COCoMoPL: A Novel Approach for Humanoid Walking Generation Combining Optimal Control, Movement Primitives and Learning and its Transfer to the Real Robot HRP-2

Debora Clever, Monika Harant, Katja Mombaur, Maximilien Naveau, Olivier Stasse, Dominik Endres

COCoMoPL: A Novel Approach for Humanoid Walking Generation Combining Optimal Control, Movement Primitives and Learning and its transfer to the real robot HRP-2

Debora Clever[1], Monika Harant[1], Katja Mombaur[1], Maximilien Naveau[2], Olivier Stasse[2], Dominik Endres[3]

***Abstract*—COCoMoPL [6] is a recently developed approach Combining Optimal Control, Movement Primitives and Learning for the generation of humanoid walking motions. It solves optimal control problems based on detailed dynamic models of the robot for a variety of walking parameters and uses the solutions as training data to create movement primitives that are very close to feasibility and optimality. These can be employed to synthesize complex walking sequences for humanoid robots online in a very efficient way. We demonstrate, for the first time, that COCoMoPL works on a real humanoid robot, here HRP-2 with 36 DOF and 30 position controlled actuators. To this end, it was necessary to significantly extend the existing approach by including transition steps into the training data, modify the movement primitives (MP) to admit these transitions, improve the representation of the ZMP MPs and tighten the transition conditions at the beginning and end of steps. We present a thorough validation of the method in simulation and on the real robot for a challenging sequence of movements. We also compare the characteristics of movements after each step of the methodology.**

***Index Terms*—Optimization and Optimal Control, Learning and Adaptive Systems, Humanoid and Bipedal Locomotion**

I. Introduction

In the development of humanoid robots one major challenge is to make these bipedal machines walk in a robust, versatile and efficient way. Nowadays, many state of the art walking motion generation methods are based on a simplified dynamics, e.g. the table cart model [14] and quite conservative stability criteria, e.g. the requirement that the simplified model's zero moment point (ZMP) must be located in a small area around the ankle joint [14].

If computing time is not an issue, then there is the possibility to model the full dynamics of the robot. This allows one to simulate its motions and to compute dynamical properties, for example the ZMP of the full model. Full body motion generation facilitates the synchronization of upper and lower body movement and usually results in a more dynamical gait [24]. Using optimal control (OC) techniques, it is even possible to compute a full body motion, which is optimal with respect to some predefined objective function, e.g. [16]. It is possible to decrease the computational time by using a flavor of DDP called iLQR, which is dropping terms that have a low impact on the cost function [26]. Further improvement in speed is achieved in [26] by using a tailored contact model. The drawback of this contact model is the difficulty in generating realistic motions for walking [19]. Therefore, to overcome the reality gap and to ensure that such motions are feasible for the real robot, a very detailed model is needed [17]. Consequently, the computational effort is much too high for real-time execution.

If optimality is a secondary concern, but small computational effort at production time is important, it is possible to simplify the problem formulation [5], [7], [12] by considering the centroidal dynamics on a small window and the whole body for instantaneous control. The problems can be disconnected [5] or coupled [7], [12]. These methods however assume heuristics to generate the end-effector trajectories, while the methods considering the whole body motion can find automatically complex feet and upper body motions.

Another interesting approach is movement primitives (MP) [8], [11]. While MPs allow for real-time computation, it is hard to guarantee optimality and feasibility of the generated motions, because the typical cost functions employed in MP learning do not measure these quantities explicitly.

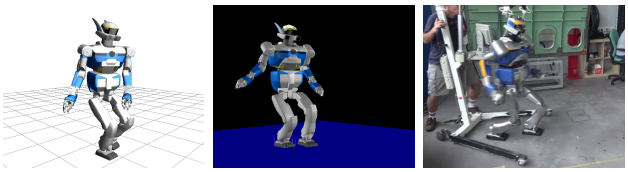

Fig. 1. **The humanoid robot HRP-2.** From left to right: optimal control model, OpenHRP model, real robot. Note that, even though the model in the robot simulator OpenHRP takes into account the full dynamics of the robot, it cannot be used for optimization purposes.

In this paper, we propose a new approach to generate full body walking motions for a bipedal robot by combining OC and MPs. Since we also employ machine learning for the extraction of MPs from OC solutions, we call our approach **C**ombination of **O**ptimal **Co**ntrol, **Mo**vement **P**rimitives and

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Re viewers' comments. *The research leading to these results has received funding from the European Union Seventh Framework Program (FP7/2007 - 2013) under grant agreement no 611909 (KoroiBot).

[1] Interdisciplinary Center for Scientific Computing (IWR), Optimization in Robotics & Biomechanics (ORB), University of Heidelberg, debora.clever@iwr.uni-heidelberg.de.

[2] CNRS - LAAS, Toulouse, France, ostasse@laas.fr.

[3] Theoretical Neuroscience Group, Dept. Psychology, Philipps-University Marburg, dominik.endres@uni-marburg.de.

Digital Object Identifier (DOI): see top of this page.

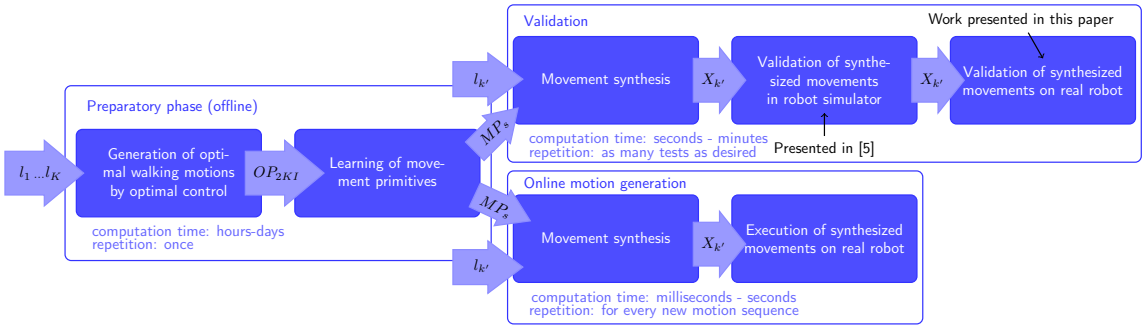

Fig. 2. **Methodology.** Optimal and dynamically feasible motion trajectories (OP_{2KI}) are computed by an optimal control approach from a given set of parameters $(l_1, ..., l_K)$. Movement primitives (MP_s) are learned in a Gaussian process framework. New motions ($X_{k'}$), parametrized by $l_{k'}$, are generated by using a small number of primitives. The resulting motions prove to be sufficiently close to optimality and dynamical feasibility, validated in the virtual robot simulator OpenHRP and run on the real robot (upper branch). After this general validation, the generated trajectories are executable online on the robot without the additional robot simulator validation (lower branch).

Learning, short COCoMoPL. Our approach inherits the key advantages of both: local-optimality of the generated motions and low computational effort for the production of novel movements. Specifically, in comparison to existing approaches that combine optimal control and movement primitives, e.g. [9], [21], our approach has the following properties:

- Optimality: due to our sophisticated optimal control model, we are able to compute locally optimal full body motions with respect to a variety of different objective functions, e.g. squared motor torques.
- Consistency: due to our detailed dynamical robot model, computed full body motions are reliably feasible in the robot simulator, which is a predictor for feasibility on the real robot, too.
- Modularity: due to our Gaussian process based primitive model, only a small set of training data is necessary to generate a large variety of new motions.
- Augmentability: our approach allows to successively include new motions into the training data, e.g. optimal transition steps that were not used in our previous work.

While the general approach has already been introduced in our previous work [6], [17], we present the following significant extensions in this paper:

- It contains the first validation of the approach on the real robot on the basis of a very challenging walking motion. To overcome the problem of large ground reaction forces and torques, a number of improvements of the method had to be implemented:
  - the inclusion of transition steps between steps of different stride lengths into the training data for MP learning,
  - the refinement of the learned ZMP trajectories by using a higher number of MPs,
  - the enforcement of a better match of the states between two subsequent differing steps.
- We show that the performance of the generated motions reaches - or even surpasses - the performance of the OC motions on the real robot.
- Finally, due to damping effects, we show that the MP motions close the reality gap even better than the OC solutions.

The paper is organized as follows: in section II, we briefly summarize the general task addressed in this paper and give some background information about the robot HRP-2 and the robot simulator OpenHRP; in section III we summarize the theoretical background of our combined optimal control, movement primitives and machine learning approach and explain how the approach is tailored to the robot; and in section IV we discuss challenges and solutions for the transfer of computational results to the real robot and the reality gap. In section V, we analyze the generated motions with respect to optimality and quality. We compare optimal control results with learned results and simulation results with their counterparts on the real robot, where the focus of the comparisons is on squared joint torques and the contact forces during touchdown and stance. We conclude our work and give future perspectives in section VI.

II. Task and Robot Description

A. Task

The challenging task discussed in this paper is the online generation of walking motions for humanoid robots with variable walking steps. This should enable robots to re-plan their walking motion on the spot, e.g. in response to perceived properties of the environment, which impose new constraints on feasible steps.

We are interested in the planning of longer walking sequences that may be quite demanding for the robot. This necessarily involves the planning of individual steps as a subproblem. The method outlined in the following sections is generally applicable to on-line movement planning. Here we focus on the example of changing step lengths, because it is a very important property to sustain in rough terrains. Other step parameters could be handled the same way. The method could also be applied to any other humanoid robot, for which a dynamic model is available. However, several choices in the method depend on the specific kinematic and dynamic structure of the robot as well as its control system, so we will briefly describe the robot first.

B. Robot

We used the humanoid robot HRP-2 at LAAS-CNRS built by Kawada Robotics [15]. It has 36 degrees of freedom, and 30 motors, one for each internal degree of freedom: $(legs, 2\times6), (chest, 2), (head, 2), (arms, 2\times7)$. To minimize impact on the ground, the robot has three shock absorbers in the ankles. The remaining 6 degrees of freedom describing the floating base are unactuated. The sensors used for balancing are an Inertial-Measurement-Unit (IMU) in the chest and two 6-axis force sensors located in each ankle of the legs. Less powerful than ATLAS, or S-One, this robot has nonetheless an industrial quality which affords good repeatability. The manufacturer provides a rigid body model for the robot, which is built upon the robot CAD model. The robot is position controlled with high gain local controllers at each joint. In addition, there is a global control system, the so-called stabilizer, which supervises the overall stability of walking motion and may interfere with any motion to improve stability. A generated motion should therefore be feasible with respect to the requirements of the stabilizer, because otherwise the motion may be significantly altered.

The robot HRP-2 also comes with the OpenHRP simulator, which uses a rigid multibody dynamics model. Rigid contacts are computed using a linear complementary problem formulation. The actuators are simulated taking into account rotor inertia and gear ratio. Springs are considered as torque generators and simulated using a forward dynamics computation. This provides a relatively faithful simulation of the humanoid robot HRP-2 [22] which provides an important step in the validation of motions. However, it is not suitable to use this robot simulator for complex optimization tasks, see Section III-B.

III. Framework

A. Overview

To quickly generate walking motions that are highly dynamical, exploit the operational range of the robot and are close to optimality with respect to a desired optimization criterion we propose an approach that combines optimal control, movement primitives and learning (COCoMoPL). We validate this approach for challenging walking sequences in simulation as well as on the real robot.

The general workflow of our approach is presented in Fig.2: in a first step, described in section III-B, training data for the MP learning is generated by solving optimal control problems that take into account a detailed model of the robot dynamics and its constraints. The resulting non-convex optimal control problems (OCPs) are solved using a direct multiple shooting approach combined with sequential quadratic programming (SQP). Even though this only ensures local optimality, in the following we refer to the solution of such an OCP simply as 'optimal solution'. Optimal solutions are computed for individual walking steps of different step length, of starting and stopping steps, as well as for acceleration and deceleration steps. Each of these solutions contains the histories of the robot's positions, velocities, forces and parameters (e.g. stride length or phase time), which are all determined simultaneously. While all these solutions form the basis of the training data, the nature of the robot's control system determines which solutions are used for MP learning (see section III-B).

In a second step, outlined in section III-C, MPs are learned with a Gaussian process approach from the trajectories in the training data. The MP model's complexity and type, here number of MPs and Gaussian process kernel type, are determined automatically by approximate Bayesian model comparison. We performed this comparison for HRP-2, it might yield different results for another robot. It is important to note that the number of MPs is much lower than the number of training trajectories. We found that with a dataset containing $\approx 10^3$ training trajectories, the optimal number of MPs is in the order of 10^1. The generated movements are controlled by morphing the weights with which the MPs are superimposed. The weights are determined by functions that take the step type and stride lengths as inputs. These functions are drawn from a Gaussian processes.

These two steps form the preparatory phase of this method. They take a significant amount of time - depending on the robot and the amount of training data generated - between many hours and up to several days. However, this effort has to be expended only once.

In our workflow, in Fig.2 we now distinguish the paths for the validation of the method (top/Validation), which we pursue in this paper and for online motion generation (bottom/Online motion generation), which describes the intended deployment of this approach outside the lab. In both cases, the third step addresses the synthesis of new walking motions by first choosing desired step parameters, then drawing the weights corresponding to these parameters, and finally superimposing the MPs with these weights. Matching the beginning of the trajectory of a step to the endpoint of the previous one is done via approximate Bayesian conditioning, see section III-C and [6] for details. In the validation case, this is executed for a set of test motions, and at deployment time it will be performed for any new motion the robot is supposed to perform.

For validation purposes, the generated motions are first tested within the robot simulator OpenHRP before being transferred to the real robot, as described in section V. The result of all these tests and their comparison with the original optimized solutions is described in section V. The computational effort for validation of a single movement ranges from several seconds to several minutes.

During deployment, the newly generated movement sequences would be transferred to the robot immediately without preliminary testing in the simulator, due to real time constraints. This step currently takes a few seconds. We are in the process of optimizing it to make it truly real-time feasible.

B. Optimal Control based Training Data Generation

The general workflow to generate the training data is depicted in Fig.3. Due to the fact that the humanoid robot HRP-2 is position controlled and takes as input joint angles, hip orientation and ZMP trajectories, we include these quantities in the training data. To be able to compute optimal and

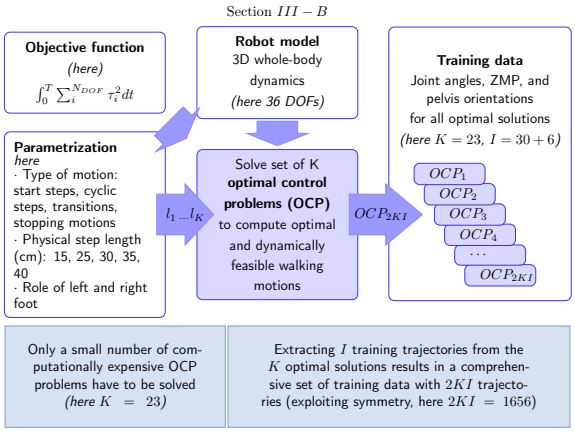

Fig. 3. **Optimal control based training data generation.** Optimal and dynamically feasible training data are computed by an optimal control approach. We set up four different kinds of OCPs: starting steps, cyclic steps, transitions steps, and stopping motions. Each OCP is solved for each training step length (150mm, 250mm, 300mm, 350mm, 400mm).

dynamically feasible full body motions, a detailed dynamics model of the robot has to be considered within the optimal control problems. Even though the model in the robot simulator OpenHRP takes into account the full dynamics of the robot, it is not usable for optimization purposes. To be able to exploit efficient mathematical optimization algorithms and ensure a suitable handling of constraints, we define an additional dynamic model in form of a hybrid system of differential algebraic equations with all relevant constraints. Such an OCP consists of different phases taking into account the change of contacts, impacts at touchdown and the resulting changes in the dynamics.

Here, the humanoid robot HRP-2 is modeled as a rigid multibody model with 36 degrees of freedom. Each step consists of two phases with varying contact sets (double and single support) and one discontinuity at the end (touchdown). Phase transitions are defined in terms of events (e.g. with respect to contact forces) to allow for a free phase timing of the motion. The resulting equation of motion is a system of hybrid differential algebraic equations (DAE) of index three. For computational efficiency, we reformulate it as a system of index one. The dynamic equations are composed analytically and converted into C-code by the dynamic model builder DYNAMOD based on 6D spatial geometry [10] and symbolic code generation following [27].

To be able to generate a wide range of walking motions, we define four different kinds of optimal control problems:

- The first type of OCP generates one optimal step of a cyclic motion with constant and predefined step length. Note, that the configuration (i.e. position, velocity and force/torque) at the beginning and the end of a step are just mirrored to each other and that the distance between the robot's feet in this configuration coincides with the physical step length.
- The second type of OCP generates optimal transition steps. They start at the final configuration of a cyclic step with a given step length, perform one transition step and end in a configuration that coincides with the initial configuration of a cyclic step of different step length.
- The third type of OCP generates optimal starting motions of a predefined step length. They begin at a static posture with both feet next to each other, perform one step and end at a configuration that coincides (i.e. same position, velocity and force/torque) with the initial configuration of a cyclic or a transition step.
- The fourth and last type of OCPs generates torque optimal stopping motions, that start at the final configuration of the cyclic or transition steps with a predefined step length, perform two steps, and finally stop at a static posture with both feet next to each other.

For the sake of smoother motor torque profiles τ, the control function u is defined by the time derivative of the motor torques in the actuated joints ($u := \dot{\tau}$). The state function x maps time onto pelvis translation, pelvis orientation, joint angles, all corresponding velocities and the actuated torques ($x := (q, v, \tau)$). The parameter set p describes model- and gait-specific quantities, where some of them are free to be determined by optimization. Finally, the objective J is defined as

$$J(x,u,p) = J(q,v,\tau,\dot{\tau},p) = \int_0^{t_f} \sum_{i=0}^{N_{\text{DOF}}} \tau_i^2 \, \mathrm{d}t + J_{\text{pen}}, \qquad (1)$$

to resolve the redundancy via torque minimization in the powered joints. The penalty term J_{pen} is a Lagrangian relaxation of inherent constraints imposed by the high level control system of the robot. Note, that for an optimal solution, where the constraints are fulfilled, this term becomes negligibly small.

Summing up, this results in an optimal control problem with 110 state and 30 control functions, which we solve with a direct approach that is based on a control discretization with local support functions, a state parameterization by multiple shooting and a structure exploiting SQP method [3], implemented in the software package MUSCOD-II [20]. For more details on the dynamic optimal control model of HRP-2, including the constraints in the penalty term, we refer to [16], [18].

Besides the parameterization by step type we also parameterize the walking motions by their physical step length. Exploiting the symmetry of the robot, we solve the involved OCPs only for steps where the first support is on the right leg and mirror the computed solutions for the corresponding OCPs with the roles of legs being exchanged. We compute optimal motions for five different step lengths (150mm, 250mm, 300mm, 350mm, 400mm) for starting, stopping and cyclic motions, increasing transitions 150mm $\rightarrow$ 250mm, 250mm $\rightarrow$ 300mm, 300mm $\rightarrow$ 350mm, 350mm $\rightarrow$ 400mm and decreasing transitions 400mm $\rightarrow$ 350mm, 350mm $\rightarrow$ 300mm, 300mm $\rightarrow$ 250mm, 250mm $\rightarrow$ 150mm. Considering right and left steps, the generated training data consists of two times 23 torque optimal and dynamically feasible motions, each defined by 30 joint angle trajectories, 3 ZMP trajectories and 3 trajectories describing the pelvis orientation. This sums up to a total number of $2 \cdot 23 \cdot (30 + 3 + 3) = 1656$ training trajectories.

C. Learning MPs and Generating New Movements

The general workflow for the MP part of our work is shown in Fig.4: we machine-learn MP models from the OCP-generated training data, select the best of these with approximate Bayesian model comparison, and validate the result by generating *novel* movements that are not part of the training data.

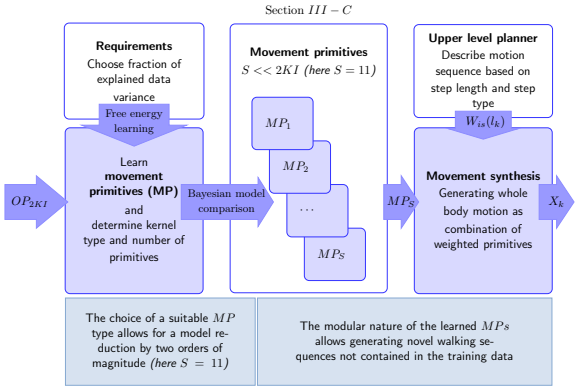

Fig. 4. **Learning movement primitives (MP) and motion synthesis.** We use variational free energy learning to extract MPs from the optimal control generated training data. MPs are drawn from Gaussian processes. The number of MPs and the kernel type are determined via approximate Bayesian model comparison.

As detailed in [6], we employ a temporal MP model, i.e. the MPs are stereotypical time courses, which are linearly superimposed to approximate the training data. A graphical

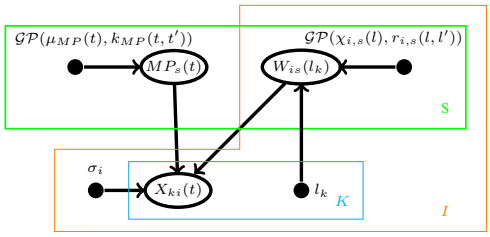

Fig. 5. The morphable movement primitive model, figure adapted from [6]. We generate time series $X_{ki}(t)$ of I signals (joint angles etc.) by multiplying S many movement primitives $MP_s(t)$ with weights W_{is}. There is one weight per signal and MP. Weights are controlled by step type/stride length parameters l_k, of which there is one instance per trial K. $\mathcal{GP}(.,.)$ indicates a Gaussian process.

model in plate notation is shown in Fig.5. We index the training movements with k, degrees of freedom (DF, joint angles, pelvis rotation and ZMP) with i and movement primitives with s. Let l_k be the parameters of movement k, $X_{ki}(t)$ be the trajectory of DF i in movement k, $MP_s(t)$ the s-th movement primitive and $W_{is}(l_k)$ the weight with which primitive s contributes to DF i in movement k. Then

$$X_{ki}(t) = \sum_s W_{is}(l_k)\, MP_s(t) \; + \eta_{ki}(t) \tag{2}$$

where $\eta_{ki}(t) \sim \mathcal{N}(0, \sigma_i)$ is Gaussian noise. To promote smooth MPs, we draw them from Gaussian process priors $\mathcal{GP}(\mu_{MP_s}(t), k_{MP_s}(t,t'))$ with a RBF kernel

$$k_{MP_s}(t,t') = \alpha_s \exp\left(-\gamma_s (t-t')^2\right) \tag{3}$$

and mean functions $\mu_{MP_s}(t)$ estimated from a PCA decomposition of the training data. Gaussian processes are a staple in modern machine learning for regression and interpolation problems [25].

The weights $W_{is}(l_k)$ are functions of the step parameters l_k. In [6], we parameterized the weights by step type and step lengths, i.e. the $l_k = (\text{type}_k, \text{length}_k)$ were tuples. In contrast, we now want to model transition steps, too. Therefore, we augment the l_k with the lengths at the beginning and the end of a step,

$$l_k = (\text{type}_k, \text{start length}_k, \text{end length}_k). \tag{4}$$

We learn the functions from these triples onto the weights in a Gaussian process framework: weight functions are drawn from $\mathcal{GP}(\chi_{is}(l), r_{is}(l,l'))$ with kernels

$$r_{is}(l,l') = \delta_{\text{type,type}'} \cdot \alpha_{is} \exp\left(-\gamma_{is}|z|^2\right) \tag{5}$$

where vector $z = (\text{start length} - \text{start length}', \text{end length} - \text{end length}')$ and Kronecker delta $\delta_{\text{type,type}'} = 1$ if type $=$ type$'$ and zero otherwise.

Since the exact posterior distributions of the $W_{is}(l_k)$ and the $MP_s(t)$ are intractable, we resort to a variational free energy approximation [2] and maximize the usual lower bound $\mathcal{L}$ on the marginal log-likelihood of the trajectory data X

$$\begin{aligned} \mathcal{L} &= \langle \log\left(P(X|H)\right)\rangle_{Q(H)} - D\left(Q(H)||P(H)\right) \\ &\leq \log(P(X)). \end{aligned} \tag{6}$$

where H are latent variables and parameters, and $Q(H)$ is an approximation to the correct posterior. We chose $Q(H)$ to be conjugate to the prior, i.e. the posterior distributions of both MPs and weights are assumed to be multivariate Gaussian. For details of the derivation of this bound, see [6]. We maximized $\mathcal{L}$ with respect to the parameters of $Q(H)$ and the kernel parameters using the Broyden-Fletcher-Goldfarb-Shanno algorithm for constrained optimization [4] implemented in the `SciPy` package [23]. Automatic gradient computation was done with the Theano library [1]. The training data was comprised of regular steps (left foot swing) and starting/stopping steps of 150mm, 250mm, 300mm, 350mm, 400mm lengths, and their mirror images (right foot swing). Furthermore, the data included the following step length transitions: 150mm $\leftrightarrow$ 250mm, 250mm $\leftrightarrow$ 300mm, 300mm $\leftrightarrow$ 350mm, 350mm $\leftrightarrow$ 400mm, and their mirror images.

To generate a new step k' with parameters $l_{k'}$, we compute the posterior means of the $X_{k'i}(t)$, conditioned on $l_{k'}$. If we are not generating a starting step, we furthermore condition the initial state of the step to be nearly equal to the end state of the previous step. For steps that could be executed by the robot without falling, we had to set the target explained variance to 99.9% for the joint angle and ZMP trajectories. A smaller explained variance target results in infeasible movements, whereas a higher one unnecessarily increases computational effort. This target was approximately reached with 11 MPs, where we found a (broad) maximum of L earlier [6]. Note that the training data for the joint angles were comprised of 1656 trajectories in total, so 11 MPs is a significant compression which the MP learning achieved with nearly no loss in accuracy.

IV. Transfer to Robot

The reality gap between a motion run on the simulated robot in OpenHRP and run on the real robot is due to a certain number of assumptions which are violated. First, our OC method assumes that the rigid multibody dynamic model and the real robot are sufficiently coherent, such that the controller on the robot is able to compensate for the differences. Our experience confirms that the model is quite close to reality. Of course, identifying the robot parameters using state-of-the-art methods helps in diminishing the differences [13]. We use a model provided by the robot manufacturer without additional corrections. Another major assumption on HRP-2 lies in the low-level position control system. We assume that defects such as elasticities in the harmonic drives, the timing belt, and possibly in the 1 KHz PID can be neglected. This assumption is reasonable due to the use of high gain controllers. The robot has three shock absorbers in its ankles [22]. In simulation they are represented by one linear and two torsional springs. However, the simulated springs do not model the limits of the shock absorbers. Indeed, when the limits are reached the shock absorbers can not be compressed anymore. For very dynamic motions, such as the ones proposed in the paper, the shock absorbers' deformation limits need to be avoided on the real robot. In OCP, these issues are taken into account by generating consistent dynamical motions, which do not deform the rotational springs. Even though such constraints can be easily formulated in the OCPs, the challenge here is to avoid such deformations when generating a new motion with machine learning algorithms which do not explicitly constrain the solutions to be feasible for the robot.

Transitions are important because they describe the landing of the swing foot and the weight shift from one foot to another. If the controller is not able to act properly the swing foot may hit the ground with an impact that is too strong for the robot. This is especially true with highly dynamic motions. In order to detect this problem, we used the simulator OpenHRP to verify the generated motions. However, due to the discrepancy explained above, the simulator is overly optimistic when the center of pressure (CoP) strays from the center of the feet. In simulation, the controller is able to recover from situations which are not feasible in reality. This happens for instance when all the contact forces appear to lie only at the border of the foot. We struggled with this discrepancy when we tried to run a 7 MP generated movement whose ZMP trajectories were feasible in OpenHRP (Fig.6, blue lines) on HRP-2: feasibility was predicted but not obtained in reality. By increasing the number of MPs from seven to eleven, we were able to generate an HRP-2-feasible movement (Fig.6, red lines). This observation coincides with one important result of our previous work [6]: the generated motions learned from optimal control results have been feasible for the robot simulator and therefore had a high potential to be executable on the real robot as well. However, analyzing contact forces and mean (squared) torques shows a significant increase of both on step transitions. Whereas the high mean torques only mean suboptimal movements (c.f. eqn.1), the high contact forces can damage the robot. Having improved the quality

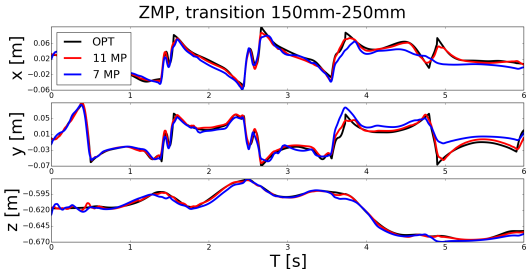

Fig. 6. ZMP trajectories for transition steps from 150mm to 250mm length. Black lines are OCP generated, red/blue lines are generated with 11 and 7 MPs, respectively. The movement with 11 MPs matches the OCP solution more closely, especially towards the end. All shown trajectories are parts of feasible movements in OpenHRP, but only the OCP and the 11 MP trajectories are feasible on HRP-2.

of these transitions significantly, the generated motions are now feasible for the real robot. Therefore, the next section is devoted to the analysis of these motions.

V. Motion Analysis

In the following section we discuss two important issues of transferring the generated motions to the real robot: feasibility and optimality. To this end, we consider a generated motion, which includes large changes in stride lengths, and several steps that were not part of the training data:

Motion M_{gen}: (150mm) $\rightarrow$ 150mm $\rightarrow$ 250mm $\rightarrow$ 350mm $\rightarrow$ 390mm $\rightarrow$ 350mm $\rightarrow$ 250mm $\rightarrow$ 150mm $\rightarrow$ (150mm),

where the values in brackets stand for the step length of the starting step and the stopping step, respectively. With the exception of the transitions 150mm $\leftrightarrow$ 250mm and the 150mm starting/stopping movement, the steps in this sequence are novel with respect to the training data. Fig.7 shows example frames of this sequence.

We compare this MP generated step sequence to the following OC generated sequence

Motion M_{opt}: (150mm) $\rightarrow$ 150mm $\rightarrow$ 250mm $\rightarrow$ 350mm $\rightarrow$ 400mm $\rightarrow$ 350mm $\rightarrow$ 250mm $\rightarrow$ 150mm $\rightarrow$ (150mm).

As noted above, most of this sequence is *not* a part of the training data and has been computed for comparison and analysis reasons afterwards. The one slight difference between a 390mm step in the generated motion and a 400mm step in the optimized motion has been chosen for the following reason: as the difference of one centimeter in physical step length (which is not explicitly set, but a result of all joint angles) lies in the range of variation, the two motions can be considered to be similar. But using 390mm instead of 400mm in the generated motion gives an additional validation of the learning results of intermediate physical steps sizes that have not been present in the training data.

From previous work [6] we have learned, that a sudden increase of contact forces up to a moderate value is compensable by the position controller in simulator but causes problems on the real robot. For the revised approach, which is now feasible for the robot, contact forces are always close to the guiding value of 600N, even on transitions between steps, see Fig.8. In comparison to the contact forces from a motion generated

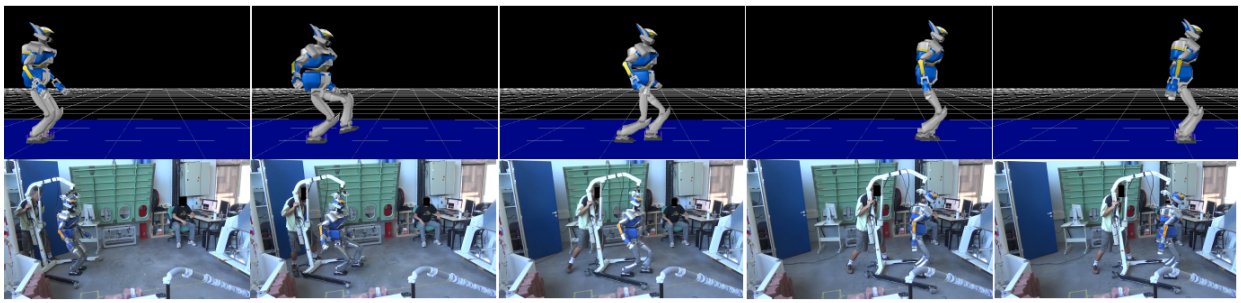

Fig. 7. **Highly dynamic motion with large step size changes.** Motion resulting form MP-learning (MP) executed in robot simulator OpenHRP (top) and on real robot HRP-2 (bottom).

with a common pattern generator (Kajita), the contact forces from the generated MP motion are even smoother.

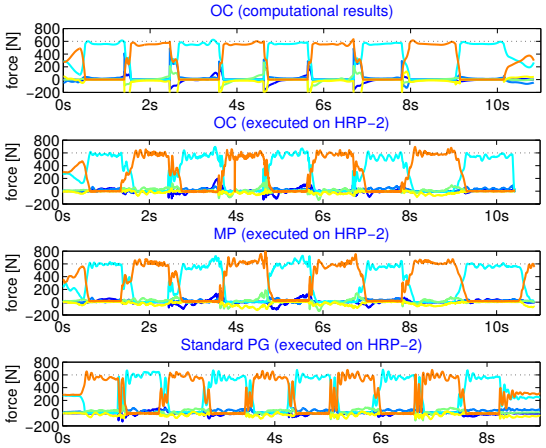

Fig. 8. Contact forces. From top to bottom: OC computational, OC executed on HRP-2, MP executed on HRP-2, pattern generator (Kajita).

Even though the most important issue is the feasibility of the generated motions on the real robot, optimality is also interesting. As our training data consists of motor torque optimal motions, minimizing (1), we are interested in the question of how well optimality carries over from optimized to generated motions. To this end, we evaluate the mean joint torques

$$\tilde{J}(q, v, \tau_{\text{act}}, \dot{\tau}_{\text{act}}) = \frac{1}{N_{\text{DOF}} \cdot t_f} \left(\int_0^{t_f} \sum_{i=0}^{N_{\text{DOF}}} \tau_{\text{act},i}^2 \right)^{\frac{1}{2}} \mathrm{d}t,$$

for optimal control results, simulation results and robot results and compare them to each other. In comparison to the computational motor and joint torques of the rigid body model in the OC context and the joint torques on HRP-2, the torques for the corresponding motions in OpenHRP are significantly damped. This behavior is shown exemplary for the knee joints in Fig.9. As a consequence, the integral over the squared torques is smaller for the executed motions in the robot simulator than for the rigid body model in the OC context and the real robot, see Table I. Even though (except for the OC motor torques) all evaluated mean torques $(\int \tau^2 dt)\frac{1}{2}/(N \cdot t_f)$ should be comparable, due to model mismatches we observe a range between 15.6Nms^{-1} and 19.8Nms^{-1}. However, it appears that the value of the mean torques with respect to

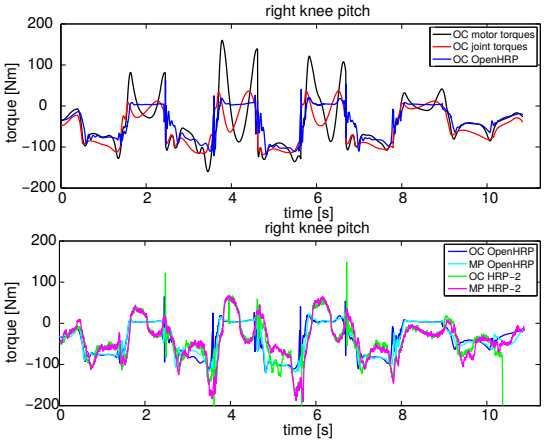

Fig. 9. **Right knee torques.** Exemplary for the behavior in all joints. Comparison between the computational motor torques of the OCP (black), the computational joint torques of the OCP (red), the torques for the OC based motion in OpenHRP (blue), the torques for the MP based motion in OpenHRP (cyan), the torques for the OC based motion on HRP-2 (green) and the torques for the MP based motion on HRP-2 (magenta).

TABLE I
MEAN TORQUES $(\int \tau^2 dt)\frac{1}{2}/(N \cdot t_f)$ [NMS^{-1}]

OC motor t	OC joint t	OC sim	MP sim	OC rob	MP rob
34.1	23.5	15.6	15.6	19.8	19.2

each device (robot or simulator) carries over from OC to MP. Finally, all test motions were repeated at least 4 times on the HRP-2 humanoid robot. In each case, there was no problem with respect to large impact at the ankles while landing. Concerning the reality gap, this is the most important improvement compared to our previous work. A persistent problem however is the divergence of ZMP at the junction of starting and ending motion primitives. This is mostly due to the fact that despite strong constraints on the ZMP included at the OC and MP level, the stabilizer is not always able to recover. As pointed out in Section IV this can be due to the motion generation, but also due to the model inaccuracies. Indeed, such problems do not only appear during the MP generated motions but also when using the OC motions. An interesting way to solve this problem would be to feed such limitations back to the MP and/or the OC level. Practically, it is usually implemented directly in the controllers using inequalities.

VI. Conclusions

We demonstrated, for the first time, that our proposed Combination of Optimal Control, Movement Primitives and Learning (COCoMoPL) yields nearly optimal movements, which can be feasible on a real robot, here HRP-2. An important contribution is the addition of transitions steps, which enable the robot to change step length during walking.

While we predicted feasibility without such transitions steps in our earlier work [6] based on OpenHRP simulations, the real robot could not execute these movements safely. This was mostly due to unbearably high contact forces at the transition points. We showed here, that the addition of a small number of transitions steps to the training data and a new kernel that explicitly accounts for transitions remedies this problem, making the MP generated movements as feasible on the robot as the OCP solutions.

Another appealing feature of COCoMoPL is the ability to generalize a small amount of training data into a continuum of new full-body movements while maintaining near-optimality and feasibility. We deem this feature very useful, because computing OCP solutions is a computationally costly off-line process, whereas MP-based generation is a cheap on-line process that could even be made real-time capable with some code optimizations.

However, our work is far from done. As next steps, we plan to extend our approach in several directions: first, we want to include more types of movements into the MP training data, such as turning, stair climbing, etc. Given that we were able to represent 1656 training trajectories with only 11 MPs, we are hopeful that the inclusion of further training data will not lead to an inflated model. Furthermore, we expect that other movements, which are not as dynamic as our OC steps, will prove to be even more easily executable on the real robot. Such an extended MP model would be an important step towards a fully on-line movement control system for HRP-2. Second, since the ZMP trajectories (Fig.6) were over-smoothed by an RBF kernel with 7 MPs, we are looking into different kernels which can model the cusps in these trajectories. Third, we aim to close the 'reality gap' between OpenHRP and HRP-2 further, so as to achieve feasibility for even more movements. To this end, we plan to incorporate feedback from trials on the robot into the MP learning process. The simplest form of such feedback would be a reinforcement learning signal, or richer feedback in the form of sensor data from the robot that might even be useful for on-line corrections.

ACKNOWLEDGMENT

The research leading to these results has received funding from the EU Program (FP7/2007 - 2013) under grant agreement no 611909 (KoroiBot), from the HGS MathComp Graduate School, from the DFG under project C06 of the SFB/TRR 135 'Cardinal Mechanisms of Perception'. We thank the Simulation and Optimization research group of the IWR at Heidelberg University for giving us the possibility to work with MUSCOD-II and H. Koch for setting up the OC model of HRP-2.

References

[1] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS Workshop, 2012.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2007.

[3] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *9th IFAC World Congress Budapest*, pages 243–247, Oxford, 1984. Pergamon Press.

[4] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[5] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard. Versatile and efficient pattern generator for generalized legged locomotion. In *ICRA*, 2016.

[6] D. Clever, M. Harant, K. H. Koch, K. Mombaur, and D. Endres. A novel approach for the generation of complex humanoid walking sequences based on a combination of optimal control and learning of movement primitives. *RAS*, 83:287–298, 2016.

[7] H. Dai, A. Valenzuela, and R. Tedrake. Whole-body motion planning with simple dynamics and full kinematics. In *Humanoids*, 2014.

[8] A. d'Avella and M. C. Tresch. Modularity in the motor system: decomposition of muscle patterns as combinations of time-varying synergies. In *Advances in Neural Information Processing Systems 14*, pages 141–148. The MIT Press, 2002.

[9] J. Denk and G. Schmidt. Walking primitive databases for perception-based guidance control of biped robots. *European Journal of Control*, 13(2-3):171–188, 2007.

[10] R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer-Verlag New York, Inc., 2007.

[11] M. A. Giese, A. Mukovskiy, A.-N. Park, L. Omlor, and J.-J. E. Slotine. Real-Time Synthesis of Body Movements Based on Learned Primitives. *Lecture Notes in Computer Science*, 5604:107–127, 2009.

[12] A. Herzog, N. Rotella, S. Schaal, and L. Righetti. Trajectory generation for multi-contact momentum control. In *Humanoids*, 2015.

[13] J. Jovic, A. Escande, K. Ayusawa, E. Yoshida, A. Kheddar, and G. Venture. Humanoid and human inertia parameter identification using hierarchical optimization. *IEEE Trans. Robotics*, 32(3):726–735, 2016.

[14] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi. *Introduction to Humanoid Robotics*, volume 101 of *Springer Tracts in Advanced Robotics*. Springer Verlag, 2014.

[15] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. Humanoid robot hrp-2. In *ICRA*, volume 2, pages 1083–1090, 2004.

[16] K. H. Koch. *Using Model-based Optimal Control for Conceptional Motion Generation for the Humannoid Robot HRP-2 14 and Design Investigations for Exo-Skeletons*. PhD thesis, Uni. Heidelberg, 2015.

[17] K. H. Koch, D. Clever, K. Mombaur, and D. Endres. Learning movement primitives from optimal and dynamically feasible trajectories for humanoid walking. In *Humanoids*, pages 866–873, 2015.

[18] K. H. Koch, K. Mombaur, O. Stasse, and P. Souères. Optimization based exploitation of the ankle elasticity of HRP-2 for overstepping large obstacles. In *Humanoids*, pages 733–740, 2014.

[19] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard. Whole-body model-predictive control applied to the hrp-2 humanoid robot. In *IROS*, 2015.

[20] D. B. Leineweber. The theory of MUSCOD in a nutshell. Master's thesis, IWR University of Heidelberg, 1995.

[21] I. Mordatch and E. Todorov. Combining the benefits of function approximation and trajectory optimization. In *RSS*, 2014.

[22] S. Nakaoka, S. Hattori, F. Kanehiro, S. Kajita, and H. Hirukawa. Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms. In *IROS*, pages 3641–3647, 2007.

[23] T. E. Oliphant. Python for scientific computing. *Computing in Science and Engineering*, 9(3):10–20, 2007.

[24] O. E. Ramos, N. Mansard, O. Stasse, C. Benazeth, S. Hak, and L. Saab. Dancing humanoid robots: Systematic use of osid to compute dynamically consistent movements following a motion capture pattern. *IEEE Robotics Automation Magazine*, 22(4):16–26, 2015.

[25] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[26] Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *IROS*, 2012.

[27] P.-B. Wieber, F. Billet, L. Boissieux, and R. Pissard-Gibollet. The HuMAnS toolbox, a homogenous framework for motion capture, analysis and simulation. In *Internal Symposium on the 3D Analysis of Human Movement*, 2006.