



A simple approach to the control of locomotion in self-reconfigurable robots[☆]

K. Støy^{a,*}, W.-M. Shen^b, P.M. Will^b

^a *The Adaptronics Group, The Maersk Institute, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark*

^b *USC Information Sciences Institute and Computer Science Department, 4676 Admiralty Way, Marina del Rey, CA 90292, USA*

Abstract

In this paper we present role-based control which is a general bottom-up approach to the control of locomotion in self-reconfigurable robots. We use role-based control to implement a caterpillar, a sidewinder, and a rolling track gait in the CONRO self-reconfigurable robot consisting of eight modules. Based on our experiments and discussion we conclude that control systems based on role-based control are minimal, robust to communication errors, and robust to reconfiguration.

© 2003 Elsevier Science B.V. All rights reserved.

Keywords: Self-reconfigurable robots; Locomotion; Role-based control

1. Introduction

Reconfigurable robots are robots made from a possibly large number of independent modules connected to form a robot. If the modules from which the reconfigurable robot is built are able to connect and disconnect without human intervention the robot is a self-reconfigurable robot. Refer to Fig. 1 for an example of a module of a self-reconfigurable robot or refer to one of the other physical realized systems described in [7,8,10–15,17,21,23].

Several potential advantages of self-reconfigurable robots over traditional robots have been pointed out in literature:

- *Versatility.* The modules can be combined in different ways making the same robotic system able to perform a wide range of tasks.
- *Adaptability.* While the self-reconfigurable robot performs its task it can change its physical shape to adapt to changes in the environment.
- *Robustness.* Self-reconfigurable robots consist of many identical modules and therefore if a module fails it can be replaced by another.
- *Cheap production.* When the final design for the basic module has been obtained it can be mass produced. Therefore, the cost of the individual module can be kept relatively low in spite of its complexity.

Self-reconfigurable robots can solve the same tasks as traditional robots, but as Yim et al. [23] point out; in applications where the task and environment are given a priori it is often cheaper to build a special purpose robot. Therefore, applications best suited for self-reconfigurable robots are applications where some leverage can be gained from the special abilities of self-reconfigurable robots. The versatility of these

[☆] The work presented here was performed while visiting USC Information Sciences Institute.

* Corresponding author.

E-mail addresses: kaspers@mip.sdu.dk, kaspers@isi.edu (K. Støy), shen@isi.edu (W.-M. Shen), will@isi.edu (P.M. Will).

URL: <http://www.mip.sdu.dk>

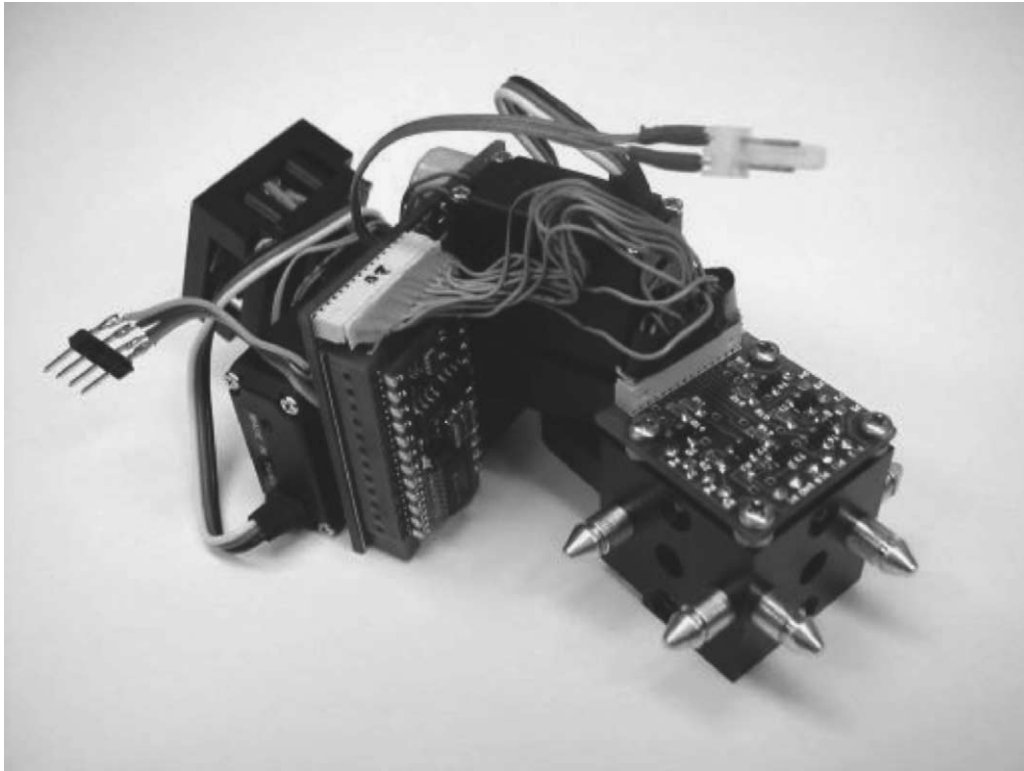


Fig. 1. A CONRO module. The three male connectors are located in the lower right corner. The female connector is partly hidden from view in the upper left corner.

robots make them suitable in scenarios where the robots have to handle a range of tasks. The robots can also handle tasks in unknown or dynamic environments, because they are able to adapt to these environments. In tasks where robustness is of importance it might be desirable to use self-reconfigurable robots. Even though real applications for self-reconfigurable robots still are to be seen, a number of applications have been envisioned [17,23]: fire fighting, search and rescue after an earthquake, battlefield reconnaissance, planetary exploration, undersea mining, and space structure building. Other possible applications include entertainment and service robotics.

The potential of self-reconfigurable robots can be realized if several challenges in terms of hardware and software can be met. In this work we focus on one of the challenges in software: how do we make a large number of connected modules perform a coordinated global behavior? Specifically we address how

to design algorithms that will make it possible for self-reconfigurable robots to locomote efficiently. In order for a locomotion algorithm to be useful it has to preserve the special properties of these robots. From the advantages and applications mentioned above we can extract a number of guidelines for the design of such a control algorithm. The algorithm should be distributed to avoid having a single point of failure. Also the performance of the algorithm should scale with an increased number of modules. It has to be robust to reconfiguration, because reconfiguration is a fundamental capability of self-reconfigurable robots. Finally, it is desirable to have homogeneous software running on all the modules, because it makes it possible for any module to take over if another one fails.

It is an open question if a top-down or a bottom-up approach gives the best result. We find that it is difficult to design the system at the global level and then later try to make an implementation at the local level,

because often properties of the hardware are ignored and a slow robotic system might be the result. Therefore, we use a bottom-up approach where the single module is the basic unit of design. That is, we move from a global design perspective to a bottom-up one where the important design element is the individual module and its interactions with its neighbors. The global behavior of the system then emerges from the local interaction between individual modules. A similar approach is also used by Bojinov et al. [1,2] and Butler et al. [4].

2. Related work

In the related work presented here we focus on control algorithms for locomotion of self-reconfigurable robots.

Yim et al. [22,23] demonstrate caterpillar-like locomotion and a rolling track. Their system is controlled based on a gait control table. Each column in this table represents the actions performed by one module. Motion is then obtained by having a master synchronizing the transition from one row to the next. The problem with this approach is that the amount of communication needed between the master and the modules will limit its scalability. Another problem is the need for a central controller, since it gives the system a single point of failure. If there is no master it is suggested that the modules can be assumed to be synchronized in time and each module can execute its column of actions open-loop. However, since all the modules are autonomous it is a questionable assumption to assume that all the modules are and can stay synchronized. In order to use the gait control table each module needs to know what column it has to execute. This means that the modules need IDs. Furthermore, if the configuration changes or the number of modules changes the table has to be rewritten.

Shen et al. [17] propose to use artificial hormones to synchronize the modules to achieve consistent global locomotion. In earlier versions of the system a hormone is propagated through the self-reconfigurable system to achieve synchronization. In later work the hormone is also propagated backward making all modules synchronized before a new action is initiated [16,18]. This synchronization takes time $O(n)$, where n is the number of modules. This slows down the

system considerably, because it has to be done before each action. Also, the entire system stops working if one hormone is lost. This is a significant problem, because a hormone can easily be lost due to unreliable communication, a module disconnecting itself before a response can be given, or a module failure. In fact, the system has n points of failure which is not desirable. The earlier version is better in this sense, but still performance remains low because a synchronization hormone is sent before each action.

Butler et al. [4] propose a method inspired by cellular automata. In their approach modules respond to state changes of neighbor modules. Their approach is a bottom-up approach related to ours, but in cellular automata there is no concept of time only of sequence. Timing is important in locomotion, because it is the key to produce smooth and life-like locomotion and avoid jerky locomotion.

In our system all modules repeatedly go through a cyclic sequence of joint angles describing a motion. This sequence could come from a column in a gait control table, but in our implementation the joint angles are calculated using a cyclic function with period T . Every time a module has completed a specified fraction d of the period a message is sent through the child connectors. If the signal is received the child module resets its action sequence making it delayed d compared to the parent. This way the actions of the individual module are decoupled from the synchronization mechanism resulting in a faster and more reliable system. Furthermore, there is no need to make changes to the algorithm if the number of modules changes.

3. Role-based control

We assume that the modules are connected to form a tree structure, that a parent connector is specified, and that this connector is the only one that can connect to child connectors of other modules. Furthermore, we assume that the modules can communicate with the modules to which they are connected.

The algorithm is instantiated by specifying three components. The first component is a cyclic action sequence $A(t)$, where $t \in [0 : T]$. T is the second component that needs to be specified and is the period of the action sequence. $A(t)$ describes the actions that each module repeats cycle after cycle. In our

implementation $A(t)$ returns joint angles to control the two degrees of freedom of the CONRO module, but the action sequence could also be used to trigger different behaviors at different times during a cycle. The third component is a delay d . This delay specifies the fraction of a period the children's action sequences are delayed compared to their parents. The skeleton algorithm looks like this:

```
t=0
while(true) {
  if(t=d)then <signal childmodules>
  if <parent signals> then t=0
  <perform action A(t)>
  t=(t+1) modulus T
}
```

Ignoring the first two lines of the loop, the module repeatedly goes through a sequence of actions parameterized by the cyclic counter t . This part of the algorithm alone can make a single module repeatedly perform the specified sequence of actions. In order to coordinate the actions of the individual modules to produce the desired global behavior the modules need to be synchronized. Therefore, at step $t = d$ a signal is sent through all child connectors. If a child receives a signal it knows that the parent is at $t = d$ and therefore sets its own step counter to $t = 0$. This enforces that the child is delayed d compared to its parent.

From the time the modules are connected it takes time proportional to d times the height of the configuration tree for all the modules to synchronize. To avoid problems with uncoordinated modules initially we make sure the modules do not start moving until they receive the first synchronization signal. After the start-up phase the modules stay synchronized using only constant time.

4. Experimental setup

To evaluate our algorithm we conducted several experiments using the CONRO (CONfigurable ROBOT) modules of which one is shown in Fig. 1. The CONRO modules have been developed at USC/ISI [5,9]. The modules are roughly shaped as rectangular boxes measuring 10 cm \times 4.5 cm \times 4.5 cm and weigh 100 g. The modules have a female connector at one end and three male connectors located at the other. Each connector

has a infra-red transmitter and receiver used for local communication and sensing. The modules have two controllable degrees of freedom: pitch (up and down) and yaw (side to side). Processing is taken care of by an onboard Basic Stamp 2 processor. The modules have onboard batteries, but these do not supply enough power for the experiments reported here and therefore the modules are powered through cables. Refer to <http://www.isi.edu/conro> for more details and videos of the experiments reported later.

5. Experiments

In this section we describe three different locomotion gaits implemented using role-based control. For each gait we have chosen to report the length of our programs as a measure of the complexity of the control algorithm. These results are used to support our claim that the implemented control systems are minimal. We also report the speed of the locomotion patterns, but this should only be considered an example, the reason being that in our system the limiting factors are how robust the modules physically are, how powerful the motors are, and how much power we can pull from the power source. To report a top speed is not meaningful before we run the robot autonomously on batteries.

5.1. Caterpillar locomotion

We connect eight of our modules in a chain and designate the male opposite the female connector to be the parent connector. We then implement the algorithm described above with the following parameters.

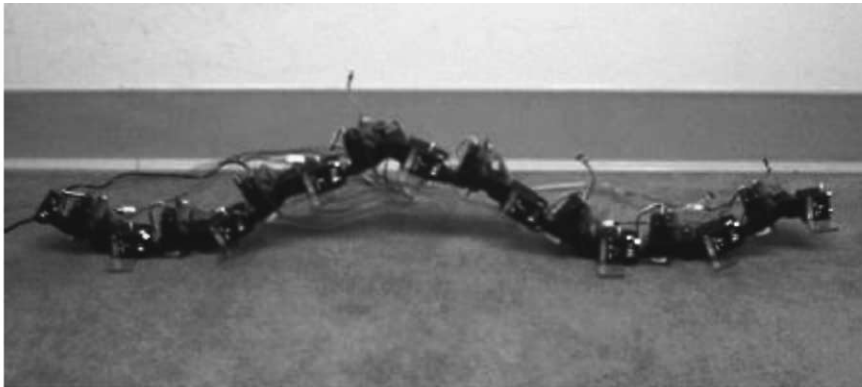
$$\begin{aligned}
 T &= 180, \\
 \text{pitch}(t) &= 50^\circ \sin\left(\frac{2\pi}{T}t\right), \\
 \text{yaw}(t) &= 0, \\
 d &= \frac{1}{5}T.
 \end{aligned} \tag{1}$$

Pitch and yaw is measured in a coordinate system where a yaw and a pitch of zero mean that the joints are straight. The motor control of our modules makes the joint go to the desired angle as fast as possible. This means that way-points have to be specified to avoid jerky motion. The period T can be used to control the

number of way-points and therefore the smoothness and speed of the motion. The action sequence is an oscillation around 0° with an amplitude of 50° for the pitch angle and the yaw joint is kept straight. Each module is delayed one-fifth of a period compared to its parent.

The modules are connected and after they synchronize a sine wave is traveling along the length of the

robot. Refer to Fig. 2. This produces caterpillar-like locomotion at a speed of 4 cm/s. Note that it is easy to adjust the parameters of this motion. For instance, the length of the wave can be controlled using the delay. The program is simple. The main loop contains 13 lines of code excluding comments and labels (shown in Fig. 2). The initialization including variable and constant declaration amounts to 18 lines of code.



SignalChild:

```

if t=d then SignalChildOn
SignalChildOff:
low TX_Port           'turns off the infra red diode
goto HandleSignalFromParent
SignalChildOn:
high TX_Port          'turns on the infra red diode
goto HandleSignalFromParent

```

HandleSignalFromParent:

```

if IN14=0 then Move   'checks if the infra red receiver
t = 0                 'is activated

```

Move:

```

pitch = 127+SIN(t*2*128/180)  'pitch is calculated and scaled
pitch = ((t*3)/4)+20
pulsout PITCH_MOTOR,(pitch*3)+350 'PWMs for motors are generated
pulsout YAW_MOTOR,(125*3)+350
t=(t+1)//T                    't is incremented
goto SignalChild

```

Fig. 2. The source code for the main loop of the caterpillar controller (bottom). The algorithm is described in Section 3. A snapshot of caterpillar-like locomotion (top).

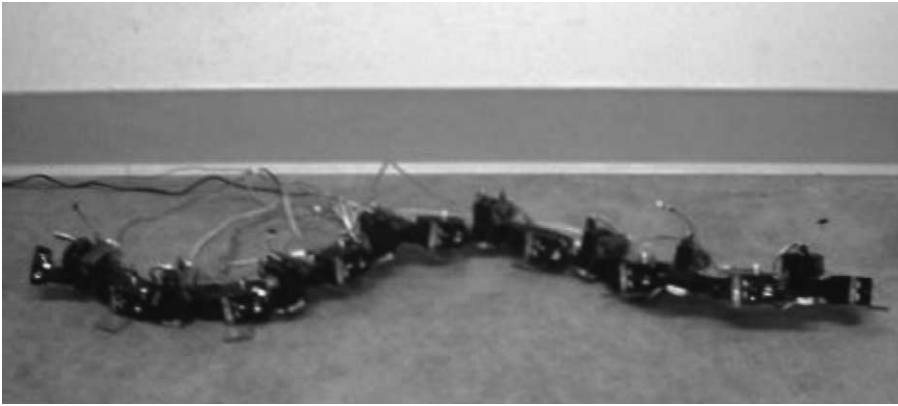


Fig. 3. A snapshot of sidewinder-like locomotion.

5.2. Sidewinder locomotion

We now turn our attention to a locomotion pattern similar to that of a sidewinding snake. A detailed mathematical analysis of this locomotion pattern has been reported in [3]. Here we just use the intuition that by having modules moving to one side lifted and those moving to the other touching the ground a sidewinder-like locomotion pattern is achieved. The result can be seen in Fig. 3. The sidewinder moves at 6 cm/s. The main loop and the initialization contain, respectively, 19 and 17 lines of code. The parameters used are:

$$\begin{aligned}
 T &= 180, \\
 \text{pitch}(t) &= 20^\circ \cos\left(\frac{2\pi}{T}t\right), \\
 \text{yaw}(t) &= 50^\circ \sin\left(\frac{2\pi}{T}t\right), \\
 d &= \frac{1}{5}T.
 \end{aligned} \tag{2}$$

5.3. Rolling track locomotion

If we maintain that each module can only have one parent, but remove the assumption that the structure forms a tree we include loops as structures that can be handled. The rolling track is an example of such a configuration. However, this poses a problem to our algorithm. In the previous experiments we have exploited the assumption that the modules form a tree to implicitly find a leader: since synchronization signals only are propagated down in the configuration tree a module

is the leader of the subtree below it. In a configuration tree this implies that the root is the unique leader of the system. In a loop configuration this is not the case.

One solution to this problem is to introduce IDs. In our implementation we just make the modules pick a random number and use that as ID. It is not guaranteed that each module has a unique ID, but it is most often the case and the shortcomings of this approach can easily be avoided if each module has a unique serial number.

The synchronization part of the algorithm now works as before, but it is combined with a simple well-known distributed leader election algorithm [6]. The signals from parent to child now contain a number which is the ID of the module originally sending the signal. Upon receiving a signal a module compares the signal's number to its ID. If it is higher the module is synchronized and the signal and its ID is propagated along with the synchronization signal. Otherwise, the module considers itself the leader and ignores the signal. After the system has settled the module with the highest ID dictates the rhythm of the locomotion pattern. The leader election algorithm runs continuously which means that the system quickly synchronizes if modules are replaced. The advantage of combining the algorithms is that there is no need to detect if the leader fails.

We used this algorithm to implement the rolling track which can be seen in Fig. 4. The rolling track is the fastest gait and achieves a speed of 13 cm/s. The program is now a little more complex and the main loop and initialization contain, respectively, 35 and 28

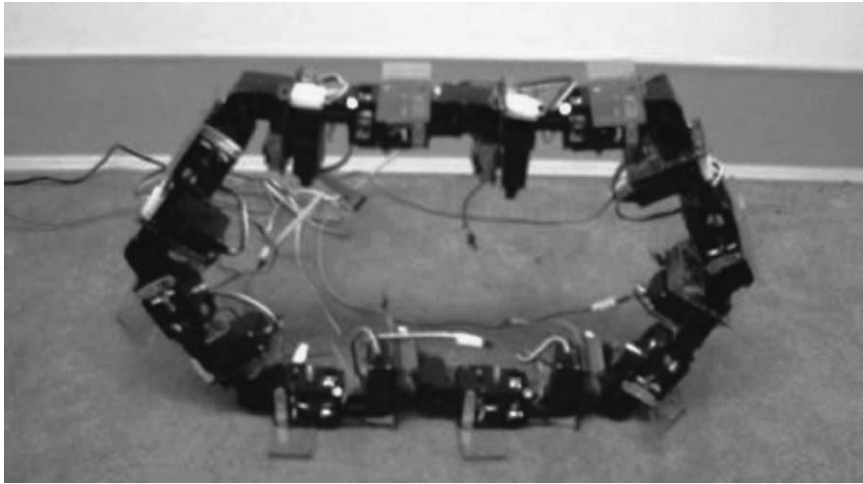


Fig. 4. The rolling track.

lines of code. The parameters for the eight module rolling track are:

$$\begin{aligned}
 T &= 180, \\
 \text{pitch}(t) &= \begin{cases} 60^\circ \left(1 - \sin\left(\frac{2\pi}{T}t\right)\right) & \text{if } t \leq \frac{1}{2}T, \\ 60^\circ & \text{if } t > \frac{1}{2}T, \end{cases} \\
 \text{yaw}(t) &= 0, \\
 d &= \frac{1}{4}T. \tag{3}
 \end{aligned}$$

Unlike the controller for the sidewinder gait and the caterpillar gait this controller only works with eight modules, because of the physical constraint. It might be possible to make a more general solution by making $\text{pitch}(t)$ and d a function of the number of modules. The number of modules in the loop could be obtained by the leader by including a hop count in the signal.

6. Handling a general configuration

We saw in the previous section that we had to introduce IDs to find a unique leader in a configuration that contains loops. Introducing the ID mechanism unfortunately ruins the opportunity to use the synchronization algorithm to automatically find a leader in a tree structure, because synchronization signals are only propagated down in the configuration tree. In fact,

the loop algorithm will fail in this situation unless the module with the highest ID also happens to be the root. In order to make a general algorithm the synchronization signal has to be propagated both upward and downward in the tree.

7. Discussion

An important issue in the design of control algorithms for self-reconfigurable robots is that the algorithms should still be efficient in systems consisting of many modules. Role-based control is only initially dependent on the number of modules, because it decides how long it takes for the synchronization signal to be propagated through the system. After this start-up phase it takes constant time to keep the modules synchronized implying that the algorithm scales.

In role-based control all modules run identical programs and there is no representation of a modules position in the configuration. Therefore, the system is highly robust to reconfiguration. In fact, the caterpillar can be divided in two and both parts still work. If they are reconnected in a different order they will quickly synchronize to behave as one caterpillar again. This also implies that the system is robust to module failure. If a module is defect and it can be detected this module can be ejected from the system and the remaining modules when reconnected can continue to perform. Finally, if a synchronization signal is lost it

is not crucial for the survival of the system. If a signal is lost it just means that the receiving module and its children will be synchronized a period later.

In role-based control the synchronization signal is only sent once per period. This means that in order for the modules to stay synchronized the time to complete a period has to be the same for all modules. In the experiments presented here the cycles take the same amount of time, but in more complex control systems where other parts of the control system use random amounts of computation time this cannot be assumed to be true. This problem can easily be handled by using timers. Even though timers are not precise enough to keep modules synchronized over a long period of time they can be used for this purpose.

In the work presented here we have shown what can be achieved using as simple a control algorithm as possible. In related work we have investigated how we can extend the algorithm to handle more complex locomotion gaits. We have shown how to implement a hexapod walking gait in the CONRO system in [19].

Another issue is that if the self-reconfigurable robot is to locomote autonomously in a real complex environment the control algorithm has to be able to take feedback from the environment into account. We have done some initial work on how sensor feedback can be included in role-based control in [20].

8. Summary

We have presented role-based control a general control algorithm for controlling locomotion in self-reconfigurable robots. The algorithm has the following properties: distributed, scalable, homogeneous, and minimal. We have shown how the algorithm easily can be used to implement a caterpillar- and sidewinder-like locomotion pattern. Furthermore, we have seen that by giving modules IDs it is possible to handle loop configurations. We have demonstrated this using the rolling track as an example.

Acknowledgements

This work is supported under the DARPA contract DAAN02-98-C-4032, the EU contract IST-20001-

33060, and the Danish Technical Research Council contract 26-01-0088.

References

- [1] H. Bojinov, A. Casal, T. Hogg, Emergent structures in modular self-reconfigurable robots, in: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00)*, San Francisco, CA, 2000, vol. 2, pp. 1734–1741.
- [2] H. Bojinov, A. Casal, T. Hogg, Multiagent control of self-reconfigurable robots, in: *Proceedings of the Fourth International Conference on MultiAgent Systems*, Boston, MA, 2000, pp. 143–150.
- [3] J.W. Burdick, J. Radford, G.S. Chirikjian, A 'sidewinding' locomotion gait for hyper-redundant robots, in: *Proceedings of the IEEE International Conference on Robotics and Automation*, Atlanta, GA, 1993, pp. 101–106.
- [4] Z. Butler, R. Fitch, D. Rus, Experiments in locomotion with a unit-compressible modular robot, in: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002, pp. 2813–2818.
- [5] A. Castano, R. Chokkalingam, P. Will, Autonomous and self-sufficient conro modules for reconfigurable robots, in: *Proceedings of the Fifth International Symposium on Distributed Autonomous Robotic Systems (DARS'00)*, Knoxville, TX, 2000, pp. 155–164.
- [6] E. Chang, R. Roberts, An improved algorithm for decentralized extrema-finding in circular configurations of processes, *Communications of the ACM* 22 (5) (1979) 281–283.
- [7] G.S. Chirikjian, Metamorphic hyper-redundant manipulators, in: *Proceedings of the 1993 JSME International Conference on Advanced Mechatronics*, Tokyo, Japan, 1993, pp. 467–472.
- [8] K. Hosokawa, T. Tsujimori, T. Fujii, H. Kaetsu, H. Asama, Y. Kuroda, I. Endo, Self-organizing collective robots with morphogenesis in a vertical plane, in: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'98)*, Leuven, Belgium, 1998, pp. 2858–2863.
- [9] B. Khoshnevis, B. Kovac, W.-M. Shen, P. Will, Reconnectable joints for self-reconfigurable robots, in: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, Maui, HI, 2001, pp. 584–589.
- [10] K. Kotay, D. Rus, M. Vona, C. McGray, The self-reconfiguring robotic molecule, in: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'98)*, Leuven, Belgium, 1998, pp. 424–431.
- [11] S. Murata, H. Kurokawa, S. Kokaji, Self-assembling machine, in: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'94)*, San Diego, CA, 1994, pp. 441–448.
- [12] S. Murata, H. Kurokawa, E. Yoshida, K. Tomita, S. Kokaji, A 3-d self-reconfigurable structure, in: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'98)*, Leuven, Belgium, 1998, pp. 432–439.

- [13] S. Murata, E. Yoshida, K. Tomita, H. Kurokawa, A. Kamimura, S. Kokaji, Hardware design of modular robotic system, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'00), Takamatsu, Japan, 2000, pp. 2210–2217.
- [14] A. Pamecha, C. Chiang, D. Stein, G.S. Chirikjian, Design and implementation of metamorphic robots, in: Proceedings of the ASME Design Engineering Technical Conference and Computers in Engineering Conference, Irvine, CA, 1996, pp. 1–10.
- [15] D. Rus, M. Vona, Crystalline robots: self-reconfiguration with compressible unit modules, *Autonomous Robots* 10 (1) (2001) 107–124.
- [16] B. Salemi, W. Shen, P. Will, Hormone controlled metamorphic robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, Seoul, Republic of Korea, 2001, pp. 4194–4199.
- [17] W.-M. Shen, B. Salemi, P. Will, Hormone-based control for self-reconfigurable robots, in: Proceedings of the International Conference on Autonomous Agents, Barcelona, Spain, 2000, pp. 1–8.
- [18] W.-M. Shen, B. Salemi, P. Will, Hormones for self-reconfigurable robots, in: Proceedings of the International Conference on Intelligent Autonomous Systems (IAS-6), Venice, Italy, 2000, pp. 918–925.
- [19] K. Støy, W.-M. Shen, P. Will, How to make a self-reconfigurable robot run, in: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02), Bologna, Italy, 2002, pp. 813–820.
- [20] K. Støy, W.-M. Shen, P. Will, On the use of sensors in self-reconfigurable robots, in: Proceedings of the Seventh International Conference on the Simulation of Adaptive behavior (SAB'02), Edinburgh, UK, 2002, pp. 48–57.
- [21] C. Ünsal, P.K. Khosla, Mechatronic design of a modular self-reconfiguring robotic system, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00), San Francisco, CA, 2000, pp. 1742–1747.
- [22] M. Yim, Locomotion with a unit-modular reconfigurable robot, Ph.D. Thesis, Department of Mechanical Engineering, Stanford University, 1994.
- [23] M. Yim, D.G. Duff, K.D. Roufas, Polybot: a modular reconfigurable robot, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'00), San Francisco, CA, 2000, pp. 514–520.



Kasper Støy is a Ph.D. student at The Maersk Institute for Production Technology, University of Southern Denmark. He received his M.S. in computer science from University of Aarhus, Denmark in 1999. Before starting his Ph.D. program he worked as a research scientist at University of Southern California's Robotics Labs conducting research on biology inspired multi-robot coordination.

As part of his Ph.D. program he visited the University of Southern California's Information Sciences Institute where the research presented here was performed. His research interests include self-reconfigurable robots, biological inspired multi-robot coordination and robot learning.



Wei-Min Shen is Director of Polymorphic Robotics Laboratory, Associate Director for Center for Robotics and Embedded Systems, and Research Assistant Professor in computer science at University of Southern California. He received his Ph.D. from Carnegie Mellon University in 1989 under the Nobel Prize Winner Professor Herbert Simon. Dr. Shen's research interests include Artificial Intelligence, Robotics, and Life Science. He has won several research awards in these fields, including USC Faculty Recognition Award in 2003, the RoboCup World Championship Award in 1997, and the AAI Robotics Competition Silver-Medal Award in 1996. He is the author of "Autonomous Learning from the Environment". He has chaired several international conferences and workshops in Robotics, Machine Learning, and Data Mining, and served on the editorial boards for two scientific books and one international journal. His research achievements have been reported by news media such as CNN, PBS, Discovery channel, LA Times, BYTE, Chinese World Journal, and SCIENCES.



Peter M. Will is an ISI Fellow, and the Director of the Distributed Scalable Systems Division at USC/Information Sciences Institute and is a Research Professor in Industrial and Systems Engineering Department and Material Science Department at USC, and has over 35 years research experience in industry. He received USC Faculty Recognition Award in 2003. He spent 16 years at IBM's Yorktown Research Laboratory, 7 years with Schlumberger, and 5 years at HP Labs. He has over 50 publications and 10 patents. He has served as chair of three NSF advisory committees and as Chair of the National Academy Study on Information Technology in Manufacturing. For 6 years he was a member of the ISAT group working with DARPA. In 1990, he was awarded the International Engelberger Prize in robotics. He received a B.Sc. degree in Electrical Engineering and a Ph.D. in non-linear Control Systems from the University of Aberdeen.