The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Automatic Gait Generation in Modular Robots: "to Oscillate or to Rotate; that is the question"

Soha Pouya, Jesse van den Kieboom, Alexander Spröwitz, Auke Jan Ijspeert

Abstract— **Modular robots offer the possibility to quickly design robots with a high diversity of shapes and functionalities. This nice feature also brings an important challenge: namely how to design efficient locomotion gaits for arbitrary robot structures with many degrees of freedom.**

In this paper, we present a framework that allows one to explore and identify highly different gaits for a given arbitrary-shaped modular robot. We use simulated robots made of several Roombots modules that have three degrees of freedom each. These modules have the interesting feature that they can produce both oscillatory movements (i.e. periodic movements around a rest position) and rotational movements (i.e. with continuously increasing angle), leading to rich locomotion patterns. Here we ask ourselves which types of movements — purely oscillatory, purely rotational, or a combination of both— lead to the fastest gaits. To address this question we designed a control architecture based on a distributed system of coupled phase oscillators that can produce synchronized rotations and oscillations in many degrees of freedom. We also designed a specific optimization algorithm that can automatically design hybrid controllers, i.e. controllers that use oscillations in some joints and rotations in others. The proposed framework is verified by multiple simulations for several robot morphologies. The results show that (i) the question whether it is better to oscillate or to rotate depends on the morphology of the robot, and that in general it is best to do both, (ii) the optimization framework can successfully generate hybrid controllers that outperform purely oscillatory and purely rotational ones, and (iii) the resulting gaits are fast, innovative, and would have been hard to design by hand.

I. Introduction

Modular robots present interesting platforms to explore locomotion strategies for robotics. Indeed, their (self-) reconfigurability allows one to explore various types of gaits in multiple types of morphologies. However, the large variety in robot configurations and having many degrees of freedom make it problematic for the user to imagine all different solutions. Hand coding and editing the gaits is tiring and time-consuming. Moreover, there might always be some solutions which are not explored by the designer. Therefore design tools are needed to help to extract the capabilities of a newly designed modular robot. Here, our goal is two-fold: (i) to present a framework for automatically designing locomotion controllers for arbitrary robot morphologies, and (ii) to use that framework to explore whether oscillations, rotations, or combinations of both, lead to the fastest locomotion. Some impressive locomotor performance can be obtained by either type of movement (see for instance the Big Dog robot for locomotion based on oscillatory movements [1] and Rhex [2] and Whegs [3] robots for rotational movements), but combinations of both are rarely used (see the salamander robot in [4] for an exception). Furthermore, the questions of which type of movements are best for a given morphology, and whether combining them could lead to even better performance, have not yet been addressed to the best of our knowledge. We use Roombots modular robots as the building blocks for exploring this question and we test different number of modules and robot shapes to be able to draw morphology-independent conclusions. However, the framework is generic and can be used for other types of robots and other types of actuations. To our best knowledge this is the first work in the field where a modular robot controller-optimizer framework can provide exploration on such a wide variety of locomotion patterns.

Authors are with Biorobotics Laboratory, Institute of Bioengineering, School of Engineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), first-name.last-name@epfl.ch

The framework we propose has two components: a distributed locomotion controller and an optimization algorithm that performs both structural and parametric optimization. The locomotion controller is implemented in a distributed system of coupled oscillators one per degree of freedom similar to the concept of central pattern generators (CPGs) found in the spinal cord of vertebrate animals. The CPGs are based on coupled phase oscillators to ensure synchronized behavior and have different output filters to allow switching between oscillations and rotations. The optimization algorithm is a modified Particle Swarm Optimization (PSO) algorithm that optimizes both the structure of the controller (i.e. which type of movement is used for each degree of freedom) and its parameters (for instance amplitude and phase difference between oscillators).

A. Related Work

Most projects in modular robotics have used oscillations for generating forward locomotion [5]–[7]. Rotational movements have shown interesting gaits using wheeled or Whegs-like propulsion. Combining these two modes sounds like an interesting approach to derive newer gaits. Similar to this idea, Hancher et al. [6] used rotation in wheeled-shape modules and oscillation in different type of modules for the rest of the robot shape. Zykov et al. [8] have recently extended Molecubes with several active and passive modules to diversify robot capabilities. In related work, we designed a salamander robot that uses rotational movements for the limbs and oscillatory movements for the spine [4]. However, it has not been explored how a combination of different movements in all the joints can influence the resulting gaits. In particular, we are interested in comparing the functionality of oscillatory and rotational movements and also their

combinations in *hybrid* solutions. The Roombots modules presented below allow to investigate these questions because of their capability to perform both oscillation and continuous rotation. Different locomotion control algorithms have been developed for modular robots. As one of the pioneers in the field, Yim [9] proposed gait table control which uses sequences of time-driven actions for modules. In a recent development, Zhang et al [10] use a new form of gait tables called Phase Automata which uses event-driven state machines. These methods usually use predefined sequences but can be adjusted during runtime [11]. Role-based control [12], hormone-based control [13] and constraint-based control [14] have been used for locomotion control. Using CPG based control as the locomotion control have been suggested in several works [7] [15] because of its capablility to produce robust, synchronized movement patterns with a few number of control parameters. This last feature makes it well-suited for optimization algorithms which are used to optimize the controller parameters. Different optimization algorithms have been used for generating optimal gait patterns in modular robots. Similar to the seminal work of Sims [16], Zykov et al. [5] used a genetic algorithm (GA) to evolve the controller parameters with rhythmic locomotion patterns. Sproewitz [7] used Powell's method as the online learning procedure for evolving the CPG parameters. Hancher and Hornby [6] used steady-state evolutionary algorithm while assuming fixed morphology and a periodic gait. Christensen et al. [17] used a reinforcement learning approach to evolve the parameters of the controller which can generate different actions of rotation and stopping. To summarize, compared to this related work, the novelty in this article is on one hand a novel distributed CPG-based controller that allows the combination of oscillatory and rotational movements, and on the other hand a modified PSO algorithm that allows one to do both structural and parametric optimization. PSO has been shown to perform as well as or better than genetic algorithms (GA) in several instances [19], [20] and [21]. In one comparative study [18], we observed the superior performance on parametric simulation in comparison with genetic algorithms and simulated annealing. Due to these studies we use PSO as the internal optimizer and a mutation operators inspired by Genetic Algorithms to switch between different controller structures.

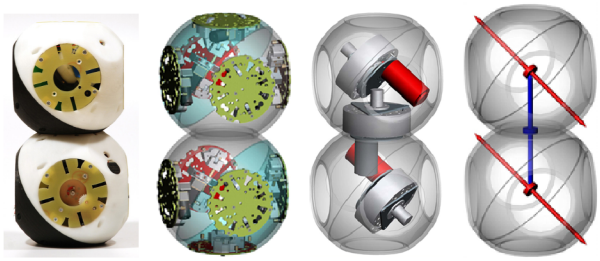
The remaining of the paper is organized as follows. We first describe the Roombots hardware in section II. Section III presents the applied controller architecture including the CPG model and topology. In section IV the proposed algorithm for optimal gait generation is presented. Sections V and VI describe the experiments and their results. The article is concluded with a discussion and an outlook on future work in section VII.

II. Roombots Hardware

A Roombots (RB) module is made of four hemispheres (Fig. 1d), which are connected and actuated by three degree of freedom (DOF). RB modules [22] fit into a regular cubic grid with $110\,\mathrm{mm}$ edge length. As the smallest possible functional unit for locomotion, we connect two RB modules serially into an RB *metamodule*. Any of the three joints of an RB unit delivers enough torque to rotate a metamodule in the "worst case scenario", i.e. it rotates it out of a horizontal stretched position (see Table I for RB module details). RB shell elements are printed in ABS plastics, plate-elements are milled out of FR4 material. An RB module weights about $1.4\,\mathrm{kg}$. This includes batteries and the estimated weight for electronic boards We use high gear ratios ($i_{Center} = 366:1$ and $i_{Outer} = 305:1$) to achieve needed torques. This limits the maximum rotational speed. The center RB joint needs $\sim 3\,\mathrm{sec}$ to rotate $360°$, both outer joints take $\sim 2\,\mathrm{sec}$. The RB active connection mechanism (ACM) is genderless, four-way symmetric, and uses four mechanical latches [23] which are retractable into the ACM. An ACM is roughly $65\,\mathrm{mm}$ in diameter, and $19\,\mathrm{mm}$ in height. The design is similar to the AMAS [24] connection mechanism. However we have integrated an altered latching mechanism [23]. Initial connector tests indicate a passive tolerance against alignment errors between ACM surfaces of roughly $1.5\,\mathrm{mm}$-$2\,\mathrm{mm}$. To built a metamodule, an RB units' foot hemisphere can be attached to the head hemisphere of a second RB unit, in one of four possible modes. We name each mode depending on the relative orientation of the involved hemisphere-axes: parallel (PAR), perpendicular (PER), shear-S (SRS), and shear-Z (SRZ) (see the four metamodules in Fig.3a).

TABLE I: Roombots module hardware specifications. Presented torque values are 150 % higher than the nominal torque. Holding torque is again $\sim 3-4$ times bigger.

Type	#	Details
Main DOF	3	Continuous rotational
Size		$110\,\mathrm{mm} \times 110\,\mathrm{mm} \times 220\,\mathrm{mm}$
Weight		$1.4\,\mathrm{kg}$
Torque_{Outer}	2	$7.3\,\mathrm{Nm}$
Torque_{Center}	1	$5.49\,\mathrm{Nm}$
Motor_{Outer}	2	12 V DC motor, FH 2342
Motor_{Center}	1	12 V DC motor, FH 2232
Outer gears	2	305 : 1, type: spur, planetary
Center gears	1	366 : 1, type: spur, planetary
ACM	4	Four-way symmetric, genderless, retractable
ACM actuator	1	Mini DC motor, 1750 : 1 GB ratio
Weight ACM		$59\,\mathrm{g}$
Others	3	12-line slip ring



(a) One RB unit. (b) ACMs. (c) Actuators. (d) 3 DOF.

Fig. 1: (a) One Roombots module. (b) Four active connection mechanisms are mounted into one RB module. (c) Custom designed gearbox. (d) All RB DOF are continuous rotational.

III. Controller Architecture

This section describes the locomotion controller. Control inputs for the CPG are high level parameters such as amplitude, offset and phase lags. Each oscillator is capable of producing either *oscillatory* or *rotational* joint angle signals.[1] We apply an oscillator network topology which matches the hardware topology, e.g. a quadruped structure or a single Roombots metamodule. An Optimization algorithm provides an automatic design of the control input parameters.

A. CPG Model

We designed a CPG controller which can produce two types of basic movements for each DOF: (i) *Rotational* movements that result from a continuously rotating (swivel) joint, and can provide wheel or Whegs-like propulsion, and (ii) *Oscillatory* movements that periodically oscillate around a resting position. Since it is important for stable, reproducible locomotion to keep all DOF synchronized whatever their mode, we built the controller as a distributed system of coupled phase oscillators, with one oscillator per DOF (joint) i:

$$\begin{aligned} \dot{\phi}_i &= 2\pi \cdot \omega_i + K_i + f_{\theta_i}(\vec{s}) && (1)\\ K_i &= \sum_j w_{ij} \cdot r_j \cdot \sin(\phi_j - \phi_i - \psi_{ij}) \\ \dot{r}_i &= a_i(R_i - r_i) + f_{r_i}(\vec{s}) && (2) \end{aligned}$$

$$\left.\begin{array}{rcll} \theta_i &=& r_i \cdot sin(\phi_i) + X_i & (Oscillation)\\ \theta_i &=& \phi_i & (Rotation)\\ \theta_i &=& X_i & (Locked) \end{array}\right\} \text{servo inputs} \quad (3)$$

where θ_i is the servo input which can be derived with different functions corresponding to the desired servo movement. Variables r_i and ϕ_i are state variables which encode amplitude and phase of the oscillation. The parameters w_{ij} and ψ_{ij} are respectively the coupling weight and phase bias of the coupling between oscillators i and j. a_i is a positive constant which determines the rise time of the amplitude to the desired value R_i. The parameters R_i, X_i, and ψ_{ij} are open parameters of which a subset (depending on the selected mode) is subject to optimization. Furthermore, this structure is capable of including sensory feedback. For this purpose the state variables can be influenced by sensory feedback signals through the functions f_{θ_i} and f_{r_i}, $\vec{s}$ being a vector of sensor states. Note that sensory feedback is not applied in this article and the methodology one can use to derive these functions is still an open question.

Equation 3 shows the output functions for three possible modes which result in oscillations, rotations or a locked condition. In the oscillation mode, the output exhibits limit cycle behavior, thus producing a stable periodic trajectory. For rotation a constant-speed profile is generated leading to a monotonic increase of the joint angle. This structure is capable of generating more complex patterns simply by designing other output functions. We also include a third mode which allows the controller to lock a joint.

With the right parameter values ($r_j > 0$ and $w_{ij} > 0$), rotational and oscillatory DOF will rapidly converge to a phase-locked regime, i.e. a regime with a constant phase difference even between phase oscillators that are in different modes. This is highly desirable for the implementation of stable, coordinated gaits. It will also ensure that several joints remain phased-locked, even if they are controlled by oscillators implemented on different micro-controllers with slightly different clocks. Fig. 2 shows this synchronization behavior between three DOFs, with two activated in oscillation mode and one in rotation mode.

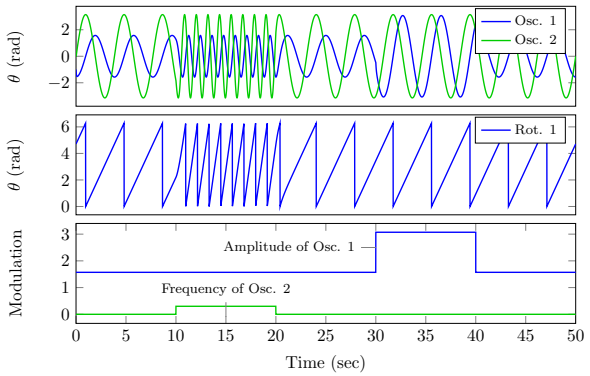


Fig. 2: Synchronization behavior of three coupled oscillators: two in oscillatory mode (upper plot) and one in rotational mode (middle plot). Frequency modulation from $t = 10$ to 20 sec, amplitude modulation from $t = 30$ to 40 sec of the simulation.

B. CPG Topology

When designing CPGs, the network coupling parameters w_{ij} and ψ_{ij} between different oscillators are of importance. For known types of locomotion gait patterns, such as quadrupedal or snake gaits, the coupling architecture can be specified based on the biological observations. Here the goal is to find different and unexpected gaits, which an arbitrarily shaped modular robot could potentially create. Hence we do not specify a pre-defined oscillator network topology. We let the coupling structure of the CPG correspond the robot's morphology, i.e. phase oscillators of neighbor DOF are coupled together.

We use one common frequency for all oscillators ($f = 0.26$ Hz, the highest frequency that fits with the hardware constraints) and symmetric bi-directional couplings. All coupling weights are set to 2, where higher values results in stronger effect of the oscillators on each other, hence they pass the transient phase of synchronization faster. In order to minimize the number of open parameters and also have a well-designed behavior for the transient phase when starting the locomotion, we set the coupling weights value such that the network phase locks in approximately two seconds. Phase biases ψ_{ij} are open parameters and subject to optimization. We do not induce symmetry artificially, i.e. we do not apply any mirroring of parameter sets along our network. Applying

[1] In the remaining part of the paper we will use *oscillator* to refer to *pattern generators* capable of producing both oscillatory and rotational output.

symmetry is usually a good strategy to reduce the number of open parameters. However it also might limit the resulting gaits, as it restricts the possible variety of parameters.

IV. Optimal Gait Generation

The optimization algorithm has two layers. The *outer* layer performs structural optimization, and is *discrete*. The *inner* layer performs parametric optimization on *continuous* valued parameters corresponding to the selected movement types.

In this paper, a modified version of the standard Particle Swarm Optimization with a constriction factor [25], [26] is used to erform the optimization process. PSO is a stochastic, population based optimization method using principles of collaboration rather than competition to evolve individuals. In PSO, each individual is represented by a position and velocity vector, representing respectively the particle's parameter values and search direction. The evolution of each particle in the swarm is then governed by Eq. 4 .

$$\begin{aligned} \vec{v_i}(t+1) &= K \cdot [\vec{v_i}(t) + c_1 r_1(\vec{p_i} - \vec{x_i}(t)) + \\ &\quad c_2 r_2(\vec{p_g} - \vec{x_i}(t))] \\ \vec{x_i}(t) &= \vec{x_i}(t-1) + \vec{v_i}(t) \end{aligned} \tag{4}$$

Where $\vec{v_i}(t)$ is the velocity vector, $\vec{x}(t)$ is the position vector, K is a *constriction* factor, c_1 and c_2 are two constants, r_1 and r_2 are two pseudo-random numbers in the range [0, 1], p_i is the best known solution vector of particle i and p_g is the global best known solution vector. The *constriction* factor and the constants c_1 and c_2 were set to ensure convergence (see for more detail, [27]).

The PSO algorithm described thus far is used for the *inner* layer optimization of the continuous parameters of a specific selection of movement types. Particles are initially uniformly distributed over the possible combinations of movement types. In each such combination, particles share the same parameters, and an independent PSO optimizes their respective solutions. The task of the *outer* layer is then to do the structural optimization and to move particles from one combination of movement types to another.

The *outer* layer consists of a set of mutation operators inspired by Genetic Algorithms. Similar to the velocity update of the PSO, the probability of mutation of each actuated degree of freedom is composed of:

- P_e: exploration probability of mutation to a movement type (oscillation, rotation or locked) other than the current one
- P_l: local probability of mutation to the movement type which is part of the selection with the best results in the particle's history
- P_g: global probability of mutation similar to P_l but from the best results taken over all the particles

Governed by these three probabilities, particles will be mutated at each iteration to different combinations of movement types. Once a particle moves to a different parameter space it is incorporated in the PSO running locally in that space. A main challenge is to choose appropriate values for the different probabilities P_e, P_l and P_g. In general, we want to stimulate exploration in the early phases of the optimization, visiting many possible combinations of movement types. Then, as the optimization progresses, particles should start exploring their local known best solutions in more detail. Finally we want the particles to converge in the best known space, as if selecting the best configuration of movement types. The system then starts behaving as a standard PSO with a fixed configuration of movement types as more and more particles are attracted. The desired behavior can be designed by varying the probabilities P_e, P_l and P_g as the optimization progresses. In this paper, the exploration and global probabilities were defined using a sigmoid function. The local probability was defined using a gaussian function. The characteristics were determined experimentally after some initial tests.

V. Experimental Setup

We performed several experiments applying our CPG- and optimization framework. Firstly we were interested in exploring the locomotion abilities of the four types of metamodules (PAR, PER, SRS, and SRZ) (Fig. 3a). Our motivation for testing metamodules is that they represent the simplest possible robot shape built from two Roombots modules (six DOF). In addition, two quadruped shapes were carefully designed featuring symmetry and DOFs allowing quadruped locomotion (using five and six modules, respectivley called Quad5 and Quad6; Fig. 3e and Fig. 3h). These structures were used to verify our approach on more complex shapes with well designed features. Finally, a more arbitrary, asymmetric shape was constructed with no specific design features in mind (called Arbit; Fig. 3c). This allows us to compare the performance of the optimization algorithm on shapes with no pre-designed characteristics, for which a well performing gait can be hard to imagine or design.

We are also interested to know which type of movement, oscillatory or rotational, would lead to the highest locomotion speed. We conducted the following optimization experiments with different possible combinations of the four joint modes: (1) *pure rotation*, all DOF are in the rotation mode, (2) *hybrid rotation* with DOF either in rotation or locked mode, (3) *pure oscillation*, all DOF are in oscillation mode, and (4) *fully hybrid*, DOF are in oscillation, rotation, or locked mode, and different modes coexist within the robot. The search space for the internal PSO optimizer depends on the joint mode; amplitude, offset and phase shift for oscillation, offset and phase shift for rotations an only offset for the locked mode. For each of the four above-mentioned experiments we perform ten optimizations with different initial conditions, each one using 50 particles and 100 iterations to search for the optimal solution. The generated solutions are evaluated in Webots [28], a simulation tool based on ODE providing simulation of collisions, rigid body dynamics and actuator properties. The fitness value to evaluate the solution is the robot speed by measuring the traveled distance over 20 seconds (approximately 5 gait cycles), which is sent back to the optimizer. A RB module/module collision detection penalizes unrealistic solutions with a zero fitness.

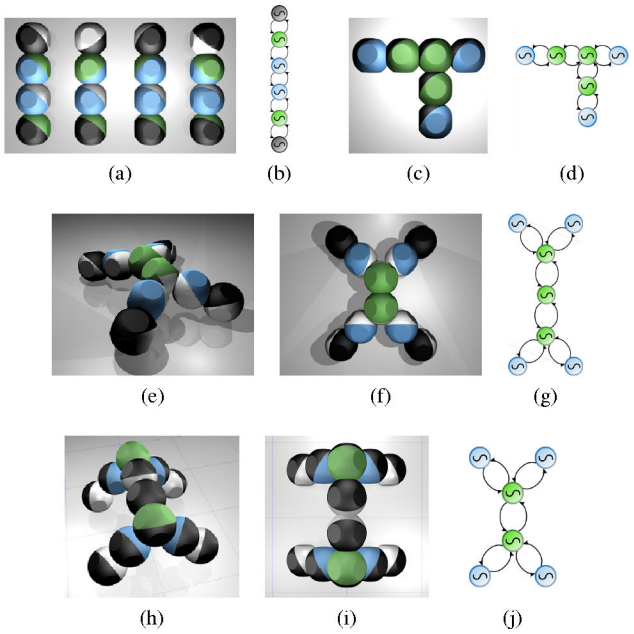


Fig. 3: Pictures of different robot shapes used for the experiments, and their CPG networks: (a,b): four metamodule configurations; (c,d): one arbitrary, asymmetric shape with three RB units (Arbit); (e,f,g): quadruped with five units (Quad5); (h,i,j): quadruped with six units (Quad6). The network mimics the physical topology by coupling neighbor oscillators. For the quadrupeds oscillators in the spine are in green and hip joint oscillators are shown in blue.

VI. Results and Discussion

a) Fitness behaviour and automatic mode selection: Fig. 5 shows the average and standard deviation of the speed of the best evolved robot over ten optimizations with different initial random population. The results illustrate three interesting properties of this framework. (i) The capability of combining different modes in the fully hybrid setting results in finding solutions with higher values for both average speed and variance in all robot structures, thus resulting in faster and more diverse solutions in the same number of simulations. (ii) Results from pure rotation show a drastic reduction in the robot performance. In the worst case, no viable gait (i.e. one without self-collision) could be found for the Quad5 and Arbit robots. While in the hybrid rotation, allowing the robot to lock some of its degrees of freedom, when the others are in rotational mode, helps avoiding self-collision during robot locomotion. The performance of this mode is comparable in terms of characteristics with the oscillation mode since both include locked joints (defined implicitly in oscillation mode due to zero amplitudes). (iii) The results indicate that the performance of oscillation and rotation modes are strongly dependent on the robot shape. In the case of PAR and Quad6, pure oscillation largely outperforms hybrid rotation. For PER, SRS, SRZ and Quad5 however, similar performance for both modes is observed. Hybrid rotation is only preferred for the Arbit robot. This shows that for a given robot shape, it is not trivial to select either oscillation or hybrid rotation. The complex interaction of the different DOF of a robot shape and the environment determine whether rotation or oscillation will provide the best performance. In almost all experiments having a mixture of both movement types (fully hybrid) yields better results. Thus, to answer the question of rotation versus oscillation, combining both modes provides the best strategy.

b) Gait diversity: Table II shows the comparison between the different movement type experiments for gait generation of the Quad6 robot. We consider four aspects; (i) the best speed driven by each experiment (ii) the mean value of the speed of the resulting gaits, (iii) the standard deviation of the speed and (iv) the number of different combinations of movement types which are used to generate the gaits. Results in table II show that fully hybrid solutions not only result in higher average speeds, but also allow the robot to derive these solutions by using combinations of joint movements (i.e. have a high "diversity"). Most of the combinations of movement types are hard to imagine and design by hand, because the locomotion behavior results from the complex interactions of the movements of all the degrees of freedom. One can imagine a situation where the algorithm provides an initial selection of possible movement type configurations. A human supervisor can then manually select a few of these configurations, which are of some particular interest, and further optimize the controller for these configurations.

TABLE II: Gait generation results from the different experiments for Quad6

Mode	Best (m/s)	Mean (m/s)	Std %	# of config.
Pure Rotation	0.22	0.17	3.1	1
Hybrid Rotation	0.24	0.21	1.4	6
Pure Oscillation	0.29	0.26	3.0	1
Fully Hybrid	0.33	0.27	3.9	10

c) Framework performance: Fig. 4 shows the fitness optimization for the best fully hybrid solution of the Quad6 robot. As discussed in section IV, the hybrid framework is capable of visiting different combinations of movement types, or *sub-swarms*, and select the best solution among them. These graphs are related to the sub-swarm with the best speed. Fig. 4a shows the number of fitness evaluations versus the number of iterations. It represents in which iterations and how many times this sub-swarm has been explored. As a result of the mutation probabilities (Fig. 4b), within the first 75 iterations, particles are divided into different sub-swarms exploring different possible solutions. After iteration 75, exploration stops and the exploitation phase starts where the particles start to be attracted to the best sub-swarm found in exploration phase. In other words, in the first phase different solutions are explored to find the best candidates and in the second phase these solution spaces are attracting more particles to optimize the open parameters. For instance, here particles first come to this space in the iteration number 63 and find high velocities inside it. At the end of the iteration 75, this space -as the globally best solution- starts attracting other particles. The plot also shows that to prevent self-collision, several solutions are being penalized by setting their fitness value to zero. Fig. 4b shows how the best fitness

of this movement type configuration is evolving through the iterations of the optimization process.

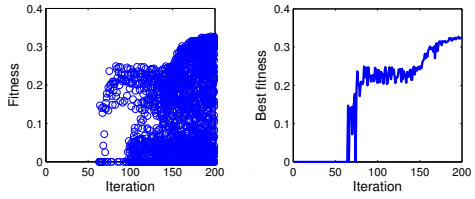


Fig. 4: Fitness optimization of the best fully hybrid solution for Quad5 robot. The figure on the left shows the number of fitness evaluations through the iterations; the figure on the right shows the best fitness for each iteration.

d) Gait pattern description: Fig. 6 shows two examples of gait patterns, namely for Quad5 and Quad6 robots. Fig.6 top shows one of the possible configuration for generating locomotion gait in Quad5 robot derived with fully hybrid framework. In this solution, optimizer developed a controller structure which uses locked mode for spine joints and rotation mode for hip joints. Then, in the next level, it optimized the internal parameters of the controller to evolve the gait with highest speed for this special configuration. This resulted in the gait with $15\,\mathrm{cm/s}$ where diagonal limbs are in phase. Similarly, in all other nine simulations of this experiment, a new joint movement configuration is introduced and then optimized. For the Quad6However, one can include the coupling weights as the open parameters. robot, the optimal locomotion gaits was derived through the hybrid structure of the controller with this setting: both spine joints are in oscillatory mode, one of the outer joints is blocked, one is in rotational mode, and the remaining two leg joints oscillate. This setting results in a wind-up like gait which propels the robot with $33\,\mathrm{cm/s}$, while the walking gait -as the first solution in mind to design gait for this robot- only leads to $21\,\mathrm{cm/s}$ maximally. Average speed for all the Quad6 robot gaits derived in fully hybrid mode are approximately $26\,\mathrm{cm/s}$ (Fig. 5)[2].

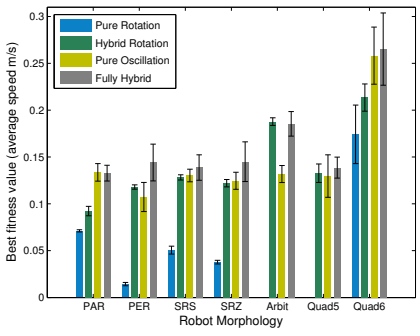


Fig. 5: The optimization results from the seven different robot structures: PAR, PER, SRS and SRZ metamodules, one asymmetric robot with three modules (Arbit) and the two quadruped shapes (Quad5 and Quad6). Each experiment was repeated ten times, with different initial random population.

[2]Locomotion videos for all the robots are available at the EPFL Locomorph webpage: *http://biorob.epfl.ch/page38289.html*

e) Advantages of the CPG Model: The CPG model has several interesting features that make it well suited for modular robotics. (i) Our model can produce stable rhythmic patterns such that the dynamical system rapidly returns to its steady state after perturbations of the state variables. (ii) The CPG model only needs a few, high level control parameters (in our case amplitude, offset and phase lag). Hence it can reduce the dimensionality of the control problem such that the optimization algorithm only needs to optimize a small number of control signals. (iii) It has the capability to generate different gaits, which can be achieved by setting the network coupling type and topology. In this way we can reproduce animal-like gaits. This has been done in previous work for quadruped, and modular robots ([7], [29], [30]). Yet one has the option to keep the network topology open, and to let new and unexpected gaits emerge. This approach is even more appealing for modular robots, where ideal gaits are initially unknown due to new robot topologies. (iv) This CPG model can be used to generate different types of locomotion patterns. In this work we used specific patterns, such as sine-waves for oscillation, and constant speed for rotation. However the framework is kept open and more complex patterns can be implemented, which could lead to an even higher versatility of derived gaits.

VII. Conclusion and Future Work

In this paper we have derived a framework for locomotion control of modular robots where the CPG based controller and the optimization algorithm are tightly connected. This framework provides an important feature: the optimization algorithm can choose and switch between oscillatory and rotational joint movements, for any joint, at any time during the optimization process, and is fully automated. The user is not required to, but can pre-assign a movement-type to a joint type. This enables us to derive gait patterns for traditional robots like quadrupeds (oscillatory and locked joint control) but also for robots featuring the more capable, continuous rotational joints, e.g. Whegs-like robots or in our case the Roombots modules. We have presented results of our optimization framework deriving pure oscillatory or rotational joint controllers based on CPGs, as well as hybrid controllers. Optimized robot gaits for the latter type often result in mixed-mode joint controllers with surprising characteristics and very competitive performance. In other words it is better to let the optimization algorithm find suitable modes for each joint rather than designing them by hand. Our research on locomotion control will be further pursued in order to address the problems of how to properly include sensory feedback for improving the efficiency and robustness of locomotion patterns against environmental disturbances. We also plan to extend the hardware by adding passive, light-weight elements like carbon-fiber plates to built more complex, and a larger variety of locomotion gaits.

VIII. Acknowledgement

This project has received funding from the EPFL and from the European Community's Seventh Framework Programme

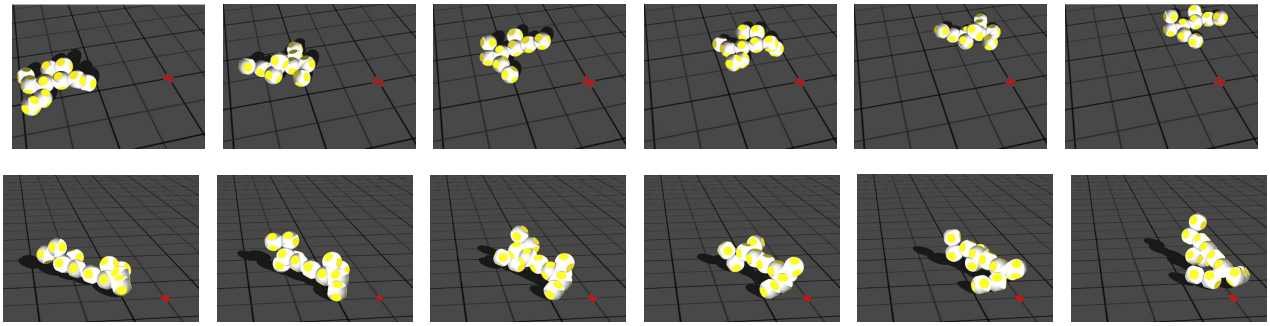


Fig. 6: top) Fully hybrid gait for the Quad5 robo. Hip joints use rotating patterns, spine joints are blocked. Diagonal limbs are in-phase, and neighboring hind or front limbs are in anti-phase. This results in a trot-like gait with $15\,\mathrm{cm/s}$. below) Fully hybrid gait for the Quad6 robot. It propels with a winding-like mechanism: by leaving two extremities on the ground it winds the remaining two of them around the body stem and vice versa ($33\,\mathrm{cm/s}$).

FP7/2007-2013 - Future Emerging Technologies, Embodied Intelligence, under the grant agreements no. 231 688 (Locomorph) and no. 231 451 (EVRYON). We gratefully acknowledge the technical support of André Guignard, André Badertscher, Peter Brühlmeier, Philippe Voessler, and Manuel Leitos in the design and construction of the robot modules.

References

[1] R. Playter, M. Buehler, and M. Raibert, "BigDog," in *Proceedings of SPIE*, vol. 6230, 2006, p. 62302O.

[2] U. Saranli, M. Buehler, and D. Koditschek, "Rhex: A simple and highly mobile hexapod robot," *The International Journal of Robotics Research*, vol. 20, no. 7, p. 616, 2001.

[3] R. T. Schroer, M. J. Boggess, R. J. Bachmann, R. D. Quinn, and R. E. Ritzmann, "Comparing cockroach and whegs robot body motion," *Proceedings of the IEEE International Conference on Robotics and Automation 2004*, pp. 3288—3293, apr 2004.

[4] A. Ijspeert, A. Crespi, D. Ryczko, and J. Cabelguen, "From swimming to walking with a salamander robot driven by a spinal cord model," *Science*, vol. 315, no. 5817, p. 1416, 2007.

[5] V. Zykov, J. Bongard, and H. Lipson, "Evolving dynamic gaits on a physical robot," in *Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO 2004*.

[6] M. Hancher and G. Hornby, "Evolving Quadruped Gaits with a Heterogeneous Modular Robotic System," pp. 3631–3638, 2007.

[7] A. Sproewitz, R. Moeckel, J. Maye, and A. J. Ijspeert, "Learning to move in modular robots using central pattern generators and online optimization," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 423–443, mar 2008.

[8] V. Zykov, P. William, N. Lassabe, and H. Lipson, "Molecubes Extended: Diversifying Capabilities of Open-Source Modular Robotics," in *IROS-2008 Self-Reconfigurable Robotics Workshop, accepted*, 2008.

[9] M. Yim, "Locomotion with a unit modular reconfigurable robot," Ph.D. dissertation, Stanford University Mechanical Engineering Dept., 1994.

[10] Y. Zhang, M. Yim, C. Eldershaw, D. Duff, and K. Roufas, "Scalable and reconfigurable configurations and locomotion gaits for chain-type modular reconfigurable robots," in *International Symposium on Computational Intelligence in Robotics and Automation*, 2004, pp. 893–899.

[11] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[12] K. Stoy, W. Shen, and P. Will, "Implementing configuration dependent gaits in a self-reconfigurable robot," in *Proceedings of ICRA2003*.

[13] W. Shen, P. Will, A. Galstyan, and C. Chuong, "Hormone-inspired self-organization and distributed control of robotic swarms," *Autonomous Robots*, vol. 17, no. 1, pp. 93–105, 2004.

[14] Y. Zhang, M. Fromherz, L. Crawford, and Y. Shang, "A general constraint-based control framework with examples in modular self-reconfigurable robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[15] A. Kamimura, H. Kurokawa, E. Toshida, K. Tomita, S. Murata, and S. Kokaji, "Automatic locomotion pattern generation for modular robots," in *IEEE International Conference on Robotics and Automation*, 2003.

[16] K. Sims, "Evolving 3d morphology and behavior by competition," in *Proceedings, Artificial Life IV*. MIT Press, 1994, pp. 28–39.

[17] D. Christensen, M. Bordignon, U. Schultz, D. Shaikh, and K. Stoy, "Morphology independent learning in modular robots," in *Proc., International Symposium on Distributed Autonomous Robotic Systems*, 2008.

[18] Y. Bourquin, "Self-organization of locomotion in modular robots," Ph.D. dissertation, Bioinspired Robotics Group (BIRG), Ecole Polytechnique Fédérale de Lausanne (EPFL), 2003.

[19] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[20] J. Kennedy and W. M. Spears, "Matching algorithms to problems: An experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator," *in procoeedings of the IEEE congress on evolutionart computation (CEC)*, pp. 78–83, 1998.

[21] P. Fourie and A. Groenwold, "The particle swarm optimization algorithm in size and shape optimization," *Structural and Multidisciplinary Optimization*, vol. 23, no. 4, pp. 259–267, may 2002.

[22] A. Sproewitz, A. Billard, P. Dillenbourg, and A. J. Ijspeert, "Roombots-mechanical design of self-reconfiguring modular robots for adaptive furniture," in *2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009, pp. 4259–4264.

[23] A. Sproewitz, M. Asadpour, Y. Bourquin, and A. Ijspeert, "An active connection mechanism for modular self-reconfigurable robotic systems based on physical latching," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 3508–3513.

[24] Y. Terada and S. Murata, "Automatic modular assembly system and its distributed control," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 445–462, mar 2008.

[25] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *"Proceedings of IEEE International Conference on Neural Networks"*, vol. 4, 1995, pp. 1942–1948.

[26] R. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 1, 2000, pp. 84 – 88.

[27] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[28] Webots, "http://www.cyberbotics.com," commercial Mobile Robot Simulation Software. [Online]. Available: http://www.cyberbotics.com

[29] H. Kimura, Y. Fukuoka, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 475–490, may 2007.

[30] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji, "Distributed adaptive locomotion by a modular robotic system, M-TRAN II," in *Proceedings of the IEEE/RSJ IROS*, 2004, pp. 2370–2377.