"Deformable Wheel"-A Self-Recovering Modular Rolling Track

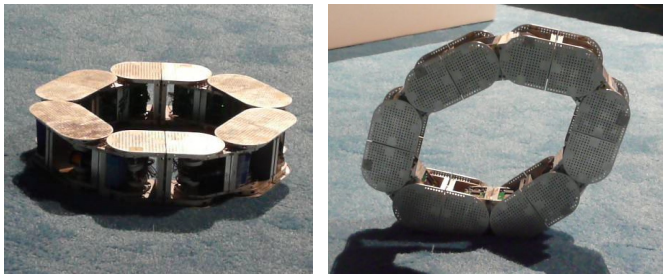
Harris Chi Ho Chiu, Michael Rubenstein, and Wei-Min Shen

Information Sciences Institute,
The University of Southern California,
4676 Admiralty Way, Marina Del Rey, CA 90292
{chichiu,mrubenst}@usc.edu, shen@isi.edu

The rolling track is an effective modular robot configuration with high maneuverability. However, one technical barrier that prevents it from practical usage is that most existing rolling track robots must start from a typical stand-up position and they are also difficult to turn while rolling. This paper presents a solution for these problems. By extending our previous work, we have developed a set of new gaits for the rolling track to self-standup, roll and turn. The combination of these behaviors on a single SuperBot rolling track has enabled it to roll from any initial conditions, steer while rolling, and recover from falling sideways. These new gaits have been demonstrated with 6 modules in hardware and 8 and 10 modules in simulation. In addition, the new gaits can be activated even reconfiguring from a snake configuration.

1 Introduction

Unlike a wheel, a modular rolling track propels forward by actively changing its shape. A common problem experienced by the wheel and rolling track is the inability to perform *self-recovery*, meaning that it cannot stand up without any external help once it has fallen sideways. An example would be a rolling track recovering from a flattened orientation as in Figure 1(a) to a stand-up posture as in Figure 1(b). A rolling track can be deemed "complete" provided it is able to self-recover, roll and turn such that the robot is able to traverse the environment. However, current research has been limited to demonstration of rolling behavior in modules with different mechanical designs and analysis of rolling gaits. Matsuda et. al.[3]controls the stiffness of the joints between modules to change shape for rolling forward. Other methods involved rotary or prismatic joints change their angles based on time [9][12] or sensor feedback[11][2][5]. In terms of speed analysis of rolling gaits, Sastra et. al[5] presented an analysis on sensor-based rolling and recorded a fast speed of rolling (1.4m/s) on CKBot. These controls are required to start from



(a) Fall sideways (b) Stand-up posture

Fig. 1. 6-Module SuperBot Rolling Track

a particular stand-up posture. Notice that their rolling motion is restricted to a single plane. The degrees of freedom and the strength of actuators only contribute to motion along the direction of travel. To perform turning and self-recovery, motions along a perpendicular plane is essential. An extra degree of freedom along a perpendicular plan allows the rolling track to steer to a different direction during rolling and lifting adjacent modules upward for self-recovery. In particular to self-recovery, the actuators providing the extra degree of freedom have to be strong enough to lift up adjacent modules while avoiding tipping over during self-recovery process. Previous work by Yim[12] has simulated rolling and turning while Shen[8] has demonstrated self-recovery in simulation. However, a control algorithm containing self-recovery together with turning and rolling has neither been proposed nor tested in hardware. The work presented here contributes by providing the control method for self-recovery, turning and finally combines these with rolling to form a "complete" rolling track demonstrated for the first time on a modular robot. The control method is scalable in number of modules and adaptable to topology changes in modular robots.

The paper is organized as follows: Section 2 introduces our hardware platform. Section 3 presents a rolling track detects its loop configuration. Section 4 presents the control used for self-recovery. In Section 5, we introduced a algorithm for turning built on top of rolling motion. Section 6 shows an integration of self-recovery, turning and rolling together with configuration detection. Experiment results are shown in Section 7. Section 7.1 demonstrates the "complete" integration using a remote controlled 6-module rolling track for rolling, turning and self-recovering. In Section 7.2, rolling tracks of size 6, 8, and 10-module have been tested. Self-recovery and rolling behaviors are shown after the loop of rolling track is completed from a chain (snake) configuration in Section 7.3. Section 8 concludes the paper with discussions and future research directions.

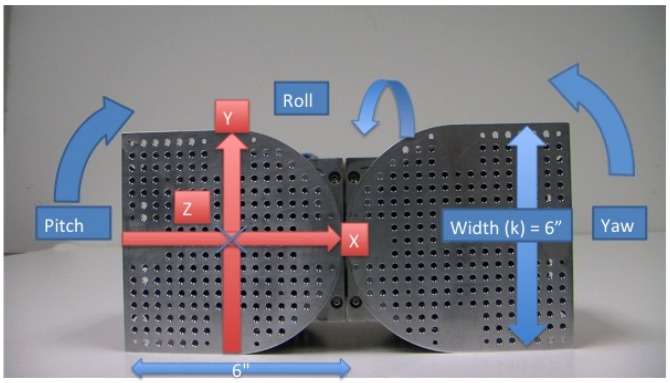


Fig. 2. A single module with local coordinate frame for gravity vector

Table 1. Comparison of current rolling track implementations

	d.o.f.	extra d.o.f. not for rolling motion	Turning	Self-Recovery
SuperBot[4]	3	2	Yes	Yes
PolyPod[12]	2	1(limited range)	Yes	No
CONRO[9]	2	1	Yes	No
MTRAN[2]	2	1(connected differently)	Maybe	Maybe
CKBot[5]	1	1(connected differently)	Maybe	No
5RChain[11]	1	0	No	No
BIYOn[3]	1	0	No	No

2 Constructing the Rolling Track

Table 1 shows the comparison of current implementations of rolling track using different modular robots. Only some are capable to do turning and most of them are unlikely to do self-recovery. PolyPod is limited by its joint angle range for self-recovery. MTRAN and CKBot are possible to turn only if they are connected differently with a 90° rotation to adjacent module. However, it requires 12 or more modules to form a feasible rolling track and requires each motor to provide enough torque to lift 3.5 times of module weight. Such configuration of rolling track might not roll in practice. SuperBot modules[4] are used as our experiment platform for its flexibility of 3-degree-of-freedom (d.o.f) and its ability to load two neighboring modules in each degree of freedom. Figure 2 shows a SuperBot module and its 3 d.o.f. (Pitch, Roll, Yaw) are highlighted. Each module also equips with a build-in 3D accelerometer for detection of its own orientation by knowing the gravity vector relative to its local coordinate frame. Therefore, SuperBot is a favorable choice to implement "complete" behaviors of rolling track.

Table 2. RULEBASED Table for Configuration Detection

Type	Recv Hormone	Action	Send Hormone
FT & BK	NOTALOOP	Reset timeout	forward hormone
BK	n/a	n/a	(NOTALOOP, ranNum)
FT	NOTALOOP	n/a	n/a

Each connection type sets $isInLoop{=}false$ upon receiving $NOTALOOP$ hormone

Our implementation connects 6 SuperBot modules together with the $FRONT$ interface connected to the $BACK$ interface of another module to form a loop. There are total 6 interfaces namely, $FRONT$, $BACK$, $LEFT$, $RIGHT$, UP and $DOWN$. Each interface can communicate to a connected neighbor through Infra-Red sensors. Modules are either connected autonomously through genderless connectors [7] or manually mounted with screws. As shown in Figure 2, each SuperBot module is of dimension $\{length, width, height\} = \{2k, k, k\}$, where $k = 6$ inches. For the rest of the paper, the joint configuration $pitch, roll, yaw$ of SuperBot shown in Figure 2 has a value of $\{0, 90, 0\}$ and is treated as default. Positive *pitch* indicates a clockwise rotation while positive *Yaw* indicates anti-clockwise rotation. For *Roll* axis, the joint configuration becomes $\{0, 180, 0\}$ if the right half of the module is rotated 90° anticlockwisely about x-axis. In the 6-module rolling track, the joint angles of all 6 modules are $\{30, 90, 30\}$. For describing the use of gravity vector, we define the local coordinate frame as show in Figure 2 with z-axis pointing inward. We also have directly adopted the asynchronous algorithm described in section 4.4.1 by Itai et. al.[1] to determine the size of rolling track before the use of any control algorithms.

3 Configuration Detection

Modular robots can formed into various topologies by connecting other modules in different ways. In previous work, loop topology is always assumed and motions for rolling track are designed based on this assumption. Therefore, it is essential for modules knowing whether they are a part of a loop before running any controls on a rolling track and detect topology changes in case of reconfiguration or module failure. We extend the identifier-free Adaptive Communication (AC) protocol by Shen et. al.[10] to detect loop formation.

Each module discovers its connection type using AC protocol by listening to probe messages periodically sent from neigboring modules. For example, the connection type of a module is *FRONT (FT)* if it has received probe messages only from its $FRONT$ interface. Similarly, the connector type is FT and *BACK (BK)* if probe messages are received from its $FRONT$ and $BACK$ interfaces. Each module looks up a pre-programmed $RULEBASED$ table (Table 2) to act accordingly based on its connection type. For BK connection type, the

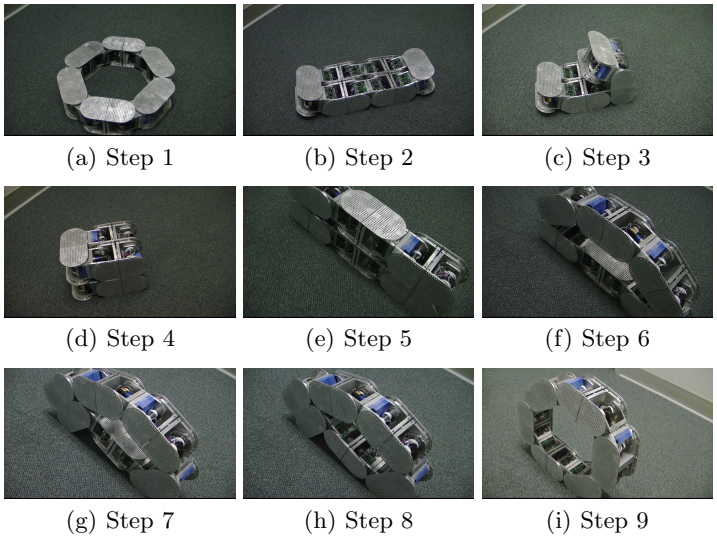
Table 3. Pre-programmed table of joint angle {pitch, roll, yaw} in degree for self-recovery. (Values changed from previous step are in bold)

	hopCount 0	*hopCount* 1	*hopCount* 2	*hopCount* 3	*hopCount* 4	*hopCount* 5
Step 1	{30,90,30}	{30,90,30}	{30,90,30}	{30, 90, 30}	{30, 90, 30}	{30, 90, 30}
Step 2	{**0**,90,**0**}	{**20**,**0**,**90**}	{**90**,**180**,**20**}	{**0**, 90, **0**}	{**20**,**0**,**90**}	{**90**,**180**,**20**}
Step 3	{0,90,**90**}	{**55**, 0,90}	{90,180,**55**}	{**90**, 90, 0}	{20, 0, 90}	{90,180, 20}
Step 4	{**90**,90,90}	{**0**,0,90}	{90,180,**0**}	{90, 90,**90**}	{**0**, 0,90}	{90,180,**0**}
Step 5	{90,90,90}	{0,0,**0**}	{**0**,180, 0}	{**0**, 90,**0**}	{ 0, 0,**0**}	{**0**,180, 0}
Step 6	{90,90,**55**}	{**35**,**90**,**35**}	{**35**,**90**,**35**}	{**55**, 90,**-90**}	{ 0, 0, 0}	{ 0,180, 0}
Step 7	{**65**,90,55}	{35,90,35}	{35,90,35}	{**90**, 90,**90**}	{**-70**, 0, 0}	{ 0,180,**55**}
Step 8	{65,90,55}	{35,90,35}	{35,90,35}	{90, 90, 90}	{-70,**90**, 0}	{ 0 ,**90**, 55}
Step 9	{**30**,90,**30**}	{**30**,90,**30**}	{**30**,90,**30**}	{**30**, 90, **30**}	{**30**, 90,**30**}	{**30**, 90,**30**}

module generates a *NOTALOOP* hormone message embeded with a random number sends to its *BK* connector. The use of random number is to prevent looping of the hormone message upon the loop is completed. The message will then continue to propagate through module with connection type *FT* and *BK*. If the loop is open, the hormone message will terminate at the one with *FT*. If the loop is closed, no *NOTALOOP* message will be generated as there is no *BK* connection type. To handle topology changes, each module also has a countdown timeout variable *notInLoopTimeout*. If a module is in *FT* and *BK* and no *NOTALOOP* hormone message has been receieved in the timeout period, the module can declare it as part of a loop and a flag *isInLoop* = *true*. If a *NOTALOOP* hormone message has been received, *notInLoopTimeout* will be reset and *isInLoop* = *false*. This timeout mechanism adds flexibility to loop detection as *isInLoop* will not be made permanent and dynamically follows topology changes.

4 Self-Recovery Control

The self-recovery reconfiguration procedure requires synchronization between modules that can be achieved using leader-based control. In a loop formation, leader is elected dynamically through probabilistic leader election[1]. Each module contains the same pre-programmed look-up table of joint angles(See Table 3) and therefore every module has to know the total number of module in advance. Angles can be looked up based on current reconfiguration *step* and *hopCount* from the leader in the recovery sequence. Leader starts with sending out a message *"step=0, hopCount=0"*. Each module passes the modified message *"step, hopCount+1"* to the next module. Module can look up the joint angles in the table with corresponding *step* and *hopCount*. The control terminates when the leader receives a message with *hopCount* equal to the number of module at the final *step*.



(a) Step 1 (b) Step 2 (c) Step 3

(d) Step 4 (e) Step 5 (f) Step 6

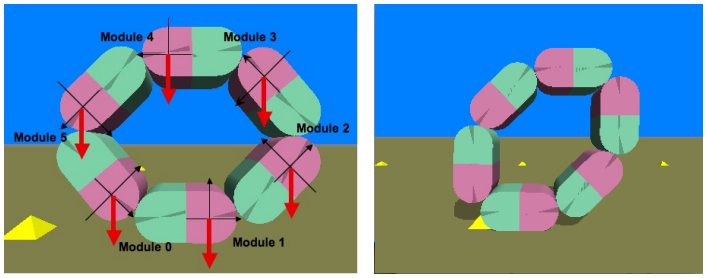
(g) Step 7 (h) Step 8 (i) Step 9

Fig. 3. A 6-module SuperBot rolling track recovers from falling down

The self-recovery is separated into 3 phases - *Initiate*, *Fold-Up-and-Expand*, and *Resume*. In the 6-Module Superbot rolling track configuration (6M-Loop), the rolling track first initializes to be a regular hexagon shape as seen in Step 1 of Figure 3. With the *Initiate* phase, the subsequent steps can always remain the same regardless of any hexagon shape it is previously. Step 2 to 5 are of *Fold-Up-and-Expand* phase. The aim is to use 3 degrees of freedom (*Pitch, Yaw, Roll*) to change from configuration having dimension {*length, width, height*} - {*6k,2k,k*} in step 2 to {*6k,k,2k*} in step 5. Three modules are lifted up in step 3 and the rolling track is folded as a block of size {*3k,2k,2k*} in step 4 and spanning horizontally in step 5. In the *Resume* phase, the rolling track has to "unwind" the twist at roll axis of 4 modules done in step 2. The "unwind" can be refered to the change of roll joint angle in step 6 for $hopCount = 1, 2$ and in step 8 for $hopCount = 4, 5$ in Table 3. It then resumes regular hexagon configuration in step 9. Table 3 shows the look-up table used in each module of 6-Module Superbot rolling track to perform recovery.

5 Turning while Rolling

Turning a rolling track into a different direction requires steering during rolling motion. The steering has to be performed just right before the module lands on the ground or it will discontinue the rolling motion or it causes the rolling track to fall sideways. Therefore, the design of control for rolling has to have turning in mind.



(a) Gravity vector relative to the local frame

(b) Rolling track changed shape for moving center of gravity

Fig. 4. 6-module rolling track changes its center of gravity

We extend our previous work [8] on dynamic rolling by adding turning to the rolling control. Figure 4 shows a simulation of 6-module Superbot rolling track (6M-Loop) rolls forward by changing its shape to move the center of gravity forward. Every time a rolling track has a horizontal orientation as in Figure 4(a), it changes the shape to a squeezed hexagon shape as shown in Figure 4(b). The unbalance causes rotation about the bottom module and thus the rolling track tips over and rolls forward. The motions repeat in cycle triggered by the orientation of modules. Turning can be done with a slight "twist" at the roll joint before a module touching the ground.

Our implementation is based on reactive control mapping the orientation of each module to a set of joint angle values *(Pitch, Roll, Yaw)*. We consider the case for n number of modules, for each module i, accelerometer values are directly mapped to joint angles as in (1). For a polygon-shaped rolling track in a up-right posture, each module will have unique orientation and hence different range of accelerometer values.

$$motor\_angles_i = f(acc\_values\_range) \tag{1}$$

In relating shape to joint angle, we consider n is even and denote the set of modules to be $module_0$, $module_1$,..., $module_{n-1}$ such that $module_i$ś $BACK$ connector is connected to $module_{(i+1)\%n}$ś $FRONT$ connector. For any $module_i$, its joint angles are denoted as $(pitch_i, roll_i, yaw_i)$. In a n-sided polygon shape, each internal angle of the polygon is governed by $\alpha_i = 180° - (pitch_i + yaw_i)$. If the internal angle of the tip of the squeezed n-sided polygon is θ, then for n-Module closed-loop configuration, the joint angles become:

For all $i = n/2 - 1$ *or* $n - 1$,

$$(pitch_i, roll_i, yaw_i) = ((180° - \theta)/2, 90°, (180° - \theta)/2) \tag{2}$$

For all other i,

$$(pitch_i, roll_i, yaw_i) = (\theta/(n-2), 90°, \theta/(n-2)) \tag{3}$$

To have the rolling track in squeezed polygon shape realizing its orientation as in Figure 4(a), values have to be obtained from the accelerometers so that a new cycle of shape changing can be invoked. To simplify the procedure in obtaining values for every module, we can measure the orientation - angle of gravity vector to local coordinate frame. For $module_k$, if angle β_k is the gravity vector relative to its local frame, the angle of gravity vector for adjacent module can be obtained using,

$$\beta_{(k+1)\%n} = \beta_k - \alpha_k \tag{4}$$

An example is shown in Figure 4(a) with $n = 6$. Red arrows represent gravity vector and same as Figure 2, black arrows represents local coordinate frame. If we have obtained angle of one red arrow to its local frame in $module_k$, the angle of red arrow of $module_{k+1}$ can be calculated by subtracting its internal angle α_k. Then, the mapping from (1) can be done by translating accelerometer values into angle of gravity vector to its local frame and corresponding joint angle can be obtained by using (2) and (3) with small tolerance on the angle of gravity vector index k for $module_k$ is incremented anticlockwisely and position dependent. That means any module in the same orientation will be having the same index and index will always start from left module first touching the ground as show in Figure 4(a).

Turning of the rolling track can be achieved by changing the roll joint of $module_1$ and $module_2$ when they are about to touch the ground. By changing their roll angle $roll_1 = 90 - \gamma$ and $roll_2 = 90 + \gamma$ for some γ, the rolling track will be able to turn. To counter balance the upper part of the rolling track, module in diagonal position can be set to have roll joint angle in counter direction $roll_{n/2+1} = 90 + \gamma$ and $roll_{n/2+2} = 90 - \gamma$. Experiment results of rolling and turning are presented in Section 7.

6 Integration of "Complete" Behaviors on a Rolling Track

To have "complete" behaviors on rolling track, controls for self-recovery, turning, rolling and also configuration detection have to be coordinated properly. Pseudocode 1 shows how the behaviors are intergrated by mode switching. Operation modes including rolling mode (*ROLL_MODE*), turning mode (*TURN_MODE*), self-recovery mode (*RECOVERY_MODE*) and idle mode (*IDLE_MODE*) are switched based on current topology, orientation and remote commands. The rolling mode is activated after a loop topology is realized through configuration detection in Section 3. Rolling mode and turning mode can be switch interchangeably using remote command. In any case the rolling track fall sideways, self-recovery mode is activated and recovery sequences will be carried out. At the end of the sequence, the rolling track will be switched back to rolling mode. In case of any topology changes, if rolling track are no longer in a loop, it will fall back to idle mode.

Pseudocode 1 Mode Switching in Distributed Control for Rolling Track

```
mode = IDLE_MODE
Loop:
 configurationDetection()//isInLoop= true if the module is in a loop
 if(isInLoop)
   switch(mode)
     case ROLL_MODE:
       if(orientation == fall sideways)
         mode = RECOVERY_MODE
       if(remoteCommand == TURN)
         mode = TURN_MODE
     case TURN_MODE:
       if(orientation == fall sideways)
         mode = RECOVERY_MODE
       if(remoteCommand == ROLL)
         mode = ROLL_MODE
     case RECOVERY_MODE:
       if(RecoveryStep >= LAST_STEP)
         mode = ROLL_MODE
     case IDLE_MODE:
       detectLoopSize();
       mode = ROLL_MODE
 else
    mode = IDLE_MODE
EndLoop
```

(a) Rolling track turns right to the door way (b) Rolling track is made flattened (c) Rolling track resumes rolling after self-recovery

Fig. 5. Snapshots of the experiment performing self-recovery, turning and rolling behaviors

7 Experimental Results

7.1 Remotely Controllable Rolling Track

Rolling, turning and self-recovery are implemented on a remote controllable 6-module SuperBot rolling track to show "complete" behaviors. In the implementation, the speed of rolling during turning is decreased by reducing the strength of the motor to lower the possibility to fall sideways. We discover

(a) 6-module track (b) 8-module track (c) 10-module track

Fig. 6. SuperBot rolling track of size 6, 8 and 10 in simulation

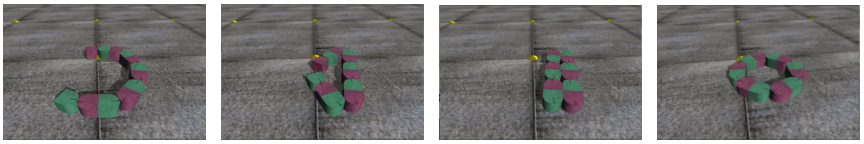
that the transition from forward rolling gait to a turning gait failed occasionally if the switch command is not issued remotely in time during falling forward. Then, a scenario of the "complete" behaviors has been carried out without turning off the power or reloading the program. At the beginning, the rolling tracks rolls forward at the same speed (about 0.35m/s) as reported in our previous work[6]. The robot is then placed flatten down and recovered successfully by carrying out the recovery steps every 3 seconds as shown in Figure 3. The rolling track starts rolling and it is commanded to turn using remote control. Figure 5 shows the self-recovering, turning, and rolling motion during the experiment. Video of the experiment can be found at the following website: http://www.isi.edu/robots/superbot/movies/rcRollingTrack.avi

7.2 Scalability for Rolling and Turning

The proposed control of rolling and turning is scalable in number of modules. Each module selects its joint angle based on its accelerometer values from its own unique orientation in a polygon, therefore, identifier is not required. The control also requires no message exchange avoiding hop delay issue. Rolling track of size 6-module, 8-module and 10-module are demonstrated in simulation. They are implemented in SuperBot simulation using Open Dynamic Engine. Control programs have been loaded into simulated modules without any modifications to the control program. As shown in Figure 6, the rolling tracks of different size are able to detect its configuration and turn while rolling. The experiment suggests the algorithm can support a higher number of modules if the joints are strong enough to support its load for rolling motion. Video of 6-module, 8-module, 10-module rolling track simulation can be viewed at: http://www.isi.edu/robots/superbot/movies/rtSimRolling.avi (or rtSimRolling.swf for faster download)

7.3 Reconfiguration Experiment in Simulation

As SuperBot is designed to be self-reconfigurable, we would like to test the future adaptability of the control to loop formation. In Figure 7, a pre-programmed reconfiguration procedure is implemented to form a 6-module



(a) initial 6-module snake (b) intermediate step 1 (c) intermediate step 2 (d) 6M-Loop

Fig. 7. Self-Reconfiguration from a 6-module snake to 6M-Loop rolling track

rolling track from a 6-module snake configuration in simulation. Every module starts with same control algorithm. Upon topology changed to a loop, the loop is detected and the size of loop is calculated by the loop verification algorithm proposed in Section 3 and self-recovery is then activated once fall-down orientation is detected. The mode switching in Section 6. It starts to roll forward again. Video of simulation on a 6-module docking, fall-down recovery and dynamic rolling can be viewed at:

http://www.isi.edu/robots/superbot/movies/rtSimReconfig.avi (or rtSimReconfig.swf for faster download)

8 Conclusion and Future Work

This paper addressed the problem of self-recovery of a rolling track and documented the first implementation of self-recovery on a moduler robot. It also provided "complete" control algorithm combining gaits performing rolling, turning and fall-down recovery on a remote-controllable Superbot rolling track. The potential to support more number of modules for rolling and turning are examined in simulation of 6, 8, 10-module configuration. Furthermore, the ability of the control to support self-reconfiguration is demonstrated with the use of loop and ring size detection in a snake to loop reconfiguration.

Future research directions include addressing problems caused by distributed control such as motor conflicts and instability of gait transition from rolling to turning, attaining higher speeds and generalizing self-recovery reconfiguration procedures for Superbot. We believe regulating motor torque using current sensors will reduce the motor conflicts.

9 Acknowledgements

We are very grateful that the SuperBot project is sponsored by NASA Cooperative Agreement NNA05CS38A. We would like to thank Duckho Kim and Dr. Behnam Salemi for tuning and maintenance of SuperBot module and the works of our colleagues at Polymorphic Robotics Laboratory at the Information Sciences Institute of the University of Southern California in carrying out

the experiments. We would also like to thank Dr. Mark Moll for building a framework on SuperBot simulation.

References

1. Alon Itai and Michael Rodeh. Symmetry breaking in distributed networks. *Information and Computation*, 88(1):60–87, 1990.
2. Akiya Kamimura, Haruhisa Kurokawa, Eiichi Yoshida, Satoshi Murata, Kohji Tomita, and Shigeru Kokaji. Automatic locomotion design and experiments for a modular robotic system,. *IEEE/ASME Transactions on Mechatronics*, 10:314–325, 2005.
3. T. Matsuda and S. Murata. Stiffness distribution control - locomotion of closed link robot with mechanical softness. *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1491–1498, May 15-19, 2006.
4. Behnam Salemi, Mark Moll, and Wei-Min Shen. Superbot: A deployable, multifunctional, and modular self-reconfigurable robotic system. In *IEEE/RSJ International Conference on Intelligent Robots*, Beijing, China, October 2006. To appear.
5. Jimmy Sastra, Sachin Chitta, and Mark Yim. Dynamic rolling for a modular loop robot. In *International Symposium on Experimental Robotics*, Rio de Janeiro, Brazil, 2006.
6. Wei-Min Shen, Harris Chiu, Michael Rubenstein, and Behnam Salemi. Rolling and climbing by the multifunctional superbot reconfigurable robotic system. Albuquerque, New Mexico, February 2008.
7. Wei-Min Shen, Robert Kovac, and Michael Rubenstein. Singo: A single-end-operative and genderless connector for self-reconfiguration, self-assembly and self-healing. In *2008 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Nice, France, September 2008.
8. Wei-Min Shen, Maks Krivokon, Harris Chiu, Jacob Everist, Michael Rubenstein, and Jagadesh Venkatesh. Multimode locomotion for reconfigurable robots. *Autonomous Robots*, 20(2):165–177, 2006.
9. Wei-Min Shen, Behnam Salemi, and Peter Will. Hormone-inspired adaptive communication and distributed control for CONRO self-reconfigurable robots. 18(5):700–712, October 2002.
10. Wei-Min Shen, Behnam Salemi, and Peter Will. Hormone-inspired adaptive communication and distributed control for conro self-reconfigurable robots. *IEEE Trans. on Robotics and Automation*, 18(5):700–712, October 2002.
11. Yamawaki Tasuku, Mori Osamu, and Omata Tooru. Dynamic rolling control of 5r closed kinematic chain with passive joints. *Journal of the Robotics Society of Japan*, 22(1):112–119, 2004.
12. Mark Yim. Locomotion with A unit-modular reconfigurable robot. Technical Report CS-TR-95-1536, 1995.