Stochastic Optimization of Bipedal Walking using Gyro Feedback and Phase Resetting

Felix Faber and Sven Behnke
Computer Science Institute
University of Freiburg, Germany
{faber | behnke}@informatik.uni-freiburg.de

***Abstract*— We present a method to optimize the walking pattern of a humanoid robot for forward speed using suitable metaheuristics. Our starting point is a hand-tuned open-loop gait that we enhance with two feedback control mechanisms. First, we employ a P-controller that regulates the foot angle in order to reduce angular velocity of the robot's body. Second, we introduce a phase resetting mechanism that starts the next step at the moment of foot contact. Using a physics-based simulation, we demonstrate that such feedback control is essential for achieving fast and robust locomotion.**

For the optimization of open-loop parameters and parameters of the feedback mechanisms, we compare Policy Gradient Reinforcement Learning (PGRL) and Particle Swarm Optimization (PSO). To make optimization more data-efficient, we extend PGRL by an adaptive step size and a sequential sampling procedure. Our experiments show that the proposed extensions increase the performance of PGRL significantly.

We selected the extended PGRL algorithm to optimize the gait of a real robot. After optimization, the robot is able to walk significantly faster.

I. Introduction

Enabling a humanoid robot to walk fast and stable is non-trivial. Such robots are highly complex, non-linear dynamical systems that are hard to control. Sensor noise, imprecise actuators, and external disturbances further complicate control.

Among the most successful approaches to biped locomotion are trajectory tracking methods that are based on precomputed trajectories of the legs or the Zero Moment Point (ZMP) [1]. The joint trajectories are generated offline, for example, by solving the dynamic equations of motion. During the walking process, precise controllers are used for following the pre-computed trajectories. Trajectory tracking, however, requires a precise model of the robot. At the same time, solving the dynamical equations of motion can be computationally very demanding if a robot has many degrees of freedom (DOF). The approach presented here does not require an accurate model of the robot and has a low computational complexity.

A completely different approach is that of passive dynamic walkers (PDW). They use the inherent machine dynamics for walking and most of them walk without actuation or control. McGeer [2] first introduced the notion of PDW and showed that unactuated and uncontrolled planar walking down a slope is possible. The main interest of the approach is the evolution of periodic gaits, without considering starting or stopping, while agility and responsiveness of motion, as required by a versatile robot such as ours, play a minor role.

In addition, biologically inspired methods have evolved in which central pattern generators (CPG) [3] generate joint trajectories by using nonlinear oscillators. CPG-driven locomotion has been successfully applied in cases where a complete dynamical model of the robot is not available or too complex to be useful [4], [5]. The disadvantage of CPG-driven methods is that it is not trivial to determine appropriate parameter settings for the oscillators in order to achieve a stable gait. Our approach is similar to CPG methods in that joint trajectories are generated and adapted online.

Although CPG-based methods do not use a model, stochastic optimization of their parameters is possible. This has been demonstrated, e.g., with the Aibo quadrupeds [6]–[8]. In order to reject disturbances, sensory feedback is essential. Consequently, several successful attempts to incorporate sensory feedback have been reported, e.g. [9], [10].

In this paper, we propose extending an open-loop gait engine with two feedback mechanisms: gyro feedback that stabilizes the trunk and phase resetting at foot contact that entrains the CPG with the natural dynamics of the robot-environment system. The parameters of these feedback mechanisms are included in a stochastic optimization procedure. We evaluate the suitability of two state-of-the-art metaheuristics for our task and propose extensions that make them more data-efficient. This includes a sequential sampling procedure, an adaptive step size, and automatic restarting.

After reviewing some of the related work, we present the humanoid robot Jupp used for the experiments and its open-loop gait engine in Sec. III and IV, respectively. The proposed feedback mechanisms are detailed in Sec. V. The optimization problem is defined in Sec. VI and Sec. VII presents the PGRL and PSO metaheuristics together with the proposed extensions. The heuristics and their extensions are evaluated in a physics-based simulation, as described in Sec. VIII. Sec. IX reports the results of transferring stochastic gait optimization to the real robot.

II. Related Work

Stochastic gait optimization has been successfully demonstrated for the Aibo quadruped, e.g. by Kohl and Stone [6] who proposed Policy Gradient Reinforcement Learning (PGRL). They approximate the gradient of the objective function by sampling in the vicinity of a parameter vector. In contrast to

our approach, the authors did not make use of sensory feedback. Röfer [11] used the readings of acceleration sensors to judge stability of Aibo gaits during evolutionary optimization. However, the acceleration sensors were not used to adapt the gait online.

Geng et al. [10] successfully used PGRL for gait optimization of a planar biped. While their approach optimizes only two gait parameters, we include many more parameters in the optimization. This is possible because of the data-efficient sequential sampling procedure.

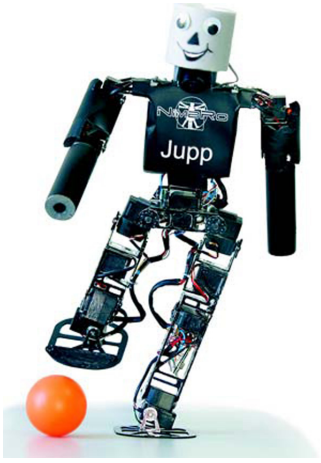
Hemker et al. [9] applied a sequential surrogate approach to optimize the speed of their humanoid robot. In contrast to our approach, the authors did not optimize feedback parameters. The gait also does not include a phase resetting mechanism.

Kosse [12] optimized the gait of a KHR-1 humanoid robot for forward speed using evolutionary strategies. His gait is completely open-loop. Niehaus et al. [13] optimized the gait of a humanoid robot using PSO. The robot also did not make use of any feedback.

Nakanishi et al. [4] proposed phase resetting to adapt the CPG-frequency of a planar biped. They reset the CPG-oscillators when the foot hits the ground. The approach learns open-loop gait parameters from demonstration, but does not include feedback parameters in the learning process.

Aoi and Tsuchiya [5] also incorporated a phase-resetting mechanism. The authors did not optimize the gait parameters and did not make use of gyroscope feedback.

III. KidSize Humanoid Robot Jupp



The robot used for our experiments is Jupp, a 19 DOF soccer robot from team NimbRo. Jupp is 60cm tall and weights only 2.3kg. Each leg has a 3 DOF hip, a knee, and a 2 DOF ankle joint. A pitch joint allows for bending the trunk, just above the hip. The arms have 2 DOF shoulders and an elbow joint. All joints are driven by RC-servo motors.

Most parts of the skeleton are made of rectangular aluminum tubes. The arms and the feet are made of carbon composite. A Pocket PC, located in the upper trunk of the robot, is the main computer. It generates target positions for all joints at a rate of 83Hz. Jupp is equipped with two gyroscopes (ADXRS, $\pm$ 150/300 °/s) that are located in the robot's trunk. Two force sensing resistors (FSR) are attached to each heel. They are used to measure foot contact with the ground. See [14] for more hardware details.

IV. Open-loop Gait Engine

The starting point for our work is a clock-driven, open-loop gait developed by Behnke [14]. The gait is driven by a central clock $-\pi \leq \phi_{\mathsf{Trunk}} < \pi$ that determines the step frequency ψ. Each leg derives its gait phase ϕ_{Leg} by shifting the trunk phase: $\phi_{\mathsf{Leg}} = \phi_{\mathsf{Trunk}} + \Lambda\frac{\pi}{2}$. The leg sign Λ is $\Lambda = -1$ for the left leg and $\Lambda = 1$ for the right leg. To abstract from the individual joints, we implemented a kinematic interface that allows for changing leg extension γ, leg angle $\theta_{\mathsf{Leg}} = (\theta^r_{\mathsf{Leg}}, \theta^p_{\mathsf{Leg}}, \theta^y_{\mathsf{Leg}})$, and foot angle $\theta_{\mathsf{Foot}} = (\theta^r_{\mathsf{Foot}}, \theta^p_{\mathsf{Foot}})$. Superscripts r, p, and y indicate the roll, pitch, and yaw direction, respectively.

Each leg generates sinusoidal trajectories for this interface from its phase ϕ_{Leg}. Although the original gait is omnidirectional, we consider here only walking in forward direction. The key ingredients of the open-loop gait are: shifting the weight from one leg to the other, shortening the leg not needed for support, and leg motion in walking direction. The gait speed and direction can be varied while the robot is walking by adjusting the leg swing amplitude $a_{\mathsf{Swing}} = (a^r_{\mathsf{Swing}}, a^p_{\mathsf{Swing}}, a^y_{\mathsf{Swing}})$. The robot starts walking by first fading in the lateral weight shift until it is stepping on the spot and then fading in a_{Swing}. The open-loop gait engine has a set of parameters that have been adapted to the robot. Many of them have an intuitive interpretation.

V. Feedback Control

Jupp reached a top speed of 22.5 cm/s using the hand-tuned open-loop gait. In order to achieve higher speeds, we augment the open-loop gait with two feedback control mechanisms using readings from the two gyroscopes and the FSRs.

A. Gyroscope Feedback

If the angular velocities measured by the gyroscopes $\omega^{r/p}_{\mathsf{Gyro}}$ become too large, the robot is likely to loose balance. In order to induce torque in the opposite direction to the possible fall, we use a P-controller to modify the foot angles $\theta^{r/p}_{\mathsf{Foot}}$ to $\theta^{r/p}_{\mathsf{FootGyro}}$. We want the angular velocity of the upper body to be zero. Thus, the gyroscope P-controllers with gains $K^{r/p}$ are

$$\begin{aligned} \theta^r_{\mathsf{FootGyro}} &= \theta^r_{\mathsf{Foot}} + \Lambda \cdot K^r \cdot \omega^r_{\mathsf{Gyro}} \quad \text{and} \\ \theta^p_{\mathsf{FootGyro}} &= \theta^p_{\mathsf{Foot}} + K^p \cdot \omega^p_{\mathsf{Gyro}}. \end{aligned}$$

B. Phase Reset

For fast and stable walking, it is essential that the frequency of the central pattern generator inside the robot and the natural frequency of the robot-environment system are tuned to each other. While it is possible to find a fixed step frequency that is on average tuned to the system, disturbances require the online-adaptation of the step duration.

In order to achieve an entrainment between the internal clock and the robot-environment system, we reset the trunk phase ϕ_{Trunk} after the foot hits the ground. When contact of the foot with the ground is sensed by the FSRs at time t_{FC},

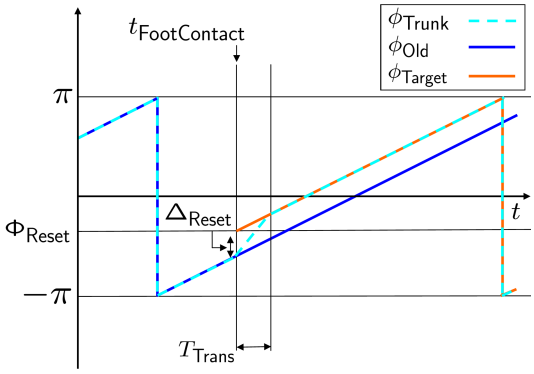


Fig. 1. Phase Resetting: After the foot hits the ground at $t_{\text{FootContact}}$, the trunk phase ϕ_{Trunk} changes by Δ_{Reset}, distributed over a time T_{Trans}.

the difference between the current trunk phase and the nominal start of the next step Φ_{Reset} is calculated:

$$\Delta_{\text{Reset}} = \Phi_{\text{Reset}} - \phi_{\text{Trunk}} \qquad \text{at } t = t_{\text{FC}}.$$

In order to avoid discontinuities in the joint trajectories, the required phase change Δ_{Reset} is distributed over a transient time T_{Trans}. Let ϕ_{Old} be the continued course of ϕ_{Trunk} if it was not reset and $\phi_{\text{Target}} = \phi_{\text{Trunk}} + \Delta_{\text{Reset}}$ be the course of the phase after an immediate jump of Δ_{Reset}. As illustrated in Fig. 1, ϕ_{Trunk} linearly fades from ϕ_{Old} to ϕ_{Target}:

$$\phi_{\text{Trunk}} = \phi_{\text{Old}} + (t - t_{\text{FC}})\frac{\Delta_{\text{Reset}}}{T_{\text{Trans}}} \quad \text{if } t_{\text{FC}} < t < t_{\text{FC}} + T_{\text{Trans}}.$$

As the phases of all other body parts are derived from the trunk phase, they reset as well. While our resetting mechanism changes the timing of the walking motion, the trajectory generation procedure is not changed.

VI. Stochastic Optimization

Both, the open-loop gait engine and the proposed feedback mechanisms have parameters that need to be adjusted to produce fast and stable walking patterns. As we do not have a good model of the robot, we follow a model-free approach.

A fitness function $f(\mathbf{x})$, $f : \mathbb{R}^N \to \mathbb{R}$, expresses our optimization goals: speed and robustness. This function evaluates a parameter vector $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ through a walking experiment, which is called a trial. The robot starts from a standing position and walks for a certain distance d_{exp} while using the gait parameters $\mathbf{x}$. The quality of the gait is judged using the time needed to reach the desired distance. In case the gait is unstable and the robot falls, the distance traveled is used. Because we prefer slower robust gaits over faster unstable ones, we formulate the fitness function in a way that any fall-free trial receives a higher score than a fall.

The speed v of the robot is divided by a maximum speed v_{max}, which will not be exceeded. In case the robot falls, this relative speed is multiplied with the relative distance traveled (d/d_{exp}). Otherwise, the robot receives an extra bonus of $+1$ for stability:

$$f(d, v) = \begin{cases} \frac{v}{v_{\text{max}}} \cdot \frac{d}{d_{\text{exp}}} & \text{if } d < d_{\text{exp}} \\ \frac{v}{v_{\text{max}}} + 1 & \text{else.} \end{cases}$$

The optimization problem is now to find the parameter vector $\mathbf{x}^*$ with the highest fitness:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} f(\mathbf{x}).$$

The optimization procedure faces several difficulties. First, f is not a deterministic function, but the fitness of a trial is a random variable that depends on disturbances encountered. Evaluating the same parameter vector twice might result in completely different fitness values, e.g. if the robot falls in one trial and walks stable in the other. To reduce the variance of the fitness, we evaluate each parameter vector three times, if not stated otherwise. Another difficulty is that the fitness function is not necessarily smooth. Slight parameter changes can make a stable gait pattern fall. Hence, common gradient descent methods can not be applied for optimizing f. Finally, the individual parameters are highly dependent on each other, which makes it impossible to tune them individually, and the mapping from parameters to the fitness is quite non-linear. Various metaheuristics have been proposed in the literature to tackle such difficult optimization problems.

VII. Metaheuristics

We evaluated two state-of-the-art metaheuristics for suitability to our task: Policy Gradient Reinforcement Learning (PGRL), introduced by Kohl and Stone [6], and Particle Swarm Optmiziation (PSO) by Kennedy and Eberhart [15]. Both metaheuristics have been successfully applied to similar optimization problems in the past [6], [16].

A. Policy Gradient Reinforcement Learning

In PGRL, each parameter vector is considered as an open-loop policy that can be executed by the robot. The algorithm randomly generates B test policies $\{\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^B}\}$ around policy $\mathbf{x}^\pi$, which is to be improved. The parameters x_j^i of the test policies $\mathbf{x^i}$ are set randomly to either $x_j^\pi - \epsilon$, to x_j^π, or to $x_j^\pi + \epsilon$, for all $1 \leq i \leq B$ and $1 \leq j \leq N$. The disturbance ϵ is a small constant value.

The test policies $\mathbf{x^i}$ are executed. The obtained fitness samples are grouped into three categories for each dimension j: S_j^-, S_j^0 and S_j^+, depending on whether the jth parameter of $\mathbf{x^i}$ is modified by $-\epsilon$, 0, or $+\epsilon$. For each set, the average fitness $\overline{z}_j^-$, $\overline{z}_j^0$ and $\overline{z}_j^+$ is computed and an adjustment vector $\mathbf{a} = (a_1, a_2, \ldots, a_N)$ is constructed as follows:

$$a_j = \begin{cases} 0 & \text{if } \overline{z}_j^0 > \overline{z}_j^+ \text{ and } \overline{z}_j^0 > \overline{z}_j^-, \\ \overline{z}_j^+ - \overline{z}_j^- & \text{otherwise.} \end{cases}$$

The adjustment $\mathbf{a}$ is normalized to a scalar step-size η and added to $\mathbf{x}^\pi$:

$$\mathbf{x}^\pi \leftarrow \mathbf{x}^\pi + \eta\frac{\mathbf{a}}{|\mathbf{a}|}.$$

PGRL continues to test the vicinity of the adjusted policy $\mathbf{x}^\pi$ for possible improvements in the next iteration.

Using average fitnesses to estimate the gradient of the fitness function and a missing memory make PGRL robust against outliers. The stochastic selection of test policies enables PGRL to escape some local minima.

Extensions to the PGRL Algorithm

1) Adaptive Step Size: To implement a coarse-to-fine search, we adapt the step size η. We start with a step size η_{max}. After g steps, we decrease η linearly to η_{min}:

$$\eta = \begin{cases} \eta_{\mathsf{max}} & \text{if } s < g \\ \eta_{\mathsf{max}} - \frac{(s-g)\cdot(\eta_{\mathsf{max}}-\eta_{\mathsf{min}})}{S-g} & \text{else.} \end{cases}$$

Here, s counts the number of fitness function evaluations and S denotes the maximum allowed number of function evaluations.

2) Sequential Sampling: Instead of sampling the fitness function for each test policy $\mathbf{x^i}$ three times (through three walking trials), we apply an approach proposed by Branke and Schmidt [17] that does not fix the number of samples in advance. Instead, samples are generated one at a time until a certain level of confidence for the fitness is achieved, or a maximum number of samples M have been generated. The target level of confidence is determined by comparing the fitness of a test policy $\mathbf{x^i}$ to that of the policy $\mathbf{x}^\pi$, which is to be improved. If the fitnesses of the two policies are very different, only few samples suffice for estimating the direction of the gradient of the fitness function. For policies with similar fitness values more samples are required to guarantee a certain level of confidence. The sequential sampling procedure is summarized in Algorithm 1.

Algorithm 1 Sequential Sampling Procedure

Input: Average policy fitness $\overline{z}_\pi$, test policy $\mathbf{x^i}$, threshold for fitness difference ν, max. no. of samplings M
Output: Average fitness $\overline{z}_i$ of policy $\mathbf{x^i}$, number of samplings used m

$\overline{z}_i \leftarrow$ fitness obtained by sampling policy $\mathbf{x^i}$ once
$d \leftarrow |\overline{z}_i - \overline{z}_\pi|$
for $m = 1$ to $M - 1$ **do**
  if $|d| > \nu$ **then**
    return $(\overline{z}_i, m)$
  else
    $z_i \leftarrow$ fitness obtained by sampling $\mathbf{x^i}$ one more time
    $\overline{z}_i \leftarrow (\overline{z}_i \cdot m + z_i)/(m+1)$
    $d \leftarrow |\overline{z}_i - \overline{z}_\pi|$
  end if
end for
return $(\overline{z}_i, m)$

B. Particle Swarm Optimization

Particle Swarm Optimization (PSO) works on a set (swarm) of parameter vectors (particles) $\mathbf{x^i}$. Each particle has a velocity $\mathbf{v^i} = (v_1^i, v_2^i, \ldots, v_N^i)$. The velocity indicates how much the value of the corresponding parameter changes and thus the position of the particle, in the next iteration of the algorithm. It is altered according to the best point in parameter space a particle has visited so far, $\mathbf{p^i} = (p_1^i, p_2^i, \ldots, p_N^i)$, and the global best point PSO has found so far, $\mathbf{p^g}$:

$$v_j^i \leftarrow w \cdot v_j^i + c_1 \cdot \chi_1 \cdot (p_j^i - x_j^i) + c_2 \cdot \chi_2 \cdot (p_j^g - x_j^i).$$

Here, parameters w, c_1 and c_2 influence the balance between local and global exploration. The weights $\chi_{1/2}$ add stochasticity to the search. They are drawn uniformly from $[0,\ 1]$.

The positions of the particles are updated accordingly:

$$\mathbf{x^i} \leftarrow \mathbf{x^i} + \mathbf{v^i}.$$

Extension to PSO

To test for convergence of the algorithm, we sum up the distances of all J particles to the current global best position:

$$q_{\mathsf{PSO}} = \sum_{\mathsf{j}=1}^{J} |\mathbf{x^j} - \mathbf{p^g}|.$$

When q_{PSO} is smaller than a threshold τ_{PSO}, the algorithm has converged and only small improvements can be expected by continuing the search. We then restart the algorithm by generating a new set of randomly generated particles.

VIII. Experiments in a Physics-Based Simulation

The simulator used in this work is based on the Open Dynamics Engine (ODE), a library for simulating rigid body dynamics [18]. For the simulation experiments the experimental distance is set to $d_{\mathsf{exp}} = 4.5$m. One trial is complete when the robot has walked this distance in any direction[1] or if it falls beforehand.

In order to be able to transfer the optimization procedure to the real robot, we limit the number of trials (evaluations of the fitness function) to 1000. We call 1000 trials an *episode*. A *configuration* for the learning problem includes a metaheuristic, parameters of the metaheuristic (e.g. step size, number of start individuals), and a parameter set subject to optimization.

When comparing different configurations, we do not only consider the overall best (fittest) parameter vector encountered during an episode. To obtain a better characterization of the optimization performance, we obtain a distribution of 1000 fitness values as follows: We evaluate each configuration 10 times. From each of the 10 episodes, we take the 10 best individuals. We then evaluate these 100 individuals 10 times. We compare the resulting distributions using notched box plots which can be viewed as a visualization of a t-test.

Because the number of evaluations of the fitness function is limited both in simulation and in the real-world experiments, not all parameters of the gait can be optimized simultaneously. To avoid the curse of dimensionality, we select nine open-loop trajectory generation parameters and four feedback parameters that we consider to be relevant candidates for optimization. For an overview of the parameters, see Table I. In order to avoid scaling problems, the ranges of parameter values considered for optimization are normalized to the interval $[0, 1]$.

[1] In the real-world experiments presented in Section IX, we use a more restrictive goal formulation.

TABLE I

OPTIMIZED PARAMETERS

Open-Loop Parameters	
* ψ	step frequency
* a_{Swing}	swing amplitude of the leg in forward direction
* a_{Shift}	amplitude for lateral weight shifting
* $\lambda_{\mathsf{FootSwing}}$	partial balance of the leg swing with the foot angle
* $\lambda_{\mathsf{FootPitch}}$	general tilt of the robot in sagittal direction
* $a_{\mathsf{FootPitch}}$	oscillates the robot in sagittal direction with every step
$\rho_{\mathsf{FootPitch}}$	phase shift of the $a_{\mathsf{FootPitch}}$ oscillation
$\lambda_{\mathsf{LegSpread}}$	offset to the lateral leg angle
a_{Arms}	amplitude of the arm movement

Feedback Control Parameters	
* K^r	gain of lateral gyro feedback
* K^p	gain of sagittal gyro feedback
* Φ_{Reset}	nominal trunk phase for phase resetting
T_{Trans}	transition time for phase resetting

* Parameters used for the real-world experiment

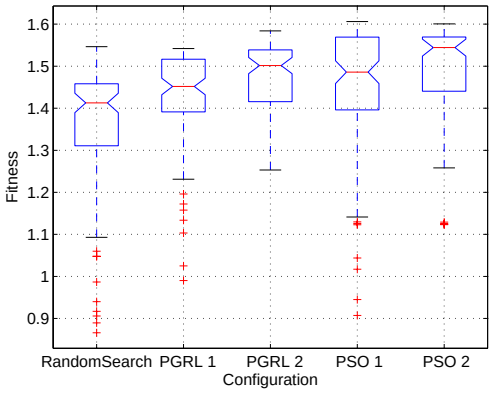


Fig. 2. Comparing metaheuristics. See text for configuration descriptions.

A. Comparing PGRL and PSO Metaheuristics

In this section, we compare two metaheuristics and their variants in order to find the most adequate one for our task. We use a uniform random search as baseline. All metaheuristics optimize the parameter set

$$P_1 = \{\psi, a_{\mathsf{Robot}}, a_{\mathsf{Shift}}, \lambda_{\mathsf{FootPitch}}, K^r, K^p, \Phi_{\mathsf{Reset}}, T_{\mathsf{Trans}}\}.$$

The hand-tuned parameters $\mathbf{x}^{\mathsf{init}}$ are used to initialize PGRL.

We evaluate two different configurations of extended PGRL. Both configurations use a threshold for selecting additional samples in the sequential sampling procedure of $\nu = 0.015$ and generate $B = 10$ test policies with a disturbance of $\epsilon = 0.015$. The step size of the first configuration (PGRL 1) falls from $\eta_{\mathsf{max}} = 0.7$ to $\eta_{\mathsf{min}} = 0.1$ after $g = 100$ steps. The second configuration (PGRL 2) uses $\eta_{\mathsf{min}} = 0.3$.

We evaluate two configurations of the PSO. The first PSO configuration (PSO 1) works on $J = 10$ particles. Both, c_1 and c_2 are set to 1. The second configuration (PSO 2) works on $J = 20$ particles with $c_1 = c_2 = 0.5$. For both configurations, a restart threshold $\tau_{\mathsf{PSO}} = 0.3$ and a weight momentum of $w = 0.3$ are used.

The box plots for comparing these six configurations are depicted in Figure 2. All tested metaheuristics perform better than the random search.

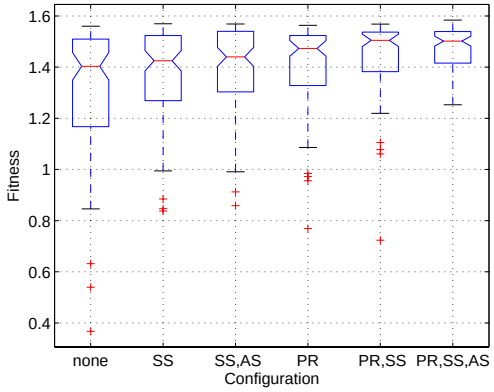


Fig. 3. Relevance of phase resetting (PR), sequential sampling (SS), and an adaptive step size (AS).

When comparing the two PGRL variants, the configuration with the larger target step size $\eta_{\mathsf{min}} = 0.3$ (PGRL 2) outperforms the one with the smaller step size $\eta_{\mathsf{min}} = 0.1$ (PGRL 1), indicating that $\eta_{\mathsf{min}} = 0.1$ is chosen too small. The PSO configurations find both a best individual with a very high fitness (1.61 and 1.6). PSO 2 performs better than PSO 1.

The results of the PSO configurations should be taken with a grain of salt: When the PSO has converged, the particles are concentrated within a small region of the search space. Thus, taking the 10 best individuals from one episode comes close to taking the same (best) individual ten times. As PGRL does not suffer from this problem and PGRL 2 performed nearly as good as the best PSO configuration, we decided to use PGRL 2 in further simulation experiments and for the real-world experiment.

B. Relevance of phase resetting, sequential sampling and an adaptive step size

To show the relevance of phase resetting, sequential sampling and an adaptive step size, we compare six different configurations. The first two configurations neither use sequential sampling nor an adaptive step size. The second two configurations make use of sequential sampling, but do not adapt the step size. The third two configurations use both, sequential sampling and an adaptive step size. One of the two configurations always uses phase resetting and the other one doesn't.

Box plots for all six configurations are shown in Fig. 3. The configurations using phase resetting (PR) outperform the configurations without phase resetting in all three cases. This indicates that the phase reset mechanism is essential to achieve fast and robust locomotion.

It can also be seen that configurations using sequential sampling (SS) outperform configurations without it and that configurations using additionally an adaptive step size (AS) outperform corresponding configurations without it. This indicates that both, sequential sampling and an adaptive step size improve the PGRL algorithm and lead to better results.

C. Parameter Selection

In the previous experiments, the parameters of set P_1 were optimized. However, P_1 might not be the optimal choice of parameters. In order to find other relevant parameters and to exclude irrelevant ones, we conduct additional experiments in the same evaluation framework.

We test the parameters $\lambda_{\mathsf{FootSwing}}$, $a_{\mathsf{FootPitch}}$, $\rho_{\mathsf{FootPitch}}$, $\lambda_{\mathsf{LegSpread}}$ and a_{Arms} by adding one parameter at the time to P_1. We evaluate these extended parameter sets using PGRL 2 and compare them to P_1. To test whether P_1 contains dispensable parameters, we construct reduced sets by excluding one of six candidate parameters. The parameters ψ and a_{Swing} are not removed, because they are clearly relevant.

The results of the experiments indicate that $a_{\mathsf{FootPitch}}$ and $\lambda_{\mathsf{FootSwing}}$ should be included in the optimization and that T_{Trans} is not relevant. Consequently, we use a modified parameter set for the real-world experiment. The parameters are marked by an * in Table I.

IX. Real-World Experiment

After evaluating metaheuristics, extensions to them, and parameter sets in simulation, we selected the most promising configuration for optimizing the gait of our humanoid robot Jupp. Fig. 4 illustrates the experimental setup. The robot starts a trial standing on one side of a RoboCup KidSize field, which is covered by carpet. A trial consists of walking across the field. This corresponds to $d_{\mathsf{exp}} = 3$m.

The robot is controlled by a standard PC, visible on the left side of the figure. Joint targets are transferred to the robot via an RS-232 serial cable. The robot reports back the readings of the FSRs and the gyros. Although Jupp's Pocket PC is not used for control during optimization, the robot carries it, because its weight significantly influences the robot dynamics.

We measure position and speed of the robot with the help of a laser range scanner [19], shown right. To ensure that some laser beams are reflected, we attach a white piece of paper on the robot's back. A trial finishes when the robot has walked the experimental distance, if it falls beforehand, or if it leaves the area of the laser range scanner.

The fitness function for the real-world experiments is more restrictive, compared to the simulation experiments: In order to reward straight walks, the robot position is projected orthogonally onto the line representing an ideal straight walk. The distance walked d is measured on this line.

During trials, the experimenter carries in one hand a lace connected to Jupp's shoulders to protect the robot when it falls. In the other hand the experimenter carries a joystick to send control commands to the PC. As a starting point for optimization, we use hand-tuned parameters that enable the robot reach a top speed of 21.3cm/s. This is close to the top speed Jupp has reached so far (22.5cm/s) using a hand-tuned open-loop gait. The initial fitness is 1.36.

The result of the real-world experiment is depicted in Fig. 5. The figure shows the sampled fitness values from all trials that did not end prematurely. The clear trend towards higher fitness values indicates that the stochastic optimization procedure

Fig. 4. Experimental setup. The robot starts at the field border and walks across the field. A white sheet is attached to its back to facilitate position tracking with a laser-range scanner (right). An external PC (left) controls the robot and executes the optimization procedure.

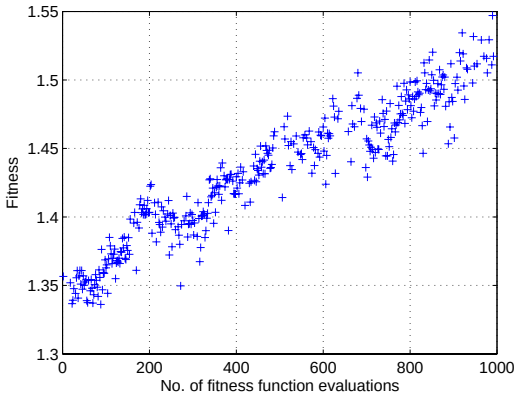


Fig. 5. Evolution of the fitness during the real-world experiment. All trials that did not end prematurely are depicted.

could be transfered successfully to the real robot. After 1000 fitness function evaluations, the best stable parameter vector found has a fitness of 1.52. With these parameters, Jupp walks the experimental distance in 11.5s. The measured top-speed of this gait is 34.0cm/s. The achieved speed is a gain of almost 60% compared to the start of the optimization and a 51% gain compared to the former top speed of Jupp. A video of Jupp walking at 34.0cm/s can be seen at `http://www.nimbro.net/movies/jupp/Jupp_Walking_Large_Steps.wmv`

Table II summarizes initial and final values of the optimized parameters. When analyzing the parameter changes, one can see that the speed gain was mainly the result of increasing the swing amplitude a_{Swing}, which results in a larger step length. Interestingly, the step frequency ψ remains close to its initial value. This indicates that there exists a natural frequency at which the robot's body oscillates in both, the sagittal and lateral direction while walking. This frequency, or a multiple of it, has to be met by the step frequency in order to walk in a stable way. One can also see that the gyro feedback gain K^p did not have an optimal initial value. This underlines the relevance of optimizing feedback parameters.

TABLE II

PARAMETER VALUES BEFORE AND AFTER OPTIMIZATION

Parameter	Range	Initial value	Final value
ψ	0.66Hz–0.93Hz	0.80Hz	0.81Hz
a_{Robot}	$[1.0,\ 3.0]$	1.3	2.22
a_{Shift}	$[0.075,\ 0.175]$	0.013	0.0133
$\lambda_{\mathsf{FootSwing}}$	$[0.15,\ 0.35]$	0.25	0.25
$\lambda_{\mathsf{FootPitch}}$	$[0.02,\ 0.08]$	0.05	0.044
$a_{\mathsf{FootPitch}}$	$[0.0,\ 0.02]$	0.01	0.008
K^r	$[0.4,\ 1.0]$	0.49	0.48
K^p	$[0,\ 0.5]$	0.2	0.29
Φ_{Reset}	$[-0.35\pi,\ 0.35\pi]$	0.0π	-0.01π

X. CONCLUSION

In this paper, we presented a stochastic optimization approach to optimize the walking pattern of a humanoid robot for forward movement. We started from an open-loop gait and enhanced it with a gyroscope P-controller and phase resetting.

We optimized both open-loop trajectory generation parameters and feedback parameters of the gait using two metaheuristics. We evaluated a policy gradient reinforcement learning approach and Particle Swarm Optimization for their applicability to our task. To improve data-efficiency, we extended the standard version of PSO by a restarting mechanism and we extended PGRL by an adaptive step size and a sequential sampling procedure.

The experiments carried out in a physics-based simulation environment suggested that PGRL is the most reliable algorithm for our task. Furthermore, the results indicated that our extensions to the PGRL improve the performance of the algorithm significantly. We also found clear evidence that the phase resetting mechanism is essential for achieving fast and robust locomotion.

We selected the extended PGRL algorithm to optimize the gait of our humanoid robot Jupp. The optimization procedure found a parameter configuration that enables the robot to walk at a speed of 34cm/s. This improves the former top speed achieved using a hand-tuned gait by more than 50%.

It is not easy to compare different approaches in humanoid robotics as no competitive standard platform like the Sony Aibos for quadrupeds exists. A robot similar to Jupp is Bruno from TU Darmstadt [9]. The 55cm tall robot reaches a speed of 40cm/s and is considered to be one of the fastest humanoids of it's size class. Please note that Bruno does not use RC-servos, but intelligent Dynamixel actuators, which produce more than three times the knee torque, compared to Jupp. At the University of Dortmund, Kosse [12] has optimized the gait of a Kondo KHR-1 robot (34cm), which reached a top speed of 22cm/s. Another robot of similar size is the Sony Qrio [20]. It is 58cm tall and can jog at a speed of 23cm/s. In comparison to Jupp, it is a significantly more expensive machine that uses proprietary intelligent servo actuators.

The experiments show that although Jupp has been used for quite some time now and its joints have significant backlash, after the proposed stochastic optimization of its gait parameters, it can compete with the fastest humanoid robots in its size class.

In future work, we would like to explore more options to improve the data-efficiency of the optimization procedure. One idea would be to employ Gaussian process optimization that showed promising results for the Aibo dogs [21]. Another direction of research would be to include more advanced feedback strategies in order to further improve walking stability. Promising strategies have been demonstrated, e.g., by the Honda Asimo robot that accelerates the gait when it detects a disturbance that would lead to toppling forward [22]. Finally, in order to make it possible to use the optimized gait for soccer, we intend to extend the stochastic optimization to omnidirectional walking.

ACKNOWLEDGMENT

This work is funded by the DFG under grant BE 2556/2-2.

REFERENCES

[1] M. Vukobratovic and B. Borovac, "Zero-moment point - thirty five years of its life," *Int. Journal of Humanoid Robotics*, pp. 131–147, 2003.
[2] T. McGeer, "Passive dynamic walking," *International Journal of Robotics and Research*, vol. 9(2), pp. 62–68, 1990.
[3] G. Endo, J. Morimoto, J. Nakanishi, and G. Cheng, "An empirical exploration of a neural oscillator for biped locomotion control," in *Proc. of ICRA*, 2004, pp. 3036–3042.
[4] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "An empirical exploration of phase resetting for robust biped locomotion with dynamical movement primitives," in *Proc. of IROS*, 2004, pp. 919–924.
[5] S. Aoi and K. Tsuchiya, "Locomotion control of a biped robot using nonlinear oscillators," *Autonomous Robots,* 19(3), pp. 219–232, 2005.
[6] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *Proc. of ICRA*, 2004, pp. 2619–2624.
[7] M. S. Kim and W. Uther, "Automatic gait optimisation for quadruped robots," in *Australasian Conf. on Robotics and Automation*, 2003.
[8] M. J. Quinlan, S. K. Chalup, and R. H. Middleton, "Techniques for improving vision and locomotion on the Sony Aibo robot," in *Proc. of Australasian Conf. on Robotics and Automation*, 2003.
[9] T. Hemker, H. Sakamoto, M. Stelzer, and O. v. Stryk, "Hardware-in-the-loop optimization of the walking speed of a humanoid robot," in *Proc. of CLAWAR*, 2006, pp. 614–623.
[10] T. Geng, B. Porr, and F. Wörgötter, "Fast biped walking with a sensor-driven neuronal controller and real-time online learning," *Int. Journal of Robotics Research*, vol. 25, Issue 3, pp. 243–2590, 2006.
[11] T. Röfer, "Evolutionary gait-optimization using a fitness function based on proprioception," in *RoboCup 2004*. Springer, 2005, pp. 310–322.
[12] R. Kosse, "Planung und Implementierung eines evolutionären Ansatzes zur Steuerung eines zweibeinigen Roboters," Univ. of Dortmund, 2006.
[13] C. Niehaus, T. Röfer, and T. Laue, "Gait-optimization on a humanoid robot using particle swarm optimization," in *Proc. of The Second Workshop on Humanoid Soccer Robots @ IEEE-RAS 7th International Conference on Humanoid Robots*, Pittsburgh, 2007.
[14] S. Behnke, "Online trajectory generation for omnidirectional biped walking," in *Proc. of ICRA*, 2006, pp. 1597–1603.
[15] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. of Int. Conf. on Neural Networks*, 1995, pp. 1942–1948.
[16] X. Zhang, L. Yu, Y. Zheng, and Y. Shen, "Two-stage adaptive PMD compensation," *Optics Communications*, vol. 231, pp. 233–242, 2003.
[17] J. Branke and C. Schmidt, "Sequential sampling in noisy environments," in *Parallel Problem Solving from Nature*. Springer, 2004, pp. 202–211.
[18] R. Smith, "Open dymanics engine," in *http://www.ode.org*.
[19] J. Stückler, "Optimierung der Laufmuster eines Humanoiden Roboters durch Reinforcement-Lernen," Study thesis, Univ. of Freiburg, 2006.
[20] Y. Kuroki, M. Fujita, T. Ishida, and K. N. und J. Yamaguchi, "A small biped entertainment robot exploring attractive applications," in *Proc. of ICRA*, 2003, pp. 471–476.
[21] D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans, "Automatic gait optimization with gaussian process regression," in *Proc. of IJCAI*, 2007.
[22] Honda Inc., "The Honda Asimo humanoid robot," 2003.