Adaptive Locomotion Control in Modular Robotics

A. Spröwitz, R. Möckel, J. Maye, M. Asadpour, A. J. Ijspeert

Abstract— **The article presents the modular robot platform YaMoR, developed at the Biologically Inspired Robotics Group at EPFL, Switzerland. We give a short overview for the mechanical and electronic design of YaMoR. For wireless communication between modules we have developed a Scatternet Protocol (SNP) based on Bluetooth communication. We are interested in applying YaMoR towards adaptive behavior, such as online learning of locomotion patterns. To create coordinated and efficient gait patterns, we combine a Central Pattern Generator (CPG) approach with a gradient-free optimization algorithm (e.g. Powell's method).**

I. Introduction

An important goal in mobile articulated robotics is the generation of robust and adaptive locomotion. Any legged robotic system has to cope with the task of limb-body coordination and the creation of rhythmic patterns. This task gets more complicated with a higher amount of degrees of freedom (dof). The resulting motion should inherit the following properties: (a) be controllable in its direction and velocity (b) be adaptive, e.g. on uneven terrain, to avoid obstacles, at slippery ground and (c) be overal stable.

In modular robots[1] additional challenges arise. Unlike monolithic robot structures the position and dimension of actuators is less optimal for a specific task. The reason lies within the nature of modular robots: they are assembled from often identical modules (homogeneous systems). Because the grid size of a modular robotic assembly depends on the size of the single modules, any assembly from rather big modules will result in an sub-optimal assembly. By using chains of modular homogeneous robots these structures become, in terms of torque, rapidly under-actuated. Robot configurations made from modular robots need therefore fine-tuned locomotion patterns. By creating optimal patterns one can compensate for higher masses along and at the ends of the limbs.

Another challenging topic is communication among the modules. More complex modular robot structures are made from a high amount of individual units, hence robust communication after reconfiguration is crucial. This is also valid after a mechanical or electronic failure of a module. Robot communication is usually implemented either by mechanical contact of electric pins or by wireless communication (such as IR, Bluetooth communication). In real-world-implementation this leads to imperfections, such as communication delays or loss of packets.

In this work we propose a combined mechanism for adaptive locomotion control: using a Central Pattern Generator (CPG) together with a gradient-free optimization algorithm (Powell's method).

The central pattern generator approach is inspired by nature, which uses CPGs to easily obtain organized, rhythmic output as basis for locomotion patterns. Central Pattern Generators are neural networks capable of producing coordinated patterns of rhythmic activity without any rhythmic inputs from sensory feedback or from higher control centers [7].

A CPG implemented as a system of coupled oscillators can be a useful building block for a robot locomotion controller. They are flexible such that speed, direction and type of gait (synchronization of phase relationships among oscillators) can be altered. We show that our CPG approach is very robust against imperfect communication due to its stable limit cycle and synchronization properties.

Here we argue that CPGs are ideal building blocks for locomotion controllers of modular self-reconfigurable (MSR) robotic systems. They can easily be combined with adaptation algorithms. We present a Central Pattern Generator based on a system of coupled amplitude-controlled phase oscillators. They are implemented and tested on the YaMoR platform, each YaMoR module running one nonlinear oscillator. All modules communicate with each other over a custom-designed transparent network of Bluetooth nodes [4].

Our CPG approach also easily fits when applying it to online learning, i.e. learning while moving. The optimization of gaits and the CPGs are running in parallel, i.e. parameters are being continuously updated without stopping the robot. All parameters needed for changing gait patterns are represented in an explicit way by our CPG, what makes it very easy to combine the CPG with any optimization method. We are using a non-stochastic, gradient-free optimization method (Powell's method shows rapid and good results), applying the above parameters directly as dimensions in our search space. Therefore no post-adaptation of results from e.g. a model-based approach is necessary (regarding friction between the robot and the environment, backlash, spring-stiffness and damping of the overall system). We believe that this approach can also be applied to changing topologies of modular self-reconfiguring robots.

In the following section we briefly present the mechanical and electronic design of YaMoR, as well as the Scatternet Protocol for communication among the modules. Section III

Rico Möckel is with the Institute of Neuroinformatics at the Uni/ETH Zürich, 8057 Zürich, Switzerland. moeckel@ini.phys.ethz.ch

Alexander Spröwitz, Jérôme Maye, Masoud Asadpour and Auke Jan Ijspeert are with the School of Computer and Communication Science at the Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland. (alexander.sproewitz, jerome.maye, masoud.asadpour, auke.ijspeert)@epfl.ch.

[1]we consider chain-lattice-mix type of modular robots, other than substrate-like modular robots

explains our Central Pattern approach and the background for Online Learning and closes with results from the experiments. We finish with a short discussion of our approach (Section IV).

II. YaMoR Design

Our modular robots consist of multiple homogeneous YaMoR modules. The electronics of the module are designed to be modular in itself. The modules can be equipped either by a micro-processor board and/or by a FPGA board. Communication among the boards is based on RS232 or I2c bus. One YaMoR unit is self-sufficient in terms of actuation, actuation control, energy supply, processing and communication. The placement of the connection mechanism classifies YaMoR as a mix of a chain-and-lattice-type modular robot (5 contact points). We use a Bluetooth-based communication, the Bluetooth stack got extended by our Scatternet Protocol.

A. Mechanical and electronic design

One YaMoR module weights approximately 0.25 kg and is 94 mm long (including the lever, see also Fig. 1). YaMoR modules have a single degree of freedom, driven by a standard RC-servo motor with a 180° working range and 110 Ncm maximum torque. This is sufficient to lift three modules. The housing is made from printed circuit boards (PCBs). The PCBs serve in addition as connection plates, the YaMoR units are plugged together and then are fixed by a screw-and-nut system. The in-between angle is determined by a pin system. A single Li-Ion battery per module

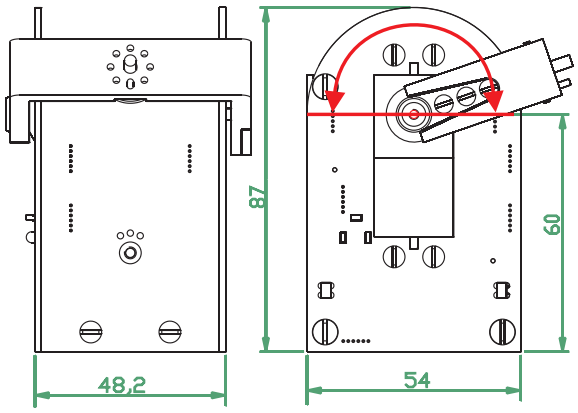

Fig. 1. CAD drawing of a YaMoR module, dimensions in (mm). The manual connection mechanism allows inter-connections of 15° between 2 YaMoR modules (a 45° partition on the male pin, 30° partition on the female connector). The RC servo motor moving range and area is marked red.

is supplying energy for 1-1.5 hours of experimental time running the (1) Power-board with step up and level-down converters, (2) Bluetooth communication board, (3) a micro-processor and/or a (4) FPGA board and the RC-servo motor. A (5) sensor board with a 3-axis accelerometer and an IR-proximity sensor is implemented, but not used in this work.

B. Module communication, Scatternet Protocol

By using Bluetooth-based communication we are able to separate our communication from the mechanical connectors. Using wireless communication is especially suitable for self-reconfiguring modular robots. Bluetooth wireless communication gives a more dynamic and adaptive framework to a future autonomously self-reconfiguring YaMoR version. Once it is set up it continues working, even after reconfigurations, splits or merges (currently reconfigurations are applied by hand). Wireless communication is also more convenient for robot-computer communication in monitoring tasks. However there is one limitation in using purely wireless communication based on Bluetooth: we need to tell each module the ID of its physical neighbor. This would not be necessary when using the above method and an additional ability to sense the ID of neighboring modules, and/or by using electrical contacts.

Bluetooth in concurrence to Infrared, WLAN or Zigbee represents a choice due to a tradeoff. On the modules only limited energy is available (Bluetooth has a relatively low power consumption of 10-100 mW, depending on its class). The usual working range of modular robot units is covered by class 2 Bluetooth devices (app. 20 m). Former Bluetooth standard 1.2 supports data rates up to 600 kbit/s and provides wireless serial links with a baud rate of 115200, the latest Bluetooth standard even up to 3 Mbit/s. In comparison Zigbee currently only provides 200-250 kbit/s. WLAN chips, that can be faster than Bluetooth, are draining more power. Bluetooth implementations are providing a robust, standard protocol.

To provide a transparent network of more than 8 Bluetooth nodes, we changed the original Bluetooth stack. By letting a slave of one Bluetooth piconet become master of the next piconet one can form a scatternet, basically a composition of piconets. It can now handle up to 256 nodes. Please see [4] for more details and information about the functionality of our Scatternet Protocol (SNP) implementation. The SNP is well suited also for sensor networks where the information of several sensors has to be collected and the maximum distance between devices is less than 100 m. Since we do not use special Bluetooth modes—like parking all sensors—the network can continuously stay active and e.g. send its data to a central monitoring device. For sensor networks that need direct communication over distances bigger than 100m other communication systems have to be used. However our SNP can help here to support transparent communication.

III. Locomotion control

Our Central Pattern Generator is implemented in a distributed manner as a set of nonlinear oscillators, one per YaMoR unit. Each oscillator can communicate with any other oscillator in the network (mainly for synchronization). In addition its parameters can be modified by a PC. To optimize locomotion patterns we use Powell's optimization method, with the covered distance over time as an optimization criterion.

A. Central Pattern Generator

We programmed one nonlinear oscillator into each module and coupled the controllers via the Bluetooth network. We implemented the CPG model as a system of n coupled

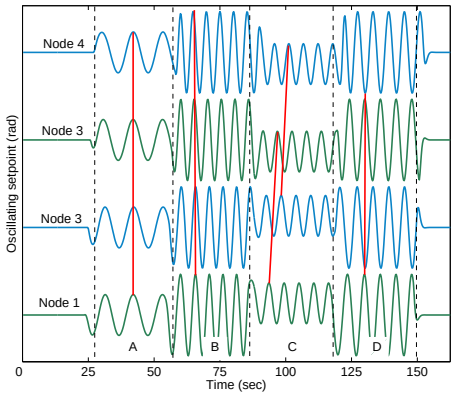

Fig. 2. Example for a CPG network using SNP communication (4 nodes in a chain: node1-node2-node3-node4 with bidirectional coupling). Changing parameters in A-B: amplitude and frequency, in B-C: amplitude, offset and phase shift, in C-D: amplitude and phase shift. Red lines indicate the phase shift.

amplitude-controlled phase oscillators, a single oscillator is described as follows:

$$\dot{\phi}_i = \omega_i + \sum_j (w_{ij}\, r_j \sin(\phi_j - \phi_i - \varphi_{ij}) \quad (1)$$

$$\ddot{r}_i = a_r(\frac{a_r}{4}(R_i - r_i) - \dot{r}_i) \quad (2)$$

$$\ddot{x}_i = a_x(\frac{a_x}{4}(X_i - x_i) - \dot{x}_i) \quad (3)$$

$$\theta_i = x_i + r_i \cos(\phi_i) \quad (4)$$

The meaning of the parameters and common parameter ranges are given in Tab. I, the equations are designed such that the output of the oscillator θ_i in Eq. (4) exhibits limit cycle behavior i.e. produces a stable periodic output. From any initial conditions, the state variables r_i and x_i will asymptotically and monotonically converge to R_i and X_i. This allows one to smoothly modulate the amplitude and offset of oscillations [2]. An example implementing the above CPG model on 4 YaMoR nodes is given in Fig. 2, showing the resulting oscillation set-points used for the RC servo motors. The phase shift between the nodes is 0 in the interval A and B, $\pi/2$ in C, and π in D. That results i.e. in an anti-phase coupling from node-to-node (D).

TABLE I

θ_i	Oscillating set-point	State variable	*radians*
ϕ_i	Phase	State variable	[]
r_i	Amplitude	State variable	*radians*
x_i	Offset of the oscillation	State variable	*radians*
ω_i	Intrinsic Frequency	Control parameter	$[0.1 : 0.3]Hz$
R_i	Intrinsic Amplitude	Control parameter	*radians*
X_i	Intrinsic Offset	Control parameter	*radians*
w_{ij}	Coupling weight	Control parameter	$[0 : 1][]$
ϕ_{ij}	Phase bias	Control parameter	[]
ϕ_j	Phase neighbor j	State variable	[]
r_j	Amplitude neighbor j	State variable	*radians*
a_r	Positive gain	Constant	$20\,rad/s$
a_x	Positive gain	Constant	$20\,rad/s$

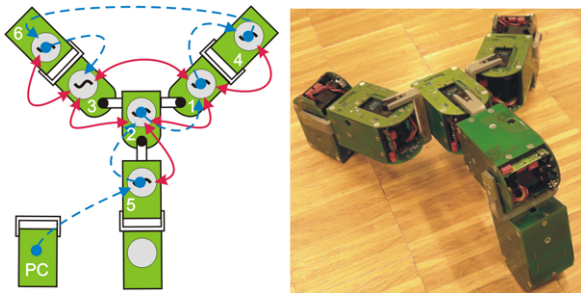

Fig. 3. Example for a tripod robot (right), the corresponding CPG configuration (red lines) and the SNP setup (blue, dashed lines). We implemented a CPG controller for each actuator and exchanged the CPG state variables via the Bluetooth scatternet.

B. Online Learning

Using our CPG approach allows us to assign explicit parameters for each of our nonlinear oscillators.

To generate organized patterns for forward locomotion (a) synchronization and (b) one or several optimal combinations of amplitude, offset and frequency of each oscillator are necessary. In our experiments we use one common intrinsic frequency (0.3 rad/s, a feasible frequency for the RC servo motor). Because of this common frequency synchronization between oscillators is rapidly, after a short transient period, obtained from any initial condition due to the inter oscillator coupling via bluetooth. For coordination among the nonlinear oscillators, phase biases between the oscillators must be tuned. Coupling weights of not connected nodes are set to 0, of connected oscillators to 1. Bi-directional couplings share the same weight with opposite sign. That sums up to a maximum of $n * 3$ free parameters, where n is the number of modules. For a configuration like in Fig. 3 with 6 actuated nodes this would result in 17 free parameters (12 plus one phase bias per red connection line minus 1 for the closed loop in the center of the robot). However by introducing symmetries and shared parameters we reduced this to 7 parameters, which finally span our vector space for the optimization.

For fitness evaluation of the resulting locomotion pattern we use the overall velocity of the robot. From previous simulations [6] we know that Powell's algorithm [8] requires rather few fitness evaluations. Indeed fast convergence is a necessity for online learning on the real hardware platform, after the batteries can provide only a maximum of 200 evaluation cycles, each about 25 s. We would like to decrease the number of evaluations more, to be able to rapidly adapt to the changes in the environment or changes in the topology of the robot. In comparison to this rather simple optimization method, stochastic optimization methods (e.g. particle swarm optimization or genetic algorithms) can find better results but are much slower. They often need 10 times more evaluations.

C. Experimental results

We tested the online optimization approach on several modular robot configurations: snake, turtle, tripod (Fig. 3)

Fig. 5. Tripod robot configuration: snapshots of the learned best gait, from left to right. Blue point shows recent position of the center module, the averaged direction of the tripod is shown by the black dashed line. The time difference between snapshot 1 and 5 are 4.3 sec (a bit more than the 3.4 sec of the cycle time for one period), time intervals between the pictures are constant.

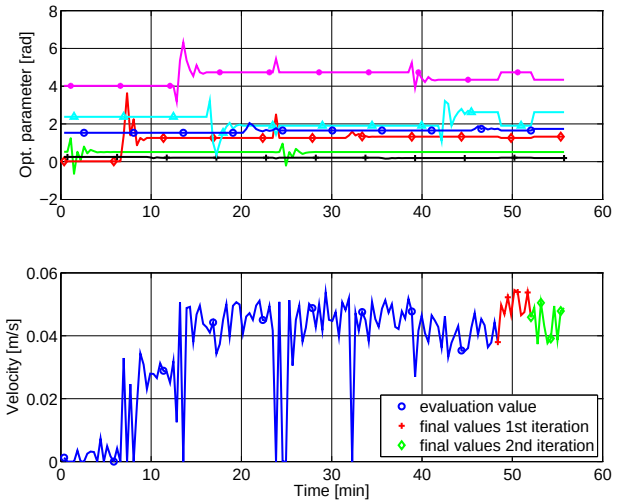

Fig. 4. Applying Powell's optimization method to Tripod configuration, 7 parameters are optimized. Top figure: green line: amplitude of the inner modules, ◇ (red) amplitude of the outer modules, + (black) offset outer modules. Remaining parameters are phase bias parameters.

and quadruped configurations. Please see [3] for more details. As visible in Fig. 4 (a plot with average performance) good gaits (5 cm/s or better) are found after 15 min for the tripod configuration. Fig. 5 illustrates snapshots of a very good solution, a straight gait. The optimization result converges after 15 min and does not improve much more after this (it has likely converged to a local optimum). Visible in Fig. 5 is a nonlinear displacement of the center module over time and place. This leads to locomotion-depending-noise when determining the velocity of a specific gait in an relatively short time window (we are using 8 sec for evaluating the velocity). Note that this noise might sometimes be useful by lifting the optimization algorithm out of a local optimum. However this noise will possibly prohibit the optimization method from finding a "narrow" global optimum. Another explanation for the convergence is that Powell's method actually found a global optimum for this robot configuration.

IV. Discussion and Future Work

We have presented a combined approach to generate optimized locomotion patterns on arbitrary modular robot configurations, using Central Pattern Generators and a gradient-free optimization algorithm. The chosen algorithm shows good results for a very small number of evaluations (compared to stochastic optimization methods). This provides a way for an online learning strategy, running CPGs and the optimization method in parallel on the real hardware and in real time. We would like to extend this adaptive locomotion learning to changing modular robot topologies and environmental situations. To provide autonomous reconfiguration we are currently developing a centralized reconfiguration strategy based on graph signatures.

V. Acknowledgment

We gratefully acknowledge the technical support of André Guignard, Andres Upegui, Elmar Dittrich, Adamo Maddalena, André Badertscher and Peter Brühlmeier in the design and construction of the robot modules.

References

[1] R. Möckel, C. Jaquier, K. Drapel, E. Dittrich, A. Upegui, and A. J. Ijspeert. Exploring adaptive locomotion with YaMoR, a novel autonomous modular robot with Bluetooth interface. *Industrial Robot*, vol. 33(4), pp. 285–290, 2006.

[2] A. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen. "From swimming to walking with a salamander robot driven by a spinal cord model", *Science*, vol. 315(5817), pp. 1416-1420, 2007.

[3] A. Spröwitz, R. Möckel, J. Maye, A. J. Ijspeert, "Learning to move in modular robots using central pattern generators and online optimization", submitted to: *International Journal of Robotics Research* (under review).

[4] R. Möckel, A. Spröwitz, J. Maye, A. J. Ijspeert, "An Easy to Use Bluetooth Scatternet Protocol for Fast Data Exchange in Wireless Sensor Networks and Autonomous Robots", submitted to: *IROS2007* (accepted).

[5] Cyberbotics (2007). Webots Reference Manual.

[6] D. Marbach, A. J. Ijspeert. "Online optimization of modular robot locomotion". *Proceedings of the IEEE Int. Conference on Mechatronics and Automation (ICMA 2005)*. pp. 248-253. 2005.

[7] F. Delcomyn. "Neural basis for rhythmic behaviour in animals". *Science*, 210, 492–489, 1980.

[8] W. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery. "Numerical recipes in C : the art of scientific computing, 2nd edition". *Cambridge University Press*, 1994.