Optimization of Humanoid Walking Controller: Crossing the Reality Gap

Miguel Alexandre Campos Oliveira[1] and Stephane Doncieux[2]
and Jean-Baptiste Mouret[3] and Cristina Peixoto Santos[4]

***Abstract*— Humanoid locomotion remains a challenge because of the inherent instability of such robotic platforms. Inspired from observations on animals, Central Pattern Generators have been proposed to support the generation of rhythmic patterns able to make a robot smoothly walk while requiring few computational power. Nevertheless, tuning such controllers is challenging, in particular because small irregularities in the walking pattern easily make the robot fall. Optimization algorithms can be used to tune them in simulation, but the transfer of such solutions to the real robot raises the *reality gap* problem, as a solution efficient in simulation may well be inefficient in reality. It is proposed here to use the transferability approach to solve this problem. Its principle is to learn a model of the transferability between simulation and reality while doing several evaluations on the real robot. This model is then used to estimate how well a controller will transfer onto the real robot and the optimization process tries to optimize it besides other cost functions related to locomotion and tested in simulation only. The approach has been applied to the DARWIN-OP robot.**

I. Introduction

Humanoid robots are highly unstable robotics platforms. Making a humanoid robot move around in a robust fashion still remains a challenge. Approaches like ZMP (Zero Moment Point) [29], [24] have proved their efficiency in a well controlled environment. Such approaches require a high computational load as they require complex geometrical computations to be performed at each time step. Likewise, they require to know beforehand where the feet should be placed. Although it may be appropriate in some situations, it would be interesting to design controllers that require less computational power and that are also more robust. Drawing inspiration from how animals do indeed walk, approaches based on Central Pattern Generators adopt a completely different point of view [8]. Instead of computing exactly the position of each part of the robot, they consist in synchronized dynamical systems linked to leg motors and that make them move in a rhythmic fashion. Using CPG for walking results then in a smoother walking pattern and also requires much less computational power as all is needed is to update the dynamics of the CPG, what is by far less demanding than other geometrical based approaches.

CPG based controllers have been used in different kind of legged robots, even quadruped [27], [18] and biped robots. Most typical approaches when implementing CPGs for bipedal walking use phase oscillators that are fed into pattern generation layers [21], [1], [3], [25] or using neural oscillators [19].

One of the main difficulties while using CPGs, consists in finding out how to tune them to exhibit a particular motion pattern. This is particularly critical when designing walking controllers for humanoid controllers as they can easily fall. Optimization tools are particularly interesting in this context to automatically find the most appropriate set of parameters and thus avoid a tedious manual tuning. Anyway, they raise a difficult problem: as the system to optimize is highly non-linear, appropriate methods mostly rely on metaheuristics, e.g. evolutionary algorithms, and thus require a large amount of tests. Such an approach is difficult to apply directly to the real robot, because it would require a huge amount of time and the robot could be damaged during the process, may it be because of the large number of tests or because some of them may put a great strain on the mechanics.

In this work, it is proposed to rely on a simulation to avoid such problems. In this case, there is a risk of over-fitting to features that are specific to the simulation, thus leading to a *reality gap*, i.e. to a situation in which a controller works well in simulation, while being completely inefficient on the real robot. To avoid this problem, the transferability approach has been used [22], [14]. This approach consists in performing some experiments on the real robot in order to learn a model of the *transferability* of controllers between simulation and reality. This model is then used in the optimization process in order to favor controllers that do transfer well. The approach is tested on a DARWIN-OP robot.

II. Related work

A. Humanoid walking with a CPG

Typical solutions to the problem of biped locomotion make extensive use of the knowledge of the robot and of its environment. Most of them are model-based solutions and employ inverse kinematics and kinetics in order to generate feet placement sequences, according to determined constraints established through some stability criterion, as the popular Zero-Moment Point (ZMP) [29], [24].

*This work is funded by FEDER Funding supported by the Operational Program Competitive Factors - COMPETE and National Funding supported by the FCT - Portuguese Science Foundation through project PTDC/EEACRO/100655/2008. Miguel Oliveira is supported by grant CRO-BI-2012

[1,4] Miguel Oliveira and Cristina P. Santos are with Industrial Electronics Department, School of Engineering, University of Minho, 4800-058 Guimaraes, Portugal mcampos@dei.uminho.pt, cristina@dei.uminho.pt

[2,3]Stephane Doncieux and Jean-Baptiste Mouret are with ISIR, Université Pierre et Marie Curie-Paris 6, CNRS UMR 7222 4 place Jussieu, F-75252, Paris Cedex 05, France doncieux@isir.upmc.fr, mouret@isir.upmc.fr

Bio-inspired approaches appear as possible alternatives to humanoid locomotion generation with quite successful results and less requirements. One such approach is based on the concept of Central Pattern Generators (CPGs), and exploits the characteristics of intraspinal neural networks in vertebrates [8]. These generate rhythmic activations for walking motor patterns.

There are several key points which make CPGs good candidates for legged robot control and can potentially generalize their ability in dynamic environments. These include (see [8] for a review): 1) CPGs produce stable rhythmic patterns with respect to their limit cycle behavior, returning to the normal stable state after transient perturbations. 2) They typically have a small number of control parameters, reducing the dimensionality of the control problem on higher level control. 3) Small changes in control parameters, even if abrupt, result in smooth modulations for the produced trajectories. 4) CPGs are well suited to integration of sensory feedback pathways [12], [1]. This provides the ability for the robots to tackle unexpected disturbances and thus make them able to work in dynamic and not completely known environments.

In [7], the authors proposed a CPG-based locomotion controller for Darwin-OP robot capable of achieving a velocity of $\approx$ 10cm/s in simulation. In [25], it is also proposed a locomotion controller based in CPGs. They achieved a 8cm/s velocity from a hand tuned CPG parameterization in simulation, and were able to optimize the robot velocity to 10(cm/s) using the multi objective NSGA-II algorithm.

However, the lack of a defined framework promoted the development of several CPG approaches [2], [26], [20], [23], [5], [28], [16]. Furthermore, many of these approaches rely on hand tuned ad-hoc solutions. Typically, the large number of parameters needed for robot locomotion and the lack of knowledge about the dynamical behavior between the robot and environment makes hand-tuning a difficult task for locomotion beyond simple flat terrain walking and navigation [15].

B. Crossing the reality gap

Many authors note that solutions optimized in simulation often do not work well once tested on the real robot [9], [30], [22], [13]. This issue, termed the "the reality gap" [9], is critical to enable the adoption of optimization algorithms in engineering: optimizing directly on the real device is too slow to be practical, but optimizing in simulation has a high probability of leading to useless solutions.

The most intuitive idea to cross this gap is to put the blame on simulators and to improve their accuracy. Several authors proposed to automatically optimize it, for instance by learning a surrogate model of the objective function [10], or to automatically improve an existing simulator [4], [30] with tests in reality. Nevertheless, automatically optimizing a simulator that is general enough to be used to optimize solutions is at least as challenging as creating good optimization algorithms for robotics. Moreover, more accurate simulators often means slower running times, which may cancel out the benefits of simulation.

In a radical proposition, Jakobi proposed to make simulator *less accurate* by hiding badly modeled phenomena in in an "envelope of noise". Despite some success with the evolution of a controller for an hexapod robot [9], Jakobi did not precisely describe how to choose what has to be noised and how this noise should be applied. For instance, it is hard to know how to add noise when optimizing locomotion controllers for humanoid robots.

The transferability approach is a recent alternative to cross the reality gap. It led to promising results when optimizing controllers for a wheeled robot [14], a quadruped [14], [22] and a hexapod [13]. This approach relies on the observation that *simulators make mistakes but they are also often right*. For instance, the rigid body simulator ODE have a very simple model of viscous joint damping, but systems with low viscous damping coefficients are accurately simulated [6]. The goal of the transferability approach is to make the optimization algorithm aware of such limits of the simulation thanks to an automated process. To this end, a few controllers are transferred during the optimization and a regression algorithm (e.g. a SVM or a neural network) is used to approximate the function that maps behaviors in simulation to the difference of performance between simulation and reality. To use this approximated *transferability function*, the single-objective optimization problem is transformed into a multi-objective optimization in which both performance in simulation and transferability are maximized.

III. Method

A. Fitness Specification

Based on the literature [15] and on our expertise in dynamical systems, the following objectives are used to evaluate the walking gait performance:

- **COP edge distance,** f_{COP}: the average distance of the COP to the edge of the foot support polygon, D_{COP}, when the foot is in stance phase is to be maximized.
- **Force,** f_{F}: a good gait should keep the feet parallel to the floor; have a minimum impact force between the feet and the floor when the stance starts, and have the measured force of the four built-in force sensors in each foot, uniformly distributed.
- **Displacement,** f_{D}: a gait is considered better if it achieves higher frontal displacements.
- **Ground Clearance,** f_{GC}: the robot should raise its feet while walking. In order to calculate the highest distance from the ground of a swing foot, we take the maximum distance to the ground of the lowest sensor of a swing foot.
- **Transferability,** f_{T}: as defined in section III-B.

Limb trajectories are shaped by varying the CPG parameters. In order to achieve biped locomotion for a biped robot that is stopped and has to start walking, we have considered a 3 staged evolution. Each stage is performed during a certain amount of time and different motions are sequentially

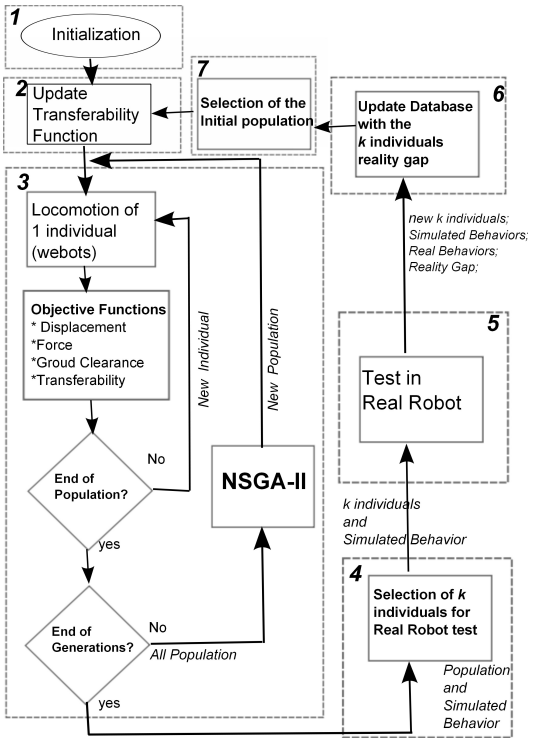

Fig. 1. Proposed Reality GAP Algorithm.

added in each stage. There is no interruption between the stages, meaning that motions set in one stage are kept to the following stages. Further, different objectives are evaluated during the stages, as follows:

- **Balancing Stage**: Balancing and Pelvis rotation motions. Objectives: max((f_{COP})), min((f_{F})) and min((f_{T})). Time: $t \leq 5$ s;
- **Stepping Stage**: add Flexion and Knee yielding motions. Objectives: max(f_{GC}), min(f_{F}) and min(f_{T}). Time: $5 < t \leq 9$ s;
- **Walking Stage**: all motions. Objectives: max(f_{D}), min(f_{F}) and min(f_{T}). Time: $9 < t \leq 20$ s.

B. Transferability for a humanoid robot

Each candidate solution i is described by a vector of 5 values $S_i = \{f_{\mathrm{Dsi}}, f_{\mathrm{Fsi}}, x_{\mathrm{si}}, y_{\mathrm{si}}, \phi_{\mathrm{si}}\}$, where f_{Dsi} corresponds to the displacement, f_{Fsi} to the force sensor value, x_{si} to the forward displacement, y_{si} to the lateral displacement and ϕ_{si} to robot's pitch, ϕ_s. If a candidate solution is tested on the real robot, the same values can be computed on the real robot, leading to an evaluation of the transferability (the "size" of the reality gap): $g_i = \sum |S_i - R_i|$, where R_i corresponds to S_i but measured on the real robot.

The set of couples (S_j, g_j), where j is a transferred controller, make a database that is used to train a Support Vector Machine (SVM). Once trained, this predictor can predict g_i for each value of S_j.

C. Optimization Loop

Figure 1 presents the algorithm's fluxogram, each step being defined as follows:

1) **Initialization**
   a) The initial population, of size n_p, is generated out of a hand tuned solution which is known to be able to walk on the real robot [25].
   b) All individuals $p_i \in P, i = \{1, \ldots, n_p\}$ are executed in simulation, yielding S_i.
   c) n_s individuals are chosen from population P. First we choose the best individual (with higher fitness functions), then, we incrementally select the most different w.r.t. those already chosen. By choosing the most different, we are increasing the diversity of the elements stored in the Database and then used to train the transferability function. The difference is computed according to: $d_i = \sum_{j=1}^{n_s} |S_i - S_j|$, that, is, we incrementally select the individuals with higher d_i. Where i indicates those from the population, $i \in \{1, \ldots, n_p\}$, that are being compared with j already chosen, $j \in \{1, \ldots, n_s\}$.
   d) Each of these selected individuals, $p_i \in P, i = \{1, \ldots, n_s\}$, is executed in reality, yielding $R_i = \{f_{\mathrm{Dri}}, f_{\mathrm{Fri}}, x_{\mathrm{ri}}, y_{\mathrm{ri}}, \phi_{\mathrm{ri}}\}$.
   e) Compute the reality gap g_i for each individual p_i, between the performance in simulation and in reality: $g_i = \sum |S_i - R_i|$.
   f) For each individual p_i, a Database entry, $D_i = \{p_i, S_i, R_i, g_i\}$, is created. This Database is used to maintain a record of all the individuals already executed in reality (and necessarily in simulation).
   g) Update the Database size, m, to the current number of elements: $m \leftarrow m + n_s$.
2) **Update Transferability function**
   The SVM is trained with all the m individuals in the Database and yields an updated transferability Function $f_T(p)$.
3) **Optimization** The NSGA-II loop is executed for n_e generations:
   a) **NSGA-II** Optimization loop for $p_i \in P, i = \{1, \ldots, n_p\}$:
      i) Evaluate individual p_i in simulation.
      ii) Acquire S_i.
      iii) Compute the objective functions.
      iv) Generate the new population based on NSGA-II algorithm.
4) **Selection**
   a) For each individual $p_i \in \{1, \ldots, n_e \times n_p\}$, calculate $d_i = \sum_{j=1}^{m} |S_i - S_j|$, where j iterates the elements in the Database.
   b) Select the k individuals with higher d_i, so as to maximize the diversity.
5) **Execution in Reality**
   a) Execute individuals $p_i, i \in \{1, \ldots, k\}$ in reality.
   b) Observe R_i.
   c) Compute the reality gap, g_i, for each individual: $g_i = |S_i - R_i|$.

6) **Update Database**
   a) Create a new entry D_i for each of the new k executed individuals: $D_i = \{p_i, S_i, R_i, g_i\}$.
   b) $m \leftarrow m + k$.
   c) If the stop criteria is not met - the number of gap iterations, n_g, is not reached - return to point 2.
7) **Selection of the Initial Population** if the number of gap iterations, n_g, is not reached, it is necessary to calculate the new population n_p for the Simulation NSGA-II loop. The selection of the individuals for the new population respect the following criterions:
   a) The individuals that were able to perform the walking stage in the real robot are selected.
   b) The individuals from all n_p populations of all n_e generations of one n_g gap iteration, with the best f_t are selected. To guarantee that only individuals with good f_t are chosen, we pick the individuals with $f_t > \frac{\sum_{i=1}^{n_e} \bar{x_i}}{n_e}$, and chose those with the best f_t and $\bar{x_i} = \frac{\sum_{j=1}^{n_p} f_{T_j}}{n_p}$, where $i = \{1, \ldots, n_e\}$

Best simulation solutions are often not transferable to reality because they typically exploit every details of the simulation, whereas differences between simulation and reality lies in these details. To be able to deal with possible conflicting solutions and search for relevant trade-off solutions, we apply a multi-objective approach,that considers several task-dependent objectives and a transferability objective.

D. CPG based control architecture

The locomotion controller used in this work relies on a Central Pattern Generator (CPG) capable of generating bipedal locomotor behaviors, such as walking forward/backwards and turning, as described in [17]. The system could easily integrate local sensory feedback. The proposed CPG controls a single leg, divided in rhythmic and unit motion pattern generators.

The use of a phase oscillator as a rhythmic generator allowed for a simple contralateral coupling between the left and right CPGs, maintaining the correct coordination between the generated locomotor trajectories in both legs by producing each a driving rhythmic signal ϕ_i in strict alternation (i for left/right leg). Motion pattern generators receive this rhythmic input and produce the corresponding joint trajectory, z, in a synergistic approach of modular motion primitives encoded as a set of non-linear dynamical equations with well defined attractor dynamics, similarly to other works [23]. Basic parameterized motion primitives, e.g. sine and bell-shaped motions (implemented as Gaussians), were considered.

The joint angle value $z_{i,j}$, for leg i and joint j, is:

$$\dot{z}_{i,j} = -\alpha\left(z_{i,j} - O_{i,j}\right) + \sum f_j^m\left(z_{i,j}, \phi_i, \dot{\phi}_i\right). \quad (1)$$

The final motor program in a single joint results from the sum of rhythmic motion primitives f_j^m around a center point $O_{i,j}$. α is a relaxation parameter for the offset. j specifies the joint: hip roll (hRoll), hip yaw (hYaw), hip pitch (hPitch), knee (kPitch), ankle roll (aRoll) and ankle pitch (aPitch); and i specifies the left or right leg.

The balancing motion is controlled as follows:

$$f_{\mathrm{hRoll}}^{\mathrm{balancing}} = -A_{\mathrm{balancing}}\omega\sin\left(\phi_i\right) \quad (2)$$
$$f_{\mathrm{aRoll}}^{\mathrm{balancing}} = -f_{\mathrm{hRoll}}^{\mathrm{balancing}} \quad (3)$$

i specifies the left or right leg, ϕ_i is the phase of left or right CPG, and parameter $A_{\mathrm{balancing}}$ specifies the amplitude of the lateral displacement motion.

Leg flexion motion is performed by the unloaded leg.This is shown in eq. (4) for hip, eq. (5) for knee, and eq. (6) for ankle.

$$f_{\mathrm{hPitch}}^{\mathrm{flex}} = \frac{A_{\mathrm{hip}}\omega\phi_i}{\sigma^2} exp\left(-\frac{\phi_i^2}{2\sigma^2}\right) \quad (4)$$
$$f_{\mathrm{kPitch}}^{\mathrm{flex}} = -\frac{A_{\mathrm{knee}}\omega\phi_i}{\sigma^2} exp\left(-\frac{\phi_i^2}{2\sigma^2}\right) \quad (5)$$
$$f_{\mathrm{aPitch}}^{\mathrm{flex}} = -\left(f_{\mathrm{hPitch}}^{\mathrm{flex}} + f_{\mathrm{kPitch}}^{\mathrm{flex}}\right) \quad (6)$$

The amplitude of the bell trajectory is specified by parameter A_{hip} for the hip and A_{knee} for the knee.

The pelvic rotation is performed at the hip yaw joints, described by :

$$f_{\mathrm{hYaw}}^{\mathrm{rotation}} = -A_{\mathrm{rotation}}\omega\sin\left(\phi_i + \frac{\pi}{2}\right) \quad (7)$$

The propulsion of the body during locomotion stems from leg motions in the sagittal plane, alternately moving the contralateral legs forward and backward, like a compass:

$$f_{\mathrm{hPitch}}^{\mathrm{compass}} = -A_{\mathrm{compass}}\omega\sin\left(\phi_i + \frac{\pi}{2}\right) \quad (8)$$
$$f_{\mathrm{aPitch}}^{\mathrm{compass}} = -f_{\mathrm{hPitch}}^{\mathrm{compass}} \quad (9)$$

The overall result of all the motions is straight bipedal walking. Here we summarize all the motions assigned to each joint:

$$\dot{z}_{\mathrm{hRoll}} = -\alpha(z_{\mathrm{hRoll}} - O_{\mathrm{hRoll}}) + f_{\mathrm{hRoll}}^{\mathrm{balancing}}, \quad (10)$$
$$\dot{z}_{\mathrm{aRoll}} = -\alpha(z_{\mathrm{aRoll}} - O_{\mathrm{aRoll}}) + f_{\mathrm{aRoll}}^{\mathrm{balancing}}, \quad (11)$$
$$\dot{z}_{\mathrm{hYaw}} = -\alpha(z_{\mathrm{hYaw}} - O_{\mathrm{hYaw}}) + f_{\mathrm{hYaw}}^{\mathrm{rotation}} \quad (12)$$
$$\dot{z}_{\mathrm{hPitch}} = -\alpha(z_{\mathrm{hPitch}} - O_{\mathrm{hPitch}}) + f_{\mathrm{hPitch}}^{\mathrm{flex}} + f_{\mathrm{hPitch}}^{\mathrm{compass}} \quad (13)$$
$$\dot{z}_{\mathrm{kPitch}} = -\alpha(z_{\mathrm{kPitch}} - O_{\mathrm{kPitch}}) + f_{\mathrm{kPitch}}^{\mathrm{flex}} + f_{\mathrm{kPitch}}^{\mathrm{yield}} \quad (14)$$
$$\dot{z}_{\mathrm{aPitch}} = -\alpha(z_{\mathrm{aPitch}} - O_{\mathrm{aPitch}}) + f_{\mathrm{aPitch}}^{\mathrm{flex}} + f_{\mathrm{aPitch}}^{\mathrm{compass}} \quad (15)$$

Further details about this approach can be found in [17].

A correct tuning of parameters, p, is necessary for achieving bipedal walking in the robot. The upper an lower bounds of each parameter are shown in Table I.

IV. Experimental setup

A. Implementation on the Darwin-OP

The Darwin-OP is a lightweight humanoid robot with 20 Degrees of Freedom (DOFs) equipped with eight Force-sensing Resistors on the soles of the feet. In this work trajectories were generated for 6 DOFs per leg.

Experiments were carried out on the Darwin-OP robot and on the Webots physics simulator. The robot is expected to move in flat floor straightforward during 20 s. At each sensorial cycle (16 ms), sensory information is acquired. The system is integrated considering the Euler method with 1 ms

TABLE I

PARAMETER BOUNDS

Parameter	$A_{\text{balancing}}$	A_{hip}	A_{knee}	σ	A_{rotation}	A_{yield}	A_{compass}	ω_{sw}(rad/s)	O_{hRoll}	O_{hPitch}	O_{kPitch}	O_{aRoll}	O_{aPitch}
Lower bound ($\mathbf{l}$)	0.01	0.01	0.01	$\pi/4$	0.01	0.01	0.01	1.57	-50	-19	0	-50	-19
Upper bound ($\mathbf{u}$)	40	80	110	$\pi/4$	100	8	80	6.28	40	19	100	40	19

fixed integration step. At the end of each solution evaluation, the robot is set to its initial position and rotation, so that the initial conditions are equal for the evaluation of all solutions of all populations in real or simulated environment.

To assure that at least one solution has a good performance in the real robot, we introduce a hand-tuned solution in the n_s solutions which we a priori know that is able to walk both in simulation and in the real robot. This hand tuned solution was determined after some time in order to produce a solution able to achieve the best displacement both in simulation and in the real robot, while still producing good results from a functional perspective. That is, a locomotion that visually looked stable and smooth without producing large oscillations in pitch or in roll. This hand-tuning required the know-how of someone used to deal both with CPGs biped locomotion.

For the reality gap process we consider, $n_p = 100$, $n_e = 30$, $k = 5$, $n_g = 12$. The SVM uses the polynomial kernel with a parameter degree of 2. To train the SVM with the first data, we test $n_s = 20$ solutions in the real robot. All of them are chosen from the initial population of the NSGA-II, in such way that the most different are selected.

The ARToolkit [11] is used to calculate the displacement of the Darwin robot. A camera is mounted on the ceiling faced downwards. We rely on 1 marker in top of the robot head. We are only interested in frontal and lateral robot displacement. In case the marker position is not acquired or the returned confidence factor is small, the system assumes it did not change.

B. Treatments

The following treatments were considered in the simulator and in the real robot, by considering the following objectives:

- **Exp1** - max(f_D), min(f_F) and min(f_T)
- **Exp2** - max(f_D), min(f_F), max(f_{GC}) and min(f_T).
- **Exp3** - max(f_D), max(f_{GC}) and min(f_T). The Force was set as a constraint.

Five experiments have been run for each treatment. Furthermore, each solution has been tested 5 times both in simulation and in the real robot.

A control treatment, **SimOnly**, was considered in which no experiment was done on the real robot. It was inspired from [25] and considered the following objectives: max(f_D), min(f_F) and max(f_{GC}). 10 experiments were performed and afterwards the transferability (f_T) for the 10 non-dominated sets of solutions were calculated. Among these, the 10 solutions with best transferability were selected and run on the real robot.

V. RESULTS

A. Walking performance achieved

Figure V-A depicts the results of the simulated displacement for the last non-dominated population of each experiment(f_{Ds}) (blue circles) and the real displacement (f_{Dr}) of all solutions that were able to walk (red crosses), for each of the treatments. Considering the walking behavior in simulation, the **SimOnly** and the **Exp 3** were able to obtain the solutions with best displacement. However, note that **SimOnly** was not able to generate a single solution that walks on the real robot. The **Exp 1** and **Exp 2** treatments obtain almost similar values of displacement in the real robot. It is important to emphasize that 1) controllers able to make the robot move were generated by the proposed approach and 2) without any manual tuning between simulation and reality.

Figure V-A(b) shows the percentage of solutions that were able to make the real robot walk without falling. The **Sim Only** experiment was unable to find solutions that walk in the real robot. **Exp 1** and **Exp 2** treatments obtain the highest number of solutions that walk. This number is lower for **Exp 3** experiment, though this was able to find solutions with better displacement (see Figure V-A(a)).

Figure 3 depicts, for the real robot tests, the values of CoM lateral and frontal displacement of the hand tuned H solution (red dashed line) and the S_i solutions.S_i are solutions that achieved the maximum displacements for **Exp**i

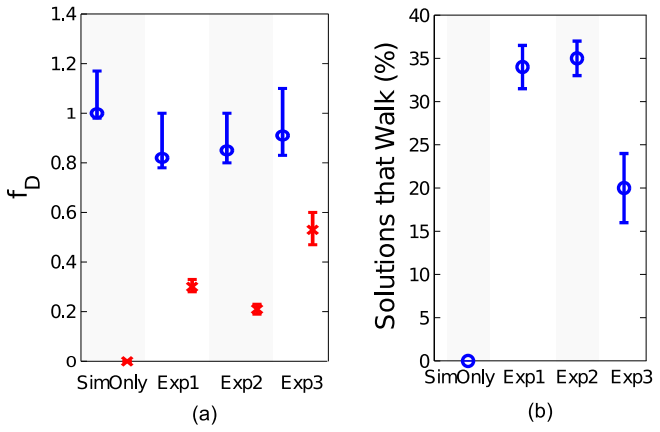

Fig. 2. Mean value (blue circle) and max and min (errorbar) of a) the displacement for the S values for the simulation. Mean value (red x) and max and min (errorbar) of the displacement values for S the real robot. b) k solutions percentage that were able to walk.

in simulation while being able to make the real robot walk. These solutions are considered as risky solutions, as the displacement was considered in priority on other functions rewarding the stability.

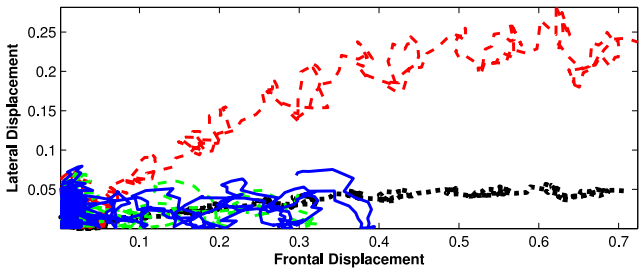

Fig. 3. Frontal and Lateral CoM displacements values of the H hand tuned solution (red dashed line), and the best S_1 solution (solid blue line), S_2 solution (dashed green line) and S_3 solution (dotted black line).

H (red dashed line) has a better frontal displacement than S_1 and S_2 solutions but similar to the S_3 solution. However, the hand tuned solution has a much higher lateral displacement compared to any of these solutions. For instance, the S_3 solution has a very low lateral displacement and a very high frontal one. These values are displayed on Table II. This shows up the obtained improvement in the performance criteria.

B. Functional Gait Analysis

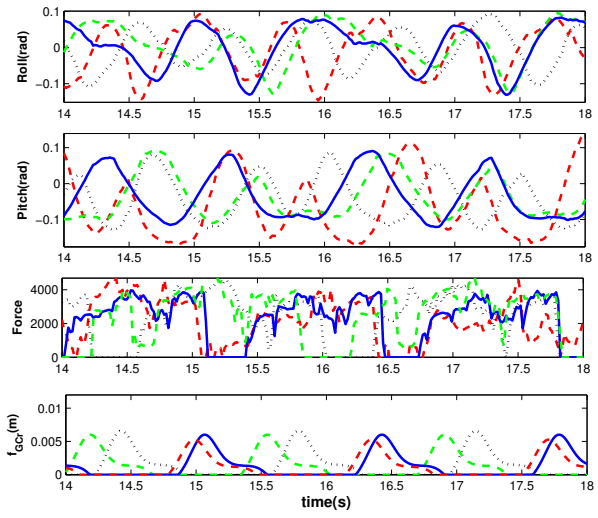

Fig. 4. Roll, Pitch and Force values of the H hand tuned solution (red dashed line), and the best S_1 solution (solid blue line), S_2 solution (dashed green line) and S_3 solution (dotted black line).

In the following, the impact of the evolutionary approach on the roll, pitch and force measures made on the real robot is studied. These are assessed through the built-in gyroscope and accelerometer sensors of the robots. In the top panel of figure 4, the mean roll of the solutions is similar (H : -0.009, S_1 : -0.0034, S_2 : 0.0016, S_3 : 0.005) but the roll of the hand tuned solution presents a higher oscillation (H : 0.25, S_1 : 0.22, S_2 : 0.24, S_3 : 0.21). The third panel depicts the values of the 4 left foot sensors of both solutions during the locomotion. In order to simplify the visualization we sum the value of the 4 left foot sensors. We can see that the S_2 solution has a tendency to have force during longer periods of time in the robot left foot, and stance periods increase. This means that the robot rises less times its foot (similar behavior for right foot), but rather drags them. On the other hand, solution S_1 and S_3 are solutions with well identified stance and swing phases.

In the second panel of Figure 4, the hand tuned solution shows a lower mean pitch (H : -0.068, S_1 : -0.03, S_2 : -0.03, S_3 : -0.039), meaning that the robot slightly tilted forward comparatively to the S solutions, which are more vertical. Furthermore, the pitch oscillation of the S solutions is lower than the one of the hand-tuned solution (H : 0.316, S_1 : 0.22, S_2 : 0.21, S_3 : 0.26). Regarding the pitch, one can see that the S solutions were able to perform a smoother locomotion in despite these are risky solutions which aim was to improve the displacement.

C. Sensitivity Analysis

In this section we are interested in verifying if there is any relationship between the CPG parameters and the obtained reality gap. In order to verify which CPG parameters most influence the gap, we verify the correlation between the gap (g) and the parameters of the solutions that were tested in the real robot and so belong to the database.

The correlation between the parameters and reality gap ($\mathbf{g}$) is assessed by computing the Pearson correlation coefficient (R) as a summary statistic. Table III presents the R coefficients between the parameters and the $\mathbf{g}$, as well as the corresponding p-values (p) for the statistical significance of the association. The statistical significant associations ($p < 0.05$) and the strongest correlations ($R > 0.5$) are highlighted. The highest correlation is obtained for O_{kPitch} ($R = -0.52$). This means that by increasing the O_{kPitch} value (flexing the knee joint) we decrease the value of the $\mathbf{g}$, therefore reducing the reality GAP. Thus, this reality GAP problem seems to be correlated to the posture.

VI. Discussion and Conclusion

The transferability approach allowed a multi-objective optimization algorithm to optimize CPG parameters in simulation while ensuring that a least some of the evolved solutions do indeed work on the real robot.

Despite the use of the transferability objective, significant differences can still be observed between the behavior in simulation and on the real robot. This suggests that the webots simulator used in these experiments could probably be improved to generate a behavior that is closer to that of the real robot. The transferability model, by highlighting what behaviors do transfer and what behaviors don't, could actually be used by the simulation designers to find out which are the less precisely modeled parts of the simulation and then update them.

TABLE II

H AND S_i SOLUTIONS WITH THE BEST DISPLACEMENT FOR EACH EXPERIMENT i.

Experiment	Pitch		Roll		COM		Objective functions			
	mean ± std	amplitude	mean ± std	amplitude	Frontal	Lateral	f_{D}	f_{GC}	f_{F}	f_{T}
S_1(run3)	-0.03±0.07	0.22	0.003±0.06	0.22	0.29	0.07	**0.33**	**0.006**	**0.55**	**0.82**
S_2(run2)	-0.03±0.06	0.21	-0.002±0.05	0.24	0.32	0.049	**0.23**	**0.006**	**0.56**	**0.85**
S_3(run3)	-0.039±0.07	0.26	-0.005±0.05	0.21	0.71	0.048	**0.6**	**0.006**	**0.6**	**0.97**
H	-0.068 ± 0.08	0.316	-0.009 ± 0.069	0.25	0.65	0.29	0.16	0.006	0.74	0.8005

TABLE III

PEARSON CORRELATION COEFFICIENTS (R) AND THE CORRESPONDING P-VALUES (p) FOR THE STATISTICAL SIGNIFICANCE OF THE ASSOCIATION BETWEEN THE PARAMETERS AND THE GAP

	$A_{\mathrm{balancing}}$	$A_{\mathrm{flexhPitch}}$	$A_{\mathrm{flexkPitch}}$	σ	A_{rotation}	A_{yield}	A_{compass}	ω_{sw}(rad/s)	O_{hRoll}	O_{hPitch}	O_{kPitch}	O_{aRoll}	O_{aPitch}
R	0.057	-0.41	-0.24	-0.029	0.027	0.33	-0.05	-0.19	0.069	0.23	**-0.52**	0.42	-0.41
P	0.74	0.011	0.15	0.866	0.873	0.048	0.750	0.257	0.687	0.167	**0.001**	0.01	0.012

Another perspective of this work would be to better exploit the non dominated set of solutions generated at the end of an experiment in order to choose, depending on the context, the fastest controller, the more stable one or a trade-off between the two. Generating all these solutions in a single optimization run is actually another advantage of multi-objective evolutionary algorithms over a careful hand tuning resulting in a single solution.

REFERENCES

[1] S. Aoi, N. Ogihara, T. Funato, Y. Sugimoto, and K. Tsuchiya. Evaluating functional roles of phase resetting in generation of adaptive human bipedal walking with a physiologically based model of the spinal pattern generator. *Biol. Cybernetics*, 102:373–387, 2010.

[2] S. Aoi and K. Tsuchiya. Adaptive behavior in turning of an oscillator-driven biped robot. *Auton. Robots*, 23(1):37–57, July 2007.

[3] S. Aoi and K. Tsuchiya. Generation of bipedal walking through interactions among the robot dynamics, the oscillator dynamics, and the environment: Stability characteristics of a five-link planar biped robot. *Aut. Robots*, 30(2):123–141, 2010.

[4] J. Bongard, V. Zykov, and H. Lipson. Resilient Machines Through Continuous Self-Modeling. *Science*, 314(5802):1118–1121, 2006.

[5] S.-J. Chung and M. Dorothy. Neurobiologically Inspired Control of Engineered Flapping Flight. *ArXiv e-prints*, May 2009.

[6] E. Drumwright, J. Hsu, N. Koenig, and D. Shell. Extending open dynamics engine for robotics simulation. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 38–50. Springer LNCS 6472, 2010.

[7] I. Ha, Y. Tamura, and H. Asama. Gait pattern generation and stabilization for humanoid robot based on coupled oscillators. In *IEEE IROS*, pages 3207–3212, 2011.

[8] A. J. Ijspeert. 2008 special issue: Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–653, 2008.

[9] N. Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive behavior*, 1997.

[10] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1):3–12, 2005.

[11] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of IWAR*, 1999.

[12] Y. Kim, Y. Tagawa, G. Obinata, and K. Hase. Robust control of cpg-based 3d neuromusculoskeletal walking model. *Biological cybernetics*, pages 1–14, 2011.

[13] S. Koos, A. Cully, and J.-B. Mouret. Fast damage recovery in robotics with the t-resilience algorithm. *arXiv preprint arXiv:1302.0386*, 2013.

[14] S. Koos, J.-B. Mouret, and S. Doncieux. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 17(1):122–145, Feb 2013.

[15] J.-Y. Lee, M.-S. Kim, and J.-J. Lee. Multi-objective walking trajectories generation for a biped robot. In *IEEE IROS*, 2004.

[16] V. Matos and C. P. Santos. Omnidirectional locomotion in a quadruped robot: A cpg-based approach. In *IEEE IROS*, pages 3392 –3397, 2010.

[17] V. Matos and C. P. Santos. Central pattern generators with phase regulation for the control of humanoid locomotion. Business Innovation Center Osaka, Japan, 2012.

[18] C. Maufroy, H. Kimura, and K. Takase. Integration of posture and rhythmic motion controls in quadrupedal dynamic walking using phase modulations based on leg loading/unloading. *Auton. Robots*, 28:331–353, April 2010.

[19] S. Miyakoshi, G. Taga, Y. Kuniyoshi, and A. Nagakubo. Three dimensional bipedal stepping motion using neural oscillators-towards humanoid motion in the real world. In *IEEE IROS*, 1998.

[20] J. Morimoto, G. Endo, J. Nakanishi, and G. Cheng. A biologically inspired biped locomotion strategy for humanoid robots: Modulation of sinusoidal patterns by a coupled oscillator model. *IEEE Transactions on Robotics*, 24(1):185–191, 2008.

[21] J. Morimoto, G. Endo, J. Nakanishi, S. Hyon, G. Cheng, D. Bentivegna, and C. Atkeson. Modulation of simple sinusoidal patterns by a coupled oscillator model for biped walking. In *IEEE ICRA*, 2006.

[22] J.-B. Mouret, S. Koos, and S. Doncieux. Crossing the reality gap: a short introduction to the transferability approach. In *Proceedings of the workshop "Evolution in Physical Systems". ALIFE'2012.*, 2012.

[23] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. Learning from demonstration and adaptation of biped locomotion. *Rob. and Aut. Systems*, 47(2-3):79 – 91, 2004.

[24] K. Nishiwaki and S. Kagami. Simultaneous planning of com and zmp based on the preview control method for online walking control. In *11th IEEE-RAS International Conference on Humanoid Robots*, 2011.

[25] M. Oliveira, V. Matos, C. P. Santos, and L. Costa. Multi-objective parameter cpg optimization for gait generation of a biped robot. In *IEEE ICRA*, 2013.

[26] L. Righetti and A. J. Ijspeert. Programmable central pattern generators: an application to biped locomotion control. In *IEEE ICRA*, 2006.

[27] L. Righetti and A. J. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *IEEE ICRA*, 2008.

[28] K. Seo, S.-J. Chung, and J.-J. Slotine. Cpg-based control of a turtle-like underwater vehicle. *Autonomous Robots*, 28:247–269, 2010.

[29] M. Vukobratović and B. Borovac. Zero-moment point—thirty five years of its life. *International Journal of Humanoid Robotics*, 1(01):157–173, 2004.

[30] J. Zagal and J. Ruiz-Del-Solar. Combining simulation and reality in evolutionary robotics. *Journal of Intelligent and Robotic Systems*, 50(1):19–39, 2007.