Online Walking Motion and Foothold Optimization for Quadruped Locomotion

Alexander W. Winkler, Farbod Farshidian, Michael Neunert, Diego Pardo, Jonas Buchli

***Abstract*— We present an algorithm that generates walking motions for quadruped robots without the use of an explicit footstep planner by simultaneously optimizing over both the Center of Mass (CoM) trajectory and the footholds. Feasibility is achieved by imposing stability constraints on the CoM related to the Zero Moment Point and explicitly enforcing kinematic constraints between the footholds and the CoM position. Given a desired goal state, the problem is solved online by a Nonlinear Programming solver to generate the walking motion. Experimental trials show that the algorithm is able to generate walking gaits for multiple steps in milliseconds that can be executed on a real quadruped robot.**

I. Introduction

The task of controlling a legged system seems trivial on first sight, as humans are able to walk with ease. However, making machines replicate this seemingly easy task is difficult, as demonstrated by the results of the DARPA Robotics Challenge. One limitation is that the underactuated base of legged systems cannot be directly controlled. Additionally, the contact forces with the ground used to generate movement of the base are restricted to pushing motions (unilateral forces) and must lie inside the friction cone.

Optimization has been successfully used in legged locomotion to generate such motions (see Fig. 1). It allows a high-level specification of a desired task and the specific motions of the joints is generated by a mathematical program. Some of the approaches are based on Trajectory Optimization (TO), e.g. transformation of a continuous time Optimal Control (OC) problem into a discrete mathematical optimization problem and solved with off-the-shelf solvers ([1]–[7]). Other approaches solve the OC problem directly through efficient solvers based on Dynamic Programing [8], [9]. These formulations are powerful, since they can deal with nonlinear dynamics and constraints and directly produce the optimal inputs $\boldsymbol{\tau}$ to the system. Some frameworks, originating more from the graphics community, directly formulate a mathematical optimization problem that generates the required motions [10]–[14].

All these successful approaches can generate complex motions for a variety of systems. However, their use to control real systems is not straight-forward. This is partial due to the discrepancy between the underlying rigid body dynamics model and the real robot, which makes it difficult to directly apply the optimized inputs to the real system. Additionally, for high dimensional system such as legged robots, the resulting optimization problems are often difficult and time consuming to solve. As they are generally solved offline, this limits their use for real robots in which continuous replanning and adapting to unknown disturbances is necessary.

Agile & Dexterous Robotics Lab, Institute of Robotics and Intelligent Systems, ETHZ Zurich, Switzerland. email: {winklera, farbodf, neunertm, depardo, buchlij}@ethz.ch

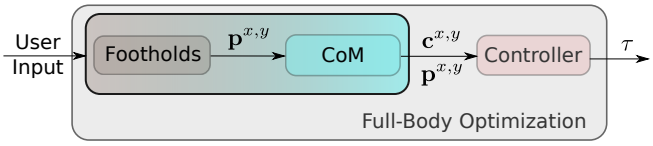

Fig. 1. Different approaches to solve a legged locomotion task: The traditional approach consists of finding footholds, a CoM trajectory and a controller that produces the control inputs τ to track this motion. Full-body Optimization combines these modules, directly producing the inputs to the system. The approach presented in this paper combines the foothold selection and CoM trajectory optimization, but leaves the generation of control inputs to the full-body controller.

A traditional way to reduce the complexity of walking motion generation is to decompose the problem into distinct sub-problems [15]–[18] as seen in Fig. 1. Contact locations are often chosen by a heuristic planner which tries to roughly approximate a path towards the goal. Given these fixed footholds, a stable Center of Mass (CoM) trajectory is found and then tracked by a low-level controller that generates the inputs $\boldsymbol{\tau}$ to the system to execute the walking motion.

Given a set of steps to execute, the generation of the CoM trajectory uses the Zero Moment Point (ZMP) [19] stability criteria: The acceleration of the CoM must be chosen so that the generated ZMP always lies inside the convex hull of the feet in contact. Since the footholds are already given, this optimization has a much smaller search space than full-body optimizations and is easier to solve. This approach is used by [20] to develop a Linear Quadratic Regulator using a preplanned ZMP trajectory for a bipedal robot. In [15] this idea is extended by not specifying the ZMP trajectory to follow in advance, but formulating the stability criteria as a constraint and then solving the Quadratic Program (QP) for the ZMP and CoM trajectory together. However, in this approach the footholds must still be known in advance (e.g. by a footstep planner), since otherwise the stability constraint is non-linear.

The drawback of these hierarchical approaches is that they decouple two inherently connected quantities: the foothold and the body movement. The contact forces and locations are the cause for the body movement, therefore specifying them beforehand based on some heuristics strongly restrains the feasible motions.

To mitigate this, approaches that optimize over both these quantities together have been proposed in [21], [22]. These successful approaches however require that the orientation

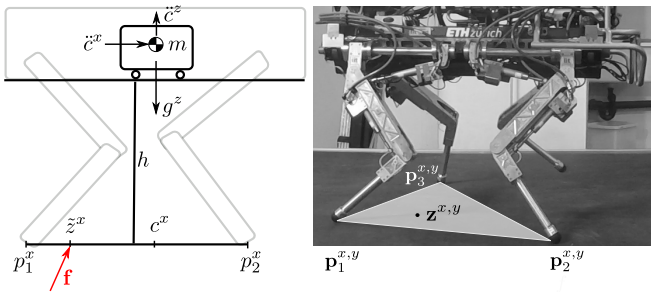

Fig. 2. A quadruped robot modeled as a cart-table. The position of the cart corresponds to the CoM and the width of the cart base (base of support) to the distance between the footholds. The quantity to control the motion of the system is the position (ZMP) $\mathbf{z}$, angle and magnitude $|f|$ of the resultant contact force $\mathbf{f}$ (red). The white triangle shows the base of support while swinging the right-hind leg.

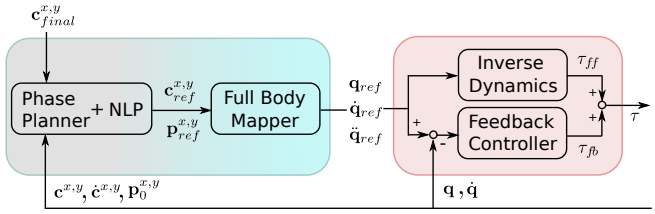

Fig. 3. The complete pipeline from planning to executing optimal walking motions: A *Phase Planner* decides on the amount and sequence of steps, an NLP then solves the optimization problem that is then mapped to the full dimensions of the system by the *Full Body Mapper* and ultimately executed on the system by the low-level controller.

of the support polygon edges is fixed in advance. This is a reasonable assumption for bipeds in single support phase, as the orientation of the feet might not be of high importance. However, for point feet quadruped robots the orientation of the support polygons change with every step.

A. Contribution

The novelties of this paper are *simultaneous* foothold and CoM motion optimization to generate quadrupedal walking, while explicitly enforcing kinematics limits (see Fig. 1). This enables to create and shift arbitrarily oriented support polygons to aid the CoM to reach a desired goal state, without the need of an explicit footstep planner. Additionally, since the generation of the full-state inputs are left to the controller, the optimization problem stays reasonably small to allow for online optimization in milliseconds. We show that the generated motions can be successfully executed on the quadruped robot seen in Fig. 2.

II. Approach

In order to understand the important connection between footholds and the CoM motion and the reasoning in optimizing these together, we briefly recap the physical aspects of legged locomotion, then present the formulation of the developed framework and describe the components.

A. An abstracted view of legged locomotion

Consider a robot modeled as a cart-table as seen in Fig. 2. By applying torque $\boldsymbol{\tau}$ in the joints, reaction forces can be generated in the footholds. The resulting force $\mathbf{f}$ is the equivalent force that has the same effect as the combination of the individual contact forces and directly influences the acceleration of the CoM. The difficulty in the control of legged system arises from the constraints on this resulting contact force. First, the force cannot pull, but only push into the ground (unilateral constraints). Secondly, a resulting force can never act outside of the footholds p_1^x and p_2^x that are producing it. Assuming sufficient friction, the resulting contact force is constrained by $\tilde{z}^x \in [p_0^x, p_1^x]$ and $f^z > 0$.

The Euler equation of motion around $\tilde{z}^x$ of the resultant contact force with no change in angular momentum can be stated as

$$\ddot{c}^x = \frac{g^z + \ddot{c}^z}{h}(c^x - \tilde{z}^x). \tag{1}$$

It can be seen that this equation is only influenced by the position $\tilde{z}^x$ of the resulting contact force, not its magnitude. In order to walk at constant height the above equation with $\ddot{c}^z = 0$ must hold. In this case, the position $\tilde{z}^x$ of the resulting contact force is called the ZMP z^x or Center of Pressure. When more acceleration is desired, the distance between the resultant contact force and the CoM must increase as well to keep the system from rotating around the foothold. If the desired acceleration is too large, the resultant contact force would have to act outside the base of support of the cart table $\tilde{z}^x \notin [p_0^x, p_1^x]$, which is physically not possible. This implies the gait is not dynamically balanced and the system is starting to fall with $\ddot{c}^z \neq 0$. There are two ways to avoid this:

1) Restrict the CoM acceleration $\ddot{\mathbf{c}}^{xy}$, so that it can be generated by a resultant contact force *inside* the base of support.
2) Modify the base of support to accommodate the CoM acceleration. This requires shifting the footholds $\mathbf{p}^{xy}$ to create the support base exactly where it is needed to generate the current body acceleration.

It becomes clear that the body acceleration and the footholds are strongly connected, the footholds serving as an aid to allow the CoM to move where it desires. The following describes the developed framework that finds optimal solutions for these two inherently connected quantities together.

B. Overview

The developed framework as seen in Fig. 3 takes as input the current state and a desired goal state to create full-body states to execute a walking motion for a legged robot. It is composed of a Phase Planner that decides on the amount and sequence of steps (not their location), a NLP solver that produces a CoM motion plan and footholds, a Full Body Mapper that maps the Cartesian output of the NLP to a full-body (base and joints) motion of the robot and finally a controller that generates the required torques to track the motion.

The complete motion is divided into separate phases, within which the feet in contact do not change (Fig. 4). The leg to use to establish contact $\mathbf{p}_i^{xy}$ is determined by the *Phase*

Planner. For quadrupedal walking, a standard sequence to follow is the lateral sequence walk left hind → left front → right hind → right front. Depending on the distance to the goal and the maximum step length the planner adds n steps following this sequence, or none if the distance is close enough to only move the body. Each of these phases j is therefore represented by a fixed set of legs in contact, e.g. in phase 3: {LH, LF, RF}, and the duration T_j of each phase. Additionally, the initial position of the legs in contact is also given and cannot be altered.

In order to generate the continuous CoM motion through optimization, we must first parametrize it by a finite number of decision variables. We therefore represent each phase j as a polynomial defined as

$$\mathbf{c}_j^{xy}(t, \mathbf{a}_j) = \begin{bmatrix} c^x \\ c^y \end{bmatrix} = \sum_{k=0}^{5} \mathbf{a}_{j,k} t^i, \tag{2}$$

where the values in xy-direction are parametrized by a fifth-order polynomial with coefficients $\mathbf{a}_{j,k} \in \mathbb{R}^2$ and the complete phase polynomial by $\mathbf{a}_j \in \mathbb{R}^{12}$. This gives the optimizer enough freedom to shape the motion in order to respect the imposed constraints. As motivated previously, the optimizer is also able to modify the position of the m footholds of each step defined as

$$\mathbf{p}_i^{xy} = \begin{bmatrix} p_i^x & p_i^y \end{bmatrix}^T.$$

Given the n phases, the solver optimizes over the decision variables to find the optimal motion and m footholds, minimizing a performance criteria while fulfilling equality and inequality constraints. This can be stated as the NLP

$$\begin{aligned} &\text{find} && \mathbf{a}_1, \ldots, \mathbf{a}_n,\ \mathbf{p}_1^{xy}, \ldots, \mathbf{p}_m^{xy} \\ &\text{subject to} && (6), (7), (8) \\ & && \mathbf{a}_1, \ldots, \mathbf{a}_n = \arg\min(3). \end{aligned}$$

The following describes the constraints (6), (7), (8) and cost (3) necessary to generate the walking motions.

C. Performance Criteria

The CoM should accelerate as little as possible during the motion, as introduced in [15]. This facilitates tracking, reduces required joint torques and energy consumption and produces more natural looking motions. Therefore, the total xy-acceleration for all phases j (see Fig. 4) is given by

$$J(\mathbf{a}) = \sum_{j=1}^{n} \int_{t=0}^{T_j} \ddot{c}_j^{x^2}(t, \mathbf{a}_j^x) + \ddot{c}_j^{y^2}(t, \mathbf{a}_j^y)\, \mathrm{d}t. \tag{3}$$

D. Dynamic Feasibility Constraint

To ensure that the planned motion is dynamically feasible, we impose constraints between the CoM motion and the footholds as first introduced in [15]. For each stance we represent the base of support by the convex hull of the current footholds. Each of the edges of the convex hull, represented by a line, is defined as $0 = \begin{bmatrix} x & y & 1 \end{bmatrix} \mathbf{n}$. For example, the edge e defined by the footholds $\mathbf{p}_1^{xy}$ and $\mathbf{p}_2^{xy}$ is calculated by

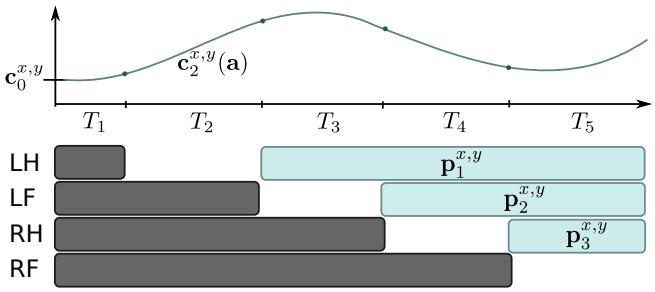

Fig. 4. The structure of the planned motion. The boxes symbolize contact of the respective leg, left-front (LF), right-front (RF), left-hind (LH), right-hind (RH). The gray boxes are the fixed initial contacts at the start of the motion. The optimizer finds the best positions for the $m = 3$ contacts $\mathbf{p}_i^{xy}$ shown in blue, as well as the $n = 5$ CoM polynomials.

$$\mathbf{n}_e = \begin{bmatrix} n^x \\ n^y \\ d_o \end{bmatrix} = \frac{1}{d} \begin{bmatrix} p_1^y - p_2^y \\ p_2^x - p_1^x \\ p_1^x p_2^y - p_2^x p_1^y \end{bmatrix} \tag{4}$$

where d is the distance between the footholds (Euclidean norm). In order to satisfy (1) without resorting to vertical $\ddot{c}^z$ accelerations, the ZMP must be inside the convex hull $\mathcal{P}(\mathbf{p}^{xy})$ of the current footholds, so

$$\mathbf{z}^{xy} = \mathbf{c}^{xy}(\mathbf{a}) - \frac{h}{g^z} \ddot{\mathbf{c}}^{xy}(\mathbf{a}) \in \mathcal{P}(\mathbf{p}^{xy}). \tag{5}$$

In the one-dimensional case this reduces to $p_1^x < z^x < p_2^x$. For two dimensions, the constraint translates to being on the inside of each line composing the convex base of support.

However, the cart-table model does not fully capture the dynamics of the real system. In order to account for this we require the ZMP to stay away from the edge of the support polygon by a margin m. Since the line equation is normalized and represents the orthogonal distance of a point (x,y) to the line, including this margin is straightforward. This condition, that depends both on the CoM and the footholds, is evaluated at every discrete time t_k. For every edge of the current support polygon, the nonlinear inequality constraint

$$\begin{bmatrix} z^x[t_k] & z^y[t_k] & 1 \end{bmatrix} \mathbf{n}_e > m \tag{6}$$

must hold. This equation highlights the strong interconnection between the location of footholds, which affect the line coefficients and the acceleration of the CoM that affects the location of the ZMP. Both of them can be used to ensure stability according and are optimized simultaneously in our formulation.

E. Kinematic Reachability Constraint

The previous constraint gives the optimizer the possibility to either adapt the CoM accelerations to ensure stability or modify the footholds to respect this constraint. This flexibility increases the range of possibly motions, but also necessitates the additional kinematic constraints that the desired foothold must be reachable with the leg from the current CoM position.

Apart from the walking height h and the mass m of the system, the algorithm so far does not need any additional knowledge about the actual system that is being controlled. This constraint however is specific to the kinematics of each system. The farther the quadruped in Fig. 2 moves its CoM towards p_2, the more the leg at p_1 must extend to remain in contact. At some point the difference between p_1 and c is too large and exceeds the kinematic range of the robot. To ensure reachability, we check at every discrete time t_k how far each foothold is from its nominal stance position ${}^{B}\mathbf{p}^{xy}_{nom}$ of the respective leg, expressed in the frame attached at the robots CoM. The constraint for each foothold at time t_k is then formulated as

$$0 < \left\| \mathbf{p}_i^{xy} - \mathbf{c}^{xy}[t_k] - {}^{B}\mathbf{p}^{xy}{}_{nom} \right\|_2 < r \tag{7}$$

This restricts the foothold to deviate less that a radius r from its nominal position. The value for this allowable deviation can be approximated using Inverse Kinematics: The limits of the joints can be mapped to Cartesian space $q_{max} \mapsto (x, y)_{max}$ to obtain an approximation for r. Again, the interconnection between the CoM and the foothold shows itself. This enables the optimizer to explicitly respect kinematics limits by knowing the actual CoM position, without resorting to heuristics used in many footstep planners.

F. CoM Continuity and Goal Constraints

We ensure continuous position and velocity at the phase junctions as well as equality with the initial and final conditions. This enables smooth motions between phases, starting the optimization from the current robot state and handling user specified goal states. Combining the CoM position and velocity as $\mathbf{x} = \begin{bmatrix}\mathbf{c}^{xy} & \dot{\mathbf{c}}^{xy}\end{bmatrix}^T$, the equality constraints of the NLP can be stated as

$$\begin{aligned} \mathbf{x}_1(0) &= \mathbf{x}_{initial} \\ \mathbf{x}_j(T_j) &= \mathbf{x}_{j+1}(0), \quad \text{for } j = 1, \ldots, n \\ \mathbf{x}_n(T_n) &= \mathbf{x}_{final}. \end{aligned} \tag{8}$$

G. Mapping CoM and Footholds to full-body states

The results of the reduced dimension optimization are now mapped back to the full body state. The base state is reconstructed using the optimized CoM motion $\mathbf{c}^{xy}$, assuming that the origin of the base frame is located at the CoM of the robot. Using the constant base height h and zero orientation, the 6 degrees of freedom base state is set to

$$\mathbf{q}_{b,ref}(t) = \begin{bmatrix}0 & 0 & 0 & c^x(t) & c^y(t) & h\end{bmatrix}^T. \tag{9}$$

The joint state of the stance legs can be constructed through each foothold, a predefined polynomial swingleg trajectory and the base position using Inverse Kinematics

$$\mathbf{q}_{j,ref}(t) = \mathrm{IK}(\mathbf{q}_{b,ref}(t), \mathbf{p}^{xy}). \tag{10}$$

Combining the above, the optimized full body state for the controller to track is given by

$$\mathbf{q}_{ref}(t) = \begin{bmatrix}\mathbf{q}_{b,ref}(t) & \mathbf{q}_{j,ref}(t)\end{bmatrix}^T. \tag{11}$$

H. Tracking the planned motion

As can be seen from Fig. 3 the optimization framework produces desired full body accelerations $\ddot{\mathbf{q}}_{ref}$ for the controller to follow. The inverse dynamics controller is responsible for generating required joint torques $\boldsymbol{\tau}$ to create a given acceleration. This is done based on the rigid body dynamics model of the system

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_{ref} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^T\boldsymbol{\tau} + \mathbf{J}_c(\mathbf{q})^T\mathbf{f}, \tag{12}$$

with the joint space inertia matrix $\mathbf{M}$, the effect of Coriolis forces on the joint torques $\mathbf{C}$, the selection matrix $\mathbf{S}$ which prohibits from actuating the floating base state directly and the contact Jacobian $\mathbf{J}_c$ which maps Cartesian contact forces $\mathbf{f}$ to joint torques. Although (6) ensures that the required contact forces $\mathbf{f}$ to produce $\ddot{\mathbf{q}}_{ref}$ are physically feasible, the actual value is not a part of the optimization. This prohibits from solving the underdetermined system of equations directly for $\boldsymbol{\tau}$ and $\mathbf{f}$. However, by knowing which feet are in contact we can calculate a projection operator $\mathbf{P} = \mathbf{I} - \mathbf{J}_c^+\mathbf{J}_c$ [23], [24]. With this we can eliminate the contact forces $\mathbf{f}$ from (12) and calculate the required joint torques through

$$\boldsymbol{\tau} = (\mathbf{P}\mathbf{S}^T)^+ \, \mathbf{P}(\mathbf{M}\ddot{\mathbf{q}}_{ref} + \mathbf{C}). \tag{13}$$

There always exist discrepancies between the used models (cart-table model, inverse dynamics model) and the real system. In order to cope with these it is essential to incorporate feedback into the control loop. We do this by adding an operational space [25] PD controller on the base position and velocity and a low gain PD controller on the joint positions and velocities for more accurate reference tracking.

III. Results

A. PC and Robot Hardware

We use the Interior Point (or Barrier) method solver Ipopt [26] to solve the NLP. The solver first finds values for the decision variables that satisfy all constraints and then gradually increases the importance of the performance metric to obtain optimal solutions. The results where obtained using C++ Code on an Intel Core i7/2.8 GHz Quadcore Laptop. The Jacobian of the constraints and the gradient of the cost function are provided analytically, which is important for performance. The Hessian of the Lagrangian is iteratively approximated by Ipopt through the quasi-Newton L-BFGS algorithm.

We evaluate the performance of this locomotion framework on the hydraulically actuated quadruped robot HyQ [27]. The robot weighs approximately 75 kg, is fully torque controlled and equipped with precision joint encoders and an Inertial Measurement Unit (IMU). State estimation is performed on board, fusing IMU and joint encoder values [28]. The lowest level torque control loop runs at 1000 Hz, while new position, velocity and torque references are set at 250 Hz. The rigid body dynamics model (12) was generated with [29].

TABLE I

RESULTS OF THE ONLINE OPTIMIZATION FOR VARIOUS TASKS

	Walking Direction			
	Stance	Side	Forward	Backward
Goal x,y [cm]	0,0	0,15	40,-20	-30,0
Footholds/Steps	0	4	8	8
Planning horizon [s]	1.5	4.3	7.1	7.1
Optimization Variables	12	68	124	124
Constraints	92	211	327	327
Iterations	2	28	29	29
Objective value	0.09	0.4	0.9	1.59
Nonzeros in Jacobian	708	1756	2844	2844
Solving Time [ms]	**3**	**58**	**72**	**75**

B. Experiments

We demonstrate the performance and robustness of the locomotion framework through a consecutive run of walking motions to different goal positions. For close goal positions the phase planner determines no steps are necessary, so only the CoM is shifted to the commanded positions. The other motions each require various optimized motions (forward walk, sideways walk) to reach the target. The time discretization to enforce the dynamics in (6) is $0.2\,\text{s}$, whereas kinematic reachability (7) is enforced every $0.3\,\text{s}$. We insert an initial stance phase of T_0=1.5 s to allow sufficient time to shift the body before taking the first step. For all tasks, the step duration is $T_i = 0.7\,\text{s}$, the walking height is h=0.58 m and a margin of m=8 cm to reduce the size of the support polygons is specified. We initialize the solver with the robot standing in default stance for the duration of the trajectory.

A quantitative summary of the optimization results for the performed tasks can be seen in Table I and a visualization of the optimized footholds and body trajectory for walking $0.5\,\text{m}$ forward is shown in Fig. 5. Additionally, the reader is strongly encouraged to view the accompanying video at `https://youtu.be/EBW3lpr1tB8`, as it provides the most intuitive way to judge the performance of our framework.

C. Discussion

The following section analyzes the experimental results, highlighting some features of the proposed framework.

1) Footstep selection towards goal: As observed in the tasks, the user specifies only the goal state. This high level input is the actual relevant information, as it is often of secondary importance with which footsteps the robot reaches this goal. The only reason the footsteps follow the direction of the goal, is because the system knows it needs to move the body there (8), but has to maintain stability (6) while staying inside the kinematic range of the legs (7). By not fixing the footsteps beforehand, we allow the optimizer to modify the base of support as needed to perform the desired motion. This allows to perform the side-, diagonal- and backwards-motions without any hand-tuned estimations of where to best place the feet.

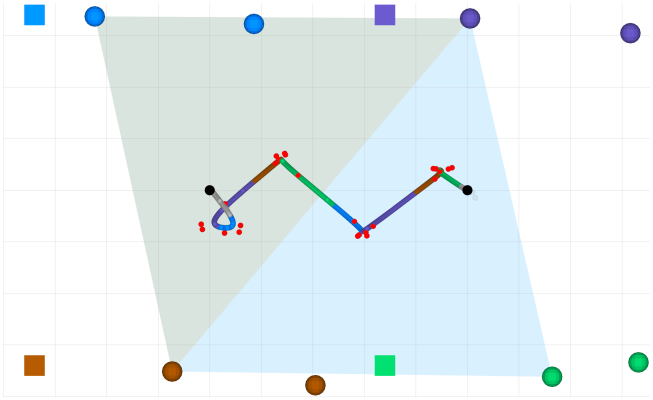

Fig. 5. 8-step quadruped walking motion from left to right to move the CoM to a goal at x=0.5 m. The colors of the CoM trajectory correspond to the swingleg at that time, gray symbolizes a four-leg support (stance) phase. The initial stance is shown by the squares, the optimized footholds by the circles. The two support areas for swinging the right-front green leg and then the left-hind blue leg are shown. The corresponding ZMP (red dots) stays inside these areas by a margin of m=8 cm. It can be seen that by manipulation of the acceleration, the ZMP accumulates towards the center of support polygons for each phase. Additionally, the footholds automatically extend towards the defined goal position in order to create appropriate support areas for the CoM.

2) Explicitly enforcing kinematic limits: In traditional approaches the footsteps are decided by a footstep planner that for instance tries to keep the step length bounded or the footholds close to an *estimated* position of the CoM. However, the actual CoM is only discovered subsequently, when these fixed foothold locations are fed to a CoM planner. This decoupling makes it impossible for the footstep planner to be sure to have chosen footsteps within the kinematic range of the robot. An important benefit of this approach is that it is able to explicitly enforce this constraint (7), as it combines footstep and CoM optimization. This is demonstrated in the in the video, showing that the chosen footsteps never overextend the legs.

3) Online optimization: The optimization on a reduced dimensional model allows us to obtain solving times of magnitudes lower than most full-body optimization approaches. As seen in Table I we are able to generate 7 s body trajectories in less than 100 ms, even though the constraints are nonlinear due to the combined optimization. This is shown in the video by specifying a new goal position on-the-fly and the robot almost immediately starting to approach it. This online planning is an essential factor in controlling real systems, where plans have to be adjusted frequently to account for changing environments or inaccurate execution. The very short solving times demonstrate that the presented framework can be used in a Model-Predictive Control fashion in the future.

IV. CONCLUSIONS

We presented an approach for online and simultaneous optimization of two core and very connected components of locomotion, namely footholds and CoM motion. The results show that our online optimization provides a variety of feasible walking motions for a quadruped to execute. It is

important to note that the demonstrated motions can also be achieved by more hand-tuned traditional ways of specifying footsteps and body movements. However, the generality of the presented approach and its more complete view on the problem has large potential for extension and future work.

As the optimization problem is formulated as an NLP, it is possible to include various types of optimization variables, costs and constraints into the formulation (linear, quadratic, nonlinear, etc.). This allows us to optimize also over the remaining degrees of freedom of the body or the phase durations. Using the body orientation for instance can help in reaching footholds that otherwise exceed the range of motion. Adapting the phase durations will allow the optimizer to take quicker steps when required (e.g. when pushed). Another feature to be explored is the inclusion of terrain costs such as slope and height deviation in the performance criteria. This can allow the robot to navigate over difficult and uneven terrain, selecting those footholds with lowest cost to generate steps to take the robot to a desired goal state. Finally, the speed of the optimization make it possible to replan the motions in a Model-Predictive Control fashion at nearly the control loop frequency. Walking to defined goal states as well as recovering from unexpected external pushes or changes in the environment can all emerge from the same controller.

ACKNOWLEDGMENT

This research has been funded through a Max-Planck ETH Center for Learning Systems Ph.D. fellowship awarded to Farbod Farshidian, a Swiss National Science Foundation Professorship awarded to Jonas Buchli and by the Swiss National Center of Competence in Research Robotics (NCCR Robotics).

REFERENCES

[1] G. Schultz and K. Mombaur, "Modeling and optimal control of human-like running," *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 5, pp. 783–792, 2010.

[2] K. H. Koch, K. Mombaur, and P. Soueres, "Optimization-based walking generation for humanoid robot," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 498–504, 2012.

[3] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of whole-body optimal dynamic multi-contact motions," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1104–1119, 2013.

[4] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2013.

[5] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body Motion Planning with Simple Dynamics and Full Kinematics," *Humanoid Robots*, 2014.

[6] D. Pardo, L. Moller, M. Neunert, A. W. Winkler, and J. Buchli, "Evaluating Direct Transcription and Nonlinear Optimization Methods for Robot Motion Planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 946–953, 2016.

[7] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, no. 3, pp. 3555–3561, 2016.

[8] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[9] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory optimization through contacts and automatic gait discovery for quadrupeds," *IEEE Robotics and Automation Letters (RA-L)*, 2017.

[10] I. Mordatch, M. de Lasa, and A. Hertzmann, "Robust physics-based locomotion using low-dimensional planning," *ACM Transactions on Graphics*, vol. 29, no. 4, p. 1, 2010.

[11] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. van de Panne, "Locomotion skills for simulated quadrupeds," *ACM SIGGRAPH*, p. 1, 2011.

[12] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 1–8, 2012.

[13] P. Hämäläinen, S. Eriksson, E. Tanskanen, V. Kyrki, and J. Lehtinen, "Online Motion Synthesis Using Sequential Monte Carlo," *ACM Trans. Graph.*, 2014.

[14] C. Gehring, S. Coros, M. Hutter, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. A. Hoepflinger, and R. Y. Siegwart, "Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot," *IEEE Robotics & Automation Magazine*, 2016.

[15] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, planning, and control for quadruped locomotion over challenging terrain," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 236–258, 2010.

[16] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," *2008 IEEE International Conference on Robotics and Automation*, pp. 811–818, 2008.

[17] A. W. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. Caldwell, and C. Semini, "Planning and Execution of Dynamic Whole-Body Locomotion for a Hydraulic Quadruped on Challenging Terrain," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 5148–5154.

[18] A. W. Winkler, I. Havoutis, S. Bazeille, J. Ortiz, M. Focchi, R. Dillmann, D. Caldwell, and C. Semini, "Path planning with force-based foothold adaptation and virtual model control for torque controlled quadruped robots," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 6476–6482.

[19] M. Vukobratović and B. Borovac, "Zero-moment point - thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[20] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," *IEEE International Conference on Robotics and Automation*, pp. 1620–1626, 2003.

[21] H. Diedam, D. Dimitrov, P.-B. Wieber, K. Mombaur, and M. Diehl, "Online walking gait generation with adaptive foot positioning through linear model predictive control," in *International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1121–1126.

[22] A. Herdt, H. Diedam, P.-b. Wieber, D. Dimitrov, and M. Diehl, "Online Walking Motion Generation with Automatic Foot Step Placement," *HAL archives-ouvertes*, 2010.

[23] F. Aghili, "A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: Applications to control and simulation," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 834–849, 2005.

[24] M. Mistry, J. Buchli, and S. Schaal, "Inverse dynamics control of floating base systems using orthogonal decomposition," *2010 IEEE International Conference on Robotics and Automation*, no. 3, pp. 3406–3412, may 2010.

[25] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Transactions on Robotics and Automation*, vol. 3, pp. 43–53, 1987.

[26] A. Waechter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming (Springer)*, vol. 106, no. 1, pp. 25–57, 2006.

[27] C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell, "Design of HyQ a hydraulically and electrically actuated quadruped robot," *Journal of Systems and Control Engineering*, vol. 225, no. 6, pp. 831–849, 2011.

[28] M. Bloesch, M. Hutter, M. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State Estimation for Legged Robots - Consistent Fusion of Leg Kinematics and IMU," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

[29] M. Frigerio, J. Buchli, D. G. Caldwell, and C. Semini, "RobCoGen : a code generator for efficient kinematics and dynamics of articulated robots , based on Domain Specific Languages," *Journal of Software Engineering for Robotics*, vol. 7, no. July, pp. 36–54, 2016.