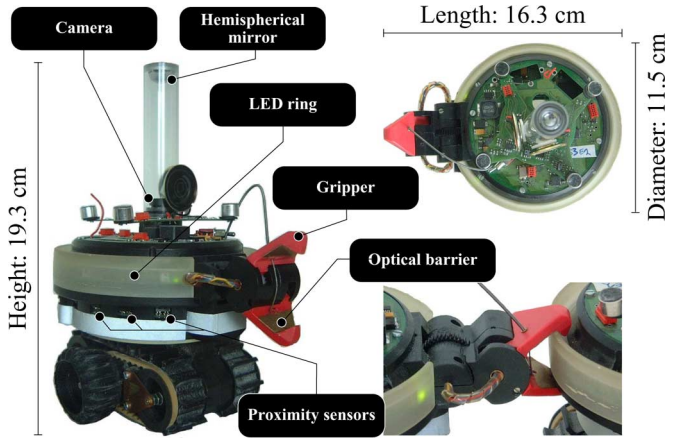

Fig. 1. S-bot: Autonomous, mobile robot capable of self-assembly. Weight: ∼700 g; battery: ∼2 h; processor: 400-MHz XScale CPU; operating system: Linux.

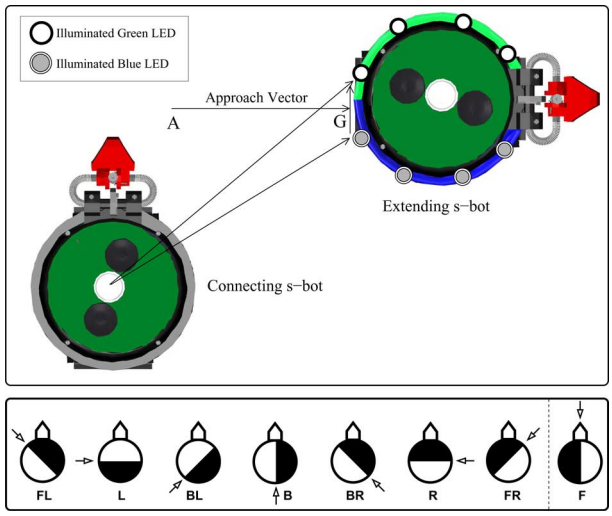

Fig. 2. (Top) Connecting s-bot calculates the approach vector for a connection slot displayed by an extending s-bot. The approach point (A), grip point (G), and approach vector $\overrightarrow{AG}$, as calculated by the connecting s-bot. The connecting s-bot calculates A and G from the closest blue LED to the closest green LED. The distance from the approach point to the s-bot with the open connection slot is 13 cm. (Bottom) Eight possible connection slots that an s-bot can open. The eighth connection slot (F) is occupied by the gripper of the s-bot displaying the connection slot and is therefore used only for signaling (see Section IV).

distributed uniformly in eight locations around the ring. A green LED, a blue LED, and a red LED are located at each LED location. The LED ring can be grasped by another s-bot using its gripper. An optical light barrier inside the s-bot gripper indicates when another s-bot's LED ring is between the jaws of the gripper. For more details, see [20].

III. Directional Self-Assembly

A. Concepts and Terminology

Directional self-assembly is a mechanism we developed to allow a self-assembling s-bot to specify the location and orientation of an inter-robot connection. SWARMORPH relies on this directional self-assembly mechanism to extend a morphology locally in a particular direction.

To grow the morphology appropriately, an s-bot that is already connected to the morphology illuminates a particular configuration of LEDs to indicate a point on its body where a new s-bot should grip and a corresponding orientation that the gripping s-bot should assume. We term such a configuration of LEDs a *connection slot*. 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. 2, top). A connection slot can be opened between any two neighboring LED locations on the s-bot LED ring, except between the two front LED locations, where the gripper is mounted. As a connection slot requires the use of all eight LED locations, an s-bot can open only one connection slot at a time. We refer to an s-bot that is displaying an open connection slot as an *extending s-bot* and an s-bot that is navigating toward or trying to connect to (grip) a connection slot as a *connecting s-bot*. A connection slot is considered "open" until an s-bot connects to it, at which point, it is considered "filled."

B. Approaching and Gripping

S-bots that are not already part of the connected morphology perform a random walk until they see a connection slot. When an s-bot can see a connection slot, it tries to connect to it. Unconnected s-bots illuminate their red LEDs and avoid each other using proximity sensors and their cameras. When two s-bots are both trying to connect to a single connection slot, they both retreat and wait for a random amount of time before trying again.

A connecting s-bot uses the illuminated LEDs of an extending s-bot's open connection slot to calculate the *approach vector* (see Fig. 2). The head of the approach vector is called the *grip point* and indicates the point on the extending s-bot's body at which the connecting 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. The heading of the approach vector matches the orientation that the connection slot specifies for the connecting s-bot. Thus, by simply following the approach vector to the grip point and then gripping, a connecting s-bot will grip the extending s-bot at the correct location and will assume the correct corresponding orientation.

A connecting s-bot first navigates to the approach point at the tail of the approach vector. It then rotates to face the grip point at the head of the approach vector and only then starts to navigate along the approach vector to the grip point. The closer an s-bot gets to the grip point, the more accurately it perceives the LEDs of the connection slot, and therefore, the more precisely it can calculate the approach vector. Corrections to the connecting s-bot's trajectory are made continuously as the s-bot approaches the grip point. During the approach, the speed of the s-bot is reduced as a linear function of the distance to the grip point (the magnitude of the alignment corrections becomes correspondingly smaller). When the s-bot determines that it is close enough to connect, it attempts to grip by closing its gripper. If the grip fails, the s-bot moves back and starts navigating to the approach point again.

C. Results

We analyzed the performance of the directional self-assembly mechanism. We ran 96 trials of an experiment in which a single s-bot connected to a stationary extending s-bot. In every trial, the extending s-bot had the same position and orientation. At the start of each trial, the extending s-bot opened connection slot B (directly behind it; see Fig. 2, bottom). For the connecting s-bot, we used 12 possible starting positions and eight possible starting orientations. Over the 96 trials, we used each possible combination of these starting positions and orientations once. The starting positions for the connecting s-bot were evenly

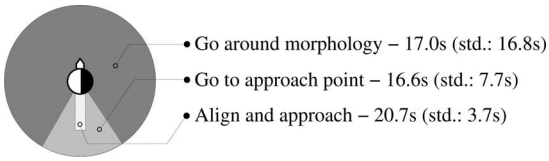

Fig. 3. Time spent in different navigation zones.

distributed around a circle of radius 35 cm centered on the extending s-bot.

In all 96 trials, the connecting s-bot successfully connected to the extending s-bot. In two of the 96 trials, the grip failed on the first attempt and the connecting s-bot retreated to try another angle. In a further four trials, the connecting s-bot retreated to try another angle before even attempting to grip, as it determined that it was approaching from an incorrect angle. In a single trial, the connecting s-bot lost sight of the connection slot and was manually replaced on its starting position.

The location on the extending s-bot's body was, on average, 0.75 cm (standard deviation 0.62 cm) from the grip point. The connecting s-bot, on average, assumed an angle of 9.9° (standard deviation 6.4°) from the optimum.

The mean times spent on the different navigation zones are shown in Fig. 3. The largest share of the time was spent on the final alignment and approach phases, during which the s-bot rotates on the approach point to face the grip point and then follows the approach vector to the grip point. Although the distances covered in the different navigation zones vary significantly, the mean times spent in the different zones are comparable. This near equivalence is a consequence of the increasing precision required as the s-bot gets closer to the connection slot—the more precision required, the slower the s-bot moves. The mean time from the start of a trial until the connecting s-bot gripped was 54.3 s.

IV. Global Morphologies From Local Rules

SWARMORPH morphology growth occurs through the repeated application of local extension rules. As each new s-bot connects to the morphology, it becomes an extending s-bot and follows its extension rules to determine how the local structure should be extended. To carry out the extension dictated by a particular rule, the s-bots use the directional self-assembly mechanism described in the previous section. By combining extension rules in different ways, we can build distributed control algorithms that grow specific morphologies.

Morphology growth starts when an initial *seed* s-bot initiates morphology growth by opening the first connection slot. The seed s-bot has its own set of extension rules. A separate rule set governs the behavior of all other extending s-bots (the seed can also be considered an extending s-bot).

A. Extension Rule Set

In this section, we describe the extension rules currently implemented in SWARMORPH. All of the rules are based purely on locally sensed information—none of the rules make reference to any kind of global blueprint, nor do they rely on any symbolic communication between modules.

The `Extend` rule uses directional self-assembly to extend the structure by opening a specified connection slot. The rule takes as a parameter the connection slot that should be opened. Using only this extension rule, we can form simple morphologies like the line and the circle in Fig. 4. For the line morphology, each connected s-bot just executes the rule `Extend(B)`, while for the circle morphology, each connected s-bot executes the rule `Extend(BL)`.

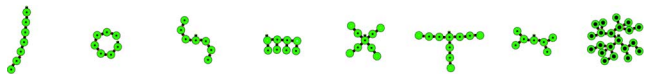

Fig. 4. Examples of different morphologies that can be made using SWARMORPH. These morphologies have been generated in simulation.

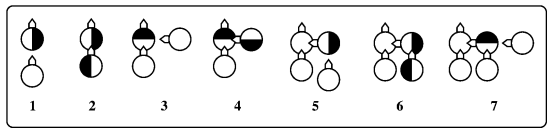

Fig. 5. Rectangle morphology growth. (1) Seed opens connection slot B. (2) Connected s-bot sends handshake signal. (3) Seed sees handshake signal and then opens connection slot R. First connected s-bot executes `Decide` rule—sees connection slot and, therefore, executes no subsequent extension rules. (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.

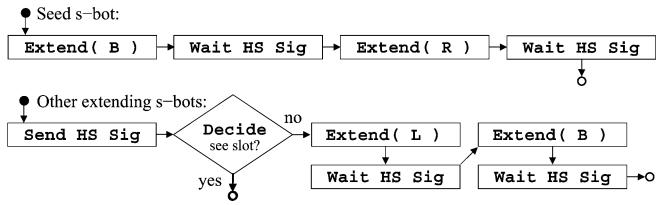

Fig. 6. Rectangle morphology extension rule sets.

The rules `Send HS Sig` and `Wait HS Sig` together make up the handshake signal. The handshake signal allows a connecting s-bot to signal to an extending s-bot that it has successfully connected to the extending s-bot's open connection slot (the s-bot hardware does not include any dedicated sensors to detect when it has been gripped by another s-bot). This signalling is required by the more complex branching morphologies that require a single s-bot to extend the local structure in more than one direction (for example, the last five morphologies in Fig. 4). The extending s-bot must know when a connection slot has been filled so that it can open the next connection slot at the right time (an s-bot can open only one connection slot at a time). The `Send HS Sig` rule tells the connecting s-bot to signal to the extending s-bot after it grips successfully. 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 `Decide` rule allows an s-bot to make conditional decisions about how to extend the local structure based on what it can see in its surroundings. In particular, this rule allows a connecting s-bot to modify its behavior based on the posthandshake actions of the extending s-bot to which it is connecting. In Fig. 5, we use the `Decide` rule to grow the rectangle morphology. The rule sets used are shown in Fig. 6.

The `Balance` rule enforces balanced morphology growth until the extremities of the morphology are no longer within visual range of each other. When we use the `Send HS Sig` and `Wait HS Sig` rules to generate branching morphologies, we run the risk that the morphology will be unbalanced—it is stochastically possible that one branch will develop much faster than another. Such imbalances are particularly problematic when the number of s-bots is limited, as is the case in our real robot experimentation. When executing the `Balance` rule, a connected robot waits with its LEDs unilluminated until it cannot see any connection slots around it. Once the s-bot can no longer see

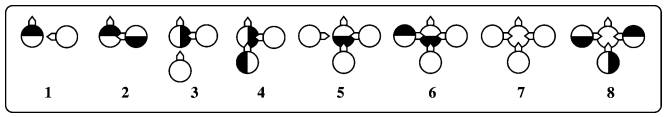

Fig. 7. T-shape morphology growth. The seed s-bot opens connection slots R, B, and L in turn (waiting each time for a handshake signal from the connecting s-bots). The connected s-bots each execute the `Balance` rule and, therefore, wait until they can see no green or blue LEDs before executing subsequent 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.

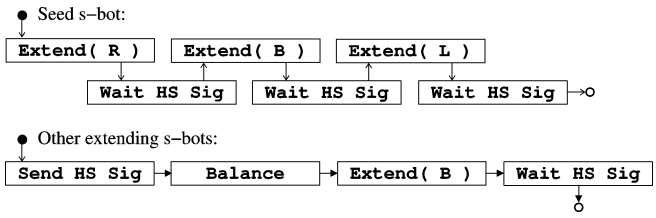

Fig. 8. T-shape morphology extension rule sets.

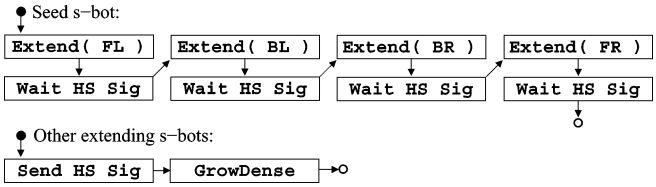

Fig. 9. Stochastic dense morphology extension rules.

any green or blue LEDs (i.e., connection slots), the connected s-bot continues executing subsequent extension rules. In Fig. 7, we use the `Balance` rule to grow a T-shape morphology. The rule sets used are shown in Fig. 8. In the T-shape morphology, as long as the s-bots at the ends of the three branches can still see each other, the branches will differ in length by at most one s-bot.

The `GrowDense` rule enables the stochastic growth of dense morphologies. When a connected s-bot executes the `GrowDense` rule, it opens a series of connection slots (L, R, BR, BL, B) in a repeating cycle. A connection slot remains open until filled or until it has been open for 120 s. After a connection slot has been filled or after the time-out, the next connection slot in the cycle is opened. The order of connection slots in the `GrowDense` cycle has been chosen to encourage dense morphology growth—connection slots near the gripper are opened first to encourage breadth-first expansion. The morphology growth is unpredictable because stochastic local conditions determine which of the opened connection slots get filled. In particular, for a connection slot to be filled, there must be an unconnected s-bot nearby (which depends on the unconnected s-bots' random walk), and there must be sufficient space available to extend the morphology (which depends on the previous random growth of the morphology). The rightmost morphology in Fig. 4 is an example of a dense morphology. The rule sets used are shown in Fig. 9.

B. SWARMORPH Generable Morphologies

Using SWARMORPH, we can generate a large number of different morphologies (see Fig. 4).[1] However, in the current rule set, there is no way to change the growth pattern of a morphology during morphology growth—all the morphologies we generated have repeating structures. This limitation could be overcome, however, if the morphologies could respond to the environment. With the `GrowDense` rule, we have already shown how a single rule set can generate different morphologies stochastically. In a real task execution scenario, it might be interesting to add an extension rule that could conditionally modify morphology extension based on what an s-bot detects in its environment. Using the s-bot infrared proximity sensors, we could imagine, for example, encircling a prey object by having each s-bot extend the morphology to match the local contours of the object. This type of rule would allow for adaptive morphologies, while still avoiding the need for any global or symbolic communication. Simple group size regulation could also be achieved with a technique similar to that used by the `Balance` rule. The seed s-bot could light up in a particular color. Morphology extension would stop (or change in nature) when s-bots lose sight of the seed robot.

[1]Each s-bot has a single gripper. This fact restricts us to the generation of tree-like structures. A morphology consisting of a single closed loop would also be possible but would require complex coordination between connected s-bots and is therefore not possible with the current implementation of SWARMORPH.

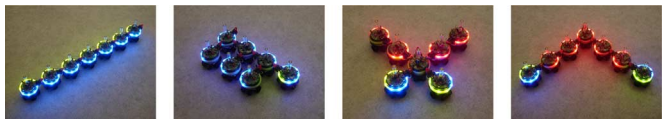

Fig. 10. Reprinted from [19]. Four morphologies grown with seven real robots: Line, arrow, star, rectangle. (The rectangle and star morphologies would become symmetric with the addition of more robots.) Videos of experiments conducted with real robots and in simulation can be found in the supplementary material online at http://ieeexplore.ieee.org or at http://iridia.ulb.ac.be/supp/IridiaSupp2008-017.

TABLE I
MEAN COMPLETION TIMES FOR DIFFERENT MORPHOLOGIES ON REAL S-BOTS AND IN SIMULATION

	Real s-bots	Simulated s-bots	Difference
Line	365.53 s (± 54.41 s)	339.76 s (± 77.53 s)	−7.0%
Arrow	346.56 s (± 58.27 s)	331.37 s (± 87.85 s)	−4.2%
Star	335.32 s (± 70.76 s)	325.31 s (± 89.98 s)	−3.0%
Rectangle	663.46 s (±227.51 s)	456.71 s (±297.97 s)	−31.2%

Numbers in brackets denote standard deviations.

V. GROWING MORPHOLOGIES WITH REAL ROBOTS

We selected four morphologies to grow in experiments with seven real s-bots. Examples of the four morphologies are shown in Fig. 10. We grew each of the four example morphologies (line, arrow, star, rectangle) ten times with seven real s-bots.

All six 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. A detailed analysis of the timing of these experiments can be found in [19].

VI. SCALABILITY

A. Simulation Environment

We conducted experiments with larger numbers of s-bots in simulation. Our simulation environment consists of a specialized software simulator with a custom dynamics engine tailored to our robotic platform [21]. We developed a control interface abstraction layer that allowed us to transfer our control programs between the simulator and the real robots without any modification.

The control abstraction layer allowed us to run and test SWARMORPH-based control programs, both in simulation and on real robots. This development model gave us a large degree of

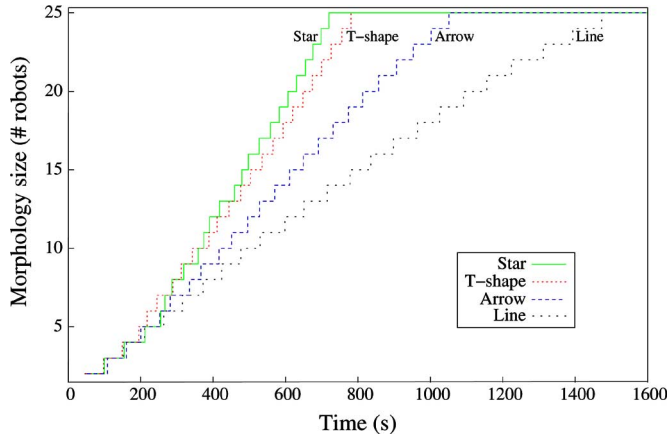

Fig. 11. Morphology growth over time for four morphologies formed with simulated robots. We fix the number of unconnected s-bots in the system at 10 by feeding a new s-bot into the simulation after every connection. Each line shows the growth over time of a single morphology averaged over 100 experimental trials.

confidence about the accuracy of our simulator. To further validate the verisimilitude of our simulation environment, we repeated in simulation the experiments that we had already done with the real robots. We set up an experiment with seven simulated s-bots for each of the four morphologies tested on real s-bots in the previous section. Table I lists the mean completion times for ten real robot trials for each morphology and the results obtained in 100 trials in simulation for the same morphologies. For the line, arrow, and star morphologies, the mean completion time in simulation is slightly lower than the mean completion time observed in the experiments with real robots. The mean completion time for the rectangle morphology is 31% lower in simulation than observed in our experiments with real robots. We believe that this difference is primarily due to the fact that when the robots are densely packed, the camera is more accurate in simulation than in reality. In the rectangle morphology, robots are located close to each other, giving rise to effects such as occlusions and reflections of the light emitted from the LEDs. These effects are not modeled by our simulator. We therefore chose to avoid the use of densely packed morphologies such as the rectangle morphology in our scalability tests.

B. Scalability

In order to test how fast various morphologies grow when more robots are available, we conducted experiments with four different morphologies. We chose the morphologies line, arrow, T-shape, and star, because they can, respectively, have one, two, three, and four connection slots open simultaneously (see Fig. 4). In order to keep our arena size manageable, we added robots to the simulation gradually. We kept the number of unconnected robots constant, at 10. Each time an unconnected robot connected to the morphology, a new unconnected robot was added to the simulation. For each morphology, we conducted 100 trials. In each trial, we varied the initial placement and orientation of the unconnected s-bots. A trial was considered finished when the morphology reached a size of 25 connected robots.

Fig. 11 shows morphology growth over time for the different morphologies. The growth profiles of the morphologies are very similar until the fifth robot connects to the morphology, at which point, they start to differ. The star morphology can have up to four connection slots open simultaneously and, on average, reaches a size of 25 robots in 720 s. The T-shape morphology can have up to three connection slots open simultaneously and reaches a size of 25 robots in 781 s. The relatively small difference (61 s) between the average completion times is probably due to the fact that the number of unconnected robots is fixed at 10. In order for the extra open connection slots to speed up morphology growth, there have to be robots in the vicinity each time a new connection slot is opened. Since the ten unconnected robots are not always evenly distributed, the star morphology completes only marginally faster than the T-shape morphology.

Comparing the completion times for the star and the T-shape morphologies with the arrow and line morphology, the benefit of multiple open connection slots becomes more apparent; the arrow morphology (which has up to two connection slots open simultaneously) and the line morphology (which has only one connection slot open at a time) reach the size of 25 robots in 1051 and 1473 s, respectively. For the line morphology, the rate of growth slows down as the morphology gets larger. The line morphology has only one connection slot open at a time, and because of its 1-D growth, for any given number of s-bots, it stretches further than the other morphologies. Thus, even once an unconnected s-bot has approached the connected structure, it can take a long time for the s-bot to navigate to the open connection slot. This delaying factor is less present for the other morphologies, partly because more connection slots are open and partly because they grow in two dimensions.

VII. Conclusion and Future Research

In this paper, we demonstrated the decentralized growth of specific morphologies using a real-world self-assembling robotic platform. Our system, called SWARMORPH, consists of a set of local extension rules built on top of a dedicated directional self-assembly mechanism. We analyzed the reliability of this low-level directional self-assembly mechanism and found it both robust and precise. We achieved a high success rate in building four different morphologies using seven real robots. The absence of symbolic communication makes our morphology growth algorithms simpler and, thus, potentially transferable to less-sophisticated self-assembling robotic systems. We demonstrated the scalability potential of our approach with experiments in simulation using larger numbers of robots.

In the future, we intend to investigate adaptive use of morphogenesis with respect to the robotic task. In an all-terrain navigation task, for example, the group could self-assemble into an elongated morphology in order to cross a ditch, while on rough and uneven terrain the robots could self-assemble into a more dense, stable morphology.

Acknowledgment

This work would not have been possible without the innovative robotic hardware developed by F. Mondada's group at the École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. 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.

References

[1] R. Gro and M. Dorigo, "Self-assembly at the macroscopic scale," *Proc. IEEE*, vol. 96, no. 9, pp. 1490–1508, Sep. 2008.

[2] R. O'Grady, R. Gro, F. Mondada, M. Bonani, and M. Dorigo, "Self-assembly on demand in a group of physical autonomous mobile robots navigating rough terrain," in *Proc. 8th Eur. Conf., Adv. Artif. Life* (Lecture Notes in Artificial Intelligence), vol. 3630, Berlin, Germany: Springer-Verlag, 2005, pp. 272–281.

[3] Y. Kawauchi, I. Makoto, and T. Fukuda, "A principle of distributed decision making of cellular robotic system (CEBOT)," in *Proc. IEEE Int. Conf. Robot. Autom.*, Piscataway, NJ: IEEE Press, 1993, pp. 833–838.
[4] M. Yim, K. Roufas, D. Duff, Y. Zhang, C. Eldershaw, and S. B. Homans, "Modular reconfigurable robots in space applications," *Auton. Robots*, vol. 14, no. 2/3, pp. 225–237, 2003.
[5] M. Yim, D. G. Duff, and K. D. Roufas, "PolyBot: A modular reconfigurable robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, Piscataway, NJ: IEEE Press, 2000, pp. 514–520.
[6] A. Castano, W. M. Shen, and P. Will, "CONRO: Towards deployable robots with inter-robots metamorphic capabilities," *Auton. Robots*, vol. 8, no. 3, pp. 309–324, 2000.
[7] S. Hirose, T. Shirasu, and E. F. Fukushima, "Proposal for cooperative robot "Gunryu" composed of autonomous segments," *Robot. Auton. Syst.*, vol. 17, pp. 107–118, 1996.
[8] H. B. Brown, J. M. V. Weghe, C. A. Bererton, and P. K. Khosla, "Millibot trains for enhanced mobility," *IEEE/ASME Trans. Mechatron.*, vol. 7, no. 4, pp. 452–461, Dec. 2002.
[9] R. Damoto, A. Kawakami, and S. Hirose, "Study of super-mechano colony: Concept and basic experimental set-up," *Adv. Robot.*, vol. 15, no. 4, pp. 391–408, 2001.
[10] K. Motomura, A. Kawakami, and S. Hirose, "Development of arm equipped single wheel rover: Effective arm-posture-based steering method," *Auton. Robots*, vol. 18, no. 2, pp. 215–229, 2005.
[11] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-tran: Self-reconfigurable modular robotic system," *IEEE/ASME Trans. Mechatron.*, vol. 7, no. 4, pp. 431–441, Dec. 2002.
[12] W. M. Shen, M. Krivokon, H. Chiu, J. Everist, M. Rubenstein, and J. Venkatesh, "Multimode locomotion for reconfigurable robots," *Auton. Robots*, vol. 20, no. 2, pp. 165–177, 2006.
[13] M. Yim, W. M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems," *IEEE Robot. Autom. Mag.*, vol. 14, no. 1, pp. 43–52, Mar. 2007.
[14] D. Rus and M. Vona, "Crystalline robots: Self-reconfiguration with compressible unit modules," *Auton. Robots*, vol. 10, no. 1, pp. 107–124, 2001.
[15] P. White, V. Zykov, J. Bongard, and H. Lipson, "Three dimensional stochastic reconfiguration of modular robots," in *Proc. Robot. Sci. Syst.*, Cambridge, MA: MIT Press, 2005, pp. 161–168.
[16] C. Jones and M. J. Matarić, "From local to global behavior in intelligent self-assembly," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, Los Alamitos, CA: IEEE Comput. Soc. Press, 2003, pp. 721–726.
[17] E. Klavins, R. Ghrist, and D. Lipsky, "A grammatical approach to self-organizing robotic systems," *IEEE Trans. Autom. Control*, vol. 51, no. 6, pp. 949–962, Jun. 2006.
[18] W. M. Shen, P. Will, A. Galstyan, and C. M. Chuong, "Hormone-inspired self-organization and distributed control of robotic swarms," *Auton. Robots*, vol. 17, no. 1, pp. 93–105, 2004.
[19] A. L. Christensen, R. O'Grady, and M. Dorigo, "Morphology control in a multirobot system," *IEEE Robot. Autom. Mag.*, vol. 14, no. 4, pp. 18–25, Dec. 2007.
[20] F. Mondada, L. M. Gambardella, D. Floreano, S. Nolfi, J.-L. Deneubourg, and M. Dorigo, "The cooperation of swarm-bots: Physical interactions in collective robotics," *IEEE Robot. Autom. Mag.*, vol. 12, no. 2, pp. 21–28, Jun. 2005.
[21] A. L. Christensen, "Efficient neuro-evolution of hole-avoidance and phototaxis for a swarm-bot," D.E.A. thesis, Inst. Recherches Interdisciplinaires Dév. Intell. Artif. (IRIDIA), Univ. Libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2005-14, Oct. 2005.

Image-Based Visual Servo Control of the Translation Kinematics of a Quadrotor Aerial Vehicle

Odile Bourquardez, Robert Mahony, Nicolas Guenard, François Chaumette, Tarek Hamel, and Laurent Eck

***Abstract*—In this paper, we investigate a range of image-based visual servo control algorithms for regulation of the position of a quadrotor aerial vehicle. The most promising control algorithms have been successfully implemented on an autonomous aerial vehicle and demonstrate excellent performance.**

***Index Terms*—Aerial robotic vehicle, visual servoing.**

I. Introduction

Visual servo algorithms have been extensively developed in the robotics field over the last ten years [7], [10], [19], [23]. Visual servo control techniques have also been applied recently to a large variety of reduced-scale aerial vehicles, such as quadrotors [1], [25], helicopters [2], [22], [26], [29], airships [4], [30], and airplanes [5], [24]. In this paper, we consider visual servo control of a quadrotor aerial vehicle.

Much of the existing research in visual servo control of aerial robots (and particularly, autonomous helicopters) has used position-based visual servo techniques [1], [2], [22], [25]–[27], [29]. The estimated pose can be used directly in the control law [1], or as part of a scheme fusing visual data and inertial measurements [29]. In this paper, we do not deal with pose estimation, but consider image-based visual servo (IBVS), similar to the approach considered in [4], [17], and [30].

The system dynamics is sometimes explicitly taken into account in IBVS. This strategy has been applied for robotic manipulators [9], [12], [20] and for aerial vehicles [15], [30]. Another popular approach (as usually done for most robotic systems such as robot arms, mobile robots, etc.) is based on separating the control problem into an inner loop and an outer position control loop. As for helicopters, the inner attitude loop is run at high gain using inputs from inertial sensors, rate gyrometers, and accelerometers acquired at high data rate, while the outer loop is run at low gain using video input from the camera [26], [27]. The outer (visual servo) loop provides set points for the inner attitude loop and classical time-scale separation and high-gain arguments can be used to ensure stability of the closed-loop system [1], [11], [15], [27].

Manuscript received June 17, 2008; revised December 4, 2008. First published February 2, 2009; current version published June 5, 2009. This paper was recommended for publication by Associate Editor P. Rives and Editor W. K. Chung upon evaluation of the reviewers' comments. This work was supported by the Centre National de la Recherche Scientifique (CNRS) under the Project Robotique et Entités Artificielles (ROBEA)–Robvolint and the International Programs for Scientific Cooperation (PICS) between France and Australia on visual servo-control of unmanned aerial vehicles.

O. Bourquardez and F. Chaumette are with the Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)–Centre National de la Recherche Scientifique (CNRS) and l'Institut National de Recherche en Informatique et en Automatique (INRIA), 35042 Rennes, France (e-mail: odile.bourquardez@voila.fr; francois.chaumette@irisa.fr).

R. Mahony is with the Department of Engineering, Australian National University, Canberra, A.C.T. 0200, Australia (e-mail: robert.mahony@anu.edu.au).

N. Guenard and L. Eck are with the Commissariat à l'Energie Atomique (CEA)/List, 92265 Fontenay-aux-Roses, France (e-mail: nicolas.guenard@cea.fr; laurent.eck@cea.fr).

T. Hamel is with the Laboratoire d'Informatique, Signaux et Systèmes de Sophia antipolis (I3S), Université de Nice Sophia-Antipolis (UNSA)–Centre National de la Recherche Scientifique (CNRS), 06903 Sophia Antipolis, France (e-mail: thamel@i3s.unice.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2008.2011419