

Scalable and Reconfigurable Configurations and Locomotion Gaits for Chain-type Modular Reconfigurable Robots

Ying Zhang, Mark Yim, Craig Eldershaw, Dave Duff and Kimon Roufas

Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA, 94304
yzhang,yim,celdersh,dduff,kroufas@parc.com

Abstract

Modular reconfigurable robots have shown the promises of great versatility and robustness; however they also impose design challenges on mechanical, electronic and software scalability. In this paper, we present a class of configurations that *are* scalable mechanically and electronically. Moreover, there exists a predefined reconfiguration sequence between any of the three configurations, namely, snakes, centipedes, and multi-loops. Locomotion gaits for the three configurations are defined and created using Phase Automata, a generalization of gait tables. The system has been tested both in simulation of 100+ modules and in hardware of 50+ modules.

Keywords: *Modular Reconfigurable Robots, Locomotion Gaits, Reconfiguration Sequences, Scalable Configurations, Phase Automata*

1. Introduction and Motivation

Modular self-reconfigurable robotic systems are those systems that are composed of modules that can disconnect and reconnect themselves automatically in different arrangements to form a new shape with different functionalities. In many cases, the total number of modules is much larger than the number of different module types within such systems, i.e., the systems tend to be more homogenous than heterogeneous. The general philosophy underlying these systems is to simplify the design and construction of components while enhancing functionality and versatility through larger numbers of modules. There are a growing number of modular self-reconfigurable robotic systems that have this design philosophy [2,3,4,5,6,8,9,10,11,12,13,14]. These systems claim to have many desirable properties including versatility, robustness and low cost. In general, there are two types of modular reconfigurable robots: chain-type and lattice-type (see <http://www.parc.com/modrobots>). Robots that use *chain* reconfiguration form serial chains (like standard robot arms) and connect and reconnect many of these chains together by forming loops. Robots that use *lattice* reconfiguration rearrange their positions by moving from one position on a lattice to a neighboring position on the lattice. Chain reconfigurable robots tend to be more easily applied

to standard robot tasks than lattice reconfigurable robots (such as locomotion and manipulation). However reconfiguration of chain-type robots is much more difficult due to the complexity in reconfiguration sequence planning [1] and six degree of freedom docking [7]. Furthermore, chain-type robots limit the general mechanical scalability in three dimensions; parallel and redundant support structures are explored to alleviate the problem.

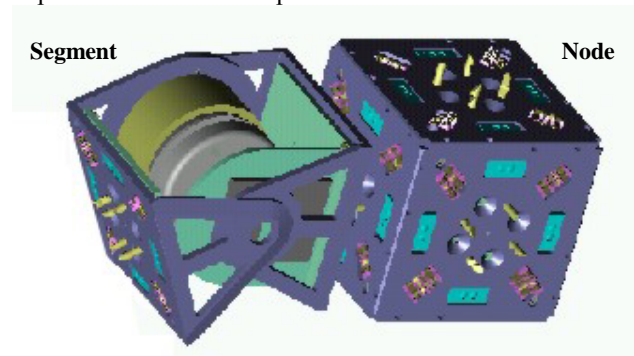


Figure 1. Two types of PolyBot modules

PolyBot [13] is a chain-type modular reconfigurable robot. Three successive generations of PolyBot have been developed at the Palo Alto Research Center; more than 100 modules have been built. One important issue for chain-type modular reconfigurable robots is the scalability with the growing number of modules. In this paper, we present a class of configurations for PolyBot. This class of configurations, called *NML configurations*, is scalable both mechanically and electronically, and can be used to represent three typical types of configuration: snakes, centipedes and multi-loops. Furthermore, there exist predefined reconfiguration sequences between any of the three forms. These can be done with docking operations restricted to 2D planes; making reconfiguration relatively simple. It is also a challenge to generate locomotion gaits when dealing with large numbers of modules. The traditional ways of programming gaits using gait tables [12,13,15] are no longer practical. We have developed a new form of gait generation, called Phase Automata, which are scalable and can be run in a

distributed fashion across the modules. Three types of gaits are discussed, one for each of the three types of configurations. The system has been tested both in simulation with 100+ modules and in hardware with 50+ modules.

The rest of the paper is organized as follows: Section 2 briefly presents the design of PolyBot, in particular, the generation one (G1) and generation three (G3) series. Section 3 discusses NML configurations and the sequences to move between them. Section 4 introduces Phase Automata, a formal model for gait programming, which can be distributed and are scalable with respect to the number of modules.

2. PolyBot Systems

Three distinct generations, and numerous variations of PolyBot have been designed. Most designs consist of two types of modules: a *segment* is a one-degree of freedom module with two connection interfaces and a *node* is a rigid cube with four-six connection interfaces. All the connection interfaces are identical and a pair can be connected in any one of four possible orientations (Figure 1).

The first generation (G1) of PolyBot was by far the most prolific, both in terms of different versions created, and the number of modules actually created in each batch. All members of this family are characterized by their use of commercial off the shelf (COTS) motors with integrated controllers (“hobby servos”) and laser-cut plastic. A total of 120 G1v5 modules (Figure 2) have been built recently, with improved overall robustness and a choice of battery power or plug-in DC-DC converter for running off a high voltage external powered bus.



Figure 2. G1v5 PolyBot segment and chain

G1 modules are simple and robust, but require manual reconfiguration, and the on-board processing is limited. Computation typically takes place in a host and communication amongst the host and modules is via a shared serial bus.

Generations two (G2) and three (G3) have more sensors, (including the IR range sensor), the latch mechanisms (which enables self-docking), powerful on-board computation and communication. Both G2 and G3 use a Motorola PowerPC MPC555 embedded processor with 448K internal flash ROM and 1M of external RAM [13]. The connection plate serves two purposes: to attach two modules

physically together as well as electrically; both power and communications are passed from module to module. Each connection plate has IR photo transistors and IR LEDs. Combinations of IR intensity measurements allow the determination of the relative 6 DOF position and orientation of mating plates. This aids in the closed loop docking of two modules and their connection plates [7,19]. Each module communicates over a global CAN bus with up to 1M bps. The node module is a rigid cube made of 6 connection plates (one for each face). For G3 the node hosts two CPUs, each with two CAN controllers. It serves three purposes: (1) to allow for non-serial chains/parallel structures, (2) to house higher power computation and power supplies, and (3) to perform transparent media access control (MAC) layer bridging between networks. Total of 22 G3 modules have been built and a 6D docking experiment [19] has been successfully conducted (Figure 3).



Figure 3. G3 segment and 6D docking platform

Software architecture for G3 features three layers of communications [16,17,18] with constraint-based embedded computation [20]. The PolyBot systems have demonstrated versatility by showing multiple modes of locomotion and self-reconfiguration.

3. Scalable Reconfigurations

Although it is unclear that the increase in the number of modules implies an increase in functionality and performance, one interesting research question for modular reconfigurable robots concerns scalability: what are the types of configurations that are scalable in terms of mechanical, electrical and software limitations. Mechanically, each module has a maximum torque, which yields a maximum length of a chain free in space. Electronically, the maximum number of modules in a single bus is limited. In G3 case, where only nodes have power supplies, the number of segments connected to a node is limited by the maximum power one node can provide. From software point of view, scalability means that the code size and the memory usage do not grow with the number of modules.

3.1 NML Configurations

In general, a PolyBot system can be represented by a graph, where each vertex represents a module, and each edge represents a connection between two modules. NML configuration is a type of compact parameterized configu-

rations. N is the number of nodes, a positive integer. M is a sequence of segments, parameterized by an array of **Boolean**, e.g., 01101011. The length of the array is the number of the segments, and each element value indicates the global orientation of that segment with respect to the node. Note that in general there are four different orientations, $n\pi/2$, $n=0\dots3$; however since segments are symmetric, inverting a module does not change the functionality of the configuration, it merely alters the sign of the motor's rotation, so a single bit of information suffices. Similar to M , L is another sequence of segments whose composition is specified by an array of **Boolean**. There are three types of NML configuration: centipedes, snakes and multi-loops.

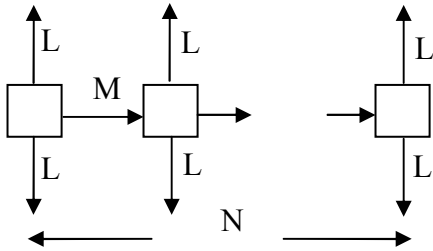


Figure 4. NML-Centipede

An NML-centipede is an N -pair legged centipede, with body configuration M and leg configuration L (Figure 4). Figure 5 shows a configuration of a centipede with $N = 4$, $M = 01$ and $L = 10101$.

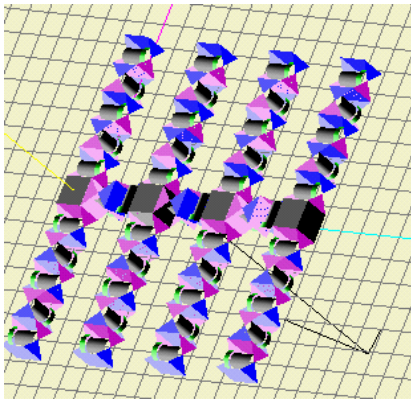


Figure 5. NML-Centipede: $N=4$, $M=01$, $L=10101$

An NML-snake is a linearly connected sequence of modules, with nodes evenly distributed (Figure 6). The left most segment sequence is L and the right most segment sequence is L' (a reversed sequence of L , e.g., if $L = 100$, then $L'=001$). The segment sequence between two nodes is LML' , which is the catenation of sequences of L , M and L' . For example, if $L=100$, and $M=10$, $LML' = 10010001$. Figure 7 shows a configuration of a snake with $N = 4$, $M = 01$ and $L = 10101$.

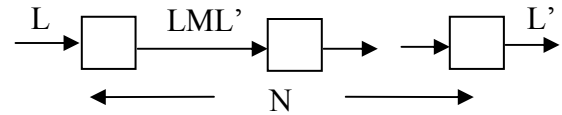


Figure 6. NML-Snake

An NML(K)-loop, where K is a positive number and N is a multiple of K , is a parallel structure with K connected loops (Figure 8). In an NML(K)-loop, nodes are arranged in a $K \times (N/K)$ mesh, the linkage between nodes in a loop is LL' , while the linkage between loops is M . There are side modules M_1 or M_2 , where $M_1M_2 = M$. Figure 9 shows the configuration of a multi-loop with $N = 4$, $M = 01$, $L = 10101$, with $K = 2$.

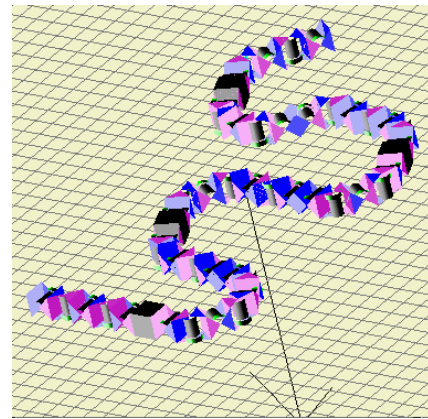


Figure 7. NML-Snake: $N=4$, $M=01$, $L=10101$

NML(K) configurations have the properties that (1) nodes are distributed evenly and (2) the maximum length of a segment chain is bounded, i.e., not growing with the number of modules. These properties guarantee the mechanical and electronic scalability for the type of configuration. Furthermore, there exists a reconfiguration sequence between any of the three types of configuration.

3.2 NML Reconfigurations

Self-reconfiguration for chain-type robots is difficult for two reasons. Firstly, reconfiguration planning between any two configurations is hard, even for a simplified configuration set [1]. Secondly, six-degree of freedom docking of two arbitrary chains in space is a hard in terms of sensing and control [7,19]. One important property for NML configurations is that there exists a predefined sequence between any two NML configurations with the same N , M and L values. In addition, all the reconfiguration steps required can be performed in a 2D plane, which makes reconfiguration significantly easier.

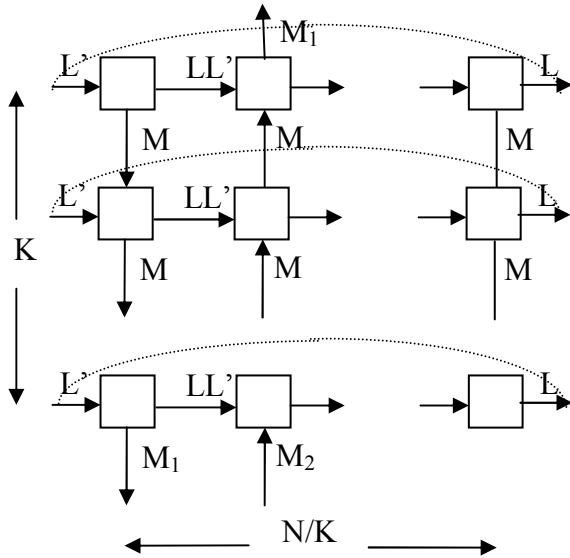


Figure 8. NML(K)-Loop

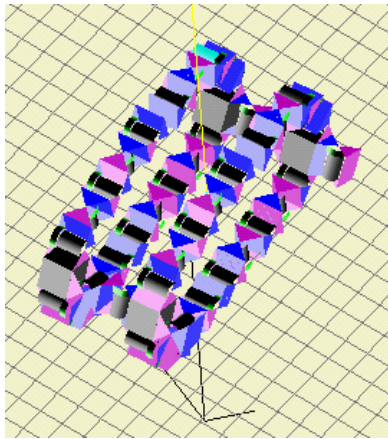


Figure 9. NML(K)-loop: $N=4$, $M=01$, $L=10101$, $K=2$

3.2.1 From a Centipede to a Snake

The reconfiguration from a centipede to a snake is basically a matter of merging the legs into the body as shown in Figure 10.

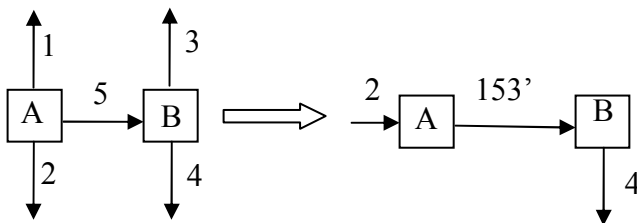


Figure 10. Merging legs into body

Let \mathbf{b} denote the beginning, and \mathbf{e} the ending, of a chain, and let the four sides of a node be denoted as North, South, East and West. The actual sequence from left to right in

Figure 10 is: attach 2e to West of A, detach 2b, **attach 4e to South of A**, detach 5b, attach 1e to 5b, detach 1b, attach 1b to East of A, detach 5e, attach 5e to 3e, detach 3b, attach 3e to West of B, **detach 4e**. The steps in **bold** are necessary to satisfy the constraint that the system must remain in one connected component throughout the reconfiguration. The total number of attach and detach steps for an N node configuration, with $N>1$, is: $4+10(N-1)$.

3.2.2 From a Centipede to a K-Loop

The reconfiguration from a centipede to a K-loop involves three phases (Figure 11). Firstly, $N/K-1$ bends are introduced into the centipede's spine. Now the $(N/K-1)K$ pairs of the legs are joined. Finally, the $N/K-1$ bends in the spine are broken. All the leg attachments may be performed in parallel followed by all the detach steps in parallel. So all can be accomplished in just two steps.

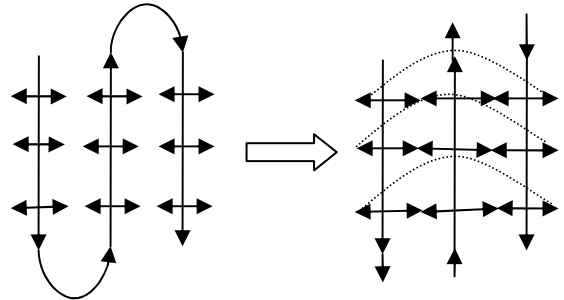


Figure 11. Bending, attaching and detaching

3.2.3 Between any NML Configurations

It is easy to see that there exists a reconfiguration sequence from any NML configuration to any NML configuration. Let S be a reconfiguration sequence from configuration A to B, then a reconfiguration sequence from B to A is S' , where S' is the reversed sequence of S . If S_1 is a reconfiguration sequence from A to B and S_2 is a reconfiguration sequence from B to C, then a reconfiguration sequence from A to C is S_1S_2 .

Note though, that even if a reconfiguration sequence exists between two given configurations, it may not be always executable due to the geometric constraints. For example, a leg with only 2 segments cannot bend $3\pi/2$, since the maximum angle of each segment is $\pi/2$.

4. Scalable Locomotion Gaits

A traditional way of programming locomotion gaits for modular robots is the use of gait tables [12,13,15]. A gait table is a two-dimensional array, with one axis representing modules, and the other axis representing time. Gait tables are simple and easy to handle, however, the size of a gait table grows linearly with increasing number of modules, so from software point of view, gait tables do not provide a scalable solution.

Most locomotion gaits exhibit two useful properties. Firstly, most are periodic - gaits with irregular intervals are the results of sensor (environment)-driven instead of time-driven. Secondly, most modules have identical behavior, subject only to phase delays. Phase Automata, presented here, is a more general and efficient way to represent gaits. Three types of gaits for the three types of NML configurations are described here using Phase Automata.

4.1 Phase Automata

A *Phase Automaton* is a regular event-driven state machine augmented with an initial *phase delay* (generally a real number between 0 and 1). Phase automata are not new in themselves, but their application to modular robotics is. A phase automaton has its own thread of control, which can be started or stopped by an external command. A phase automaton may or may not have finite states; however, the transition from one state to another is discrete, driven by events. A phase automaton can be associated with a set of events; these can be timers or conditional sensory triggers. Different types of events may result different actions and state transitions. The effect of the initial phase delay can be to literally delay of the start of the thread, or else cause the automaton to start at a particular state corresponding to the phase delay (Figure 12).

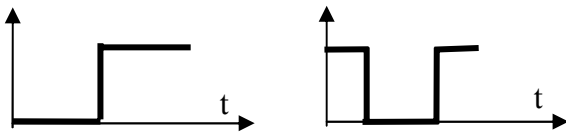


Figure 12. A phase delay of 0.25 for a two-state (0, 1) automaton

A phase automaton can be applied to a single module, or to a set of modules.

4.2 Scalable NML Gaits

For any NML configuration, a segment whose rotation axis is parallel to the ground is termed a *horizontal* segment, and whose rotation axis is vertical to the ground is termed a *vertical* segment.

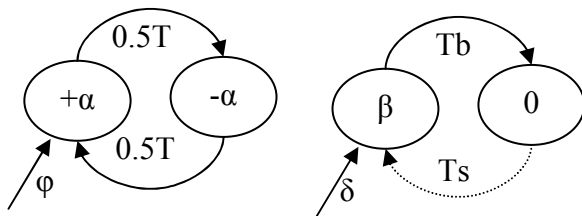


Figure 13. Left: forward; Right: turning

4.2.1 Snake Gait

There are two types of automaton for snake locomotion, one is applied to horizontal segments for forward motion,

and another is applied to vertical segments for turning motion.

The forward motion automaton is applied to horizontal segments (Figure 13 left) and has two states: one for a positive angle and one for a negative angle. The transition between the two states is triggered by time. Let the total cycle time be T , the transition time is $0.5T$. The initial phase delay ϕ varies from segment to segment, and is determined by the position of the horizontal segment in the chain. The difference in phase delay $\Delta\phi$ between any two neighboring horizontal segments is a constant. Figure 14 shows a chain of 12 horizontal segments with $\alpha = 0.25$ and $\Delta\phi = 0.125$. The parameters α and $\Delta\phi$ control the shape of the snake. Given $\Delta\phi < 0.5$, the number of horizontal segments between two ground points is roughly $1/\Delta\phi$. In this case the amplitude of the snake wave is proportional to $\sin\alpha/\Delta\phi$ and the width of the wave is proportional to $\cos\alpha/\Delta\phi$, for small α .

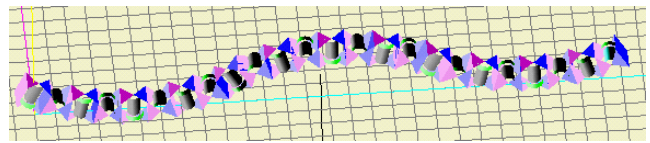


Figure 14. A snake of 12 horizontal segments

The turning automaton is applied to vertical segments (Figure 13 right) and has two states: one is for bending to an angle β and one for strengthening to 0. The transition between the two states can be triggered by time or sensor signals. Turning can be either periodic or non-periodic. For non-periodic turning, T_s is infinite. In this case, the phase delay δ is defined to be the ratio between the actual delay of bending and the time of bending T_b for the segment; in which case, δ can be greater than 1. Assume that the difference in phase delay between neighboring vertical segments is a constant $\Delta\delta$, the radius of the turning, i.e., the number of modules in bending state, is $1/\Delta\delta$, and the total turning angle is $\beta/\Delta\delta$. Figure 15 shows two turning actions, one with radius 3 and total turning angle $\pi/2$ and the other with radius 4, and total turning angle $\pi/3$. Given a required radius R and a total turning angle A , $\Delta\delta$ and β can be obtained: $\Delta\delta = 1/R$; $\beta = A/R$. The bending time T_b is related to the forward motion speed; it can also be sensory driven.

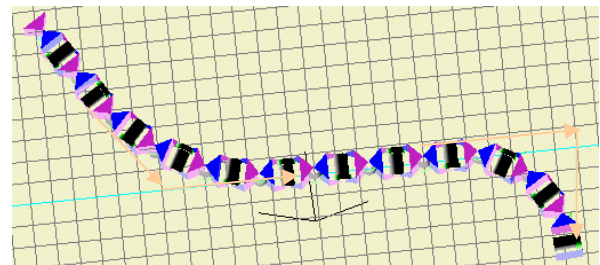


Figure 15. Turning snake

The phase automata can be run independently in each module. The initial phase delays for modules can either be set globally when starting the forward or turning locomotion, or else determined locally if each module knows its position in the chain.

4.2.2 K-loop Gait

Loop locomotion is one of the most efficient motions over flat terrain. As each of the parallel loops in a K-loop is linked at several points, then the motion of each loop must obviously maintain synchronized.

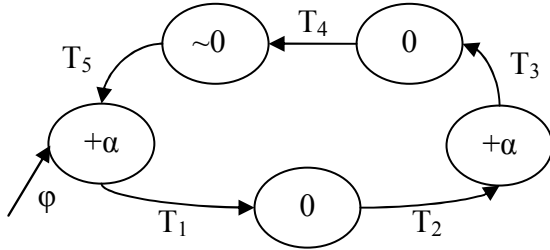


Figure 16. Forward loop automaton

The forward motion loop automaton applies to horizontal segments in a loop and in general has five state transitions (Figure 16): T_1 : head strengthen, T_2 : tail bend, T_3 : tail strengthen, T_4 : head relax (~ 0 denotes a compliant state), T_5 : head bend. The transitions between states are time-driven. The phase delay between neighboring horizontal segments is $1/N$ where N is the total number of horizontal modules. Given that T is the total cycle time for a loop, the transition times for $T_1 = T_3 = (\pi/\alpha)T/N$, $T_2 = (1/2)T - (\pi/\alpha)T/N$, and $T_4 + T_5 = (1/2)T - (\pi/\alpha)T/N$. The ratio between T_4 and T_5 can be varied. A number of modules are always maintained in a compliant state to avoid internal conflict due to mechanical error and the over constrained nature of the loop.

Turning gaits for multi-loop are difficult and have yet to be developed.

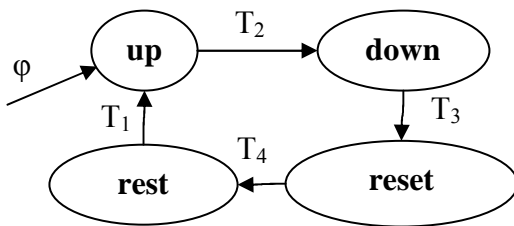


Figure 17. Leg automaton

4.2.3 Centipede Gait

Centipedes are the most flexible of the three configurations in the number of possible gaits. A centipede leg automaton applies to a leg, which is composed of a set of segments. A leg automaton in general has four states (Figure 17). The **up** state moves the leg up and forward, the **down** state

moves the leg down in a forward position, and the **reset** state moves the leg back to its original position, thus shifts the body forward. If the cycle time for a leg is T , then $T_2 + T_3 = (1 - \beta)T$ and $T_4 + T_1 = \beta T$, where β is called *duty cycle*. Given an N -pair legged centipede, β should be greater than or equal to $3/(2N)$, to ensure static stability.

The step height and stride size are other parameters that can be set. While a general inverse kinematics routine could be applied to legs, this is not efficient. Rather, three active joints are selected and a closed form solution is obtained. The selection of joints, called *leg configuration*, results in different gait properties. Figure 18 shows two different leg configurations for the same centipede: the left (*horizontal configuration*) is wider and lower, increasing stability and forward speed, and the right (*vertical configuration*) is thinner and taller, applying a lower torque for the leg.

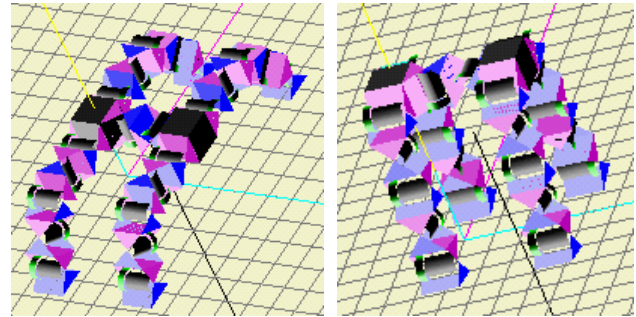


Figure 18: Two different centipede leg configurations

Centipede gaits also vary in terms of different phase delay between legs. In general, the left and right legs of a pair for an N -pair legged centipede always have 0.5 phase difference. Let the left legs be numbered in $2i$, and right legs be numbered as $2i+1$, for $i=0, \dots, n-1$. Three types of this gait (based upon the phase ϕ) have been particularly experimented:

1. **wave**: $\phi[2i] = F(i\beta)$;
2. **equal phase**: $\phi[2i] = 1 - i/N$;
3. **half equal phase**: $\phi[2i] = 1 - i/(2N)$;

In all cases: $\phi[2i+1] = F(0.5 + \phi[2i])$, where $F(x)$ is a fractional part of a real number x .

There are two ways that turning gaits can be implemented. For a centipede with short body length, it is enough to change relative stride sizes of the left and right legs, i.e., for turning left, the stride size for the left legs should be smaller than that for the right legs, and *vice versa*. For a long centipede, a turning gait for the spine should be applied, similar to the snake's turning strategy described in 4.2.1. Figure 19 shows a 14-legged centipede with 55 G1v5 modules ($N=7$, $M=0$, $L=011$).



Figure 19. A 55 module real centipede

5. Conclusions

Scalability is an important issue for modular reconfigurable robots. In this paper, we have explored scalability in terms of configurations and locomotion gaits. We have shown that NML specification is a compact representation of three interchangeable configurations. These special configurations are scalable both mechanically and electronically. We have also illustrated standard reconfiguration sequences between NML configurations. We have developed Phase Automata as a way of programming scalable locomotion gaits and demonstrated the use of Phase Automata to design locomotion gaits for NML-configurations.

Acknowledgements

This work is funded in part by the Defense Advanced Research Project Agency (DARPA) contract #MDA972-98-C-0009.

References

- [1] A. Casal, "Reconfiguration Planning for Modular Self-Reconfigurable Robots", PhD Thesis, Stanford, 2002
- [2] T. Fukuda, S. Nakagawa, "Dynamically Reconfigurable Robotic System," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 1581-1586, 1988.
- [3] K. Kotay, D. Rus, M. Vona, C. McGray, "The Self-reconfiguring Robotic Molecule," *Proc. of the IEEE International Conf. on Robotics and Automation*, pp424-431, May 1998.
- [4] S. Murata, H. Kurokawa, S. Kokaji, "Self-Assembling Machine," *Proc. of the IEEE International Conf. on Robotics and Automation*, pp441-448, May 1994.
- [5] S. Murata, H. Kurokawa, E. Yoshida, K. Tomita, S. Kokaji, "A 3D Self-Reconfigurable Structure," *Proc. of the IEEE International Conf. on Robotics and Automation*, pp432-439, May 1998.
- [6] A. Pamecha, C. Chiang, D. Stein, G.S. Chirikjian, "Design and Implementation of Metamorphic Robots," *Proc. of the 1996 ASME Design Engineering Technical Conf. and Computers in Engineering Conf.*, Irvine, California, August 1996.
- [7] K. Roufas, Y. Zhang, D. Duff, M. Yim, "Six Degree of Freedom Sensing for Docking using IR LED Emitters and Receivers," *Seventh International Symposium on Experimental Robots*, Dec. 2000.
- [8] D. Rus, M. Vona, "Self-reconfiguration Planning with Compressible Unit Modules," *Proc. of the IEEE International Conf. on Robotics and Automation*, pp2513-2520, May 1999.
- [9] K. Tomita et al, "Development of a Self-Reconfigurable Modular Robotic System," *Proc. of SPIE Sensor Fusion and Decentralized Control in Robotic Systems III*, Vol. 4196.
- [10] C. Unsal, P.K. Khosla, "Solutions for 3-D Self-reconfiguration in a Modular Robotic System: Implementation and Path Planning," *Proc. of SPIE Sensor Fusion and Decentralized Control in Robotic Systems III*, Vol. 4196.
- [11] P. Will, A. Castano, W-M Shen, "Robot modularity for self-reconfiguration," *SPIE Intl. Symposium on Intelligent Sys. and Advanced Manufacturing, Proceeding Vol. 3839*, pp. 236-245, Sept. 1999.
- [12] M. Yim, "New Locomotion Gaits," *Proc. of the IEEE International Conf. on Robotics and Automation*, pp. 2508-2514, May 1994.
- [13] M. Yim, D. Duff, K. Roufas, "PolyBot: a Modular Reconfigurable Robot" *Proc. of the IEEE Int. Conf. on Robotics and Automation*, April 2000.
- [14] M. Yim, Y. Zhang, J. Lamping, E. Mao, "Distributed Control for 3D Metamorphosis," *Autonomous Robots 10, special issue on self-reconfigurable robots*, pp41-56, 2001.
- [15] M. Yim, Y. Zhang, D. Duff, "Modular Robots", Cover Story on *February 2002 issue of IEEE Spectrum Magazine*.
- [16] Y. Zhang, K. Roufas, M. Yim, "Software Architecture for Modular Self-Reconfigurable Robots", *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Hawaii, USA, Oct. 2001
- [17] Y. Zhang, K. Roufas, M. Yim, C. Eldershaw, "Massively Distributed Control Nets for Modular Self-Reconfigurable Robots", *2002 AAAI Spring Symposium on Intelligent Distributed and Embedded Systems*.
- [18] Y. Zhang, M. Yim, K. Roufas, C. Eldershaw, D. Duff, "Attribute/Service Model: Design Patterns for Efficient Coordination of Distributed Sensors, Actuators and Tasks in Embedded Systems", *Workshop on Embedded System Codesigns 2002*
- [19] Y. Zhang, K. Roufas, C. Eldershaw, M. Yim, D. Duff, "Sensor Computation in Modular Self-Reconfigurable Robots", *Experimental Robotics VIII*, Advanced Robotics Series, Bruno Siciliano Ed. Springer, 2003
- [20] Y. Zhang, M. Fromherz, L. Crawford, Y. Shang,, "A General Constraint-Based Control Framework with Examples in Modular Self-Reconfigurable Robots", *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Lausanne, Switzerland, Oct. 2002