2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Gait Optimization for Roombots Modular Robots - Matching Simulation and Reality

Rico Moeckel, Yura N. Perov, Anh The Nguyen, Massimo Vespignani, Stéphane Bonardi, Soha Pouya, Alexander Sproewitz, Jesse van den Kieboom, Frédéric Wilhelm, Auke Jan Ijspeert

Abstract— **The design of efficient locomotion gaits for robots with many degrees of freedom is challenging and time consuming even if optimization techniques are applied. Control parameters can be found through optimization in two ways: (i) through online optimization where the performance of a robot is measured while trying different control parameters on the actual hardware and (ii) through offline optimization by simulating the robot's behavior with the help of models of the robot and its environment.**

In this paper, we present a hybrid optimization method that combines the best properties of online and offline optimization to efficiently find locomotion gaits for arbitrary structures. In comparison to pure online optimization, both the number of experiments using robotic hardware as well as the total time required for finding efficient locomotion gaits get highly reduced by running the major part of the optimization process in simulation using a cluster of processors. The presented example shows that even for robots with a low number of degrees of freedom the time required for optimization can be reduced by a factor of 2.5 to 30, at least, depending on how extensive the search for optimized control parameters should be. Time for hardware experiments becomes minimal. More importantly, gaits that can possibly damage the robotic hardware can be filtered before being tried in hardware. Yet in contrast to pure offline optimization, we reach well matched behavior that allows a direct transfer of locomotion gaits from simulation to hardware. This is because through a meta-optimization we adapt not only the locomotion parameters but also the parameters for simulation models of the robot and environment allowing for a good matching of the robot behavior in simulation and hardware.

We validate the proposed hybrid optimization method on a structure composed of two Roombots modules with a total number of six degrees of freedom. Roombots are self-reconfigurable modular robots that can form arbitrary structures with many degrees of freedom through an integrated active connection mechanism.

I. INTRODUCTION

With an increasing number of degrees of freedom it becomes challenging and often even impossible to design and tune efficient locomotion controllers by hand. Scalable controllers like Central Pattern Generators (CPGs) in combination with learning and optimization techniques allow for an automatic exploration of efficient locomotion gaits in simulation [1] and hardware [2]. With their relatively low number of control parameters, CPGs can reduce the time required for gait optimization. However, also CPGs cannot fully solve the problems that come with optimization techniques that are purely based on hardware or software experiments.

Online optimization, where the optimization process is performed on the robotic hardware, is typically too time consuming for robotic structures with many degrees of freedom. The parameter space exploration requires experiments running in real time and unless many robots with well-matched behavior are available the optimization process cannot be parallelized. Furthermore, online optimization can be dangerous for the robotic hardware since high impacts between robot and ground often cannot be predicted and get detected only during the actual experiment.

Offline optimization allows the exploration of a variety of control parameters in simulation often faster than real time and in parallel since the optimization process can be performed on a cluster with many processors. Furthermore, time consuming processes including resetting the robot after each experiment as well as charging and replacing batteries can be avoided. Control parameters can be explored safely without the risk of damaging expensive robotic hardware. This is why the exploration of robot behavior in simulation is so popular. However, offline optimization has one major drawback that can make it poorly suited for finding control parameters for robotic hardware: Due to a lack of precision in the robot and environmental models, the optimized control parameters are typically not transferable from simulation to robotic hardware, a problem known as the "reality gap".

A variety of researchers has been studying pure online and offline optimization of locomotion patterns for legged and modular robots [3]–[10].

Several other researchers have started targeting the problem of reducing the reality gap. Lipson et. al [11], Glette et.al [12], and Coros et. al [13] have been presenting studies using quadruped robots. Adams has been using artificial evolution as a tool for generating controllers for physical robots [14]. Bongard et. al studied self-modeling machines [15]. A comparison of different strategies for simulator tuning was presented by Klaus et. al [16].

This paper explores the method of hybrid optimization as a solution to combine the advantages of the online and offline optimization process applied to a modular robot. Hybrid optimization is a cyclic method that avoids time consuming parameter optimization with hardware. Instead hybrid optimization aims at finding optimal control parameters in simulation through simulation models that match well the robotic hardware and the environment (Fig. 1). In

All authors are with the Biorobotics Laboratory, Ecole Polytechnique Fédérale de Lausanne, Switzerland. Yura Perov is also with the Siberian Federal University, Institute of Mathematics and Computer Science, Russia, Krasnoyarsk. Corresponding authors: {rico.moeckel, auke.ijspeert}@epfl.ch

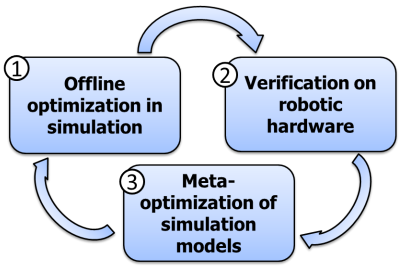

Fig. 1: Hybrid optimization is a cyclic process combining (1) offline optimization,(2) exploration of control parameters found in simulation on hardware, and (3) meta-optimization to improve matching of software models and hardware.

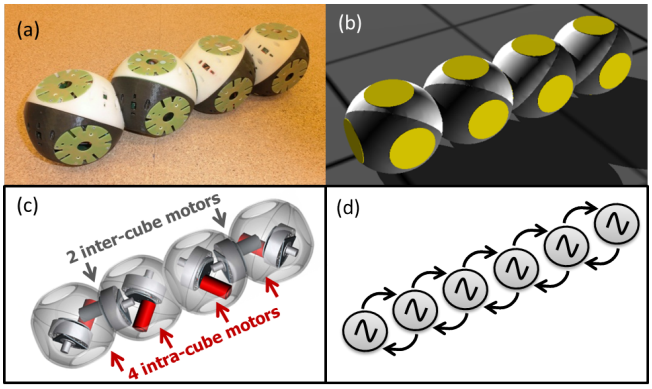

Fig. 2: (a) Picture of the Roombots (RB) meta-module hardware. (b) Snapshot of the simulation model implemented in Webots. (c) Transparent CAD drawing. A RB meta-module is composed of two individual RB modules that form a rigid connection. Thus a RB meta-module contains 4 cubes and six degrees of freedom (DOF). Four intra-cube motors actuate the four DOFs within the cubes. Two inter-cube motors actuate the inner DOFs of the two individual RB modules forming the meta-module. (d) Each DOF is controlled by an oscillator. The six oscillators are bidirectionally phase coupled to form a central pattern generator.

other words, hybrid optimization improves both control and model parameters. In the first step (1) during an optimization cycle, control parameters leading to optimized robot gaits are extracted in simulation using existing simulation models. In a second step (2), a selection of optimized gaits found during the first step is verified on robotic hardware by testing optimized control parameters on the actual robot while the robot's behavior is recorded. The third step of the hybrid optimization cycle (3) is the meta-optimization, where the behavior of the simulation model and robotic hardware for each selected gait is compared and model parameters are updated to achieve better behavioral matching. Afterwards, a new optimization cycle using the updated models can be started in simulation.

Modular robots that allow the rapid assembly of a variety of morphologies with many degrees of freedom present a valuable and challenging platform for the exploration of locomotion control and learning strategies. This is why we chose to test our hybrid optimization methods on our modular robot system Roombots.

In this paper we present only one hybrid optimization cycle since our study concentrates on reducing the gap between software simulation and reality. So instead of running a second offline optimization, we verified the parameters found during the first meta-optimization by simulating again the gaits that have been found during the initial offline optimization and selected for verification on robotic hardware but this time using the updated simulation models. The behavior of the robot model simulated with the meta-optimized model parameters and the actual robotic hardware already matched very well after the first optimization cycle such that differences in the behavior are difficult to identify by pure human observation. Several fast and robust robot gaits could be identified.

In Section II we present the Roombots hardware. The sections III and IV describe the central pattern generator controller and optimization method, respectively. Section V gives details on the offline optimization process while Section VI presents our setup for robotic experiments and Section VII explains the meta-optimization process. Section VIII discusses experimental results and Section IX concludes and describes future work.

II. ROOMBOTS HARDWARE

We conduct our simulation and hardware experiments using a Roombots (RB) meta-module - a structure composed of two individual Roombots modules. Fig. 2 shows a picture of a RB meta-module (Fig. 2a), of its simulation model (Fig. 2b) as well as a transparent CAD (Computer Aided Design) drawing depicting the meta-module's six degrees of freedom (DOF) (Fig. 2c). Specifications for the meta-module are given in Table I. Further details about the Roombots hardware can be found in [17] and [18].

A RB meta-module is composed of four cubes. Due to space constraints, different motors have been selected for the DOFs within a cube (intra-cube DOFs, see Fig. 2c) and DOFs between cubes (inter-cube DOFs, see Fig. 2c) resulting in the different torque and speed constraints given in Table I. The connection between the two RB modules forming the meta-module is rigid through an active connection mechanism (ACM) based on mechanical grippers. These ACMs give the robot its ability for self-reconfiguration and allow for a rapid assembly of complex structures.

RB meta-modules are operated on two Lithium Polymer battery packs giving the robot autonomy for about 1 hour. The brushed motors are position and speed controlled in real-time through a PID loop implemented on custom motor driver electronics. The motors' relative encoders are used for feedback. Real-time control through central pattern generators (CPG) is implemented on custom electronics that communicate with the motor driver electronics via a RS485 communication bus protocol at a baudrate of 1MBAUD. CPG values get updated at a frequency of 20Hz. We control the RB meta-module and configure the CPG controller from a PC using a wireless Bluetooth communication module. At each time step a real-time collision checker simulates the next moves of the RB meta-module before they are executed. Only if no internal collisions are detected the set-points of

TABLE I: Roombots Metamodule Specifications

Specification	Value
Degrees of freedom	6 (continuous rotation)
Intra-cube DOF	
Motors	Faulhaber 2342 012 CR
Gearbox reduction	305:1
DOFs speed (No load)	26.6 RPM
DOFs nominal torque	4.9 Nm
Inter-cube DOF	
Motor	Faulhaber 2232 012 SR
Gearbox reduction	366:1
DOFs speed (No load)	19.4 RPM
DOFs nominal torque	3.6 Nm
Overall dimensions	110x110x440 mm
Weight	2.8 kg
Energy source	2x 2-cell LiPo battery with 1200 mAh autonomy $\sim$1 hour

the motor position get updated.

III. CENTRAL PATTERN GENERATOR CONTROLLER

We control Roombots structures through central pattern generators (CPG) [19] - networks of coupled nonlinear oscillators - for the following reasons: (i) CPGs are capable of generating synchronized movement patterns using only a few control parameters. Thus only a few parameters have to be learned during the optimization process to achieve complex behavior. (ii) CPGs provide a scalable control scheme that can be easily adapted to the number of modules forming the robotic structure by simply adding or removing oscillators from the network. Thus the concept of CPGs is well suited for the control of modular robots especially when using a distributed implementation. (iii) CPGs ensure smooth transitions of motor set points after a modification of control parameters. This avoids abrupt changes of the motor states which is typically preferred to achieve stable gaits and expand motor and robot life time.

The RB meta-module presented in this paper (Fig. 2) is controlled by six phase oscillators each producing the position set points for one of the meta-module's six DOFs. The coupled phase oscillators are implemented in form of the following coupled differential equations:

$$\dot{\phi}_i = 2\pi \cdot f \cdot \sum_j w_{ij} \cdot r_j \cdot \sin(\phi_j - \phi_i - \psi_{ij}) \quad (1)$$

$$\dot{r}_i = a_i (R_i - r_i) \quad (2)$$

$$\theta_i = r_i \cdot \sin(\phi_i) + X_i \quad (3)$$

where i and j are the indexes of the oscillator. The oscillators output θ_i directly controls the position set point of the motor actuating DOF number i. θ_i follows a harmonic oscillation with an amplitude r_i, a phase ϕ_i, and an offset X_i. The constants $w_{ij} = 0.5$ and $a_i = 2$ control how quickly oscillators synchronize and reach a modified amplitude, respectively. The frequency $f = 0.2Hz$ was chosen to respect the maximum rotation rate and acceleration of the inter-cube DOFs. In our experiments oscillators are bidirectionally coupled ($\psi_{ij} = -\psi_{ji}$) in a chain configuration as shown in Fig. 2d. We did not couple all oscillators with all other ones. Thus each oscillator i has a maximum of three parameters that are subject to optimization: the desired amplitude R_i, offset X_i and the coupling phase ψ_{ij} to the following neighbor $j = i+1$.

IV. PARTICLE SWARM OPTIMIZATION

We use Particle Swarm Optimization (PSO) [20], [21] to optimize CPG control parameters during offline optimization as well as simulation model parameters during meta-optimization. PSO is a stochastic optimization method that uses so-called particles to explore the parameter search space. The position $\mathbf{x}_i$ and velocity $\mathbf{v}_i$ attributed to each particle i represent the particle's parameter values and search direction, respectively. At each iteration t, both $\mathbf{x}_i$ and $\mathbf{v}_i$ get updated as follows while continuously taking into account the best known solution vector of particle i, $\mathbf{p}_i$, and the global best known solution vector $\mathbf{p}_g$:

$$\mathbf{v}_i(t+1) = K \cdot [\mathbf{v}_i(t) + c_1 r_1 (\mathbf{p}_i - \mathbf{x}_i(t)) + c_2 r_2 (\mathbf{p}_g - \mathbf{x}_i(t))] \quad (4)$$

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) \cdot \mathbf{v}_i(t) \quad (5)$$

where the constriction factor K, the cognitive factor c_1, the social factor c_2, and the two pseudo-random numbers r_1 and r_2 were set according to [22] to ensure convergence. We chose PSO over other optimization techniques for the following reasons: (i) With its relatively low number of computations, PSO is a light-weight optimization method that can even be run directly on the robot if required. (ii) In contrast to other optimization techniques, PSO does not require gradients and performs well also on more complex search spaces.

V. OFFLINE OPTIMIZATION

We identified 17 fast gaits that show low impact between the robot and the ground and no internal collisions by simulating a model of the RB meta-module in the robot simulator Webots [23] based on ODE (Open Dynamics Engine). The parameters of the CPG controlling the RB meta-modules were optimized with PSO using 300 particles and 200 iterations using 30 cores of a cluster of 2.00GHz quad-core Intel Xeon E5504 processors. To make the hybrid optimization as efficient as possible it is important to simulate in a non-perfect virtual world to (i) avoid overfitting and (ii) to receive gaits that are robust to noise and thus can be transferred to the real world [24]. This is why we added noise to the outputs of the CPGs and repeated each iteration three times. The fitness of a trial was evaluated as follows: If during a trial a collision between the half-spheres of the RB meta-module occurred, the fitness of this

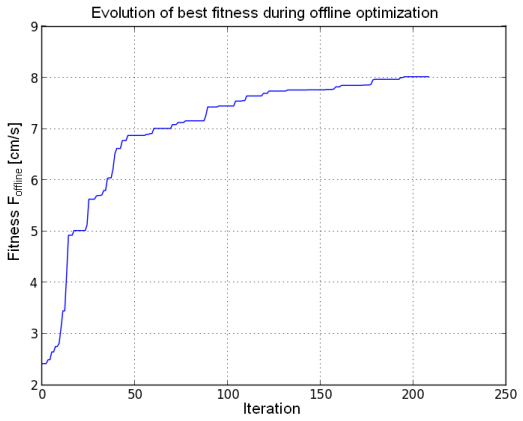

Fig. 3: Evolution of particle with the highest fitness during offline optimization.

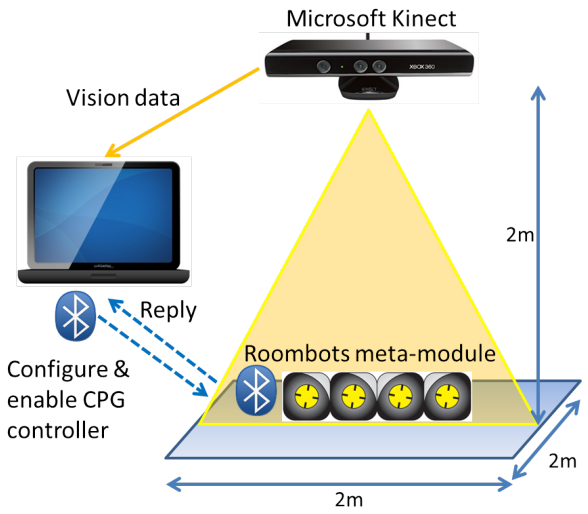

Fig. 4: Setup for hardware experiments. The behavior of the Roombots meta-module is recorded by a Kinect camera. Both the camera and robot are controlled from a PC.

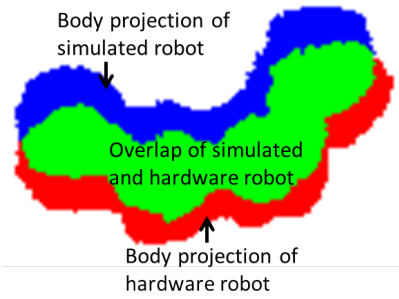

Fig. 5: We measure the similarity between the motion exhibited by a simulated and hardware robot by monitoring the overlap of the robots' projections onto the ground as observed by an overhead camera.

trial was set to 0. Otherwise, the fitness of collision-free trials was set to be proportional to the speed of the meta-module averaged over three repetitions of the iteration. For each iteration, we performed a pairwise comparison between the robot trajectories to penalize those gaits that led to large deviations in the presence of noise.

The evolution of the particle with the highest fitness during the offline optimization process is shown in Fig. 3. The plot shows the asymptotic increase of the best fitness over the iterations while the PSO particles are evaluating the search space for the CPG control parameters.

Since previous experiments with a simplified robot simulation model could not be transfered to the robotic hardware, we used a detailed robot model with realistic object boundaries for the ODE collision checker. This detailed model was directly imported from the CAD tool SolidWorks that was used to design the RB hardware. Due to the detailed model, simulation speed was decreased in comparison to former Roombots simulation studies to about 2x real time so that each trial simulating 30 seconds in real time took about 15 seconds simulation time on the 2.00GHz processors. The total time required for offline optimization on the cluster simulating the 300 particles, 200 iterations, and 3 repetitions was about 22 hours.

VI. VERIFICATION THROUGH HARDWARE EXPERIMENTS

For verification of CPG control parameters found during offline optimization we build the setup shown in Fig. 4. During locomotion, the RB meta-module was constantly observed by a Microsoft Kinect camera. We decided for the Kinect since it allows for simple and efficient tracking of robotic structures in real time. The Kinect provides depth images that support an easy removal of the background from the recordings. Before each trial CPG parameters were loaded into the RB meta-module via a wireless Bluetooth communication. Once the internal CPG network was configured the robot became fully autonomous and tried out the control parameters for 30 seconds.

We tested and recorded in total 17 gaits that we selected from the offline optimization. Six gaits were used to optimize simulation parameters during the meta-optimization. We validated the new simulation parameters with the remaining 11 gaits to test for overfitting in the meta-optimization. The selection of gaits included both those with the best fitness but also several hand selected gaits with medium fitness that clearly used different strategies for locomotion than the fittest gaits. The medium fitness gaits were not selected for their speed but to improve the variability of the gaits used for the following meta-optimization process.

VII. META-OPTIMIZATION

During the meta-optimization, we aimed at optimizing the RB model, the environmental parameter of the Webots simulation tool as well as the parameters of the virtual camera to match the behavior of the simulated robot with the recordings from the hardware experiments. Table II gives an overview of the 22 parameters that we optimized through PSO as well as the search space for each parameter, its value before and after meta-optimization.

TABLE II: Model parameters subject to meta-optimization

#	Parameter description	Search range	Hand tuned value	Value after meta-optimization
	Parameters of simulation environment			
1	Constraint force mixing (CFM) in Webots[1]	[0.0; 0.001]	0.00001	0.00014892
2	Error reduction parameter (ERP) in Webots[1]	[0.1; 0.8]	0.2	0.3652
3	Coulomb friction between Roombots body and ground surface[2]	[0.1; 5.0]	1	0.8018
4	Bounce between Roombots body and ground surface[2]	[0.0; 1.0]	0.5	0.8040
	Parameters of Roombots model			
5	Mass of meta-module's half-spheres. [kg]	[0.280; 0.370]	0.325	0.3038
6	Mass of meta-module's half-spheres. [kg]	[0.280; 0.370]	0.325	0.3142
7	Mass of meta-module's half-spheres. [kg]	[0.280; 0.370]	0.325	0.2939
8	Mass of meta-module's half-spheres. [kg]	[0.280; 0.370]	0.325	0.3308
9	Mass of meta-module's half-spheres. [kg]	[0.280; 0.370]	0.325	0.3079
10	Mass of meta-module's half-spheres. [kg]	[0.280; 0.370]	0.325	0.3316
11	Mass of meta-module's half-spheres. [kg]	[0.280; 0.370]	0.325	0.3438
12	Mass of meta-module's half-spheres. [kg]	[0.280; 0.370]	0.325	0.3017
	Intra-cube DOF			
13	Maximum velocity[3]	[1.0; 15.0]	2.5	6.8721
14	Maximum torque[3]	[2.5; 15.0]	4.8	9.0643
15	Control parameter P[3]	[5.0; 15.0]	10	6.9873
16	Damping constant[3]	[0.0; 1.0]	0	0.1172
	Inter-cube DOF			
17	Maximum velocity[3]	[1.0; 15.0]	1.5	7.5193
18	Maximum torque[3]	[2.5; 15.0]	3.2	12.2080
19	Control parameter P[3]	[5.0; 15.0]	10	12.8652
20	Damping constant[3]	[0.0; 1.0]	0	0.5622
	Camera parameters			
21	Height of the virtual Kinect depth camera [cm]	[210.0; 240.0]	260	227.8160
22	Time shift between movies taken by hardware and virtual software camera [ms]	[0.0; 1200.0]	900	424.1056

We quantify the match of behavior of the robot in simulation and hardware through the following procedure: We record the hardware through a real and the simulation model through a virtual overhead mounted Kinect camera. The camera removes the background and only keeps those camera pixels of the robot body. We thus gain two sequences of snapshots or movie frames, the hardware (H) and simulation movie (S). Fig. 5 shows two overlaid snapshots. The camera projection of the simulated robot body is shown in blue while the projection of the robotic hardware is shown in red and the overlap of the robot bodies is shown in green. For the meta-optimization we defined a *similarity ratio*, R, that is equal to the overlapped area divided by the total area of the robot projections. Areas are measured in number of pixels. If both robot projections match perfectly R reaches its maximum equal to 1.

During the meta-optimization we want to maximize R for all selected gaits found during offline optimization while R is measured frame by frame. Since the frames of the hardware and simulation trials need to be synchronized we added the time shift parameter #22 (Table II), s, to the optimization process. Thus the final fitness measure F_{meta} for meta-optimization becomes:

$$F_{meta} = \frac{\sum_{gaits}\left[W_{gait}\cdot\frac{1}{\sum_{frames}}\cdot\sum_{frames}R\left[H(t),S(t+s)\right]\right]}{\sum_{gaits}W_{gait}} \quad (6)$$

where $H(t)$ is the movie frame at time t from a hardware experiment and $S(t+s)$ is the possibly corresponding movie frame of a software experiment at time t shifted in time by the time shift s. For each gait the similarity rate R is modified by a gait dependent weight W_{gait}.

We aim at gaits that are fast, robust to noise, that do not generate internal collisions, and that lead to low impacts between the robot and the ground. When selecting the gaits for meta-optimization from the offline optimization process

[1]For explanation see: "http://ode-wiki.org/wiki/index.php?title=Manual:_Concepts"

[2]For explanation see: "http://www.cyberbotics.com/dvd/common/doc/webots/reference/section3.12.html"

[3]For explanation see: "http://www.cyberbotics.com/dvd/common/doc/webots/reference/section3.41.html"

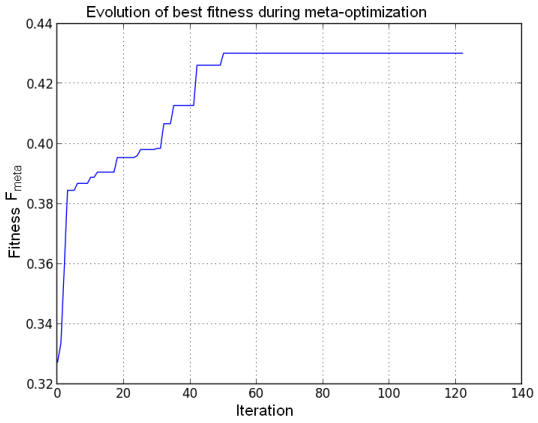

Fig. 6: Evolution of the particle with the highest fitness value during meta-optimization.

we selected only fast gaits that showed no internal collision and low impacts with the ground. To ensure that fast gaits with a high robustness to noise have greater influence on the selection of parameter values during the meta-optimization, we weighted the influence of gaits differently in the fitness function through the weight W_{gait}:

$$W_{gait} = \overline{v_{gait}} \cdot R_H \tag{7}$$

$$R_H = \frac{\sum_k^K \sum_{n>k}^K \frac{1}{\sum_{frames}} \cdot \sum_{frames} R(H_k, H_n)}{\sum_k^K \sum_{n>k}^K} \tag{8}$$

where $\overline{v_{gait}}$ is the speed of the robot achieved during the hardware verification averaged over three trials. R_H measures the similarity of different trials of the same gait tested on the hardware by taking the pairwise similarity ratio SR between different trials $(k,n \in [1,2,3])$. $K = 3$ is the total number of trials.

For the meta-optimization on the cluster we were using 100 particles and 100 iterations. Fig. 6 shows the evolution of the best fitness or similarity over iterations. The convergence of the fitness after about 60 iterations can be clearly seen. The total simulation time was about 20.2 hours.

VIII. EXPERIMENTAL RESULTS AND DISCUSSION

Since we wanted to understand how well simulation and hardware behavior got matched after the first meta-optimization we did not continue the hybrid optimization cycle with another offline optimization but instead re-simulated the previously 17 selected in hardware tested gaits using the meta-optimized simulation parameters. Table III depicts the average similarity rates achieved with the hand-tuned and the meta-optimized simulation parameters. The similarity rates of the 6 gaits that belonged to the training set for the meta-optimization was improved by more than 65%. A perfect match with a similarity rate of 1 could not be achieved for the following reasons: (i) Although both simulation and hardware were recorded at the same frequency a perfect synchronization between the virtual and hardware camera

TABLE III: Improvements of similarity between robot behavior in simulation and hardware due to meta-optimization

	R of hand-tuned simulation	**R of meta-optimized simulation**	**Improvement**
Training set	0.250317	0.414293	65.51 %
Test set	0.325153	0.402104	23.67 %
All gaits (training + test set)	0.281152	0.409271	45.57 %

is not possible and possibly also not meaningful since also (ii) noise in the hardware experiments will remain preventing a perfect match. (iii) Furthermore, the parameters and simulation models provided by ODE probably will not allow a much better match since for instance the ODE friction model will not allow to recreate exactly the ground reaction forces and sliding generated by the hardware and the bounce parameter will not be sufficient to exactly simulate the properties of the ground. More detailed models might allow for better matching but will also lead to an increase in simulation time. (iv) Also the ground is not exactly homogeneous during a gait cycle and might vary over time.

Fig. 7 shows the fastest gait we found during the experiment that according to our optimization function received the highest weight during the meta-optimization and thus achieved the best matching properties. The gait that achieved an average speed of 6.0 cm/s in simulation and 5.8 cm/s in hardware could be robustly repeated at numerous occasions. We considered the matching between hardware and simulation that led to this gait good enough to stop the optimization process after one iteration. As Fig. 7 and the movie submitted with this publication show[5], simulation and hardware results are well matched and sufficiently difficult to distinguish through human observation.

To further validate the improvement of the matching between the behavior of the simulated robot and hardware we used the additional 11 gaits (called the test set) from the offline simulation and previously performed hardware experiments that were not part of the meta-optimization process. When tested in hardware and simulation the gaits achieved a similarity rate of more than 0.4 - similar to the gaits that were part of the training set. These results make us confident that the parameters obtained during meta-optimization are robust and can be used to predict the behavior of the robotic hardware for gaits that were not part of the meta-optimization process.

With our hybrid optimization process the time for obtaining fast and robust gaits could be highly reduced in comparison to an online-learning approach while obtaining realistic results as if the experiments would have been carried out purely in hardware. Learning the required 17 control parameters (6x oscillator amplitudes, 6x oscillator offsets, 5x coupling phases) for a RB metamodule as stated in Section

[5]The movie can also be found at "http://biorob2.epfl.ch/utils/movieplayer.php?id=257".

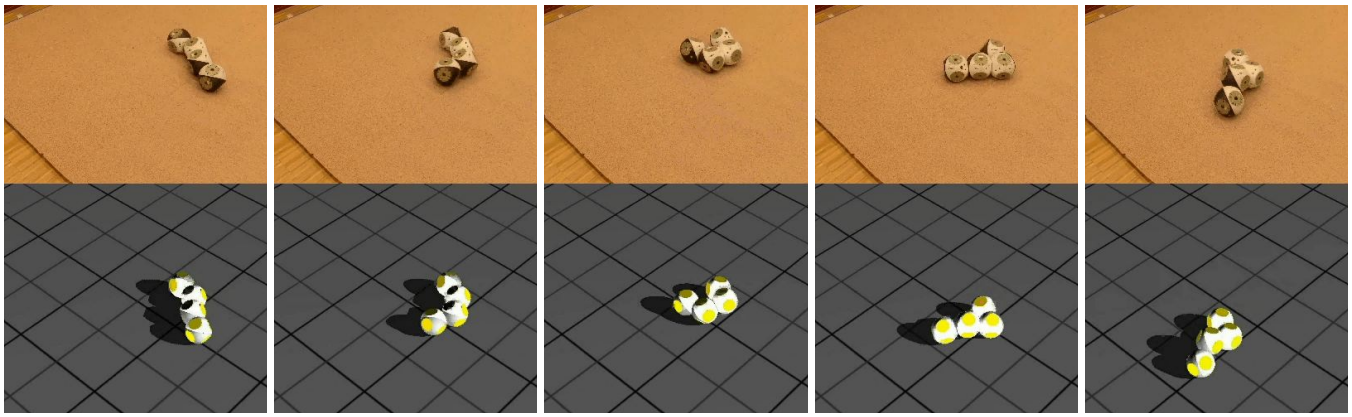

Fig. 7: Snapshots of a meta-optimized RB meta-module gait with a similarity rate of about 0.4. Time increasing from left to right. Top: hardware experiment, Bottom: simulation. The time between snapshots is 12 seconds.

III with PSO typically takes about 50 particles and more than 100 iterations. A single experiment requires at least 1 minute including the 30 seconds for recording the robot's behavior as well as the additional average time for resetting and maintaining the robot. If the effect of noise on the experimental results should be reduced, the experiment has to be repeated for at least 3 times resulting in a minimum time for the online optimization of about 250 hours. An extensive search as we performed during the initial offline optimization with 300 particles, 200 iterations and 3 repetitions would require about 3000 hours.

The hybrid optimization process requires simulation time for the (i) offline and (ii) meta-optimization as well as a minimal set of (iii) hardware experiments. (i) With our detailed robot simulation model, an offline optimization with PSO using 300 particles, 200 iterations and 3 repetitions took about 22 hours. In contrast to online optimization, the offline optimization process could be parallelized and the total simulation time could be easily distributed among several processor cores as we did with our distributed simulation framework[6]. In theory each PSO particle can be simulated on a separate processor core - in practice we used 30 cores heavily reducing the actual required simulation time.

(ii) We recorded 17 gaits on the hardware that we repeated 3 times. With the typical overhead that comes with hardware experiments the total time required for was less than 2 hours.

(iii) The meta-optimization was again distributed on 30 processor cores. We chose 120 particles and 100 iterations to optimize the 22 parameters shown in Table II. The total simulation time was about 20.2 hours.

To allow a fair comparison between online and meta-optimization one has to take into account the time required for building the initial robot simulation model required for optimization in simulation. The Roombots simulation model was improved during many studies and experiments. A fair upper bound taken for building the model is about 50 hours.

So in the described case the meta-optimization took about 94 hours in comparison to the 250 to 3000 hours for the online optimization which corresponds to a reduction of a factor of at least 2.5 to 30 depending on how extensive the search for optimized control parameters should be. If 100 processor cores were used for simulation the optimization time excluding the design and hand tuning of the initial model can most likely be reduced to less than 10 hours even for extensive parameter searches. More importantly, the time experimentalist have to be physically present to perform experiments was reduced to 2 hours. Furthermore, possibly dangerous gaits that would have caused high impacts between the robot and the ground and thus possibly might have damaged the hardware could be discarded before being tried on the robot. To conclude, our study shows that the time spent to improve the simulation model in addition to the control parameters is well invested. We believe that the more degrees of freedom future robots will have, the more different structures should be explored and thus the more different control parameters will have to be extracted, the more experimentalist will benefit from the hybrid optimization approach.

[6]See "http://biorob2.epfl.ch/users/jvanden/docs/optimization-concept/concept.pdf" for an overview.

IX. CONCLUSIONS AND FUTURE WORK

We presented a hybrid optimization process for efficient learning of fast locomotion gaits that are robust to noise, that do not generate internal collisions, and that lead to low impacts between the robot and the ground. The method is verified using a Roombots meta-module with six degrees of freedom. Our locomotion control and optimization approach combines a scalable central pattern generator controller that is capable of generating coordinated complex movements for robots with many degrees of freedom with particle swarm optimization for learning parameter values for the controller as well as for the robot simulation software. Although optimization is performed offline in simulation, we achieve well-matched behavior of the simulated robot and hardware through meta-optimization. We show that, through hybrid optimization, experiment time using robotic hardware and thus the time where the experimentalist has to be physically present can be easily reduced by a factor of more than 100. If the optimization process in simulation is parallelized

as presented, simulation time can be reduced by a factor greater than 2.5 to 30 depending on how extensive the search for optimized control parameters should be. With an increase in robot complexity we expect experimentalists to benefit even more from the proposed hybrid optimization approach. Furthermore, in contrast to online optimization, hybrid optimization allows avoiding the test of gaits with high impacts that could possibly damage the robot.

We are currently testing the hybrid optimization process on more complex Roombots structures as well as on more dynamic legged modular robots. Futhermore, we are extending the proposed methods to more complex, less homogeneous terrains.

X. ACKNOWLEDGMENT

The authors gratefully acknowledge the technical support of Andre Guignard, Andre Badertscher, Peter Bruhlmeier, Philippe Voessler, and Manuel Leitos in the construction of the robot modules. This project has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 - Future Emerging Technologies, Embodied Intelligence, under the grant agreement no. 231688 (Locomorph). This research was also supported by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics.

REFERENCES

[1] S. Pouya, J. Van Den Kieboom, A. Spröwitz, and A. Ijspeert, "Automatic gait generation in modular robots: to oscillate or to rotate? that is the question," *Proceedings of IEEE/RSJ IROS 2010, Taipei, Taiwan, October 18*, vol. 22, 2010.

[2] 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, 2008.

[3] O. Miglino, H. H. Lund, and S. Nolfi, "Evolving mobile robots in simulated and real environments," *Artificial life*, vol. 2, no. 4, pp. 417–434, 1995.

[4] G. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata, "Autonomous evolution of gaits with the sony quadruped robot," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2. Citeseer, 1999, pp. 1297–1304.

[5] A. J. Ijspeert and J. Kodjabachian, "Evolution and development of a central pattern generator for the swimming of a lamprey," *Artificial Life*, vol. 5, no. 3, pp. 247–269, 1999.

[6] K. Wolff and P. Nordin, "Learning biped locomotion from first principles on a simulated humanoid robot using linear genetic programming," in *Genetic and Evolutionary ComputationGECCO 2003*. Springer, 2003, pp. 199–199.

[7] 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*, vol. 4, 2004.

[8] G. S. Hornby, S. Takamura, T. Yamamoto, and M. Fujita, "Autonomous evolution of dynamic gaits with two quadruped robots," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 402–410, 2005.

[9] D. Christensen, D. Brandt, K. Stoy, and U. P. Schultz, "A unified simulator for self-reconfigurable robots," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 870–876.

[10] D. J. Christensen, U. P. Schultz, and K. Stoy, "A distributed strategy for gait adaptation in modular robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2765–2770.

[11] H. Lipson, J. Bongard, V. Zykov, and E. Malone, "Evolutionary robotics for legged machines: from simulation to physical reality," in *Proceedings of the 9th Int. Conference on Intelligent Autonomous Systems*, 2006, pp. 11–18.

[12] K. Glette, G. Klaus, J. C. Zagal, and J. Torresen, "Evolution of locomotion in a simulated quadruped robot and transferral to reality," in *Proceedings of the Seventeenth International Symposium on Artificial Life and Robotics*, 2012.

[13] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. Van De Panne, "Locomotion skills for simulated quadrupeds," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, p. 59, 2011.

[14] B. Adams, "Evolutionary, developmental neural networks for robust robotic control," Ph.D. dissertation, Massachusetts Institute of Technology, 2006.

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

[16] G. Klaus, K. Glette, and J. Tørresen, "A comparison of sampling strategies for parameter estimation of a robot simulator," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2012, pp. 173–184.

[17] A. Sproewitz, "Roombots: Design and Implementation of a Modular Robot for Reconfiguration and Locomotion," Ph.D. dissertation, EPFL, Switzerland, 2010.

[18] A. Sprowitz, S. Pouya, S. Bonardi, J. Van den Kieboom, R. Mockel, A. Billard, P. Dillenbourg, and A. J. Ijspeert, "Roombots: reconfigurable robots for adaptive furniture," *Computational Intelligence Magazine, IEEE*, vol. 5, no. 3, pp. 20–32, 2010.

[19] A. J. Ijspeert, "2008 special issue: Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

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

[21] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

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

[23] Webots, commercial Mobile Robot Simulation Software. Available online at http://www.cyberbotics.com.

[24] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *Advances in artificial life*. Springer, 1995, pp. 704–720.