Transformation of Control in Congruent Self-Reconfigurable Robot Topologies

Jacob Everist, Feili Hou, and Wei-Min Shen
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
Email: everist@usc.edu, fhou@usc.edu, shen@isi.edu

***Abstract*— Much work on self-reconfigurable robotics has been focused on motion planning and physical reconfiguration of the robot. Using the Superbot self-reconfigurable robot, we focus on the details of realizing locomotion gaits given that a single robot topology can be realized in a large number of different ways. That is, each module in the robot topology has 4 symmetric orientations that are functional and shape equivalent. Once a role is selected for each module, such as through the use of hormone-inspired control, each module's role is supplied with a gait template which then must be transformed to suit the local configurations of each module with respect to the global topology. We provide a theoretical framework for which this can be accomplished.**

I. Introduction

Much of the past work in self-reconfigurable robots has been attended to developing methods for reconfiguration planning or gait locomotion. However, little attention has been paid to the number of possible configurations that achieve the same global shape but require different sets of control algorithms. Using the Superbot self-reconfigurable robot system, a single module within a robot topology has 4 symmetric orientations that can achieve the same required function. Most locomotion methods assume a canonical orientation for all the modules in the robot and thus miss the important details of actually implementing the locomotion on all possible permutations of symmetric module configurations.

To understand what we mean, see Figure 1 which shows different views of a Superbot module [15], a type of self-reconfigurable robot. As one can see, there are four different possible module orientations that are functional and shape equivalent. However, to produce the same movement pattern on each of the four orientations requires a different set of motor commands because the direction and location of the physical motors have changed. The software of the control algorithm must transform the motor commands somehow to achieve the functionally equivalent actions. A systematic methodology for achieving this control transform is what we seek in this paper.

If we can find this transformation methodology, we need only specify a single gait template for a given topology. For example, if we want to make a 4-legged walker gait, we need only specify the gait for one instance of the 4-legged walker shape with its composed modules in any given configuration.

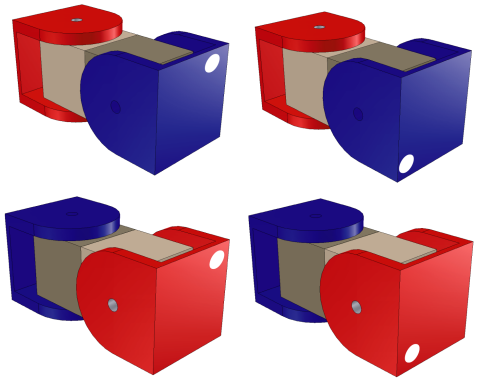

Fig. 1. Superbot Module - 4 Functionally Equivalent Orientations

Then, whenever we make a congruent topology of the 4-legged walker, we simply apply the control transforms for each individual module as it differs from the canonical module configuration.

Using this same methodology, we can transform global robot behaviors if the whole robot is rotated through its symmetric orientations. For instance, if the 4-legged robot in Figure 2 were flipped over, we could transform its global behavior to walk in this upside-down position just as if it hadn't been flipped at all. This gives self-reconfigurable robots the robustness that we have come to expect, being completely functional in the event of a fall or tumble without the need for reconfiguration. This ability only exists in a few fixed form robots such as Whegs [1] or the Scout [2] series of robots.

Conceptually, this approach is easy to understand, but the details of its implementation are difficult to articulate. A systematic methodology for determining transforms in all robot shapes is still elusive. In this paper, we provide a number of basic concepts and some functional examples using the Superbot system that should serve as building blocks to a completely automatic method for deriving control transforms for classes of congruent robot morphologies.

Many different self-reconfigurable robot systems have been proposed. Good survey articles have been written about the field in [3] [4] [5] [6] [7]. Self-reconfigurable robots are either lattice-based or chain-based. Lattice-based systems are fixed

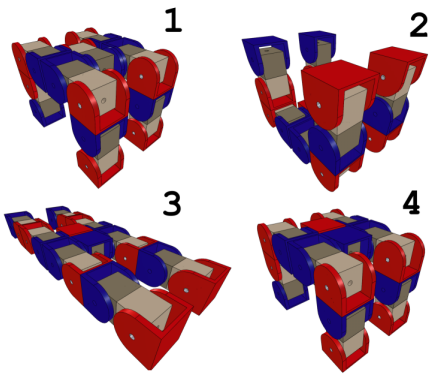

Fig. 2. 4-Legged Robot Recovering from Flip-Over by Transforming Its Control

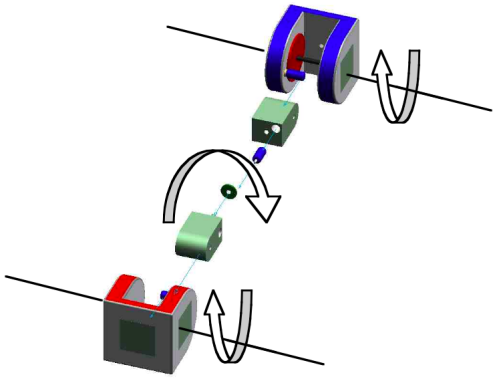

Fig. 3. Superbot - 3 Degrees of Freedom

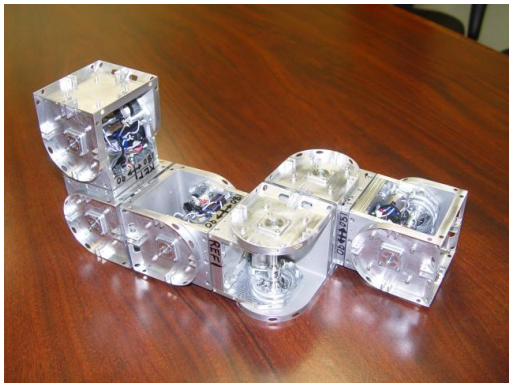

Fig. 4. Superbot - Arbitrary 3-Module Configuration

into geometric relationships with their neighbors and have discrete positions in a lattice structure. Such systems include Crystal [9], Molecule [8], and ATRON [10]. Chain-based systems are robots in tree or loop structures with continuous motor positions. Such systems include Polybot [11], CONRO [13], M-TRAN [14], and Superbot [15].

Various locomotion control methods have been developed for chain-based self-reconfigurable systems which include hormone-inspired control [16], role-based control [17], constraint-based control [19], nonlinear oscillators [21], phase automata [20], gait tables [22], and central pattern generators [18]. An overview of different locomotion methods can be found in [23]. This work derives from the mathematical foundations described in [12].

This work is different from role-selection in hormone-inspired control and role-based control but naturally extends them since they only specify the roles the modules are supposed to take and assume canonical module orientations for the motor commands. We could naturally place a control transformation algorithm on top of role-selection to give hormone-inspired control even more power.

We divide this paper into the following sections. In section 2 we describe the Superbot system and the locomotion algorithm we are using. Section 3 discusses control transforms and physical rotations that achieve shape congruence. Section 4 discusses the symmetries in robot morphologies that show functional and shape congruence. Section 5 illustrates some experiments of control transformations in the Superbot system. Section 6 discusses the experiments and lessons learned. Section 7 concludes and sets the agenda for future work.

II. Superbot and Locomotion Control

The Superbot system is the testbed and subject of our experiments. Each Superbot module is composed of three individual motors as seen in Figure 3. Each module has 6 genderless docking connectors and two onboard ATMega128 microcontrollers. Any connector of a module can be connected to any of another module in 4 different 90 degree rotations about the connector face normal to form arbitrary configurations such as in Figure 4. Inter-module communication is achieved through sets of infrared emitter-detector pairs on each of the docking faces.

For the purposes of this paper, we use the simplest locomotion control method possible which are gait tables. Gait tables are a matrix of motor position values with the columns representing intervals in time and the rows representing different motors in the robot. An example of a simple gait table for a single Superbot module gait is below. The middle row represents the positions of the roll motor.

$$\begin{bmatrix} 0 & 0 & -60 & -60 \\ 0 & 0 & 0 & 0 \\ 0 & -60 & -60 & 0 \end{bmatrix} \quad (1)$$

For multi-module gaits, we simply include large matrices with more rows. For each additional module to the system, we add three more rows. The details of communicating these motor commands to all the modules and synchronizing their actions we omit here but refer the reader to another paper [16] for a current effective method.

For the sake of convenience which will become clear in the next section, we wish to designate two types of modules whose only difference is their roll motor home position. As seen in Figure 5, we wish to call the module whose roll axis is at zero degrees an M-type module and the module whose roll axis is offset by 90 degrees the C-type module. In the current implementation, the roll motor only has 270

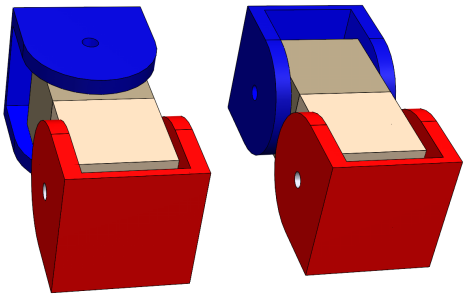

Fig. 5. Modules - C-Type module (left) and M-Type module (right)

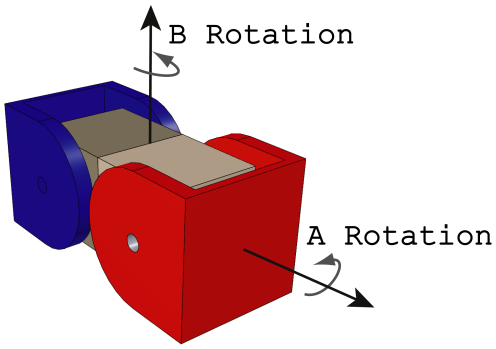

Fig. 6. A and B Rotation Types

degrees of motion, so both the M-type and C-type modules have the freedom to move 90 degrees in either direction from their home position. M stands for M-TRAN which is the self-reconfigurable robot system whose left and right motor axes are parallel as in the M-type module. C stands for CONRO which is the self-reconfigurable robot system whose left and right motor axes are orthogonal as in the C-type module. Superbot can achieve the range of motion of both the M-TRAN and CONRO systems by offseting the roll by 90 degrees.

III. Transforms

First we must define the types of rotations that can occur for a single module. In Figure 6, we define the A rotation and the B rotation. Any combination of the A and B rotations can reach all 4 congruent module shapes. For instance, for the M-type module, we can define the group of congruent modules as $\{e, A_{180}, B_{180}, A_{180}B_{180}\}$ where e is the empty rotation and the other elements are combinations of rotations by 180 degrees. Likewise, for the C-type module, we can define the group of congruent modules as $\{e, A_{180}, A_{90}B_{180}, A_{270}B_{180}\}$. We can see the four cases of shape and functional congruence in the M-type module as shown in Figure 7.

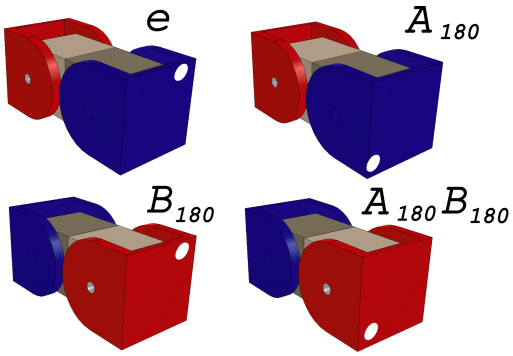

Fig. 7. M-Type Module - 4 Congruent Orientations

Given that we now know all the possible physical orientations for a single Superbot module, we can now demonstrate how control transformations are accomplished with gait tables. All control transformations can be achieved with a simple matrix multiplication. For instance, if an M-type module performing a single module gait were rotated by A_{180}, equivalent motion would be achieved by a selective negation of rows on the gait table:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & -60 & -60 \\ 0 & 0 & 0 & 0 \\ 0 & -60 & -60 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 60 & 60 \\ 0 & 0 & 0 & 0 \\ 0 & 60 & 60 & 0 \end{bmatrix} \quad (2)$$

On the other hand, if an M-type module were rotated by B_{180}, a row permutation would be required:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & -60 & -60 \\ 0 & 0 & 0 & 0 \\ 0 & -60 & -60 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -60 & -60 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -60 & -60 \end{bmatrix} \quad (3)$$

To generalize this concept, we must recognize that there are two distinct classes of situations we want to use control transforms for. One case occurs while the robot is in operation, while one only occurs during self-reconfiguration.

The first case is the situation of global robot reorientation or tumbling. That is, the whole robot has been rotated or flipped from the canonical direction that its gait is setup for. If the robot's current orientation is topologically congruent with the canonical shape, a control transform is possible to get it locomoting again in the desired direction.

The second case is during reconfiguration when the robot has achieved a topology for a given gait. However the orientation of one or more of the modules locally deviates from the canonical shape. Individual control transformations for each of the deviating modules is required to achieve a globally consistent behavior. For instance, in a legged walker, if one of the legs was attached backwards, the desired gait could still be achieved by applying a transform to the leg's local gait table.

We will demonstrate each of these cases in the experiments in section 5.

IV. Symmetries

Different types of symmetry can be discovered for a given robot topology and its associated gait. Understanding the symmetries is helpful for designing custom gaits as well as applying control transforms to achieve the correct behavior in a given robot orientation.

The most important symmetry of a global robot shape is whether or not it has form symmetry under given transformations. That is, under what sets of rotation operations on the robot will the shape be topologically congruent to its original form. If we consider the joints to be "loose" and just compare based on the topology, we can know whether the robot is form symmetric under the given set of rotation operations. If the robot is form symmetric under this rotation, it is possible to perform a control transformation to elicit the equivalent behavior. An example of a form symmetric robot shape is shown in Figure 8. Form symmetry implies that the robot has shape congruence under the given rotation operations.

Of the robot topology and associated rotation operations that are form symmetric, some of them may still produce the same behavior if the gait tables remain untouched. We call this type of phenomenon task symmetry, in that the robot will still move in the exact same direction. No control transformations are required if the robot happens to be put in this situation. Task symmetry implies that the robot has shape and functional congruence under the given rotation operations.

Of the form symmetries that are not task symmetric, their shapes and associated rotation operations are called task asymmetric. That is, if the robot is put under the given rotation, it will not produce the equivalent behavior using the same gait table. A control transform is required for the equivalent behavior to be produced. Task asymmetry implies that the robot has shape congruence but not functional congruence under the given rotation operations.

The family of symmetries can be visually represented as sets with Venn diagrams as seen in Figure 9. The entire space is the set of all topologies T transformed by the group of all rotations $SO(3)$. Only a small set of possible topologies and associated rotations are form symmetric. Of this form symmetric set, only a smaller subset is task symmetric. The larger remainder is task asymmetric. Our control transformations are mostly focused on the task asymmetric remainder.

V. Experiments

In this section we demonstrate control transformations on three different Superbot gaits and examine all the different symmetries that they have. Each of these gaits has been tested in simulation and on real Superbot hardware. We tested the control transformations in simulation using Open Dynamics Engine [24].

A. Single Module Gaits

The Push-Pull gait is a single module gait using the M-type module. The gait is demonstrated in Figure 10. We can see by inspection that the Push-Pull gait is form symmetric under the rotations $\{e, A_{180}, B_{180}, A_{180}B_{180}\}$. Since the gait

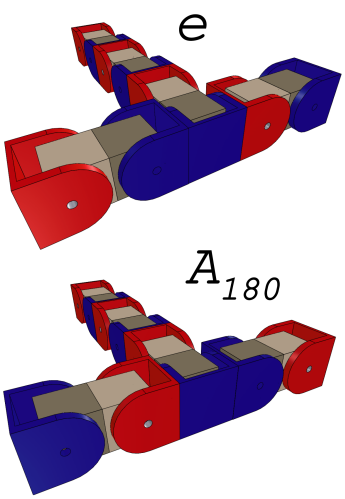

Fig. 8. Form Symmetric Configuration over A_{180}

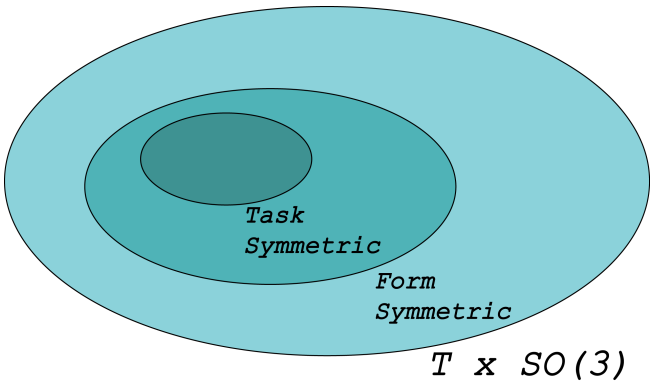

Fig. 9. Set representation of symmetric topologies in the space of all topologies transformed by the group of all rotations $SO(3)$.

is asymmetric in nature, that is, it is pushing and pulling in one direction against the ground, the Push-Pull gait has no task symmetry. Therefore, it is fully task asymmetric. The gait table is the same as equation 1.

$$\begin{bmatrix} 0 & 0 & -60 & -60 \\ 0 & 0 & 0 & 0 \\ 0 & -60 & -60 & 0 \end{bmatrix} \tag{4}$$

The control transformations for each of the associated rotations, $\{e, A_{180}, B_{180}, A_{180}B_{180}\}$, are respectively:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \tag{5}$$

The Salamander gait is a single module gait also using the M-type module. The gait is demonstrated in Figure 11. We can see by inspection that the Salamander gait is also form symmetric under the rotations $\{e, A_{180}, B_{180}, A_{180}B_{180}\}$. However, unlike the Push-Pull gait, its gait is not necessarily asymmetric and under experiment, we find that the Salamander

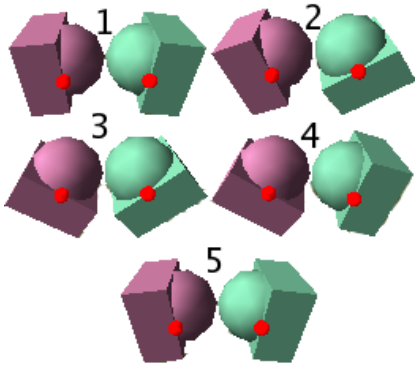

Fig. 10. Push-Pull Gait

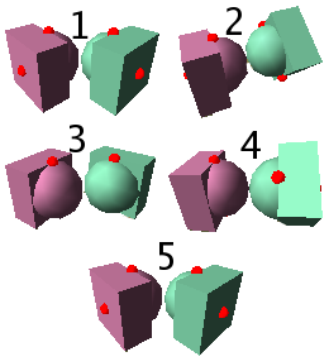

Fig. 11. Salamander Gait

is task symmetric under the rotations $\{e, A_{180}B_{180}\}$. The gait table for the Salamander is below:

$$\begin{bmatrix} 0 & 45 & 0 & -45 \\ 45 & 0 & -45 & 0 \\ 0 & -45 & 0 & 45 \end{bmatrix} \tag{6}$$

The control transformations for each of the associated rotations, $\{e, A_{180}, B_{180}, A_{180}B_{180}\}$, are respectively:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

The reason these transforms differ from the Push-Pull gait transforms are two-fold. In the B_{180} transform for the Push-Pull gait, a permutation of roles between motors is necessary because one must do the "pull" and the other must do the "push". However, the ambulatory motion of the salamander is more symmetric between motors and a mere negation of the motor positions is required. For the $A_{180}B_{180}$ transform, the Push-Pull gait requires a permutation of roles and a negation because it is asymmetric in this rotation, but the Salamander gait has task symmetry under this rotation, so only the identity matrix is required. These are unique properties of these two gaits which must be predetermined in order to create a proper control transformation matrix.

B. Two-Module Gaits

The 2-Creep gait is composed of two C-type modules connected at their tails to perform a type of sideways crawling gait. An example of the 2-Creep gait can be seen in Figure 12. We can see by inspection that the 2-Creep gait is form symmetric under the rotations $\{e, A_{180}, B_{180}, A_{180}B_{180}\}$. 2-Creep has no task symmetry. The gait table for the 2-Creep is below:

$$\begin{bmatrix} 20 & -20 & -20 & 20 \\ 0 & 0 & 0 & 0 \\ -60 & -60 & 60 & 60 \\ -20 & 20 & 20 & -20 \\ 0 & 0 & 0 & 0 \\ -60 & -60 & 60 & 60 \end{bmatrix} \tag{8}$$

For the sake of space, we only wish to show the control transformations under the rotations $\{A_{180}, B_{180}\}$. The transformation under A_{180} is simply $-I$ where I is the identity matrix. The transformation under B_{180} is:

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

Finally, we must consider the 2-Creep gait should one of the modules not be attached in the canonical orientation. Consider the rightmost module in Figure 12 and if it were rotated by A_{180}. This would mean that the gait table for the rotated module would be changed but not the other. The control transform for the whole gait table would be:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{10}$$

Likewise, if the rightmost module were rotated by B_{180}, switching its head and tail, the control transform would be:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \tag{11}$$

This results in the head and tail of the rightmost module switching their roles to accomodate their new position.

VI. Discussion

One thing that can be clearly seen from the transition from 1-module gaits to 2-module gaits are the required transform matrices grow in $O(n^2)$. This further requires multiplication of these matrices, and this system clearly does not scale with even moderately large topologies. On the other hand, the transform matrices are very sparse and easily represented with elementary operations such as permutations and negations

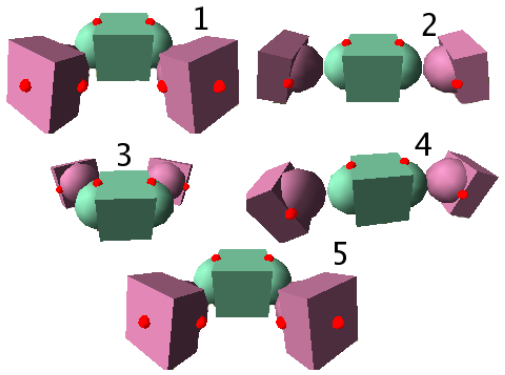

Fig. 12. 2-Creep Gait

of rows. Computational trimming is possible which would significantly increase the useful range of topologies for this system.

We also illustrated the two distinct situations in which one would want to make a control transform. The first being that the whole robot has changed its orientation because of a fall or because of its initial deployment. The other case for needing a control transform would be that at least one module is out of the canonical orientation specified by the template for the particular locomotion pattern. This allows us to avoid wasting time doing needless reconfiguration when a simple software fix is possible to make the locomotion pattern work.

We also notice that in all the control transforms, there are only two basic operations being used. These are the permutation and negation of rows. It is quite interesting that we can achieve so much with only these two elementary operations. However, this type of power is only possible when we segregate modules into C-type and M-type. The only difference between these two modules is the 90 degree offset of the home position on the roll motor. If we did not do this, we would have 90 degree offsets to transform and matrix multiplication is not capable of addition. Keeping the gait tables centered around 0 degrees allows us to achieve most transformations with simple permutation and negation operators.

Finally, detecting the global orientation of the robot and local orientation of its individual modules requires sensors. Global orientation can be detected with 3-axis accelerometers such as those currently on the Superbot system. These give us a best estimate of the direction of gravity and guidance for which control transform to perform for the current orientation. Individual module orientation in the local configuration can be detected through dock sensors and by whom our neighbors are. Alternatively, in place of dock sensors, 3-axis accelerometers can also be used and subtracted with the neighbor's readings to detect relative orientation.

VII. Conclusion

In this paper we examined the neglected topic of how to achieve locomotion for all possible configurations of a single canonical shape without needless physical reconfiguration. This includes all the forms of global robot orientation as well as all the forms of local module orientation. We then provided a theoretical framework for transforming template gait tables to meet the real configuration. We examined various types of symmetry in robot form and locomotion patterns and what they mean for achieving control. We demonstrated with a set of real Superbot locomotion gaits how our approach would work.

Future work will involve designing an algorithm which will automatically create control transforms for any arbitrary shape and locomotion gait. Furthermore, we wish this system to be applicable for locomotion strategies beyond gait tables such as harmonic oscillators, central pattern generators, and coupled nonlinear oscillators. Hopefully, further in-depth study of this topic will yield a more insightful theory on shape and locomotion in self-reconfigurable robots.

VIII. Acknowledgements

This research is supported in part by NASAs Cooperative Agreement NNA05CS38A, and in part by US Army Research Office under the grants W911NF-04-1-0317 and W911NF-05-1-0134. We are also grateful for other members in our Polymorphic Robotics Laboratory, such as Mark Moll, for their useful comments on earlier drafts of the paper.

References

[1] R.D. Quinn, G.M. Nelson, R.J. Bachmann, D.A. Kingsley, J. Offi, and R.E. Ritzmann, "Insect Designs for Improved Robot Mobility," In *Proc. of Climbing and Walking Robots Conference (CLAWAR01)*, Karlsruhe, Germany, Sep. 2001.

[2] D. F. Hougen, S. Benjaafar, et. al., "A Miniature Robotic System for Reconnaissance and Surveillance," *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pp. 501-507, San Francisco, CA, April 2000.

[3] D. Rus, Z. Butler, K. Kotay, and M. Vona, "Self-Reconfiguring Robots," *Communications of the ACM*, vol. 45, no. 3, pp. 39-45, Mar. 2002.

[4] M. Yim, Y. Zhang, and D. Duff, "Modular Robots," *IEEE Spectrum*, vol. 39, no. 2, pp. 30-34, 2002.

[5] P. Jantapremjit and D. Austin, "Design of a Modular Self-Reconfigurable Robot," in *Proc. 2001 Australian Conf. on Robotics and Automation*, Sydney, Australia, Nov. 2001, pp. 38-43.

[6] D. Mackenzie, "Shape Shifters Tread a Daunting Path Toward Reality," *Science*, vol. 301, no. 5634, pp. 754-756, Aug. 2003.

[7] W.-M. Shen and M. Yim (editors), "Special Issue on Self-Reconfigurable Modular Robots," *IEEE Transaction on Mechatronics*, vol. 7, no. 4, 2002.

[8] K. Kotay and D. Rus, "Locomotion Versatility Through Self-Reconfiguration," *Robotics and Autonomous Systems*, vol. 26, pp. 217-232, 1999.

[9] D. Rus and M. A. Vona, "Crystalline Robots: Self-Reconfiguration with Compressible Unit Modules," *Autonomous Robots*, vol. 10, pp. 107-124, 2001.

[10] M.W. Jorgensen, E.H. Ostergaard, H.H. Lund, "Modular ATRON: Modules for A Self-Reconfigurable Robot," In Proceedings of IEEE/RSJ International Conference on Robots and Systems (IROS), pps. 2068-2073, Sendai, Japan, 2004.

[11] M. Yim, D. G. Duff, and K. D. Roufas, "PolyBot: A Modular Reconfigurable Robot," in *Proc. 2000 IEEE Intl. Conf. on Robotics and Automation*, 2000, pp. 514-520.

[12] F. Hou, W.-M. Shen, "Mathematical Foundation for Hormone-Inspired Control for Self-Reconfigurable Robotic Systems," *Proc. 2006 IEEE Intl. Conf. on Robotics and Automation*, May 2006, Orlando, USA.

[13] A. Castano, W.-M. Shen, and P. Will, "CONRO: Towards Deployable Robots with Inter-Robots Metamorphic Capabilities," *Autonomous Robots*, vol. 8, no. 3, pp. 309-324, June 2000.

[14] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji, "M-TRAN: Self-Reconfigurable Modular Robotic System," *IEEE/ASME Trans. on Mechatronics*, vol. 7, no. 4, no. 4, pp. 431-441, Dec. 2002.

[15] B. Salemi, M. Moll, and W.-M. Shen, "Superbot: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System," in *Proc. 2006 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2006 (submitted).

[16] W.-M. Shen, B. Salemi, and P. Will, "Hormone-Inspired Adaptive Communication and Distributed Control for CONRO Self-Reconfigurable Robots", IEEE Transactions on Robotics and Automation, Volume 18, Issue 5, Oct. 2002.

[17] K. Stoy, W.-M. Shen, P. Will, "Using Role-Based Control to Produce Locomotion in Chain-type Self-Reconfigurable Robots," IEEE Transactions on Mechatronics, Volume 7, Issue 4, pps. 410-417, Dec. 2002.

[18] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, S. Kokaji, "Distributed Adaptive Locomotion by a Modular Robotic System, M-TRAN II (From Local Adaptation to Global Coodinated Motion using CPG Controllers)," Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, 2004.

[19] Y. Zhang, M. Yim, C. Eldershaw, D. Duff, K. Roufas, "Phase Automata: a programming model of locomotion gaits for scalable chain-type modular robots," Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, USA, 2003.

[20] Y. Zhang, M.P.J. Fromherz, L.S. Crawford, Y. Shang, "A General Contraint-Based Control Framework with Examples in Modular Self-Reconfigurable Robots," Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, Oct 2002.

[21] D. Marbach, "Evolution and Optimization of Central Pattern Generators for Modular Robot Locomotion", Masters Thesis, Lausanne, Switzerland, 2005.

[22] M. Yim, Y. Zhang, D. Duffy, "Exploring and Programming Modular Reconfigurable Robotics Tutorial", http://www2.parc.com/spl/projects/modrobots/chain/polybot/IROS.html, Las Vegas, 2003.

[23] J. Everist, and W.-M. Shen, "Comparative Study of Locomotion Methods for Modular and Self-Reconfigurable Robots", In *Proc. of Eighth International Symposium on Distributed Autonomous Robotic Systems (DARS)*, Minneapolis, MN, July, 2006 (submitted).

[24] R. Smith, Open Dynamics Engine, http://ode.org/.