Auton Robot (2006) 20:165–177
DOI 10.1007/s10514-006-6475-7

Multimode locomotion via SuperBot reconfigurable robots

Wei-Min Shen · Maks Krivokon · Harris Chiu · Jacob Everist · Michael Rubenstein · Jagadesh Venkatesh

Published online: 22 April 2006

Abstract One of the most challenging issues for a self-sustaining robotic system is how to use its limited resources to accomplish a large variety of tasks. The scope of such tasks could include transportation, exploration, construction, inspection, maintenance, *in-situ* resource utilization, and support for astronauts. This paper proposes a modular and reconfigurable solution for this challenge by allowing a robot to support multiple modes of locomotion and select the appropriate mode for the task at hand. This solution relies on robots that are made of reconfigurable modules. Each locomotion mode consists of a set of characteristics for the environment type, speed, turning-ability, energy-efficiency, and recoverability from failures. This paper demonstrates a solution using the SuperBot robot that combines advantages from M-TRAN, CONRO, ATRON, and other chain-based and lattice-based robots. At the present, a single real SuperBot module can move, turn, sidewind, maneuver, and travel on batteries up to 500 m on carpet in an office environment. In physics-based simulation, SuperBot modules can perform multimodal locomotions such as snake, caterpillar, insect, spider, rolling track, H-walker, etc. It can move at speeds of up to 1.0 m/s on flat terrain using less than 6 W per module, and climb slopes of no less 40 degrees.

Keywords Modular · Multifunctional and Self-reconfigurable robots · Space robots · Multimode gaits

W.-M. Shen (✉) · M. Krivokon · H. Chiu · J. Everist · M. Rubenstein · J. Venkatesh
Information Sciences Institute and Computer Science Department, University of Southern California

1. Introduction

Multimode locomotion is an essential ability for any self-sustaining robotic system. Many tasks, such as transportation, assembly, and exploration, require a robot to travel through terrains that may not be fully characterized ahead of time. In such cases, a robot must use different moving modes in different environments. For example, a robot must "climb" if it is to go up a slope, must "run" if it is to cover more distance with less energy, must "balance" if the terrain is rugged and uneven, and must "get up on its feet" if it fell down by mistake. We call such an ability multimode locomotion.

To support multimode locomotion, a robot must have at least four capabilities. First, it must be able to perform different locomotion modes. Second, it must be able to recover from unexpected locomotion failures. Third, it must be able to shift from one mode to another. Finally, it must be able to choose the correct mode for the correct environment. This paper will focus on the first and the second topic and we will leave the third and fourth for a future paper. Specifically, this paper is to propose a concept and a system for how a single robot system can perform as many locomotion modes as possible (run, climb, fly, jump, reach, crawl, etc.).

It is a great challenge for a single robot to achieve multimode locomotion because the robot must simultaneously satisfy two competing and even conflicting criteria. On the one hand, the robot must be as *general* as possible so that it can deal with many types of environments and difficult tasks. On the other hand, the robot must be as *specialized* as possible so that it can achieve goals (such as speed and distance) with greater efficiency.

Over the past decades, it has been shown to be very hard for a conventional robotic system to achieve both criteria, since in such systems optimization of one criterion inevitably leads to deterioration of the other. For example, improving efficiency of locomotion on particular terrain type leads to decreased efficiency on other terrain types and therefore lowers the generality of the robot.

Reconfigurable and modular robotic systems provide a new approach to this challenge and allow simultaneous optimization on both criteria. Generality is inherent in design of such systems because modules can form different configurations. Thus, improvements in individual modules contribute to efficiency of all configurations and behaviors, while improvements in a particular configuration/behavior do not affect negatively on other configurations/behaviors.

This paper demonstrates a case study of multimode locomotion with a reconfigurable modular robot called SuperBot. The SuperBot robot demonstrates a diverse set of locomotion modes, which will enable the robot to traverse in different types of environment with non-deteriorating efficiency. The advantages of SuperBot modules include the integrated capability of chain-based and lattice-based robots, the full set of locomotion primitives (moving and turning) of any single module, the maneuverability and reconfigurability from any initial configuration with single or multiple modules, and the recoverability from unexpected failures in unknown environments. It also uses a totally distributed control method for locomotion and reconfiguration that is capable of supporting arbitrary reconfiguration of modules from one mode to another.

The multimode locomotion of SuperBot is demonstrated in both real world and simulation. In reality, a single SuperBot module can move, turn, sidewind, maneuver with other gaits, and travel on batteries up to 500 m on carpet in an office environment. In physics-based simulation, SuperBot modules can support many different locomotion modes (such as rolling track, snake, caterpillar, insect, spider, H-walker, etc.). Some of these modes are energy-efficient (less than 6 W per module), some can recover from falling down on unknown terrain, and others can move at speeds up to 1.0 m/s on flat terrain or climb slopes that are no less than 40 degrees.

The rest of the paper is organized as follows. Section 2 describes the related work on multimode locomotion. Section 3 presents the design of SuperBot modules. Section 4 describes the physics-based simulation in detail. Section 5 presents a set of multimode locomotions and describes their characteristics. Section 6 concludes the paper with future work and acknowledgements.

2. Related work

Existing reconfigurable robots can be classified into two general categories: the lattice-based and the chain-based reconfigurable robots. The distinct mechanical feature of a lattice-based robot is that a module is designed to reach only its adjacent modules. Examples of such robots include 3D Fracta (Moruta et al., 1998), Molecule (Kotay et al., 1998), I-Cubes (Unsal et al., 2001), ATRON (Jorgensen et al., 2004), Molecube (Lipson et al., 2005), etc. Such a design makes reconfiguration simple but relies on an assumption that neighboring modules are always aligned and docking can be done without alignment. For locomotion, such a robot will "flow" its modules on top of each other and several algorithms have been demonstrated in simulation (Butler et al., 2002). However, the assumption for automatic-alignment may not be always true in reality. Two neighboring modules may not be able to dock because uncertainties among modules in a large configuration could accumulate and become large enough to prevent neighboring modules from docking to each other without active alignment. Similarly, the flow-style locomotion may face the same problem when the size and distribution of obstacles in the environment do not match the lattice grid of the robot. Furthermore, the flow-style locomotion is extremely slow and inefficient because the movement of the robot is accomplished as a side effect of many docking and undocking between modules. For example, for a robot of N modules in a snake configuration to move forward for a distance of one module, it would require $N+2$ docking/undocking operations. Because of these reasons, the flow-style movement for lattice-based robots is not yet practical at the present time.

On the other hand, the chain-based modules have mechanical features that are designed to support efficient gaits and offer flexible reconfigurations. Each module can bend its body to form shapes with arbitrary angles and align modules to dock. The alignment of docking would require modules to have sensors and to control their movements based on the sensor information. Chain-based modules are also efficient for locomotion because the modules can bend their shapes to accomplish movement. For example, modules in a snake-shaped robot can bend themselves in coordination to move. Many such locomotion modes have already been demonstrated. A PolyBot (Yim et al., 2003) rolling track can run for 500 m on batteries and can climb stairs and fences; a M-TRAN robot (Kamimura et al., 2003) can move as a rolling track, H-walker, snake, caterpillar, and others; and a CONRO robot (Shen et al., 2002) can move as snake, caterpillar, insect, spiders, and others.

Although many case studies in locomotion for chain-based robots are already in place, no systematic study of locomotion modes has been done before. As a result, no existing reconfigurable robot possesses all the necessary features that would allow a robot to move in an arbitrary configuration. For example, a M-TRAN robot cannot turn if its current configuration does not contain any "helper" modules. A Polybot robot cannot survive a dynamic reconfiguration without reloading software. A CONRO robot cannot bend its module flexibly into a 180° U-shape, and as a result, it requires more modules to perform certain locomotion

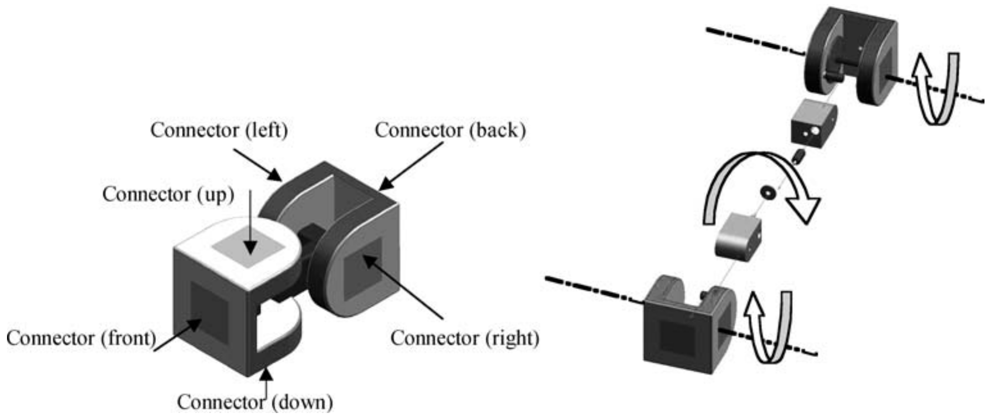


Fig. 1 The mechanical design of a SuperBot module

modes (such as rolling tracks) than other robots. In addition to the above disadvantage, most existing robots cannot recover from unexpected failures without reconfiguration. For example, if a rolling track of M-TRAN, Polybot, or CONRO modules falls down, it will not be able to stand up and run again without rearranging their configuration.

3. The design of SuperBot modules

The SuperBot modules described in this paper integrate the necessary features from both M-TRAN and CONRO to accomplish multimode locomotion. Shown in Fig. 1, each SuperBot module has three joints. The middle joint can mechanically rotate continuously in both directions (currently it is limited by the electronic wires going through the joint), and the two joints at the end can each rotate $\pm 90^\circ$, respectively. The three joints provide enough flexibility for a single SuperBot module to act as a gimbal mechanism, enabling a module to change its shape dynamically and perform locomotion and change its moving directions without any external assistance.

Shown in Fig. 2, a single SuperBot module can change its shape dynamically to provide the needed flexibility for the multimode locomotion and self-reconfiguration. For example, by rotating the middle joint for 90°, a single SuperBot module can transform itself into a M-TRAN shape (called “M-module”) or a CONRO shape (called “C-module”), respectively. Also shown in Fig. 2, each SuperBot module has six reconfigurable connectors on the six surfaces of the two linked cubes to allow connections in any of the six directions in 3D (front, back, left, right, up, and down). The design requirements for these connectors include being genderless so that any connector can dock to any other connector (such a connector is also necessary for modules to dock with heterogeneous devices and tools that use the same connector), strong mechanical endurance, power sharing, communication, dock guidance, single-side release mechanism, and reliability in rough environments. At the present time, the connectors shown in Fig. 2 are still primitive and require manual

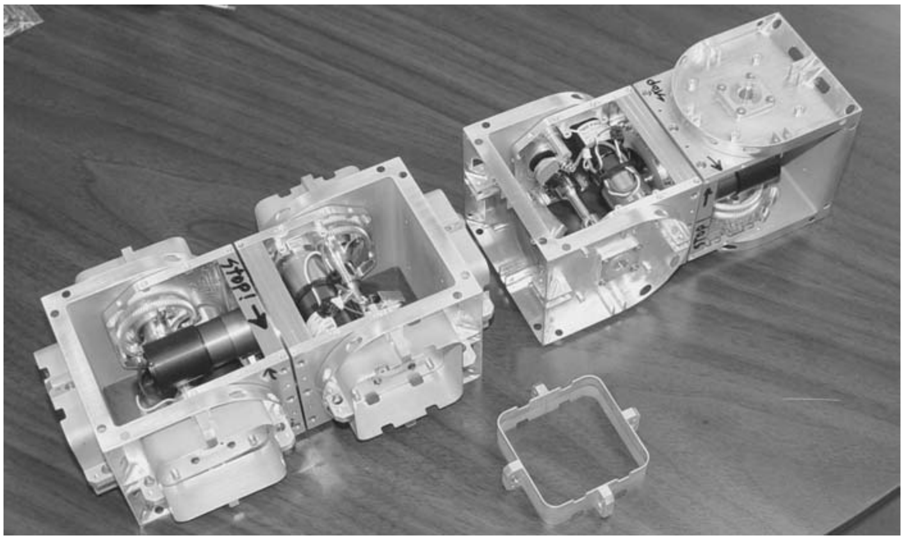


Fig. 2 Module shape changing (M-module and C-module) and connectors

docking and de-docking, thus all reconfigurations in this paper are done by hand. The details of the connector design and performance will be described in a separate paper.

Different from all existing reconfigurable modules, a single SuperBot module can perform many different gaits (e.g., caterpillar, sidewinder, push-and-pull, etc.) and turn and flip without any external help. For example, a M-module can move as a "caterpillar" by bending itself into a U-shape and then straightening out and perform this process repeatedly. A C-module can change its moving directions by twisting its two end joints in rhythm, and move sideways by performing a "sidewinder" movement. It can also flip back from side to side to change its orientation relative to the gravity field. Movies of these movements can be found at the website http://www.isi.edu/robots/superbot. With such locomotion abilities, a single SuperBot module is a miniature robot by itself and can move around and search for other modules to join and form big robots. If a module is connected to a thruster with sufficient thrust and flexible controllability, it can be made to fly in micro-gravity environments for space applications.

SuperBot modules are designed to be multifunctional and form robots that have many different configurations and functionalities. Figure 3 shows some example configurations formed by multiple modules. Different configurations can provide different functions. For example, one configuration may be suitable for tool using, while another may be used for transportation. When a long chain configuration (possibly with several branches) can serve as a crane to lift objects with remote control. A configuration with legs and limbs can climb slopes that are too steep for wheeled robots. A rolling track or wheel-like configuration can move efficiently on flat terrain with minor obstacles.

Since the modules can reconnect among themselves, they can be packaged in a way that is appropriate for transportation. Once at the destination, they can separate and scatter themselves to conduct search in a large area, and then later come back and join together to form different shapes for different tasks. In the same spirit, modules can also recharge their batteries by first communicating with a recharging station, then going to the station and connecting for recharging. In addition, the modules can interact with humans to detect and recover from unexpected failures in their configurations. For space applications, they can be used as structural components of a landing vehicle to planetary surfaces. They then become an useful tool for loading and unloading or a set of agile autonomous rovers for other surface operations.

4. Physics-based simulation

A physically accurate simulation of robotic systems provides a very efficient way of prototyping and verification of control algorithms, hardware design, and exploring system deployment scenarios. It can also be used to verify the feasibility of system behaviors using

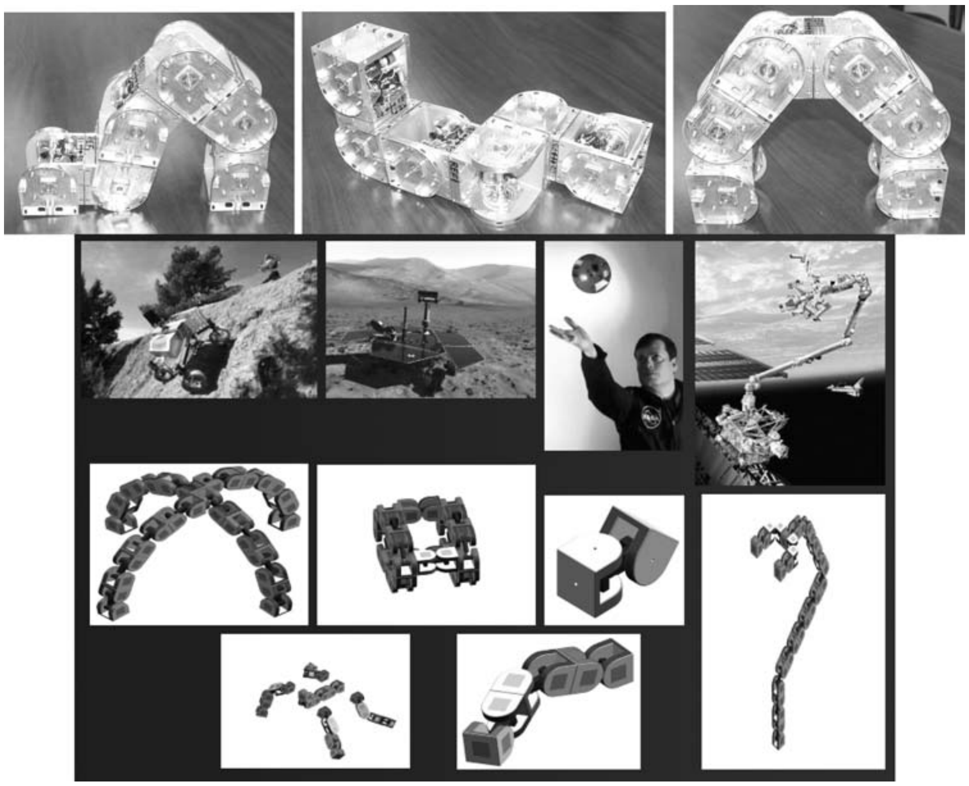


Fig. 3 SuperBot configurations and their potential functions

realistic morphology, body mass and torque specifications for servos.

For SuperBot, we have developed the Galina simulator to create modules and testing environments as realistically as possible. It contains collision detection and rigid body dynamics algorithms for SuperBot modules. It is built upon an existing open source implementation of rigid body dynamics, the Open Dynamics Engine (ODE), and we have tailored it for the SuperBot robot. ODE was selected for its popular open-source physics simulation API, its online simulation of rigid body dynamics, and its ability to define wide variety of experimental environments and actuated models.

ODE engine is very flexible in many respects. It allows user to control many parameters of simulation, such as gravity, Constraint Mixing Force, Error Reduction Parameter, etc. ODE also does not have any fixed system of measurement units, and therefore accommodates systems of different scales and ratios that could be more appropriate for a particular setup. This flexibility however makes it quite difficult to come up with a set of parameters that result in stable and adequate simulation environment. We have spent considerable time testing different combinations of these settings and used the experience to produce a tuned simulation that models most accurately the possible real settings for SuperBot behaviors. One example of such subtle simulation setting is definition of friction direction. By default the direction of friction force that is applied to every body on contact is not defined and therefore is arbitrary. However, this setting produced inconsistent locomotion results for a simple chain configuration of SuperBot modules. Being unconstrained, the direction of friction was chosen by the engine in arbitrary manner, which led to "turning" effects in a caterpillar movement that was supposed to be straight. These problems led us to explicitly determine and define direction of friction force based on contact information for each body, and it resulted in consistent results for caterpillar and other locomotion behaviors.

In order to implement multifunctional behaviors of the SuperBot system in simulation, the morphology of SuperBot modules had to be described in a form compatible with the ODE API. For the purposes of better performance and stability, the model of SuperBot was simplified to a set of standard geometrical primitives (such as spheres and cubes) connected by three degrees of freedom, which were defined as powered joints. This simplification of changing odd shapes into standard shapes was necessary to make simulation scalable (collision detection with odd shapes are very expensive in ODE). Since some behaviors we planned to implement needed a large number of modules and adding more bodies and degrees of freedom to the simulated environment affects the performance in a polynomial fashion, we needed simple shapes for large simulations.

The created geometric morphology model was assigned dynamic properties that correspond to SuperBot design specifications. Masses for each body part were assigned values estimated by mechanical engineers. Degrees of freedom were limited by maximum torque and speed available from specifications of servomotors selected to be deployed in each SuperBot module. To ensure proper interaction of modules with the simulated environment, friction coefficients were set to values estimated for materials to be used for module manufacturing and possible surface materials.

To be capable of producing behaviors with different functionalities, SuperBot modules should be able to dock to each other forming different complex configuration shapes. In design specification, SuperBot module has three docking faces on each side and a docking mechanism that allows considerable tolerance in unknown angle when docking. In the simulated environment, the docking capability of SuperBot was implemented in the form of two features. One is the ability of two modules to be attached to each other and maintain the relative positions fixed. This was implemented using a fixed joint that is created, connecting two sides of different modules. The second feature is dynamic docking of two modules when distance and approach angle errors between their corresponding faces are less than predefined values. This feature had to be implemented in the form of an additional level of collision detection. Space around the dock faces of each module is simulated to be a virtual sphere body that can collide with other similar spheres. When collision is detected, distance and angle between the two faces is computed and compared to error margins. If computed errors are small enough, a docking event is registered and necessary simulation changes are introduced before next simulation step: Namely, a fixed joint is created between corresponding bodies and their docking pseudo spheres are disabled for collision detection process.

To perform certain functions successfully, a SuperBot configuration needs the capability of acquiring some information about itself and its environment. In a real SuperBot module, this is achieved through sensors. The most important and simple sensor that is required for almost all behaviors is the servo position sensor. It is used in many cases to make decisions if the action is done or which action should be selected next. In a simulated environment, this sensor is easily implemented through accessing the current state of the powered joint and retrieving the angle parameter. Another sensor that is used often for dynamic locomotion and detecting abnormal configuration position is a gravity sensor. Real SuperBot modules are equipped with a three dimensional accelerometer, readings of which will be accumulated over some time and filtered to determine direction of gravity. For the purposes of this paper, we did not simulate noise that is present in real accelerometer readings that comes from such factors as movement of the module itself, collisions with other modules, etc. Thus, in the simulated SuperBot model, the gravity sensor is simplified to a three dimensional vector in the frame of reference of the module.

The core of every SuperBot behavior is the distributed control algorithm that determines how modules coordinate their actions to perform behavior functionality. In reality, the configuration of SuperBot modules is a distributed system that has *n* independent processors running the same control program and exchanging messages through the infrared or other sensors placed on the dock faces. However the physics-based simulation runs only on one computer and executes control programs for each simulated module along with solving the dynamics equations. Thus, to achieve realistic results, the simulation environment has to emulate concurrent execution of control programs for different modules and the resulting communication issues. Ideally this emulation should be micro-processor specific, namely the simulation time of execution for a particular program instruction should be equivalent to the real time it takes for SuperBot processors to process that instruction. This approach however introduces another level of simulation fidelity, and therefore, considerable overhead. We have decided to follow a simpler route and use concurrency mechanisms provided by the operating system—namely threads—to emulate simultaneously running modules. Each simulated SuperBot control program has its own independent thread of execution which runs in an infinite loop. The physics simulation engine spawns all the SuperBot threads in the setup routine and then proceeds to the simulation loop. Each SuperBot thread yields execution control at the end of its program loop to give control to the simulation thread which thus has highest execution priority. This helps to make simulation smooth and reduce CPU load, since instead of busy waiting, the SuperBot programs go to "sleep" mode until enough simulated time has passed.

The emulated concurrency also forces discipline on control program developers. The fact that each simulated module runs an independent piece of code requires deep consideration of synchronization and sensor data propagation among modules in configuration. This realistic approach makes the developed control algorithms much more suitable for transferring them onto real SuperBot modules.

The physics-based SuperBot simulation was developed using Object-Oriented design to allow intuitive programming of behaviors that map conceptually to programming of SuperBot hardware. Thus, control of a module was abstracted as a set of methods to set servo target position and get current position. Other sensor readings were also implemented as simple access methods. We developed simulation API concurrently by several people to develop different locomotion behaviors and obtain test results, which are described in the following sections.

5. Classification of locomotion modes

The key feature of the SuperBot platform is to support many different locomotion modes on a wide range of terrain types. Before we describe each mode in detail, it is useful to discuss how we classify the locomotion modes in theory, so that they can be used appropriately for future applications.

Table 1 shows a list of locomotion modes in terms of 8 parameters. The "mode" and "config" show the name and configuration of the mode. The "slope" and "obstacle" parameters show the type of environment that the mode can cope with. The value for slope is a range of degree of the slope that the robot can handle. For example, [− 60, 40] means that the mode can go down on slopes of 60 degree and climb up slopes of 40 degree on carpet. The "speed" tells how fast the mode can move, the "turn" parameter indicates if the mode can make turn or not, the "energy" parameter specifies the efficiency of the energy consumption in terms of Watts per module (W/mdl). Finally, the "recover" parameter indicates if the mode can recover from failures or not.

For each locomotion mode, it is a challenge to optimize all the parameters simultaneously. For example, efficiency and adaptability are two complete groups of parameters. When optimizing the efficiency parameters such as speed and energy consumption, it is hard to keep the adaptability parameters such as the range of slope and obstacle height. Using the advantage of reconfiguration to change shape, one can change mode and select the most effective and efficient mode for the current task and environment. This way, we can

Table 1 Classification of locomotion modes.

Mode	Config	Slope	Obstacle	Speed	Turn	Energy (W/mdl)	Recover
6M	Loop	[− 40, 10]	0	1.0 m/s	no	5.8	non
10C	Loop	[− 40, 10]	1/3 high	0.3 m/s	yes	∼ 5.0	yes
9M	H-Walker	[− 10, 10]	1/1 high	0.36 m/s	yes	4.35	yes
6M4C	T-Wheel	[− 40, 10]	0	0.6 m/s	yes	∼ 6.3	yes
2M4C	Loop	[− 40, 10]	0	0.70 m/s	yes	∼ 6.0	yes
8M	Climber	[− 60, 40]	0	0.1 m/s	yes	∼ 6.5	yes
Other	...	[− ..., ...]	...	...	...	...	...

avoid the tradeoff between efficiency and adaptability for a conventional robotic system.

Thus, a reconfigurable robot can optimize traversal of two different terrain types. Optimizing and enhancing functionality of a SuperBot module will lead to improvement in each locomotion mode, given a properly designed controller. Optimizing control of a specific mode does not affect efficiency of other modes so that the robot can guarantee good performance on speed and energy consumption for all locomotion modes. In the next several subsections, we describe the locomotion modes in Table 1 in detail.

5.1. The 6M-loop mode

The first locomotion mode we consider is the 6M-Loop mode, which consists of six M-modules in a loop configuration. This mode can cope with relative flat terrain (with -10° to 10° slopes) with minimal or no obstacles. This type of environment is traditionally dominated by wheeled systems in terms of locomotion efficiency. Wheeled mode of locomotion is very energy efficient and allows high speeds. However the tolerance to environmental obstacles is limited by the size of the wheel and increase in wheel size leads to other undesirable features. Thus, a wheel-actuated system usually uses obstacle avoidance instead of increased wheel size.

In order to achieve comparable locomotion performance results on this type of terrain, the SuperBot system has to mimic wheeled locomotion using its hyper-redundant configuration. Dynamic behavior must be introduced for this rolling-track to accomplish a wheel-like movement. The idea is to configure the modules in a ring configuration of a hexagon shape and put it vertically as a rolling traveler. It rolls along a vertical plane propelling itself by changing its shape. It continues the rolling movement by contracting and relaxing its configuration based on gravity sensors in the modules. The number of 6 modules is chosen for the stability and efficiency of movement.

Figure 4 shows a sequence of the fast rolling movement, and the implementation of the dynamic control for each step. The mode alternates its shapes between a regular hexagon (shown in the left diagram) and a deformed hexagon that tends to fall forward (shown in the right diagram). Starting from the regular hexagon, the movement is controlled by the deformation of the shape to change the center of gravity of the robot. There are 2 commands governing the shape transformation. One is to retain the regular hexagon shape. The other is to let the rolling robot "squeeze" itself to a deformed hexagon. The deformed shape shifts the center of gravity forward creating torque about the joint of a module (module F in the right diagram) and causes the robot to roll. When at top speed, the regular round shape helps to maintain the momentum.

Fast rolling movement is achieved by co-operating these two commands with gravity sensors. Gravity sensors are put on both segments of each module. Each segment of a module knows the vector of gravity with respect to its own coordinate frame. Initially, the traveler is set in a regular hexagon shape. The lower left module touching the ground (module A in the diagram) will check the gravity sensor of its segment 2. If it indicates the module lies flat on the surface, a "squeezing" command is issued to deform the traveler and the robot starts to roll. The command to retain regular hexagon shape is issued when segment 1 of the lower left module (module A) knows it is in vertical. The duty of determining commands based on gravity sensor is transferred counter-clockwise to

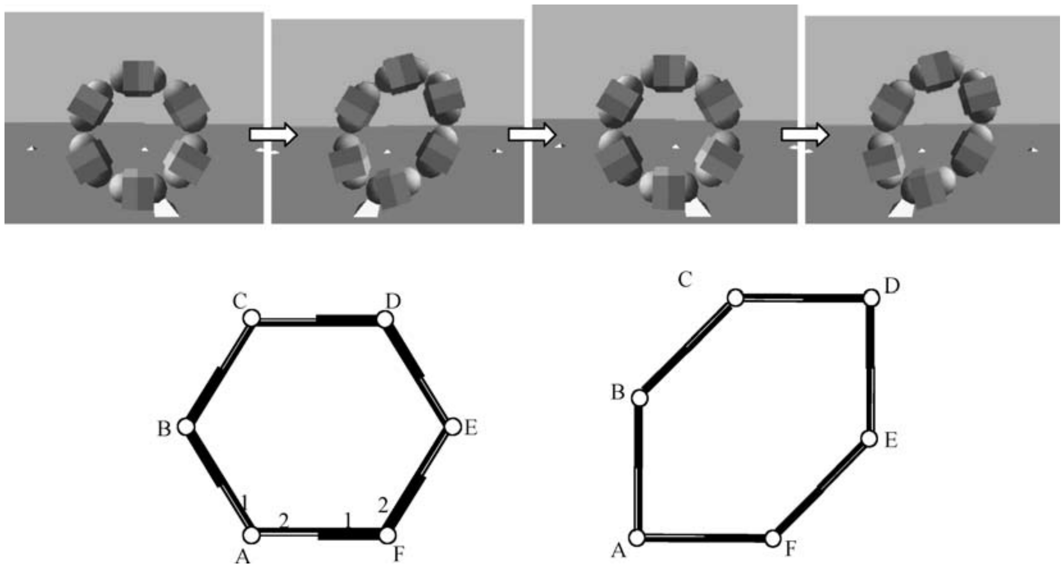


Figure 4. 6M-Loop mode for fast movement with dynamic control.

the module next to it. For example, after gravity sensor of segment 1 of module A indicates the segment is vertical and the traveler transforms back to a regular hexagon shape, the lower left module will be module F, which will start to examine its gravity sensor on its segment 2 and check if it is touching the ground. The duty of determining commands continually transfers in a counter-clockwise direction. In this way, the rolling traveler keeps accelerating until the servo reaches its speed limit for responding to shape changing commands. This particular controller is dynamic and sensor based. Note that a similar gait has been achieved before by the M-TRAN robot (Kamimura et al., 2003). However, the control of M-TRAN gait was open loop, while the gait described here uses gravity sensors so that it may adapt to changes in the environment.

In terms of energy efficiency, this locomotion mode allows fast traversal of terrains with minor obstacles. With 6 modules in a loop configuration the robot can roll with average speed of 0.93–1.04 m/s. This was measured in simulation for a distance of 1 km traveled in 18 min. Maximum bound of energy usage was estimated by assuming each change of servo angle uses maximum torque of 1.8 Nm. Sum of angle changes was accumulated then used to calculate energy consumption of 35 W for 6 modules, thus 5.83 W/module in average. To travel 1 km, the total energy required by the entire robot for 18 min is 37.776 KJ.

5.2. The 10C-Loop Mode

Although the 6M-Loop can move fast, the robot cannot stand up again once it falls down. This is due to the fact that all modules in that mode are in M-TRAN shape, and they can only move in the same plane they were configured. To overcome this limitation, the 10C-Loop mode uses all CONRO-like modules so that each module can control its pitch and yaw movement. As a result, the robot is much more flexible and can run, turn, and recover from falling down. This mode can deal with environments where obstacles do not exceed in size the height of the robot configuration.

To cope with obstacles in the environment, this rolling track is particularly effective and efficient. With a flexible track, it can roll over obstacles that are comparable to the size of the robot (Yim et al., 2003). The 10C-Loop mode demonstrates the rolling track locomotion while retaining its ability to use more efficient configuration for flat terrains.

When a single rolling track is fast moving, it is a challenge to keep it balanced when there are obstacles in the environment. The most common solution is to have two tracks in parallel, but it doubles the energy consumption if the task at hand requires only a single track. For a single track to survive in an obstacle-rich environment, it must be able to turn and avoid obstacles. To the best of our knowledge, the turn capability has been lacking in all previous single rolling track movements in modular robotics.

As shown in Fig. 5, the 10C-Loop mode has 10 CONRO-like modules connected in a loop. The forward straight movement starts in the position shown in the left picture. At a fixed time interval, or when all modules have bent forward to the desired angle, each module begins to bend forward again to reach the angle that is equal to the current angle of the module that is in front of it. When this process repeats, the rolling track will move forward in a straight path. To make the robot turn, the top center and bottom center modules will bend sideway (shown on the right-hand side in Fig. 5) while all modules maintaining the same process of described above. Because every module desires to reach the same angle position as the current position of the module in front of it, the entire rolling track will turn and move forward. Notice that in comparison with the dynamic control of the 6M-Loop mode, this movement is static stable and every module moves from one statically state to another.

Even with the ability to turn to avoid obstacles, there is no guarantee that the robot will never fall down in an unfamiliar and unstructured environment. Thus, the ability to recover from failure is essential. The most common locomotion fault is "tipping over" or more generally, change of robot's orientation and position relative to terrain such that interaction of the locomotion gait and environment does not produce expected

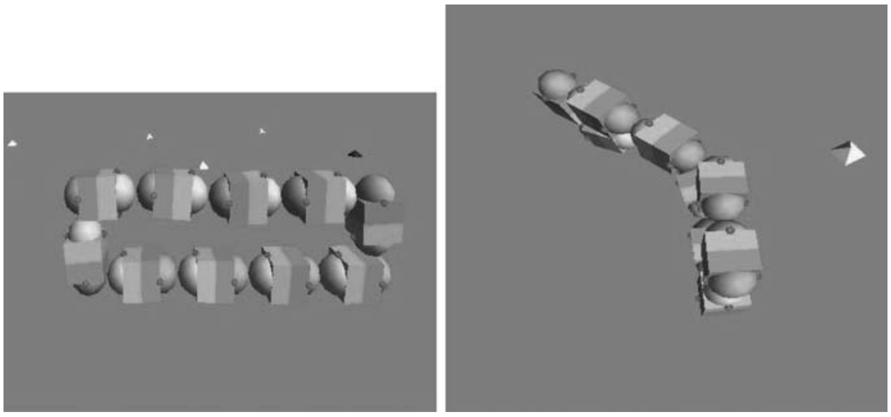


Fig. 5 The implementation of 10C-Loop locomotion mode

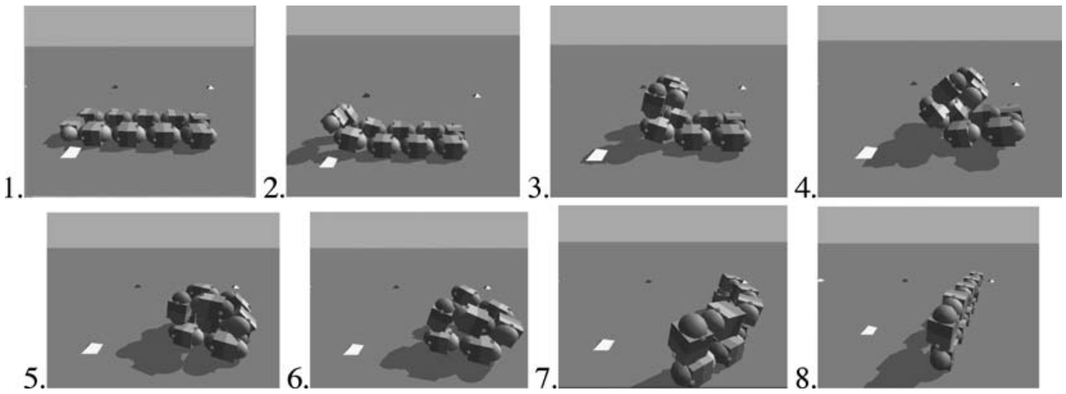


Fig. 6 Recovery of 10C-Loop from a falling position

propelling effect. Conventional robotic systems usually use elaborate preventive strategies to avoid such faults, because recovery from most faults is extremely difficult or impossible. This leads to very conservative results in terms of locomotion efficiency. For example, a Mars Rover's slow movement is partially due to this reason. However, modular reconfigurable robots have a great advantage in this respect because they can change configuration shape and recover from locomotion faults much easier than the conventional robots. This justifies the fact that a reconfigurable robot can employ more aggressive locomotion strategies for uneven and unknown terrain.

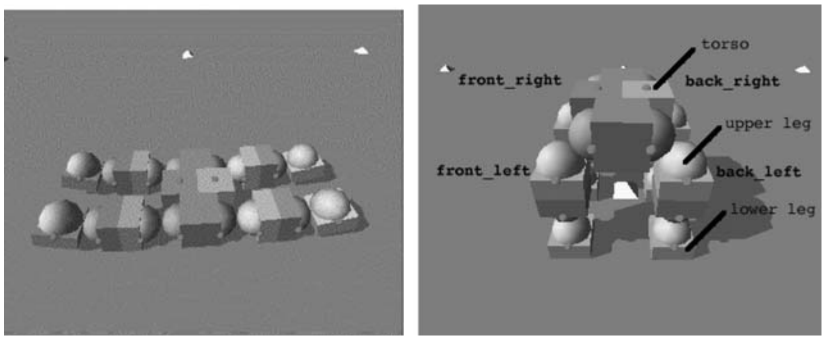
10C-Loop mode represents one fault recovery strategy for SuperBot, and this strategy can be generalized for all modes that have the similar configuration. When the robot fall down, it can restore its original normal orientation by going through a sequence of motions shown in Fig. 6. In step 1–4, the fallen-down loop first bends itself upward to form a folded loop in a vertical position (step 5), and then unfold this vertical loop horizontally (step 6–8) so that the loop stretched into a new rolling track. Then it can turn itself back to the original moving direction (not shown). The 10 modules are experimentally determined to be minimal for C-modules. If we use M-modules, then two M-modules must be used to achieve the effect of one C-module's pitch-yaw movement, so the total number of M-modules would be 20.

5.3. The 9M-walker mode

In many deployment scenarios, terrain to be traversed is highly irregular and has many obstacles that are of size comparable to that of the robot itself. In such environments, even hybrid wheeled locomotion becomes impossible and limbed morphology has to be used to allow for "walking", "climbing" and similar gaits. For specialized robotic systems with relatively monolithic design, this drastic change in morphology makes it impossible to use any other gaits that are suitable for other types of environments described above. However modular systems, such as SuperBot, allow using limbed modes of locomotion without giving up other more efficient gaits, since they allow reuse of the same hardware through reconfiguration. Furthermore, Digital Hormone distributed control used in SuperBot allows reuse of the same software, which dynamically detects changes in configuration topology and adjusts control accordingly. We present a limbed locomotion mode—"H-Walker" that is implemented using SuperBot simulated modules.

The H-Walker is a 4-legged walker using two degrees of freedom on each module. It is composed of two M-modules for each leg and a single M-module representing the torso as seen in Fig. 7. This locomotion design was inspired from the M-TRAN experiments in automatic locomotion generation (Kamimura et al., 2003). The naming convention for the modules and the control strategy are also illustrated in

Fig. 7 The H-Walker configuration and naming convention



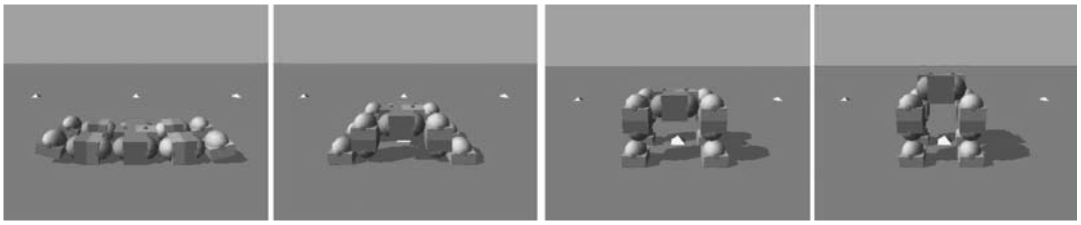


Fig. 8 H-Walker standing sequence

Fig. 7. There were three possible local topologies for a SuperBot module in this robot: torso, upper leg, and lower leg. Distributed locomotion control was achieved using the digital hormone method (Shen et al., 2002) based on these different topological types. Four hormones were used in this robot and each one is to control a different leg (front-left, front-right, back-left, and back-right). The torso was responsible for sending hormone messages to each of the legs and synchronizing their coordinated actions. Each leg receives one of four hormones representing whether it is the right-front leg, left-front leg, right-back leg, or left-back leg. The torso sends a message to each leg once every locomotion period to reset and synchronize the local timers of all the leg modules.

Both the upper and lower legs, based on the local topology and the hormone message they receive, reset their local timer to zero and move the pan and yaw servos based on a sinusoid with a given phase offset and amplitude. This way, the motions of all the legs are reasonably coordinated so long as the hormone messages are received quickly to synchronize the clocks of all the modules.

Using this control method, the H-Walker can move forward and backward, and can turn to change its moving directions. It can reach speeds of 0.6 m/s, with average speed 0.36 m/s. The total energy required is 39.07 W for 9 modules, thus 4.35 W per module in average.

The H-Walker locomotion mode demonstrates a conceptually different approach to locomotion fault recovery. Since in a limbed configuration, a change in morphology can be costly, the initial design itself should include features facilitating recovery. In H-Walker mode, it is achieved through symmetry of design and control. Because the H-Walker's topology is in the shape of an 'H', the SuperBot can walk forwards and backwards using the same control strategy, as well as upside down with the legs re-positioned below the torso. In fact, this symmetry prevents the H-Walker from ever falling into any unrecoverable position because the robot does not need to flip itself over to resume walking.

Should the H-Walker fall, it is easy to achieve the relaxed position in which the legs are straightened out to the sides in a double-caterpillar shape. Then H-Walker proceeds to stand up using the steps as seen in Fig. 8. These steps perform well in simulation, and further tests are required on a physical system in various terrains.

5.4. The 6M4C-training-wheel mode

As we mentioned before, the fastest mode is the 6M-loop but it cannot turn or recover from falling. The 6M4C-training-wheel mode is a modified version of 6M so that it can run fast, and can turn and recover from falling. To do so, we add four extra legs as "training wheels" to the 6M-loop. Shown in Fig. 9, one pair of "legs" is attached to each side of the loop, perpendicular to the hexagon plane and in opposite directions. The other pair of legs is attached in the same fashion but at the other end of the loop.

To turn and change the traveling direction of the loop, the "leg" modules on one side, which is a C-module, can lower down its "feet" by rotating the pitch servo, and at the same time, the 6M-Loop continues to roll forward. The 6M-Loop will be tilted and begin turning. This sequence is shown in Fig. 9.

Figure 10 shows how this locomotion mode recovers from falling down. The robot first straightens all the "leg" modules and collapses the hexagon to a flat loop. The hexagon plane can then be made vertical and the flat loop will change back to its hexagon shape and continue to roll. Note that this recovery procedure is simpler than that of 10C-Loop because it is accomplished through changes of contacting points

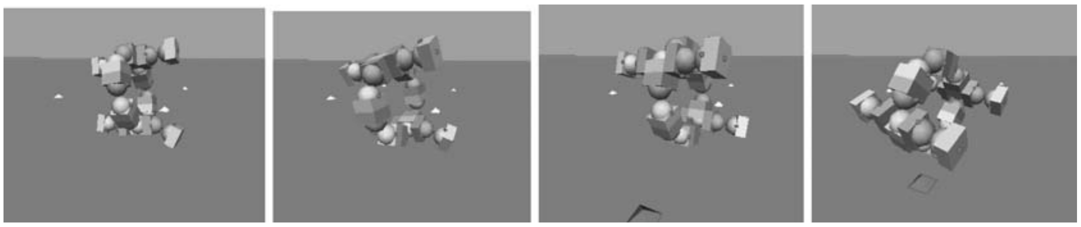


Fig. 9 The training-wheel configuration and its turning sequence

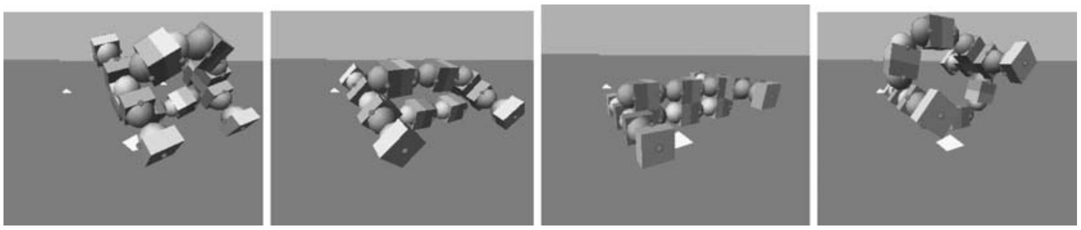


Fig. 10 Recovering from a fall

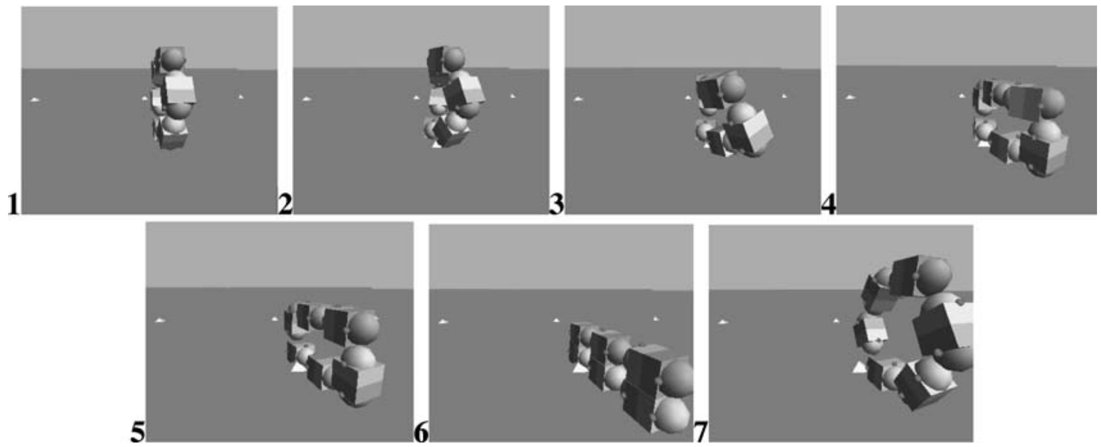


Fig. 11 The 2M4C-Loop mode and its turning operation

between the robot and the ground. Efficiency-wise, adding the 4 extra leg modules reduces the average rolling speed down to 0.77 m/s.

5.5. The 2M4C-loop mode

The training-wheel mode uses extra modules to allow a 6M-loop to turn and recover. However, there exist other modes that can accomplish the same effect without extra modules. The 2M4C-Loop is one of such modes. It still uses 6 modules for the loop, but alternates the type of module to enable the loop to turn and recover from falling.

Figure 11 illustrates this new locomotion mode and its turning sequence. The 6 modules in the loop are arranged to be M-C-C-M-C-C. To run, this mode uses the same dynamic control as 6M-Loop but each module will only use one servo (the C-modules will use their pitch motor) for the rolling action (see step 7). To turn, the hexagon of the 2M4C-Loop first bends into a rectangular shape where the C-modules are on the top and bottom of the rectangle and M-modules are at the vertical sides of the rectangle. Then, it uses the yaw servos of the C-modules to changes its direction. After that, the rectangle will go back to the original hexagon shape to run again. Note that this turning is statically stable and possible only when the robot is not rolling.

Figure 12 illustrates how the 2M4C-Loop recovers from falling. First, the loop straightens itself by bending the 2 M-modules into 180 degrees (note that this can be done only by the M-modules) and reset the shape of all 4 C-modules (step 2). The C-modules then change their yaw servos so that the robot is rising up yet unbalanced (step 3). The unusual movements of the C-modules will cause the robot to fall sideways for 90 degrees (step 4). The loop will then straighten up again (step 5) and go back to its original hexagon shape (step 6).

An interesting observation about 2M4C-Loop is its relation to 6M-Loop. In the 6M-Loop, by rotating the roll motors of two consecutive modules simultaneously, the two modules can become two consecutive C-modules without any disconnecting and connecting actions. Thus, a 6M-Loop can become a 2M4C-Loop, and vice versa. This makes a single configuration of SuperBot capable of fast running, turning, and recovery from failures.

5.6. The 8M-climbing mode

Previous locomotion modes show how different combinations of modules and gaits tackle the exploration issue on a relatively flat terrain. To tackle the climbing problem, a rolling track gait has been simulated to climb up a slope of 40 degrees, with the friction simulated as if the robot is climbing a slope covered by regular office carpet.

Shown in Fig. 13, the mode consists of 8 M-shape SuperBot modules forming a rolling track that is only 1.5-module

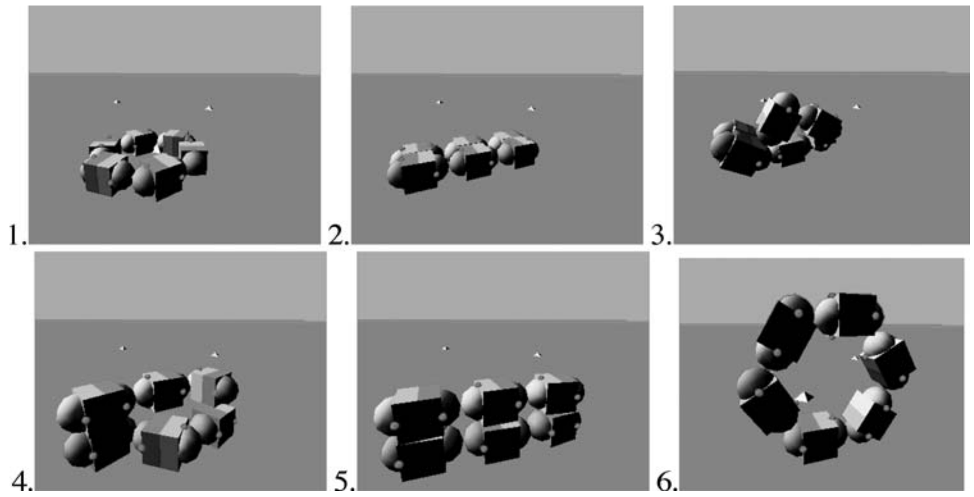


Fig. 12 The recovery sequence of 2M4C-Loop locomotion mode

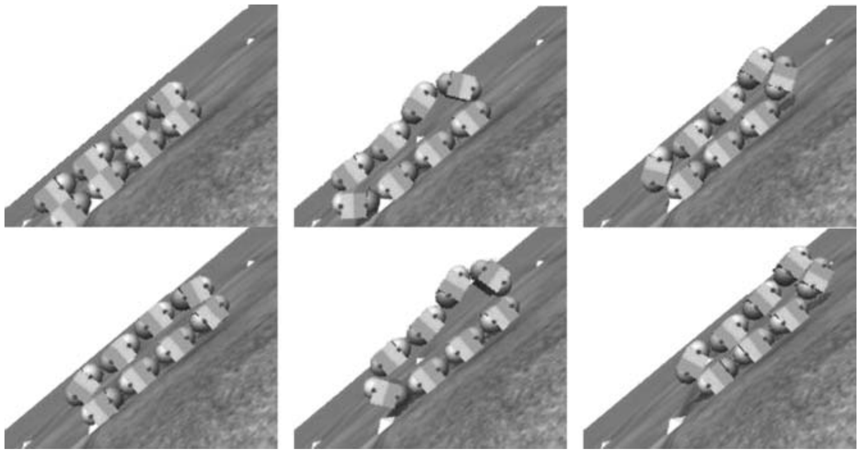


Fig. 13 The 8M-Climbing locomotion mode

in height. The advantage of this configuration is to make use of its low height property to stabilize it on the slope. In this simulation we assume the friction between the skin of the module and the slope is about 1.5 times higher than the rest of the modes in the paper (similar to climbing on carpet). In reality, this friction ratio can be achieved by allowing modules to dock with some special material on their outside docking faces. The mode will climb up the slope slowly by moving module by module. The simulation shows the configuration can climb up a slope of 40 degrees straight with sufficient friction, but it also show a ± 10 degrees of tolerance in the direction of travel before it falls sideway.

6. Conclusions

This paper presents the concept of multimode locomotion for robotic systems, and illustrates the concept of SuperBot modules that are designed to combine the advantages of several existing reconfigurable robotics systems. The paper also describes a list of locomotion modes, and each mode is classified by a set of realistic parameters to specify the environment properties, the flexibility of the locomotion for moving and turning, as well as the ability to recover from unexpected failures. The effectiveness of these modes are demonstrated by the SuperBot modules and configurations in simulation and the details of the moving, turning, recovering, and energy-efficiency of these locomotion modes are described. To the best of our knowledge, these results are very difficult to achieve by conventional robotic systems and provide convincing evidence for the concept of multimode locomotion and the value of modular and reconfigurable robots and distributed control mechanisms for these robots. It is our intension to show that it is possible to accommodate different locomotion modes without increasing the hardware and software complexity of the robots, and therefore avoiding the inescapable tradeoff between the generality for adaptation and the specialty for high-performance for conventional robotic systems. Our future work will be addressing the process of how to reconfigure the robot from one mode to another through self-reconfiguration. We will also implement all these modes on SuperBot modules and verify them through field tests in different terrain types and travel distances.

This research is supported in part by US Army Research Office under the grants W911NF-04-1-0317 and W911NF-05-1-0134, and in part by NASA's Cooperative Agreement NNA05CS38A. We thank Alliance Spacesystems Inc. and USC Engineering Machine Shop for the fabrication of prototype SuperBot modules, and we are also grateful for Peter Will, Berok Khoshnevis, Yigal Arens, and other members in the Polymorphic Robotics Laboratory for their intellectual and moral support.

References

Butler, Z., Kotay, K., Rus, D., and Tomita, K. 2002. Generic decentralized control for a class of self-reconfigurable robots. In *IEEE International Conference on Robotics and Automation*.

Jorgensen, M.W., Ostergaard, E.H., and Lund, H.H. 2004. Modular ATRON: Modules for a self-reconfigurable robot. In *IEEE/RSJ International Conference on Robots and Systems*.

Kamimura, A., Kurokawa, H., Yoshida, E., Tomita, K., Murata, S., and Kokaji, S. 2003. Automatic Locomotion Pattern Generation for Modular Robots, In *Proceedings of IEEE International Conference on Robotics and Automation* (ICRA'03), pp. 714–720.

Kotay, K., Rus, D., Vona, M., and McGray, G. 1998. The self-reconfiguring robotic molecule: Design and control algorithms. In *IEEE International Conference on Robotics and Automation*.

Lipson, H., White, P., Zykov, V., and Bongard, J. 2005. 3D Stochastic Reconfiguration of Modular Robots. Presentation at the *Workshop on Self-reconfigurable Robotics at the Robotics Science and Systems Conference*, MIT.

Moruta, S., Hurokawa, H., Yoshida, E., Tomita, K., and Kokaji, S. 1998. A 3D self-reconfigurable structure. In *IEEE International Conference on Robotics and Automation*.

Shen, W.-M., Salemi, B., and Will, P. 2002. Hormone-Inspired Adaptive Communication and distributed control for CONRO self-reconfigurable robots. *IEEE Transactions on Robotics and Automation*, 18(5).

Stoy, K., Shen, W.-M., and Will, P. 2002. Using role-based control to produce locomotion in chain-type self-reconfigurable Robots. *IEEE Transactions on Mechatronics* 7(4):410–417.

Unsal, C., Kiliccote, H., and Khosla, P. 2001. A modular self-reconfigurable bipartite robotic system: Implementation and motion planning. *Autonomous Robots*, 10(1):23– 40.

Yim, M., Roufas, K., Duff, D., Zhang, Y., Eldershaw, C., and Homans, S. 2003. Modular Reconfigurable robots in space applications. *Autonomous Robots Special Issue on Space Applications*, 14(2/3).

Harris Chi Ho Chiu is a PhD Student in Computer Science at the University of Southern California and a research assistant in Polymorphic Robotics Laboratory of Information Science Institute. He received his Master in Computer Science from the University of Southern California and his Bachelor of Engineering from the University of Hong Kong. His research interests include intelligent automated systems, modular self-reconfigurable systems, artificial intelligence, and machine learning.

Michael Rubenstein is currently a PhD student at the Polymorphic Robotics Laboratory, working on the CONRO and Superbot self-reconfigurable robotic systems. He has received his bachelors in Electrical Engineering from Purdue University, and his masters in Electrical Engineering from the University of Southern California, and is currently working towards his PhD in Computer Science from the University of Southern California. His interests include modular self-reconfigurable systems, autonomous robots, self-healing systems, and self-replicating systems.

Jagadesh B Venkatesh is a member of the Polymorphic Robotics Laboratory at the Information Sciences Institute. He is currently a Master's candidate in the Product Development Engineering program at the University of Southern California. He received his MS in Computer Science with specialization in Intelligent Robotics, also at the University of Southern California in 2005. His current interest is the commercialization of robotic technologies, specifically in the consumer robotics sector.