On the Use of Sensors in Self-Reconfigurable Robots

K. Støy* **W.-M. Shen**** **P. Will****
*The Maersk Institute,
University of Southern Denmark,
Campusvej 55,
DK-5230 Odense M,
Denmark
kaspers@mip.sdu.dk

**Information Sciences Institute
University of Southern California
4676 Admiralty way,
Marina del Rey, CA 90292,
USA
{shen,will}@isi.edu

Abstract

In this paper we investigate the use of sensors in self-reconfigurable robots. We review several physically realized self-reconfigurable robots and conclude that little attention has been paid to the use of sensors. This is unfortunate since sensors can provide essential feedback that can be used to guide self-reconfiguration and control. In the systems that do use sensor feedback, the feedback is used locally on each module. However we identify a need in some situations to use sensor feedback globally. We therefore propose an approach where raw sensor values are abstracted and propagated to all modules. The sensor values are abstracted differently depending on the position of the producing sensor on the robot. We combine this approach with role based control, a control method for self-reconfigurable robots that we have developed earlier. We demonstrate that by combing these two approaches it is possible to make a self-reconfigurable robot consisting of six modules walk and avoid obstacles. However the reaction time of the robot is slow and therefore we discus possible ways of reducing the reaction time.

1 Introduction

In this paper we focus on self-reconfigurable robots built from a possibly large number of physically independent modules. The modules of these robots are able to connect and disconnect autonomously and can have sensors, actuators, a processor, a power source, and a communication system.

Self-reconfigurable robots have several advantages over traditional fixed shaped robots. 1) The modules can connect in many ways making it possible for the same robotic system to solve a range of tasks. This is useful in scenarios where it is undesirable to build a special purpose robot for each task. 2) Self-reconfigurable robots can adapt to the environment and change shape as needed. This could for instance be useful in a retrieval scenario where the robot has to snake its way through the rubble of a collapsed building and at some point change shape to recover an object from the rubble. 3) Since the robot is build from many independent modules it can be robust to module failures. If one module is defect it can be ejected from the system and the robot can still perform its task. 4) Self-reconfigurable robots are built from many identical modules. These modules can be mass produced and therefore the cost can be kept low despite their complexity. The advantages of self-reconfigurable robots can be summarized as: versatility, adaptability, robustness, and cheap production compared to the complexity and versatility of the resulting robot.

2 Research Challenges

In order to realize the potential of self-reconfigurable robots a number of research challenges have to met. There are interesting challenges to be met both in hardware and software.

2.1 Hardware Issues

In hardware some of the fundamental questions are: Does each module need computation power on-board? Where from do the modules get their energy? What kind of sensors are needed? What kind of communication system does the modules need? Several systems have been build to try to answer these questions and their properties are summarized in Table 1.

In Table 1 the systems are approximately sorted in order of increasing autonomy. It can be can seen from the table that very few of these systems are fully autonomous. This is not encouraging, because in order to realize the potential of self-reconfigurable robots it is important that they are autonomous. One important step toward achieving autonomy is to understand how to use sensors to make the robot able to sense and react to its environment. In this paper we take one small step toward understanding this.

Robot	CPU on-board	power on-board	sensors used for	Communication	Reference
JHU Hexagonal	yes[1]	no	n/a	n/a	(Pamecha et al., 1996)
JHU Rectangular	yes	no	n/a	n/a	(Pamecha et al., 1996)
MEL 3d unit	no	no	joint position	serial w. host	(Murata et al., 1998)
RIKEN vertical	no	yes	none	radio w. host	(Hosokawa et al., 1998)
PolyPod	yes	no	force/torque joint position	bus	(Yim, 1994)
Xerox PARC PolyBot	yes	no	joint position docking aid	CANbus	(Yim et al., 2000)
MEL 2000	yes[1]	no	none	serial bus with IDs[2]	(Murata et al., 2000)
MEL fractum	yes	no	none	inter-unit optical	(Murata et al., 1994)
Dartmouth molecule	yes[1]	no	not reported	serial w. host	(Kotay et al., 1998)
CMU ICES-cube	yes[1]	yes	joint position	serial w. host	(Ünsal and Khosla, 2000)
Dartmouth crystalline	yes	yes	joint position	sync. signal f. host	(Rus and Vona, 2000)
USC CONRO	yes	yes	docking aid	inter-unit optical	(Shen et al., 2000a)
CEBOT	yes	yes	docking & obst. avoid.	on-board	(Fukuda and Nakagawa, 1990)

Table 1: Overview of physically realized self-configurable robots.

2.2 Software Issues

There are two main categories of approaches to the control of self-reconfigurable robot: centralized control and distributed control.

In centralized control a central host dictates the actions of each module (Castano et al., 2000b). The advantage of this approach is that it is potentially easier for a controller to handle the complexity of the system when global knowledge is available. The disadvantage is that when the number of modules increase the host becomes the bottleneck. This scalability problem can be reduced by having modules do low-level control and having the central host coordinate the actions of the modules (Yim, 1994). The problem can also be side stepped by having a reconfiguration sequence computed off-line and afterward downloaded into the modules (Rus and Vona, 2001). The central host in this situation works as a conductor telling the modules how far they are in their action sequences. Calculating the reconfiguration sequence off-line unfortunately has the disadvantage that no adaptation can be made when the system is online. Furthermore systems based on centralized control are not robust since they all rely on a single host to function.

In order to address these problems distributed control has been used. Two classes of distributed control systems exist: synchronous and asynchronous distributed control systems. In synchronous control the strict control of the system is maintained, but the problem of robustness is removed. In synchronous control a distributed synchronization algorithm can be used to synchronize the actions of the modules connected in a low bandwidth network. Basically the synchronization signal that before came from a central host now is produced using a distributed synchronization algorithm. The hormone based algorithms developed by Shen, Salemi, and others are examples of this approach (Shen et al., 2000a, Shen et al., 2000b, Salemi et al., 2001). This solves the problem of robustness: there is no central host and the system still works even though a module is taken out. However insisting that all the modules should be strictly synchronized has a cost in terms of efficiency.

Another approach to distributed control is asynchronous distributed control. In synchronous control the modules were considered part of the whole. In asynchronous control each module is considered the whole. A module can be combined with more modules to make a bigger whole, but it in itself represents the whole. In this approach the asynchronous nature of the system is embraced and the idea of having a strictly controlled system is abandoned. The autonomy of each module is increased and the focus is on the local interaction between modules (Murata et al., 1994). The problem in asynchronous control is how to get coherent global behavior to emerge out of local interactions between many modules. The advantage of these systems

[1] but controlled off-board.
[2] local communication under development.

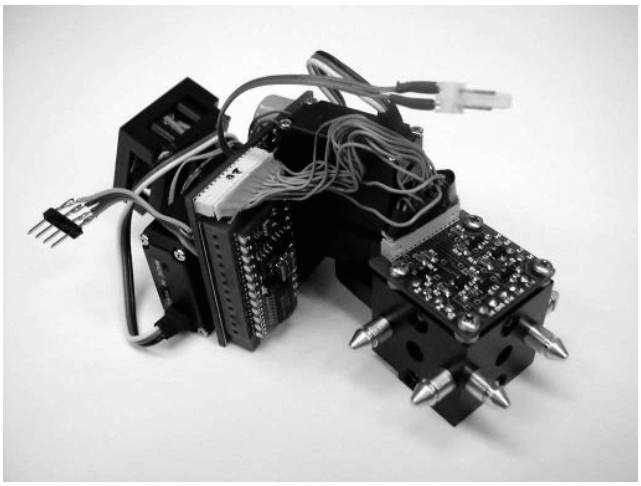

Figure 1: A CONRO module.

is that since all information is handled locally the systems scale. These methods take their inspiration from multi-agent systems (Bojinov et al., 2000b), and cellular automata (Hosokawa et al., 1998, Butler et al., 2001). Their similarities to minimalist collective robotics are also apparent see for instance (Beckers et al., 1994, Beckers et al., 2000, Støy, 2001).

In *role based control* which we will present in detail in Section 5 we combine some of the ideas from synchronous and asynchronous control. We acknowledge the need for each module to only use local information to insure scalability and robustness. However in situations where the modules are cooperating tightly, for instance to produce a locomotion gait, there is also a need to keep the modules synchronized. In role based control synchronization is achieved by having modules synchronize with neighbors from time to time. Over time this leads the entire system to be synchronized. This way the synchronization mechanism is decoupled from the control of the individual module and a robust, scalable, and synchronized system is the result.

3 Sensors and Self-Reconfigurable Robots

In research on sensor fusion there has been some work on how to combine and abstract sensor values into logical (Henderson and Shilcrat, 1984) and virtual sensors (Rowland and Nicholls, 1989). This work has been further extended with a commanding sensor type (Dekhil et al., 1996). The focus was on improving fault-tolerance of sensor systems and aiding development by making the sensor systems modular (Hardy and Ahmad, 1999). These ideas are relevant to the use of sensors in self-reconfigurable robots. However using sensors in self-reconfigurable robots is different because of the unique features of self-reconfigurable robots.

In a self-reconfigurable robot it can not be assumed that the position of the sensor is fixed. It can be moved through reconfiguration or maybe just by movements of the modules. This means that we need to understand how to extract meaningful sensor data from a network of sensors connected in time-varying ways. The previously proposed approaches also mainly deal with one consumer of the sensor date. If distributed control is employed there are many controllers that act on the sensor data. This means that system should be able to deal with inconsistent sensor data.

In distributed systems the problem of many decision makers can be dealt with in two ways. One way is to consider the modules independent and always handle sensor information locally. That is, on the module that receives the sensor input. Butler et al (Butler et al., 2001) have in simulation made a system where each module is a cellular automaton that reacts to its local configuration and surrounding obstacles. Using seven rules the modules are able to role over and across each other to produce "water-flow" like locomotion through an environment with obstacles. A similar idea was explored earlier on a real robot by Hosokawa et al (Hosokawa et al., 1998). Another approach explored by Bojinov et al (Bojinov et al., 2000a, Bojinov et al., 2000b) is to have the structure of the robot grow from seed modules. The growth is accomplished by having the seed module attract spare modules to a specific position with respect to the seed by using a virtual scent. When a spare module reaches that position the old seed module stops being a seed and the newly arrived module becomes the seed. The behavior of the seed module is controlled based on events it can sense in the environment. In these approaches the modules are decoupled in the sense that the modules only interact through stigmergy (Beckers et al., 1994).

In some systems the modules are highly coupled and sensor information can not always be handled locally: a sensor input might have effects in other modules than in the one in which it originated. This raises a fundamental questions which is the main focus of this paper: how do we distribute sensor information in order for it to arrive at the modules that need it? In this paper we present a system where sensor information is abstracted and propagated to all modules in the systems. Each module in the system then independently decides what action to take based on these propagated sensor values. Our use of sensors is inspired by the use of sensors in behavior based robotics (Arkin, 1998, Matarić, 1997) where sensors are used directly to control motors and not to build a geometrical model. We combine this communication system with role based control which we have developed earlier for the control of self-reconfigurable robots (Støy et al., 2002a, Støy et al., 2002b).

4 The CONRO module

Before describing this approach in more detail we will describe the CONRO self-reconfigurable robot. The CONRO modules were developed at University of Southern California's Information Sciences Institute (Castano et al., 2000a, Khoshnevis et al., 2001) (see figure 1). The modules are roughly shaped as rectangular boxes measuring 10cm x 4.5cm x 4.5cm and weigh 100grams. The modules have a female connector located at one end facing south and three male connectors located at the other end facing east, west, and north. Each connector has an 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. For the experiments reported here we also equipped the modules with flex sensors. The flex sensors are mounted on small circuit boards and Velcro is used to attach them to the modules. This mounting strategy is not very robust, but it is very flexible making it easy to experiment with different sensor morphologies. The flex sensors are 11cm long. Their resistance increase as they are bent and can therefore be used for rich tactile sensing. Refer to http://www.isi.edu/conro for more details and for videos of the experiments reported later in this paper.

5 Short Introduction to Role Based Control

In previous work we have introduced role based control. Role based control is a simple minimalist approach to the control of self-reconfigurable robots. We have shown earlier how this control method can be applied to chain and tree configurations to implement caterpillar like locomotion, locomotion similar to that of a sidewinding snake, and rolling track locomotion (Støy et al., 2002a). We have also used the method to make the CONRO robot configured as a hexapod and a quadruped robot walk (Støy et al., 2002b). Here we summarize role based control.

5.1 A Role

A role r consists of three components. The first component is a function $A(t)$ that specifies the joint angles of a module given an integer $t \epsilon [0:T]$. Where T is the period of the motion and the second component that needs to be specified. The third component is a set of delays D. A delay $d_i \epsilon D$ specifies the delay between the child connected to connector i and the parent. That is, if the parent is at step $t_{parent} = t_1$ the child is at $t_{child} = (T + t_1 - d_i)\, modulus\, T$. Below is some examples of roles used in the quadruped robot shown in Figure 4. The spine modules play the spine role:

$$A(t, spine) = \begin{cases} pitch(t) = 0^\circ \\ yaw(t) = 25^\circ \cos(\frac{2\pi}{T}t + \pi) \end{cases} \quad (1)$$

$$d_{east} = \frac{T}{4} \quad (2)$$

$$d_{south} = \frac{2T}{4} \quad (3)$$

$$d_{west} = \frac{3T}{4} \quad (4)$$

$$T = 180 \quad (5)$$

The legs play the forward role below or the backward role where t is replaced by $2\pi - t$ giving the same motion, but in the opposite direction.

$$A(t, forward) = \begin{cases} pitch(t) = 35^\circ \cos(\frac{2\pi}{T}t) - 55^\circ \\ yaw(t) = 40^\circ \sin(\frac{2\pi}{T}t) \end{cases} \quad (6)$$

$$T = 180 \quad (7)$$

5.2 Playing a Role

The algorithm that we now will describe is used to make a module play a role. However first some assumptions need to be made: a parent connector is specified and the remaining connectors are considered child connectors, connections can only be made between a parent connector and a child connector. Furthermore we assume that there are no loops in the configuration. These assumptions limit the configurations the algorithm can handle to tree configurations.

The algorithm has two components. One component makes sure that the actions are executed as specified in the role definition. This component also is responsible for synchronization with neighboring modules. The second component is discovering what role the module should play in case it can play more than one role. What role to play is discovered based on information propagated down from the parent and the local configuration.

The role playing component is visualized in Figure 2. The algorithm starts by setting $t = 0$ and continues to the main loop. Here the algorithm first checks if t is equal to the delay specified for each connector. In case t equals one of these delays d_i a signal is send through the corresponding child connector i. If the module has received a signal from its parent, t is reset. After that the joints are moved to the position described by $A(t)$. Finally t is incremented unless a period has been completed in which case t is reset and another iteration of the loop is initiated.

In some situations it is desirable for a module to be able to play different roles depending on its location in

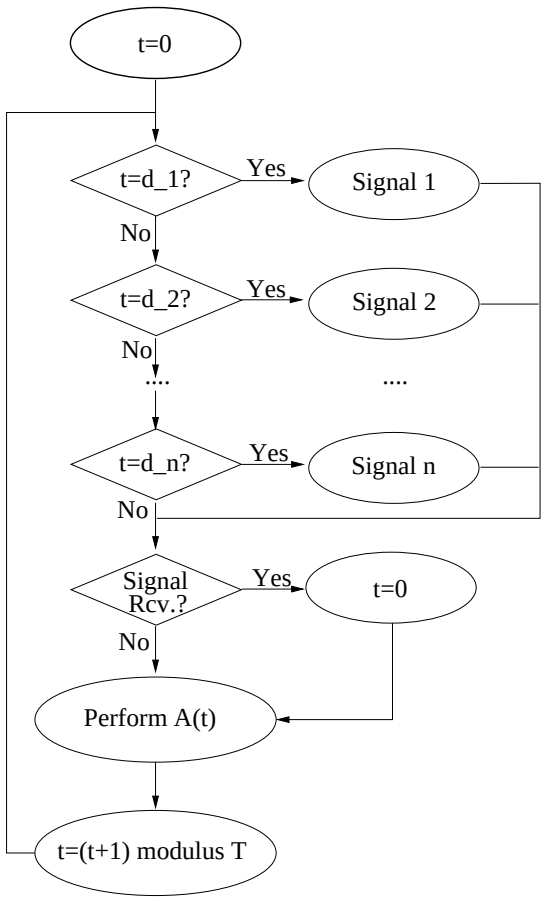

Figure 2: Visualization of the role playing part of the algorithm. See section 5.2 for an explanation.

the configuration tree. This is taken care of by the role selection component of the algorithm. The role can be selected based on the local configuration. Therefore every time a signal has successfully been sent through a child connector meaning that a child module is connected to that connector there is a check to see if the role should be changed because of that. In the basic algorithm the parent signaled the children to keep them synchronized the parent now sends a message which is a function of the parents role and the connector to which the child is connected. These two algorithmic components combined are shown in pseudo code in figure 3.

We have used this algorithm to make the CONRO self-reconfigurable robot walk. In the walker two modules are connected to form a spine. One module is connected on each side of the spine modules (see Figure 4). In this configuration the modules can play three different roles: east leg, west leg, and spine. A modules decide which role to play using the following rules: if communication to the sides (to the legs) is successful the module plays the spine role. It plays the role of a west leg if its parent is a spine module and it received the synchronization mes-

```
r = <start role>
t = 0
while(1)
  if (t=d(r)_1) then
    <send message M(r,1) to child connector 1>
    <update r>
  endif
  ...
  if (t=d(r)_n) then
    <send message M(r,n) to child connector n>
    <update r>
  endif

  if <message m received from parent connector> then
    t=0
    <update r based on m>
  endif

  <perform action A(r,t)>
  t = (t+1) modulus T(r)
endwhile
```

Figure 3: The algorithm used to play multiple roles. Refer to section 5.2 for further explanation.

sage through the west connector of the parent module. A module plays east leg if the synchronization message was send through the east connector.

Role based control is an example of an synchronous control method that does not insist on all the modules being synchronized at each step, but achieves this over time. This makes a role based system like the walker efficient because the modules work independently most of the time and only occasionally share information and synchronization information with neighboring modules. In this system all modules run identical programs and therefore modules can be interchanged and switch roles accordingly resulting in a very robust system. For more information on this system and role based control refer to (Støy et al., 2002b).

We have now summarized how role based control can be used to make the CONRO self-reconfigurable robot walk. However the system is open-looped in the sense that no sensor input from the environment is used in the control. Therefore we want to extend role based control to include sensor feedback. This is the subject of the following sections.

6 Role Based Control using Propagated Sensor Information

The CONRO self-reconfigurable robot is now configured into a quadruped robot as shown in Figure 4. Two flex sensors are attached to the front spine module and one is attached to each of the front legs. We now want to use feedback from these sensors to make the robot steer away from obstacles. In general the direction of loco-

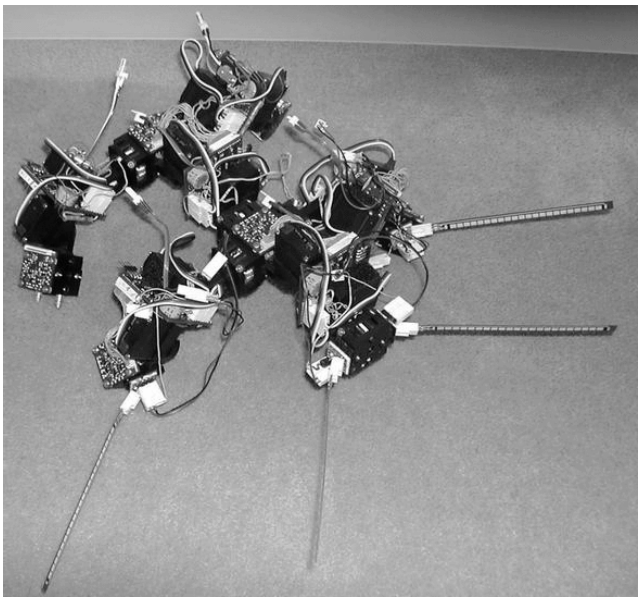

Figure 4: The CONRO robot in a walker configuration. The spine is made from two modules and the legs are made from one module each.

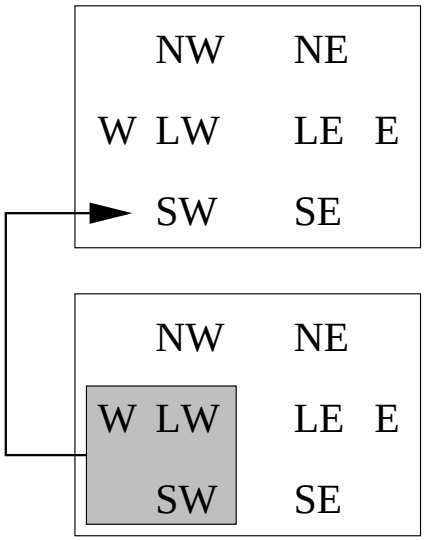

Figure 5: This figure shows how sensor values are propagated north from one spine module to the next. The south module (bottom) sums up the variables in the gray box and sends the sum to the north module (top). The north module receives this sum and write it in the variable indicated by the arrow.

motion can be changed in two ways: the motion of the legs can be biased so legs on the side pointing away from the obstacle take shorter steps and those on the other take longer steps. This approach will enable the robot to make soft turns away from obstacles. An alternative is to have legs on the side pointing away from the obstacle to move backwards and thus producing a turning on the spot motion. We found in initial experiments that the sensor based bias of the locomotion pattern does not produce a sharp enough turn to avoid obstacles. Therefore we decided to implement roles that make it possible for the robot to turn on the spot.

The goal is to make the quadruped robot turn on the spot away from an obstacle detected using one of the four flex sensors. Below we describe how this is achieved. A module has up to two flex sensors mounted: the front legs have one each, the front spine module has two, and the rest zero. The modules continuously sample these sensors and write the analog value into a local variable. What variable dependents on the position of the sensor. If the flex sensor is pointing toward the east the sensor value is written in a variable named local east (LE) and if it points west in local west (LW). If there are no sensors attached these variables contain zero. Each module has an additional six variables: northeast (NE), northwest (NW), east (E), west (W), southeast (SE), southwest (SW). These variables represent the sensor activity in the direction indicated by the names. For instance if all the west variables including local west are added up it will give the sum of the sensor activity on the west side of the robot. The same is true for the east values.

We will now describe how the sensor values are propagated in the system to produce the contents of the variables as it is described informally above. When a spine module sends sensor information to a module connected to its north connector it works as follows. The south module adds up the variables west, southwest, and local west and sends the sum to the north module. The sum is received by the north module and is written in the southwest variable. Note that this satisfy the invariant that the southwest variable of the north module now contains the sum of the sensor activity to the southwest. This mechanism is summarized in figure 5. At the same time the sum of the east variables is propagated and written in the southeast variable of the north module.

This mechanism will sum up the sensor inputs all along the spine and the northern most module will have information about the sensor activity to the southeast and southwest. However we want all the modules to have information of sensor activity in all directions therefore a similar, but independent mechanism is also propagating sensor activity southward. These two propagation mechanisms are summarized in figure 6.

All the spine modules now have information about sensor activity along the spine. For instance the modules can sum the northwest, local west, and southwest variables to find the sensor activity on the west side of the robot. This is not enough in our situation, because there are also two legs attached to the spine modules. Therefore sensor information should also be propagated from and to them. In our setup an east leg can only have one piece of sensor information that the spine does not have:

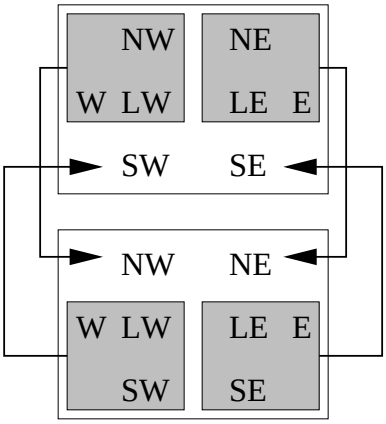

Figure 6: This figure shows how sensor values are exchanged between two spine modules. The modules sum the variables in the gray boxes and send these two values to the other module. The receiving module then writes these values in the variables as indicated by the arrows.

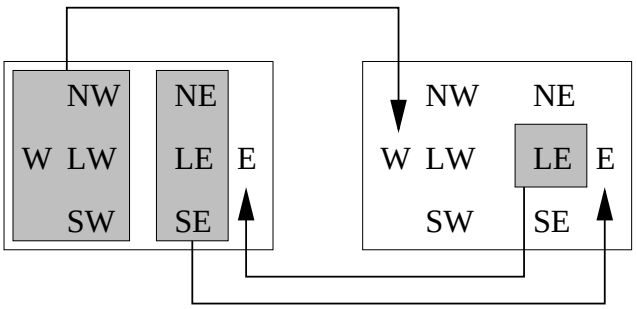

Figure 7: This figure show how sensor values are exchanged between a spine module (left) and an east leg module (right). The modules adds up the numbers in the gray boxes and send these values to the other module. The receiving module then write these values in the variables as as indicated by the arrows.

the value of the sensor connected to that leg. Therefore the local east value is propagated from the leg to the spine. The leg on the other hand receives the sum of all the sensor activity on the west side of the robot and writes that in the west variable. The sum of the variables northeast, local east, and southeast of the spine is written in the east variable of the leg. This is summarized in figure 7.

In role based control synchronization information from the parent module is sent to child modules each period of locomotion. The sensor information is also exchanged at this time as described above. How the sensor information is exchanged depends on what role the module plays. This is decided by the role playing algorithm. Therefore modules can still be exchanged and the system will continue to function if the sensors are placed correctly.

All the modules now have access to global sensor information and can make their decisions based on this information. In order to make a decision we sum the variables for the west and east side of the robot to have a measure of the activity on each side of the robot. A leg then decides to move backward if the sensor activity on the other side of the robot is above a small threshold and higher than the sensor activity on the leg's side. Otherwise it will move forward.

7 Results

First we will note some general properties of the system. One step of the robot corresponding to one period of locomotion takes two seconds. The step length is 15cm. Note that a step is quite long compared to the length of a module (10cm). The long steps are achieved by actively using the spine to make the steps longer. The robot achieves a speed of 7.5cm/second.

In four separate experiments the robot was placed so it approached an obstacle from four different angles. These experiments were videotaped using an overhead camera. We then manually analyzed the tape and for every two seconds recorded the position of the front end of the robot, the rear end, and whether a flex sensor was touching the obstacle. The results of this analysis can be seen in Figure 8.

The sensor values are exchanged when modules synchronize. We know the spine synchronizes with the east leg at $T/4$, the spine module to the south at $2T/4$, and the west leg at $3T/4$. We can use this information to calculate upper and lower bounds on communication delays. In the worst case where the sensor change happens just after synchronization it takes 2 periods to get sensor information from a front leg to the rear leg on the other side. In the best case where the sensor change happens just before synchronization it takes 1 period. This means that the whole system has a reaction time between two and four seconds or a reaction distance of 15cm to 30cm. Note that the reaction time is much better for the front legs. We can see these slow reaction times in Figure 8. The robot can only successfully avoid the obstacle when it approaches at an angle. In trial number four where the robot does not have time to react it bumps into the obstacle. This also explains why we decided to implement the turning on the spot behavior.

8 Discussion

If we look at how our approach can be used in general. We note that there are two things that make out system work. 1) The sensor data is abstracted based on the sensors position in a way that is useful for the receiving modules. 2) The abstracted sensor values are propagated at a constant slow rate to all modules.

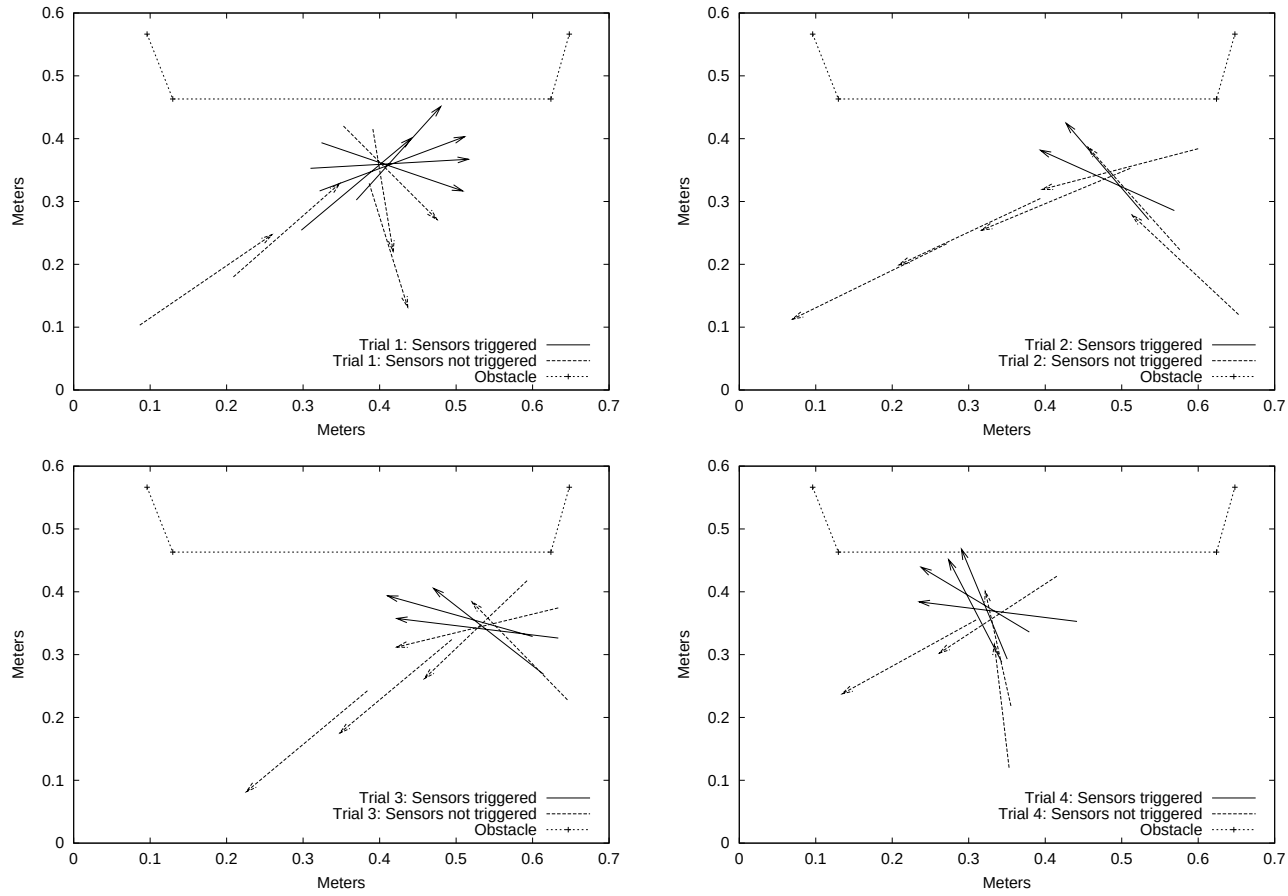

Figure 8: These figures show the robot approaching an obstacle, turning on the spot, and moving away. The arrows represent the positions of the front end and rear end of the robot recorded every 2seconds. The direction of the arrow shows the direction of movement. A solid arrow indicate that a flex sensor was triggered in that time step. A dashed that none were triggered.

In order to keep the amount of communication manageable we abstract sensor values to maintain an estimate in each module of the sensor activity to the left and right of the robot. It is possible for all the modules to agree on left and right, because of the properties of the CONRO hardware. The modules can only be connected in a tree structure (with one loop) and be connected in four ways and therefore the transformation of direction from module to module is easy. In general it seems to be important to have some relative position information about the sensor with respect to the acting module. This means that in systems where the relative position of two connected modules can be found it is possible to abstract the sensor information in a useful way. This is also what makes this approach different from previous work on sensor fusion. The position of the sensor is variable and this is in our case handled by the abstraction mechanism.

Sensor values flow around in our system at a constant slow rate. This rate could be increased significantly to reduce the reaction time. The problem of doing this is that our modules have limited resources and therefore if time is spent on communication less time can be spent on control of the motors resulting in a decrease in speed. Therefore in order to decrease the reaction time of the system without sacrificing speed we need to use less communication and achieve a shorter reaction time at the same time. One solution would be to have the module monitor a sensor and if it goes above a certain threshold it can be propagated. When the sensor later drops below the threshold another message can be propagated. This might improve the response time of the system, because when communication takes place only when it is needed it can be made efficient.

Another orthogonal way to decrease the amount of communication would be to only propagate sensor information to the modules that need it. For instance in the walker sensor information from one side could be propagated to the other side. In this way the sensors on the left control the legs on the right and the other way around. In general directed diffusion (Intanagonwiwat et al., 2000) could be used for this. In directed diffusion information is propagated from the producer to the consumer through networks with a time varying configuration. In this framework it is possible

for the consumer to show his interest in a specific kind of data and have that routed to it from the producer.

9 Summary

In a self-reconfigurable robot sensors can be used in two ways. One way is to use the sensor information locally: sensors can be used to bias the motions of modules to produce the desired global behavior. The advantage of this is efficiency since the sensor values do not need to be communicated anywhere.

In some situations it is not possible to handle all sensor information locally. Therefore another way to handle sensor information has been presented. We have experimented with an approach where sensor information is abstracted based on the sensors position on the robot. This abstracted sensor information then flows from the modules that produce the information to all the modules of the system. An important aspect of this system is that the acting modules have abstract information about the position where the sensor value originated. We have shown that by combining this communication mechanism with role based control we were able to make a six module self-reconfigurable robot walk and avoid an obstacle.

Acknowledgments

This research is funded 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

Arkin, R. (1998). *Behaviour-Based Robotics*. MIT Press.

Beckers, R., Holland, O., and Deneubourg, J. (1994). From action to global task: Stigmergy and collective robotics. In *Proceedings of Artificial Life 4*, pages 181–189, Cambridge, Massachusetts, USA.

Beckers, R., Holland, O., and Deneubourg, J.-L. (2000). From local actions to global tasks: Stigmergy and collective robotics. *Prerational intelligence: Adaptive behavior and intelligent systems without symbols and logic*, 2:549–563.

Bojinov, H., Casal, A., and Hogg, T. (2000a). Emergent structures in modular self-reconfigurable robots. In *Proceedings of the IEEE int. conf. on Robotics & Automation*, volume 2, pages 1734 –1741, San Francisco, California, USA.

Bojinov, H., Casal, A., and Hogg, T. (2000b). Multiagent control of self-reconfigurable robots. In *Proceedings of the Fourth int. conf. on MultiAgent Systems*, pages 143 –150, Boston, Massachusetts, USA.

Butler, Z., Kotay, K., Rus, D., and Tomita, K. (2001). Cellular automata for decentralized control of self-reconfigurable robots. In *ICRA 2001 Workshop on Modular Self-Reconfigurable Robots*, Seoul, Korea.

Castano, A., Chokkalingam, R., and Will, P. (2000a). Autonomous and self-sufficient conro modules for reconfigurable robots. In *Proceedings of the 5th int. Symposion on Distributed Autonomous Robotic Systems*, pages 155–164, Knoxville, Texas, USA.

Castano, A., Shen, W.-M., and Will, P. (2000b). Conro: Towards deployable robots with inter-robot metamorphic capabilities. *Autonomous Robots*, 8(3):309–324.

Dekhil, M., Sobh, T., and Efros, A. (1996). Commanding sensors and controlling indoor autonomous mobile robots. In *Proceedings of the 1996 IEEE International Conference on Control Applications*, pages 199 –204, Dearborn, Michigan, USA.

Fukuda, T. and Nakagawa, S. (1990). Method of autonomous approach, docking and detaching between cells for dynamically reconfigurable robotic system cebot. *JSME int. Journal*, 33(2):263–268.

Hardy, N. and Ahmad, A. (1999). De-coupling for reuse in design and implementation using virtual sensors. *Autonomous Robots*, 6:265–280.

Henderson, T. and Shilcrat, E. (1984). Logical sensor systems. *Robotic Systems*, 1(2):169–193.

Hosokawa, K., Tsujimori, T., Fujii, T., Kaetsu, H., Asama, H., Kuroda, Y., and Endo, I. (1998). Self-organizing collective robots with morphogenesis in a vertical plane. In *Proceedings of the IEEE int. conf. on Robotics & Automation*, pages 2858–2863, Leuven, Belgium.

Intanagonwiwat, C., Govindan, R., and Estrin, D. (2000). Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00)*, pages 56–67, Boston, Massachussetts, USA.

Khoshnevis, B., Kovac, B., Shen, W.-M., and Will, P. (2001). Reconnectable joints for self-reconfigurable robots. In *Proceedings of the IEEE/RSJ int. conf. on Intelligent Robots and Systems*, Maui, Hawaii, USA.

Kotay, K., Rus, D., Vona, M., and McGray, C. (1998). The self-reconfiguring robotic molecule. In *Proceedings of the IEEE int. conf. on Robotics & Automation*, pages 424–431, Leuven, Belgium.

Matarić, M. J. (1997). Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2–3):323–336.

Murata, S., Kurokawa, H., and Kokaji, S. (1994). Self-assembling machine. In *Proceedings of the IEEE int. conf. on Robotics & Automation*, pages 441–448, San Diego, USA.

Murata, S., Kurokawa, H., Yoshida, E., Tomita, K., and Kokaji, S. (1998). A 3-d self-reconfigurable structure. In *Proceedings of the IEEE int. conf. on Robotics & Automation*, pages 432–439, Leuven, Belgium.

Murata, S., Yoshida, E., Tomita, K., Kurokawa, H., Kamimura, A., and Kokaji, S. (2000). Hardware design of modular robotic system. In *Proceedings of the IEEE/RSJ int. conf. on Intelligent Robots and Systems*, pages 2210–2217, Takamatsu, Japan.

Pamecha, A., Chiang, C., Stein, D., and Chirikjian, G. (1996). Design and implementation of metamorphic robots. In *Proceedings of the ASME Design Engineering Technical conf. and Computers in Engineering conf.*, pages 1–10, Irvine, USA.

Rowland, J. and Nicholls, H. (1989). A modular approach to sensor integration in robotic assembly. In Puente, E. and Nemes, L., (Eds.), *Information control problems in Manufacturing Technology*, pages 371–376. IFAC, Pergamon Press, Oxford, UK.

Rus, D. and Vona, M. (2000). A physical implementation of the crystalline robot. In *Proceedings of the IEEE int. conf. on Robotics & Automation*, pages 1726–1733, San Francisco, USA.

Rus, D. and Vona, M. (2001). Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1):107–124.

Salemi, B., Shen, W., and Will, P. (2001). Hormone controlled metamorphic robots. In *Proceedings of the IEEE int. conf. on Robotics & Automation*, pages 4194–4199, Seoul, Korea.

Shen, W.-M., Salemi, B., and Will, P. (2000a). Hormone-based control for self-reconfigurable robots. In *Proceedings of the int. conf. on Autonomous Agents*, pages 1–8, Barcelona, Spain.

Shen, W.-M., Salemi, B., and Will, P. (2000b). Hormones for self-reconfigurable robots. In *Proceedings of the int. conf. on Intelligent Autonomous Systems*, pages 918–925, Venice, Italy.

Støy, K. (2001). Using situated communication in distributed autonomous mobile robots. In *Proceedings of the 7th Scandinavian conf. on Artificial Intelligence*, Odense, Denmark.

Støy, K., Shen, W.-M., and Will, P. (2002a). Global locomotion from local interaction in self-reconfigurable robots. In *Proceedings of the 7th int. conf. on Intelligent Autonomous Systems IAS-7 (to appear)*, Marina del Rey, California, USA.

Støy, K., Shen, W.-M., and Will, P. (2002b). How to make a self-reconfigurable robot run. In *Proceedings of the First International Joint Conference on Autonomous Agents & Multiagent Systems (to appear)*, Bologna, Italy.

Ünsal, C. and Khosla, P. (2000). Mechatronic design of a modular self-reconfiguring robotic system. In *Proceedings of the IEEE int. conf. on Robotics & Automation*, pages 1742–1747, San Francisco, USA.

Yim, M. (1994). New locomotion gaits. In *Proceedings of int. conf. on Robotics & Automation*, pages 2508 –2514, San Diego, California, USA.

Yim, M., Duff, D., and Roufas, K. (2000). Polybot: A modular reconfigurable robot. In *Proceedings of the IEEE int. conf. on Robotics & Automation*, pages 514–520, San Francisco, USA.