Comparative Study of Locomotion Methods for Modular and Self-Reconfigurable Robots

Jacob Everist[1] and Wei-Min Shen[2]

[1] Information Sciences Institute, University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90007 everist@usc.edu
[2] Information Sciences Institute, University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90007 shen@isi.edu

Abstract - Modular and self-reconfigurable robots can take any form by connecting modules together in different configurations. Each module has its own set of motors and due to the high amount of redundancy and distributed processing, many different methods have been developed to control them. We extract some basic low-level methods that are common to all locomotion strategies developed by researchers. We compare and contrast them to see their advantages and disadvantages.

1 Introduction

Modular and self-reconfigurable robots are composed of many homogeneous building blocks that each contain one or more actuated degrees of freedom. The modules can be assembled together in any arbitrary configuration and self-reconfigurable robots have the ability to change their configuration autonomously. Each module has its own on-board computer and ability to communicate with other modules in the configuration. Also, due to the reconfigurable nature of modular robots, the kinematics have a great deal of error introduced by tolerance in the connectors and compounded uncertainty of motor positions. Error, high degrees of freedom, and distributed processing make it difficult to use simple inverse-kinematic control algorithms for global control.

Some examples of self-reconfigurable and modular robots are the CONRO [17] self-reconfigurable robots seen in figure 1a, the new Superbot system seen in figure 1b, MTRAN [7], YaMoR [2], ATRON [15], and PolyBot [16].

Many different approaches have been taken to the problem of modular and self-reconfigurable robot control. Most methods focus on the problems of reconfiguration planning or robot locomotion. We will focus on only locomotion applications in this paper. We wish to closely analyze the properties of the locomotion methods currently being used. We wish to come up with

(a) **(b)**

Fig. 1. Self-Reconfigurable Robots - a) CONRO, b) Superbot

definitive answers about what methods are good in what situations and when they should and should not be used.

Many high-level control frameworks have been developed in the literature. Digital hormone control [5] is a software framework which provides reconfigurable topology discovery, modular behavior synchronization, and distributed decision-making support. The key feature is the passing of "hormone messages" that modulate the robot behavior and assist in synchronization. Role-based control [6] is a subset of digital hormone control which consists only of a synchronization method with carefully delayed communication messages and a small set of possible module roles. The roles are decided by where the module finds itself in the robot morphology such as being a leg, a head, or a tail. Constraint-based control [1] methodology is also a software framework for designing new control strategies. It distributes various services across different modules posing everything as a constraint problem where operations are either control regulators or state estimators and planning or problem-solving select the next motor positions.

We choose to analyze only the low-level locomotion algorithms that are subsets or independent of the high-level methods. Low-level locomotion methods are responsible for specifying motor positions of the modules at any given time and are usually computationally simplistic, repetitious, and quick. These include harmonic oscillators, nonlinear coupled oscillators, central pattern generators, gait tables, and phase automata. These methods will be discussed in detail in section 2.

Section 3 discusses our experimental evaluation of the locomotion methods. Section 4 introduces our evaluation metrics for comparison of the methods. Section 5 discusses the evaluation with metrics and comparison, and section 6 concludes the paper.

2 Locomotion Methods

We now discuss the five basic low-level locomotion methods we will be analyzing.

2.1 Gait Tables

Gait tables are essentially lookup tables of motor positions over time. They are designed or generated and loaded a priori to the robot. Each module looks up its desired motor position at each time interval and cycles through the tables as necessary for a continuous locomotion pattern. An example of a caterpillar locomotion pattern can be seen below:

$$\begin{vmatrix} 0 & 0 & -60 & -60 \\ 0 & 0 & 0 & 0 \\ 0 & -60 & -60 & 0 \end{vmatrix}$$

Gait tables have been used in the CONRO digital hormone experiments [5] and the Polybot system can use gait tables that are interactively created by the user [19]. Gait tables are almost the simplest locomotion method to use to do whatever task is desired.

2.2 Harmonic Oscillators

Harmonic oscillators are standard sinusoidal curves over time. Each module only needs three parameters for each motor: frequency, amplitude, and phase offset. This completely describes any sinusoidal curve that may be desired. Each module should have it's own set of parameters, oscillating at different phase offsets. Frequency is usually the same across all modules and is often proportional to locomotion speed. The simple equation is as follows:

$$x = A\sin(2\pi ft + \phi) \tag{1}$$

Harmonic oscillators are commonly used as a baseline method to compare against when developing new locomotion strategies. They are used as primitives in role-based control [6] and were tested in the Marbach experiments [2].

2.3 Central Pattern Generators

Central pattern generators are artificial neural network models of how vertebrate motor control systems work. The primary neural circuit used in modular robots has been the model developed by Matsuoka [10] and implemented on modular robots by Kamimura [7] [8]. Other neural network control systems such as Gas Networks [13], Continuous-Time Recurrent Neural Networks [11],

and Plastic Neural Networks [12] have potential as modular robot control methods but have never been tried.

The model used in the MTRAN is described by the following coupled equations:

$$\tau \dot{u}_{1i} = -u_{1i} - w_0 * y_{2i} - \beta * v_{1i} + u_e + f_{1i} + a * s_{1i} \tag{2}$$

$$\tau' \dot{v}_{1i} = -v_{1i} + y_{1i} \tag{3}$$

$$y_{1i} = max(0, u_{1i}), i = 0, \cdots, num - 1 \tag{4}$$

$$\tau \dot{u}_{2i} = -u_{2i} - w_0 * y_{1i} - \beta * v_{2i} + u_e + f_{1i} + a * s_{2i} \tag{5}$$

$$\tau' \dot{v}_{2i} = -v_{2i} + y_{2i} \tag{6}$$

$$y_{2i} = max(0, u_{2i}), i = 0, \cdots, num - 1 \tag{7}$$

Coupling between the different CPG systems is accomplished with the s variables computed by the following:

$$s_{1i} = 2.0 * \left\{ 1 + exp\left(-\sum_j (weight_{ij} * u_1 j)/num \right) \right\}^{-1} - 1.0 \tag{8}$$

$$s_{2i} = 2.0 * \left\{ 1 + exp\left(-\sum_j (weight_{ij} * u_2 j)/num \right) \right\}^{-1} - 1.0 \tag{9}$$

Since this is a differential system, the equations need to be solved with a numerical method such as Runge-Kutta which creates a heavy workload. The equations need to be called at at least 15ms intervals to create desirable behaviors. Further explanation of these equations is provided in [7]. MTRAN is the only known modular robot system to use CPGs as a locomotion method.

2.4 Nonlinear Coupled Oscillators

Nonlinear coupled oscillators are simplified models of CPGs and have been in use by Ijspeert [4] and others [2] [3]. They have some nice synchronization properties which make them well suited for modular robots. With the recent work of Marbach [2] to easily specify phase offsets, nonlinear coupled oscillators have also become intuitive to use.

Nonlinear couple oscillators are described by the following differential system:

$$\tau \dot{x}_i = v_i \tag{10}$$

$$\tau \dot{v}_i = -\alpha \frac{x_i^2 + v_i^2 E_i}{E_i} v_i - x_i + p_i \tag{11}$$

Where the coupling terms between oscillators are:

$$p_i = \sum_j \left(\frac{a_{ij}x_j + b_{ij}v_j}{\sqrt{x_j^2 + v_j^2}} \sqrt{a_{ij}^2 + b_{ij}^2} - \frac{v_i}{\sqrt{x_i^2 + v_i^2}} \right) \tag{12}$$

a and b are the cartesian coordinates in phase space that describe the converging phase offset between two nonlinear coupled oscillators. One of the unique properties of these oscillators is that they converge to a stable synchronous limit cycle specified by the phase offsets. These oscillators converge even if the communication is intermittent. This makes them particularly suited to distributed robotics applications.

2.5 Phase Automata

Phase automata [9] are event-driven automata with the ability to have phase delays between different automata. Phase automata provide some decision making support and facilities for sensor input by treating everything as an event and having appropriate responses. The full description of a phase automata is provided by a finite state machine, phase offsets between other phase automata, and the initial conditions. An example of a phase automata performing a caterpillar gait is shown figure 2:

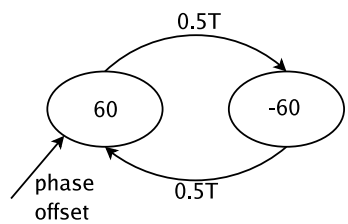

Fig. 2. Phase Automata - Basic Caterpillar Gait

The phase offset varies depending on which segment in the caterpillar it is in. According to the implementation in [9], it appears that you need a centralized system to control the robot. No distributed control method is prescribed for phase automata but it appears with some extra work, it could be transplanted to a distributed algorithm.

3 Experiments

We implemented each of these methods except phase automata on the Superbot self-reconfigurable robot system. Most of our experiments were performed in the Open Dynamics Engine [18] simulating the Superbot hardware. We present them here.

3.1 Gait Tables

Gait tables were relatively straight forward to implement. You merely need to create a matrix with the desired gaits within them and an algorithm to access the matrix members at the right time. The period between motor positions was about half a second and generally had four positions per motor. So, for a 3 motor module like the Superbot, this requires a 3x4 matrix. Gait tables required 247 SLOC (Source Lines Of Code).

An example plot is shown in figure 3a. The robot starts in the initial position of (pi/2,0) and the motors gradually converge to the desired steady state prescribed by the gait tables. One can see brief periods of stability when the motors reached the commanded position before the next gait phase.

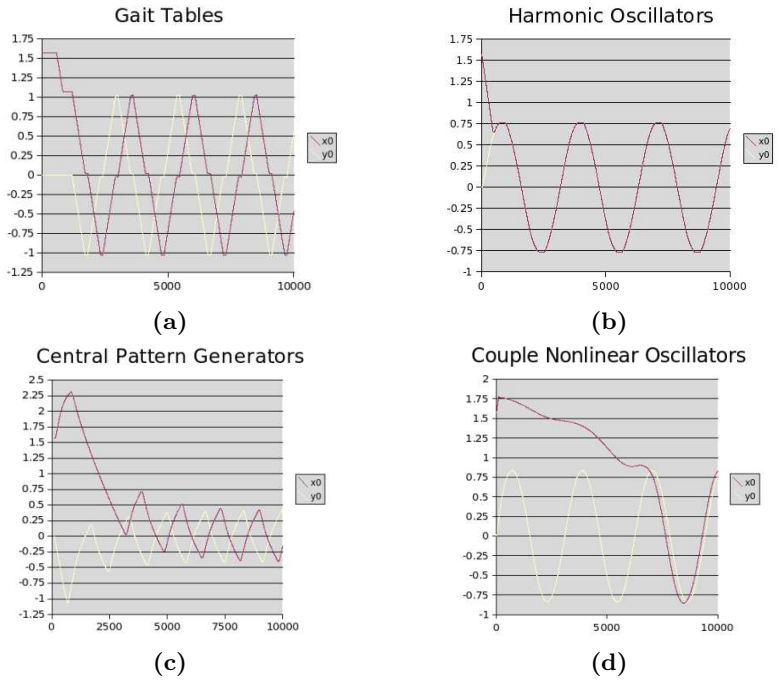

Fig. 3. Motor Plots - a) Gait Table, b) Harmonic Oscillator, c) CPG, d) Nonlinear Oscillators

3.2 Harmonic Oscillators

Harmonic oscillators were also very easy to implement. This merely required calling a sine function with the time input increasing over time. The code size

and effort was comparable to gait tables at 243 SLOC. The example plot is shown in figure 3b.

3.3 Central Pattern Generators

Central pattern generators were by far the most difficult locomotion method to implement due to the complexity and sheer number of variables to deal with. Also, how the CPGs work is difficult to comprehend, so when they are not producing the expected behavior, as happened in our case, it is difficult to debug the problem. CPGs require integration with the 4 point Runge-Kutta method, executed at least every 15ms which is difficult to do with simple microcontrollers. More high speed microcontrollers are needed such as the ATMega128 at 16MHz which is used on the Superbot modules. CPGs required the most effort and code at 1169 SLOC.

CPGs have some stability issues, and it is difficult to hand program gaits with them, so we were only able to demonstrate their use with small module gaits. An example motor plot is shown in figure 3c. As can be seen, some time is required for the motor patterns to converge to a stable limit cycle. In the case of the CPG, this is because the output is in motor torques instead of motor positions.

3.4 Nonlinear Coupled Oscillator

Nonlinear oscillators were also difficult to implement. However, they were far easier to understand and thus easier to debug. Nonlinear oscillators suffer from the longest time to converge to a steady state locomotion gait, but they make it up with their novel and robust synchronization method that requires only intermittent local neighbor communication. Nonlinear oscillators also require a numerical integration method. The Euler numerical method usually suffices for this system and makes it easy to load into microcontrollers. The coding effort was about half as difficult as the CPGs at 776 SLOC.

An example motor plot is shown in figure 3d. As can be seen, the nonlinear oscillator takes the longest time to converge but makes up for it in its stability and robustness. The two motors dont completely converge to the same trajectory because one is slightly delayed behind the other. This a fundamental limitation in the synchronization method which could probably be rectified with faster communication and shorter integration time steps.

4 Evaluation Metrics

We have devised a number of metrics or features in which to compare the different low-level locomotion methods. These features were selected based on various research claims and arguments about why these locomotion methods

should be used. Other features were created based on the practical considerations encountered during our experience implementing four of the five methods. The evaluation of these methods are necessarily subjective based upon control experiments and understanding of the literature. However, a comprehensive evaluation like this has never been performed before, so we believe it has value.

We separate the evaluation metrics into 3 different categories: Requirements, Features, and Miscellaneous. Requirements are the resource requirement necessary to effectively run the locomotion method. Features are a general set of metrics that are basic to distributed control algorithms. Miscellaneous are other metrics that dont fit into the previous two categories. We explain each metric and why it was chosen.

4.1 Requirements

Computation requirements are a measure of the necessary components and computational time needed to execute the locomotion method. Some methods are more intensive than others since they require numerical integration techniques while others are simple quick computations or table lookups.

Data requirements are the memory and storage requirements for a locomotion algorithm. It is also a measure of the number of constants that need to be specified by the programmer. These include things like amplitude, frequency, and phase offset among others.

Communication requirements are a measure the necessary communications required between modules. The amount and type of communication varies between locomotion methods and are usually required for locomotion synchronization. Modular robot locomotion is not effective unless the modules are synchronized in their actions. Some broadcast global synchronization signals while others require a communications bus or need only local communications.

Synchronization is a measure of how quickly the locomotion methods will synchronize the module behaviors. Some are capable of instantaneous synchronization once a signal has been broadcast, while the methods that require numerical integration take some time to converge to stable limit cycles.

4.2 Features

Communication failure fault tolerance is a measure of how well a locomotion method performs under intermittent communication failures or congestion. All locomotion methods will fail under complete communications loss, but it is important to know how each method will fail and how badly.

Stability is a measure of how stable the underlying locomotion algorithms are. Whenever dealing with complex systems, there is always the risk of instability, particular in the methods that require numerical integration techniques.

Scalability is a measure of how well the locomotion algorithm will scale up with the number of modules in the robot. The limiting factor of scalability in

these algorithms is the communication requirement and therefor this metric closely corresponds to the communication requirements.

Distributivity is a measure of how well distributed the locomotion control is. Most methods do local computation, but some require global communication while others only need local communication. In all cases, some amount of communication is required to synchronize which is the upper limit in distributivity.

4.3 Miscellaneous

Biological metric is simply an indicator of whether or not the locomotion method is biologically inspired or not.

Sensor feedback indicates whether or not the locomotion algorithm makes use of sensor feedback. This sensor feedback is usually the motor position over time, but can extend to general sensor input.

Gait transition metric is how the locomotion algorithm can transition from one gait to another. That is, whether or not it can make smooth transitions or whether the transition is abrupt. This can cause jerkiness in the robot and cause falls if not carefully monitored.

Ease of implementation metric is the degree of difficulty in implementing the locomotion algorithm on ones own robot hardware. This metric is subjective to our own experience, and we believe this will hold for others as well.

Understandability is the degree to which a locomotion algorithm is human-understandable and how easy it is to hand-design locomotion gaits. This is also closely correlated with ease of implementation since understandability is key to smooth implementation.

Output commands are the form of motor commands provided by the locomotion algorithm. These are either motor position, motor velocity, or motor torque.

Optimizability is a measure of how well-suited locomotion gaits are to optimization algorithms such genetic algorithms or reinforcement learning. Our results for this metric are a result of reading the literature and also one of the results of our experiments.

5 Method Evaluation

The results of our evaluation can be seen in table 1.

It's clear from the previous section that these features are not all independent from each other. There is a great deal of interdependence especially with regards to communication, stability, synchronization, and failure. Any loss of communication will lead to the eventual failure of any of the control methods. The only difference is when the failure will occur and how catastrophic it will be.

Table 1. Locomotion Method Evaluation

Attribute	**Gait Table**	**Harmonic Oscillator**	**Nonlinear Oscillator**	**CPG**	**Phase Automata**
Computation	Lookup Table	Sine/15ms	RK4/15ms	RK4/15ms	FSM
Data	Gait Table	f, a, ϕ	f, a, ϕ, $O(n-1)$ cpl. terms	9 consts., IC, $O(n^2)$ cpl. terms	Transition Table
Communication	global synch signal	global synch signal	local comm.	bus comm.	bus comm.
Synchronization	neg.	neg.	1-20s, $O(n)$	1-6s, $O(1)$	neg.
Comm. Failure Fault Tolerance	clock drift	clock drift	clock drift	unstable!	alg. failure
Stable	yes	yes	divide by zero	no	yes
Scalable	$O(1)$	$O(1)$	$O(1)$	$O(n^2)$	$O(n)$
Distributed	mostly	mostly	yes	yes	no
Biological	no	no	yes	yes	no
Sensor Feedback	no	no	motor position	motor position	event triggers
Gait Transitions	abrupt	abrupt	smooth	smooth	abrupt
Ease of Implementation	easy	easy	medium	hard	?
Understandable	easy	easy	medium	hard	easy
Output Commands	position	position	position, velocity	torque	position
Optimizable	slow	yes	yes	yes	?

It seems from the above analysis that the CPG is the least promising low-level control method of all since it is so difficult to use and suffers from problems of instability. However, this is the only control method that uses actual feedback from the motors and has adaptive capabilities that no other control method possesses. It will require further work to see how best this capability can be exploited.

Harmonic oscillators and gait tables seem to be neatly matched in their capabilities. The only major differences being that sinusoids need only 3 parameters whereas gait tables require a lot more. A more subtle difference might be that not all locomotion gaits can be described in sinusoidal motions and can only be described by a gait table.

Finally, both nonlinear oscillators and CPGs require real-time numerical integration and timely communication in order to function successfully. This may not be possible on some modular robot systems. However, nonlinear oscillators are less taxing and more forgiving about communication failures,

so this might be the best choice for frugal robot resources. Both methods require computation iterations of about 1 to 15ms each time step.

6 Conclusion

Many different control systems for modular and self-reconfigurable robots have been proposed, built, and analyzed in past research. We separated these control systems into high-level software frameworks and low-level locomotion algorithms and chose to focus more detailed analysis on the locomotion methods. We devised a set of features that can be used to measure and compare the various control methods and identify their strengths and weaknesses. We then discussed the differences between the control methods one feature at a time.

Future work will involve studying these low-level methods in real robot systems under different high-level software frameworks and different tasks. We hope that this analysis will serve as a guide and a road map for future research in the control of modular and self-reconfigurable robots.

References

1. 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.
2. D. Marbach, "Evolution and Optimization of Central Pattern Generators for Modular Robot Locomotion", Masters Thesis, Lausanne, Switzerland, 2005.
3. B. von Haller, "Framework for the Simulations of Modular Robots and Self-Organization of Locomotion under Water", Masters Thesis, Lausanne, Switzerland, 2005.
4. A.J. Ijspeert, A. Crespi, J.-M. Cabelguen, "Simulation and Robotics Studies of Salamander Locomotion: Applying Neurobiological Principles to the Control of Locomotion in Robots", Neuroinformatics, Volume 3, Issue 3, pps. 171-196, Fall 2005.
5. 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.
6. 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.
7. A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, S. Kokaji, "Automatic Locomotion Design and Experiments for a Modular Robotic System", IEEE Transactions on Mechatronics, Vol. 10, Issue 3, pp. 314-325, 2005.
8. 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.

9. 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.
10. K. Matsuoka, "Mechanisms of Frequency and Pattern Control in the Neural Rhythm Generators,", Biological Cybernetics, Volume 56, pps. 345-353, 1987.
11. R.D. Beer, "On the Dynamics of Small Continuous-Time Recurrent Neural Networks," Adaptive Behavior, Volume 3, Issue 4, pps. 469-510, 1995.
12. D. Floreano, J. Urzelai, "Evolution of Adaptive-Synapse Controllers," Advances in Artificial Life, Proceedings of the 5th European Conference on Artificial Life, Berlin, 1999.
13. P. Husbands, "Evolving Robot Behaviours with Diffusing Gas Networks," In EvoRobot98: Proceedings of 1st European Workshop on Evolutionary Robotics, pp 71-86, Springer-Verlag LNCS, 1998.
14. A.J. Ijspeert and J.-M. Cabelguen, "Gait Transition from Swimming to Walking: Investigation of Salamander Locomotion Control Using Non-Linear Oscillators," Proceedings of Adaptive Motion in Animals and Machines, 2003.
15. 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.
16. M. Yim, K. Roufas, D. Duff, Y. Zhang, C. Eldershaw, S. Homans, "Modular Reconfigurable Robots in Space Applications," Autonomous Robot Journal, Special Issues for Robots in Space, Spring-Verlag, 2003.
17. A. Castano, A. Behar, P. Will, "The CONRO Modules for Self-Reconfigurable Robots," IEEE Transactions on Mechatronics, Volume 7, Issue 4, pps. 403-409, Dec. 2002.
18. R. Smith, Open Dynamics Engine, http://ode.org/.
19. 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.